UNIVERSITÉ PARIS 7 – DENIS DIDEROT
UFR D'INFORMATIQUE

Théorie et Pratique de la Cryptanalyse à Clef Publique

MÉMOIRE

présenté et soutenu publiquement le 23 novembre 2007

pour l'obtention du

Diplôme d'Habilitation à Diriger des Recherches
(Spécialité Informatique)

par

NGUYỄN Phong Quang

Composition du jury

Rapporteurs :	Christian Choffrut Arjen K. Lenstra Nigel P. Smart
Examinateurs :	Antoine Joux Jean-François Mestre Adi Shamir Jean Vuillemin
Directeur de recherche :	Jacques Stern
Présidente :	Brigitte Vallée

Recherches effectuées au Laboratoire d'Informatique de l'École Normale Supérieure — UMR 8548

Avant-Propos

Ce document constitue le dossier en vue de l'obtention du diplôme d'habilitation à diriger des recherches, soumis à l'Université de Paris 7 – Denis Diderot. Il est composé de trois parties :

1. Une synthèse sur le thème principal de mes travaux effectués au cours de ces dernières années : *Théorie et pratique de la cryptanalyse à clef publique*. Cette synthèse se compose de trois chapitres visant à préciser le contexte dans lequel s'inscrivent mes contributions : un premier chapitre introduisant la cryptologie ; un second chapitre présentant la cryptanalyse à clef publique ; et un troisième chapitre consacré à la géométrie des nombres algorithmique, qui est peut-être l'outil le plus populaire en cryptanalyse à clef publique ;
2. Un curriculum vitæ et une liste complète de mes publications ;
3. Une annexe regroupant une sélection de mes articles depuis ma thèse de doctorat, soit dans leur version originale, soit dans leur version complète. Ces articles illustrent les différents aspects de la cryptanalyse moderne des systèmes à clef publique :
   – L'algorithmique, notamment celle des réseaux euclidiens (page 95). De nombreux algorithmes sont particulièrement utiles en cryptanalyse.
   – L'étude du problème calculatoire sous-jacent, par exemple via des réductions efficaces à d'autres problèmes mieux connus (page 207).
   – L'exploitation de failles de conception (page 241).
   – L'exploitation de failles d'implémentation (page 299).

Bien que ce document traite essentiellement de la cryptanalyse des systèmes à clef publique, la liste complète de mes publications montre que mes contributions ne se limitent pas à ce seul domaine. J'ai en effet travaillé sur plusieurs autres sujets, dont :

– La conception et l'analyse de sécurité de systèmes à clef publique ;
– La cryptanalyse de systèmes à clef secrète : codes d'authentification à base de fonction de hachage, et chiffrement par flot.

Table des matières

Partie I Théorie et pratique de la cryptanalyse à clef publique

Chapitre 1
La Cryptologie Moderne : Fondements et Perspectives **1**

1.1 Introduction 2
1.2 Qu'est ce que la cryptologie ? 3
1.2.1 Quelques définitions 3
1.2.2 Repères historiques 3
1.2.3 Qu'est-ce que l'impossible ? 4
1.3 La cryptographie sans secret 4
1.3.1 Fonction à sens unique 4
1.3.2 Fonction de hachage 5
1.3.3 Générateur d'aléa 6
1.4 La cryptographie symétrique ou à clef secrète 6
1.4.1 Chiffrement et déchiffrement 7
1.4.2 Décryptement 7
1.4.3 Chiffrement par bloc 8
1.4.4 Chiffrement par flot 9
1.4.5 Codes d'authentification 10
1.5 La cryptographie asymétrique ou à clef publique 10
1.5.1 Chiffrement asymétrique 10
1.5.2 Signature numérique 11
1.5.3 Preuve de connaissance 12
1.5.4 Quelle sécurité ? 12

1.5.5 Quelles alternatives à RSA ? 12
1.5.6 La cryptographie asymétrique avec fonctionnalités spécifiques 14
1.6 Conclusion 15

Chapitre 2
La Cryptanalyse à Clef Publique **17**

2.1 Introduction 17
2.1.1 Les hypothèses calculatoires de la cryptographie à clef publique 18
2.1.2 Qu'est-ce que la cryptanalyse ? 19
2.1.3 Organisation 21
2.2 Cryptosystèmes classiques 21
2.2.1 RSA 21
2.2.2 Elgamal 23
2.3 Notions de sécurité 25
2.3.1 Sécurité des signatures numériques 26
2.3.2 Sécurité du chiffrement asymétrique 27
2.3.3 Etat de l'art 27
2.4 Attaques élémentaires 28
2.4.1 Attaques contre les signatures numériques 28
2.4.2 Attaques contre le chiffrement asymétrique 31
2.5 La cryptanalyse moderne 33
2.5.1 L'algorithmique de la cryptanalyse 33
2.5.2 Failles de conception 35
2.5.3 Failles d'implémentation 36
2.6 Conclusion 38

Chapitre 3
La Géométrie des Nombres Algorithmique **39**

3.1 Introduction 40
3.1.1 Les empilements de sphères 40
3.1.2 Généralisations géométriques de l'algorithme d'Euclide 41
3.1.3 Organisation 42
3.2 La géométrie des nombres 42
3.2.1 Réseaux euclidiens 42
3.2.2 Le théorème fondamental de Minkowski 45
3.2.3 Minima successifs et constante d'Hermite 46
3.3 La réduction de réseau 47
3.3.1 Bases réduites 47
3.3.2 Réduction en dimension deux 48

3.3.3 La réduction de Minkowski . . . 49
3.3.4 L'orthogonalisation de Gram-Schmidt . . . 49
3.3.5 La réduction faible . . . 50
3.3.6 La réduction de Hermite, Korkine et Zolotareff . . . 51
3.4 Réseaux aléatoires . . . 52
3.4.1 L'ensemble des réseaux . . . 52
3.4.2 Mesure naturelle sur l'ensemble des réseaux . . . 53
3.4.3 Cas de la dimension deux . . . 53
3.4.4 Propriétés des réseaux aléatoires . . . 54
3.4.5 Génération de réseaux aléatoires . . . 55
3.5 Problèmes algorithmiques . . . 55
3.5.1 Représentation . . . 55
3.5.2 Problème du plus court vecteur . . . 56
3.5.3 Problème du plus proche vecteur . . . 57
3.6 Algorithmes de réduction de réseau ou de plus court vecteur . . . 58
3.6.1 L'algorithme de Lagrange et sa généralisation gloutonne . . . 59
3.6.2 Gram-Schmidt et la réduction faible . . . 61
3.6.3 Les algorithmes d'Hermite et de Lenstra-Lenstra-Lovász . . . 61
3.6.4 Applications à la recherche de plus court vecteur et à la réduction HKZ . . . 69
3.6.5 Généralisations par blocs de LLL . . . 72
3.6.6 Les algorithmes de Babai . . . 77
3.6.7 Faits expérimentaux . . . 78
3.7 Conclusion . . . 79

Partie II Curriculum vitæ et publications

Chapitre 1
Curriculum vitæ **83**

1.1 Formation 83
1.2 Distinctions 84
1.3 Valorisation de la recherche 84
1.3.1 Activités éditoriales et organisation de conférences 84
1.3.2 Conférences invitées 84
1.4 Enseignement 85
1.5 Encadrement de la recherche 85
1.6 Projets de recherche 86

Chapitre 2
Publications **87**

2.1 Édition d'actes de conférence 87
2.2 Chapitres de livre 87
2.3 Articles de revue 87
2.4 Conférences internationales avec comité de lecture 88
2.4.1 Articles invités 88
2.4.2 Congrès FOCS et STOC 88
2.4.3 Congrès CRYPTO et EUROCRYPT 88
2.4.4 Congrès ASIACRYPT 89
2.4.5 Autres congrès de cryptologie 89
2.4.6 Congrès de théorie des nombres algorithmique 90
2.5 Vulgarisation 91
2.6 Brevets 91
2.7 Thèse 91

Partie III Articles joints

Algorithmique des réseaux euclidiens **95**

Annexe A Low-Dimensional Lattice Basis Reduction Revisited **97**
ANTS-VI (2004) . . . . . .

Annexe B An LLL Algorithm with Quadratic Complexity **135**
EUROCRYPT 2005 . . . . . .

Annexe C LLL on the Average **161**
ANTS-VII (2006) . . . . . .

Annexe D Symplectic Lattice Reduction and NTRU **175**
EUROCRYPT 2006 . . . . . .

Annexe E Rankin's Constant and Blockwise Lattice Reduction **191**
CRYPTO 2006 . . . . . .

Réductions polynomiales **207**

Annexe F Noisy Polynomial Interpolation and Noisy Chinese Remaindering **209**
EUROCRYPT 2000 . . . . . .

Annexe G Adapting Density Attacks to Low-Weight Knapsacks **227**
ASIACRYPT 2005 . . . . . .

Failles de conception **241**

Annexe H Learning a Parallelepiped : Cryptanalysis of GGH and NTRU Signatures **243**
EUROCRYPT 2006 . . . . . .

Annexe I The Impact of Decryption Failures on the Security of NTRU Encryption **259**
CRYPTO 2003 . . . . . .

Annexe J On the Insecurity of a Server-Aided RSA Protocol **275**
ASIACRYPT 2001 . . . . . .

Annexe K Cryptanalysis of the RSA Schemes with Short Secret Exponent from Asiacrypt '99 **287**

ASIACRYPT 2000 . . . . . .

Failles d'implémentation **299**

Annexe L The Insecurity of the Digital Signature Algorithm with Partially Known Nonces **301**

Journal of Cryptology (2002) . . . . . .

Annexe M Experimenting with Faults, Lattices, and the DSA **323**

PKC 2005 . . . . . .

Annexe N Can We Trust Cryptographic Software ? Cryptographic Flaws in GNU Privacy Guard v1.2.3 **333**

EUROCRYPT 2004 . . . . . .

Bibliographie **345**

Première partie

Théorie et pratique de la cryptanalyse à clef publique

Table des figures

2.1 Formatage de chiffrement PKCS#1 v1.5, type 02. 37

3.1 L'empilement le plus dense en dimension deux : l'empilement hexagonal. 40
3.2 L'empilement le plus dense en dimension trois. 41
3.3 Un réseau de dimension 2, et une base. 43
3.4 Interprétation géométrique du déterminant. 44
3.5 Valeurs connues de la constante d'Hermite. 46
3.6 Probabilité expérimentale (en pourcentage) que l'algorithme LLL (avec facteur optimal) renvoie un plus court vecteur d'un réseau aléatoire, en fonction de la dimension. 68
3.7 Exponentielle de la valeur moyenne de $\log(\|\mathbf{b}_1\|/\mathrm{vol}(L)^{1/d})/d$ où $\mathbf{b}_1$ est le premier vecteur renvoyé par l'algorithme LLL (avec facteur optimal) sur un réseau aléatoire, en fonction de la dimension. 68
3.8 Temps en secondes (échelle logarithmique) de l'énumération de Schnorr-Euchner, en fonction de la dimension et de la qualité de la base de départ (LLL-réduite ou BKZ-20 réduite). Comparaison avec l'énumération récursive de Kannan. La ligne horizontale indique 1 minute. 70
3.9 Exponentielle de la valeur moyenne de $\log(\|\mathbf{b}_1\|/\mathrm{vol}(L)^{1/d})/d$ où $\mathbf{b}_1$ est le premier vecteur renvoyé par l'algorithme BKZ sur un réseau aléatoire de dimension 220, en fonction de la taille de bloc. 76
3.10 Exponentielle de la valeur moyenne de $\log(\|\mathbf{b}_1\|/\mathrm{vol}(L)^{1/d})/d$ où $\mathbf{b}_1$ est le premier vecteur renvoyé par l'algorithme BKZ sur un réseau aléatoire, en fonction de la dimension, pour des tailles de bloc de 20, 24 et 28. 77
3.11 Temps en secondes (échelle logarithmique) de l'algorithme BKZ sur un réseau aléatoire, en fonction de la dimension, pour des tailles de bloc de 20, 24 et 28. 77
3.12 Comparaison de la courbe de la figure 3.9 avec la courbe correspondant à l'inégalité de Mordell. La courbe Mordell est générée à partir des valeurs exactes connues de $\gamma_\beta^{1/(\beta-1)}$, et des bornes supérieures de [66]. 80

A.1 Lagrange's algorithm. 103
A.2 The η-Lagrange algorithm. 104
A.3 The greedy lattice basis reduction algorithm in dimension d. 105
A.4 The iterative description of the greedy algorithm. 106
A.5 The η-greedy lattice basis reduction algorithm in dimension d. 110
A.6 The Gap Lemma in dimension 2. 117
A.7 A generalisation of the greedy algorithm. 134

B.1 Comparison of different L^3 algorithms. 137
B.2 The L^3 algorithm. 141
B.3 The size-reduction algorithm. 142
B.4 The Cholesky factorization algorithm (CFA). 144

B.5 The lazy size-reduction algorithm. . . . 145
B.6 The L^2 algorithm. . . . 146
B.7 A floating-point size-reduction algorithm. . . . 151
B.8 The modified L^2 algorithm. . . . 158

C.1 The L^3 Algorithm. . . . 163
C.2 The size-reduction algorithm. . . . 164
C.3 Variation of $\frac{1}{d}\log\frac{\|\mathbf{b}_1\|}{(\det L)^{1/d}}$ as a function of d. . . . 167
C.4 Distribution of the local bases after L^3 (left) and deep-L^3 (right). . . . 168
C.5 Distribution of $\mu_{i,i-1}$ (top left), $\mu_{i,i-2}$ (top right), $\mu_{i,i-5}$ (bottom left) and $\mu_{i,i-10}$ (bottom right) for 71-dimensional lattices after L^3. . . . 169
C.6 Number of loop iterations of L^2 as a function of d, for Ajtai-type random bases. . . . 170

D.1 Quality of the input basis ($\log_2 \|\mathbf{b}_i^*\|$ as a function of i) . . . 190

E.1 Transference reduction . . . 202
E.2 "Exponents" of the upper-bound on β_k . . . 204

H.1 The Hidden Parallelepiped Problem in dimension two. . . . 245
H.2 The Hidden Hypercube Problem in dimension two. . . . 250
H.3 The fourth moment for $n = 2$. The dotted line shows the restriction to the unit circle. 251
H.4 Experiments on NTRUSign-251 without perturbation, and without using NTRU symmetries. The curve shows the average number of random descents required to recover the secret key, depending on the number of signatures, which is in the range 80,000–300,000. . . . 253
H.5 Average number of GGH signatures required so that ten random descents coupled with Babai's nearest plane algorithm disclose with high probability a secret vector, depending on the dimension of the GGH challenge. . . . 254

I.1 Wrap probability on a single test for strings of length 45 . . . 270

K.1 *Example of the lattice formed by the vectors $g_{k,i,b}(xX, yY, zZ)$ and $h_{k,j}(xX, yY, zZ)$ when $m = 3$, $t = 1$, and $a = 1$. The matrix is lower triangular. Entries marked with "–" indicate off-diagonal quantities whose values do not affect the determinant calculation. The polynomials used are listed on the left, and the monomials they introduce are listed across the top. The double line break occurs between the $g_{k,i,b}$ and the $h_{k,j}$, while the single line breaks occur between increments of k. The last single line break separates the helper polynomials (top) from the two primary polynomials (bottom).* . . . 295
K.2 *Largest δ (where $d < N^\delta$) for which our attack can succeed, as a function of the system parameters.* . . . 296

M.1 A Modified CLIO Reader . . . 326
M.2 Experimental Set Up . . . 327
M.3 I/O and Current Consumption (Beginning of the Trace of the Command Used to Generate Signatures). . . . 328

N.1 Session key format in OpenPGP. . . . 336
N.2 PKCS#1 v1.5 encryption padding, block type 02. . . . 336
N.3 PKCS#1 v1.5 signature padding, block type 01. . . . 336
N.4 The Wiener table used to generate ElGamal primes. . . . 337
N.5 An attack using a single GPG–ElGamal signature. . . . 339

N.6 The RSA key generation in GnuPG. 341

Chapitre 1

La Cryptologie Moderne : Fondements et Perspectives

« Lots of people working in cryptography have no deep concern with real application issues. They are trying to discover things clever enough to write papers about. »

Whitfield Diffie

Sommaire

1.1 Introduction 2
1.2 Qu'est ce que la cryptologie ? 3
1.2.1 Quelques définitions 3
1.2.2 Repères historiques 3
1.2.3 Qu'est-ce que l'impossible ? 4
1.3 La cryptographie sans secret 4
1.3.1 Fonction à sens unique 4
1.3.2 Fonction de hachage 5
1.3.3 Générateur d'aléa 6
1.4 La cryptographie symétrique ou à clef secrète 6
1.4.1 Chiffrement et déchiffrement 7
1.4.2 Décryptement 7
1.4.3 Chiffrement par bloc 8
1.4.4 Chiffrement par flot 9
1.4.5 Codes d'authentification 10
1.5 La cryptographie asymétrique ou à clef publique 10
1.5.1 Chiffrement asymétrique 10
1.5.2 Signature numérique 11
1.5.3 Preuve de connaissance 12
1.5.4 Quelle sécurité ? 12
1.5.5 Quelles alternatives à RSA ? 12
1.5.6 La cryptographie asymétrique avec fonctionnalités spécifiques 14
1.6 Conclusion 15

Ce chapitre est une version légèrement modifiée du panorama [269] écrit avec Jacques Stern.

1.1 Introduction

Dans la société de l'information en émergence, l'usage de la cryptologie s'est banalisé. Téléphones mobiles, cartes bleues, titres de transports, cartes vitales, décodeurs, Internet, consoles de jeu, on ne compte plus les objets de la vie courante qui incorporent des mécanismes cryptographiques. Les algorithmes cryptographiques nous assurent que personne ne peut téléphoner à nos frais, intercepter notre numéro de carte de paiement sur la toile, accéder aux données confidentielles de notre carte vitale, *etc.* Sans le recours à des mécanismes cryptographiques implantés de façon correcte, il est impossible d'éliminer les fraudes les plus sérieuses et les atteintes les plus graves à la confidentialité.

Si la cryptologie joue un rôle essentiel dans la sécurité des systèmes d'information, son impact doit cependant être relativisé. Il ne peut y avoir de sécurité sans cryptologie, mais la cryptologie ne suffit pas à elle seule à garantir la sécurité. Bien que les failles de sécurité n'aient souvent aucun rapport avec la cryptographie mise en œuvre, on rencontre dans la pratique beaucoup de mauvaise cryptographie, pour diverses raisons :

- La cryptographie est une discipline fascinante, qui tend à attirer des autodidactes peu au fait des progrès scientifiques des trente dernières années ; leurs algorithmes fétiches peuvent typiquement être cassés en quelques minutes par un expert.
- La cryptographie est un domaine de haute technologie, à la frontière entre plusieurs disciplines, en particulier les mathématiques et les sciences informatiques. Il est malheureusement très facile de faire des erreurs, or la moindre erreur peut être fatale.
- La cryptographie peut être volontairement bridée. C'est le cas lorsqu'il existe des restrictions réglementaires sur l'utilisation, la fourniture, l'importation ou l'exportation de produits cryptographiques qui limitent le niveau de sécurité. Si heureusement ces contraintes sont de plus en plus allégées (l'exemple des États-Unis et de la France en témoigne), la mise-à-jour des produits peut prendre beaucoup de temps et avoir un coût prohibitif. C'est aussi le cas quand des contraintes technologiques antérieures ont fortement limité le niveau de sécurité obtenu.
- Progrès en cryptanalyse : des techniques de plus en plus sophistiquées sont développées pour attaquer des systèmes cryptographiques, mais aussi heureusement pour améliorer leur sécurité.
- Progrès en puissance de calcul : la loi de Moore prédit qu'à coût égal, la puissance de calcul des ordinateurs classiques double tous les 18 mois. On sait en outre que si des moyens de calcul alternatifs tels que des ordinateurs quantiques ou certains circuits dédiés voient le jour, on pourra résoudre certains problèmes de façon beaucoup plus efficace, ce qui remettrait en cause la sécurité de certains systèmes cryptographiques.

Ces dernières années, le phénomène le plus marquant en cryptologie a sans doute été un effort de normalisation exceptionnellement enraciné dans les recherches les plus récentes : plusieurs organisations gouvernementales ou internationales ont sélectionné des algorithmes cryptographiques totalement spécifiés, suite à des appels d'offres ouverts. Du coup, la recherche académique s'est rapproché du monde industriel, ce qui a permis de renouveler des pans entiers de la cryptographie, ouvrant au passage de nouvelles perspectives, tant au niveau de la sécurité et des performances, que de l'éventail des services. Après trois décennies riches en découvertes, la recherche en cryptologie aurait-elle atteint l'âge de raison, ou la normalisation marquerait-elle le début d'un nouvel âge d'or ?

Dans ce chapitre, nous tenterons de faire un état de l'art de la cryptologie, et d'esquisser les perspectives de la recherche fondamentale. Auparavant, nous rappelerons les fondements de la cryptologie et ses principales fonctionnalités.

1.2 Qu'est ce que la cryptologie ?

1.2.1 Quelques définitions

La *cryptologie* est informellement la science des messages secrets. Longtemps restreinte aux usages diplomatiques et militaires, elle est maintenant une discipline scientifique à part entière, dont les applications sont si vastes aujourd'hui qu'il est difficile de définir *a priori* ce qui relève ou non de la cryptologie. Historiquement, la cryptologie a pour objet l'étude des méthodes permettant d'assurer les services d'intégrité, d'authenticité et de confidentialité dans les systèmes d'information et de communication. Elle recouvre aujourd'hui l'ensemble des procédés informatiques devant résister à des adversaires, aussi redoutables et aussi mal intentionnés soient-ils, qui ne respectent pas les « règles du jeu ».

Un service d'*intégrité* garantit que le contenu d'une communication ou d'un fichier n'a pas été modifié de façon malveillante. Par exemple, on peut souhaiter vérifier qu'aucun changement du contenu d'un disque dur n'a eu lieu, ou qu'un fichier téléchargé n'a pas été corrompu lors de la transmission.

Un service d'*authenticité* garantit l'identité d'une entité donnée ou l'origine d'une communication ou d'un fichier. Lorsqu'il s'agit d'un fichier et que l'entité qui l'a créé est la seule à avoir pu apporter la garantie d'authenticité, on parle de *non répudiation*. Le service de non-répudiation est réalisé par une signature numérique, qui a une valeur juridique en France depuis la loi du 20 mars 2000 [275].

Un service de *confidentialité* garantit que le contenu d'une communication ou d'un fichier n'est pas accessible aux tiers. Des services de confidentialité sont offerts dans de nombreux contextes, notamment :

- en téléphonie mobile, pour protéger les communications dans la partie "aérienne" ;
- en télévision à péage pour réserver la réception des données aux abonnés ;
- dans les navigateurs, par l'intermédiaire du protocole SSL (*Secure Socket Layer*), dont l'activation est généralement indiquée par l'apparition d'un cadenas dans la fenêtre.

La cryptologie se partage en deux sous disciplines, également importantes : la *cryptographie* dont l'objet est de proposer des méthodes pour assurer les services définis plus haut et la *cryptanalyse* qui recherche des failles dans les mécanismes ainsi proposés.

1.2.2 Repères historiques

L'ouvrage [330] distingue trois périodes dans l'histoire de la cryptologie. L'âge *artisanal* part des origines : Jules César utilisait, semble-t-il un mécanisme de confidentialité rudimentaire, où chaque lettre d'un message était remplacée par celle située trois positions plus loin dans l'alphabet. La méthode se généralise en opérant une permutation quelconque de l'alphabet, et prend le nom de *substitution*. Une autre méthode, dite de *transposition*, change l'ordre des lettres ; elle a été mise en œuvre au Moyen Age notamment par un dispositif appelé "grille de Cardan". De façon générale, jusqu'au début du vingtième siècle, la cryptographie était affaire de substitutions et de transpositions. On opérait d'ailleurs fréquemment des substitutions non seulement sur des lettres mais sur des mots, en s'aidant d'une sorte de dictionnaire à double entrée, nommé *code* ou *répertoire*. La cryptanalyse, quant à elle, utilisait des méthodes statistiques simples, fondées principalement sur la fréquence des lettres ou des suites de deux lettres (digrammes) dans un texte.

L'âge *technique* garde les substitutions et les permutations mais les met en œuvre à l'aide de machines mécaniques ou électro-mécaniques. Les plus célèbres sont la Hagelin et l'Enigma utilisée par l'armée allemande durant la seconde guerre mondiale. La complexité des méthodes rendues ainsi accessibles étant plus grande, la cryptanalyse devient plus conceptuelle et a aussi recours à des machines. Pour venir à bout de l'Enigma, les Alliés réunissent à Bletchley Park un groupe de scientifiques, dont Alan Turing, inventeur des machines qui portent son nom si chères à l'informatique

théorique. Turing parvient à réaliser une spectaculaire cryptanalyse en la réduisant à une recherche de cas suffisamment restreinte pour être menée par une machine spécialement construite à cet effet. C'est aussi Turing qui, dans le cadre d'une autre cryptanalyse également réussie, fit construire le Colossus, doté d'électronique, et qui peut être considéré comme l'un des ancêtres de nos ordinateurs modernes.

L'âge *paradoxal* couvre les trente dernières années. Il voit l'introduction de mécanismes donnant des réponses positives à des questions *a priori* hors d'atteinte telles que :

- Comment assurer un service de confidentialité sans avoir au préalable établi une convention secrète commune ?
- Comment assurer un service d'authenticité - basé sur la possession d'un secret - sans révéler la moindre information sur le secret ?

La période récente est également marquée par le développement d'une importante communauté de recherche, représentée par l'association internationale pour la recherche cryptologique (IACR). Cette communauté a largement transformé l'image de la cryptologie : elle a apporté une plus grande rigueur à la cryptographie en essayant de produire, autant que possible, des preuves partielles de sécurité, de type mathématique. Elle a également donné un statut nouveau à la cryptanalyse, destinée maintenant à valider les méthodes proposées par une approche systématique, plutôt qu'à donner un avantage compétitif ou stratégique.

1.2.3 Qu'est-ce que l'impossible ?

Pour quantifier la sécurité, il nous faut prédire ce qu'un adversaire ne peut pas faire. En cryptologie, l'impossibilité n'est souvent que calculatoire : elle n'est vraie que si l'on suppose qu'une certaine puissance de calcul est inaccessible dans le monde réel. Il était coutume il y a dix ans d'identifier l'impossibilité calculatoire à un niveau de sécurité de 80 bits : en se fondant sur des ordres de grandeurs physiques, on supposait qu'aucune entité ne pouvait effectuer plus de 2^{80} opérations dans un temps réaliste. Aujourd'hui, le consensus se situe plutôt aux alentours de 128 bits, c'est-à-dire qu'une puissance de calcul de 2^{128} est supposée inatteignable. À titre indicatif, le record de calcul le plus important connu à ce jour est de l'ordre de 2^{64} opérations « élémentaires » : il fut établi en septembre 2002 lors de la recherche exhaustive distribuée d'une clef 64 bits de chiffrement symétrique RC5, qui mobilisa pendant quatre ans jusqu'à trois cent mille machines. On peut en outre estimer la puissance de calcul disponible sur la toile. En effet, un microprocesseur typique est aujourd'hui cadencé à 2GHz, ce qui signifie qu'il dispose de 2 milliards de cycles par seconde, soit $2 \times 10^9 \approx 2^{31}$ opérations élémentaires par seconde. Le nombre d'ordinateurs personnels vendus par an est estimé à environ 200 millions, soit de l'ordre de 2^{28}. Comme une année dure $24 \times 3600 \times 365 = 31536000 \approx 2^{25}$ secondes, on en déduit que la puissance de calcul vendue par an sous formes d'ordinateurs personnels est de l'ordre de $2^{25} \times 2^{28} \times 2^{31} = 2^{84}$ opérations élementaires. Si la loi de Moore continue à se vérifier, ce nombre doit être multiplié par deux tous les dix-huit mois.

1.3 La cryptographie sans secret

Aussi paradoxal que cela puisse paraître, la science du secret s'appuie en partie sur des primitives ne faisant appel à aucun secret : les fonctions de hachage et les générateurs d'aléa, qui sont des cas particuliers de fonctions dites à sens unique.

1.3.1 Fonction à sens unique

Une *fonction à sens unique* est informellement une fonction f simple à calculer mais qui est impossible à inverser d'un point de vue calculatoire : pour tout y choisi aléatoirement dans l'ensemble image, on ne peut calculer un antécédent x tel que $y = f(x)$. La théorie de la complexité en propose

une définition rigoureuse : une fonction f est à sens unique si toute machine de Turing polynomiale probabiliste ne peut inverser cette fonction qu'avec probabilité de succès négligeable à mesure que la taille de l'entrée augmente, l'entrée étant la donnée y, et la sortie souhaitée un élément x tel que $y = f(x)$. L'impossibilité calculatoire devient ainsi une notion asymptotique. Concrètement, si l'on souhaite un niveau de sécurité d'au moins 128 bits, l'ensemble de départ d'une fonction à sens unique doit au moins contenir 2^{128} éléments, et l'on veut que la méthode la plus efficace pour inverser la fonction soit la recherche exhaustive sur l'ensemble de départ.

Il arrive fréquemment qu'une fonction à sens unique doive satisfaire des propriétés plus fortes. Intuitivement, on aimerait qu'une fonction à sens unique masque toute information sur ses entrées : la sortie devrait idéalement se comporter comme une variable parfaitement aléatoire. Si un adversaire prend connaissance de la valeur $f(x)$, il ne devrait obtenir aucune information sur x ou tout autre antécédent de $f(x)$.

1.3.2 Fonction de hachage

Une *fonction de hachage cryptographique* H est une fonction à sens unique calculant efficacement un condensé de taille fixe à partir d'un message formé d'une suite de bits quelconque. Par définition, une telle fonction ne peut être injective : elle comporte même une infinité de *collisions*, c'est-à-dire des paires de messages m et m' distincts mais tels que $H(m) = H(m')$. D'un point de vue cryptographique, on souhaite cependant qu'il soit impossible de calculer en pratique de telles collisions. Cette propriété empêche ainsi la substitution d'un message à un autre, si le condensé est conservé séparément, ce qui permet d'assurer un service d'intégrité, objectif initial des fonctions de hachage.

Le célèbre paradoxe des anniversaires de la théorie des probabilités montre qu'une fonction de hachage cryptographique doit renvoyer des condensés d'au moins $2n$ bits si l'on veut un niveau de sécurité supérieur à 2^n. Ainsi, pour une sécurité en 128 bits, il faut des condensés d'au moins 256 bits. Cette condition nécessaire sur la taille des condensés n'est bien entendu pas suffisante. Dans le modèle dit de l'*oracle aléatoire*, une fonction de hachage idéale agit comme une boîte noire renvoyant des données ayant une distribution parfaitement aléatoire, décorrélée de celle des entrées. Par exemple, les condensés de deux messages ne diffèrant que d'un seul bit devraient n'avoir aucun rapport, et un condensé ne devrait révéler aucune information sur ses antécédents.

Les fonctions de hachage sont d'une importance cruciale pour de nombreuses applications cryptographiques, notamment les signatures numériques, le chiffrement asymétrique. En signature, une fonction de hachage permet de compresser les messages sans dégradation de la sécurité : au lieu de signer des messages de taille quelconque, il suffit de signer des condensés (de taille fixe) des messages, ce qui est utile parce qu'un algorithme de signature n'accepte souvent pas des messages de taille arbitraire, et requiert en général beaucoup plus de temps qu'un simple hachage. En chiffrement asymétrique, les fonctions de hachage servent à renforcer le niveau de sécurité en rendant bien plus imprédictibles les messages clairs à l'aide de transformations bien choisies. On peut aussi utiliser les fonctions de hachage pour mettre en gage des valeurs : dévoiler $H(x)$ et seulement plus tard x permet de s'engager sur la valeur x sans révéler la moindre information sur x entre les instants où l'on révèle $H(x)$ et x, les autres pouvant vérifier *a posteriori* que $H(x)$ est bien le condensé de x.

Pour souligner l'étendue des applications des fonctions de hachage, notons qu'il a récemment été proposé d'utiliser les fonctions de hachage pour endiguer les fameux *spams* (courriers électroniques non sollicités) : imaginons que tout message électronique m soit nécessairement accompagné d'un certificat c tel que le condensé de la concaténation de m avec c commence par 20 bits nuls. Si la fonction de hachage se comporte comme une fonction aléatoire, il faut en moyenne 2^{20} évaluations de la fonction de hachage pour pouvoir calculer un c convenable pour un m donné, ce qui assure au destinataire que l'expéditeur a passé au moins un certain temps avant de pouvoir envoyer le message m, rendant ainsi coûteux les envois massifs.

La théorie des fonctions de hachage n'est pas encore arrivée à maturité dans la recherche aca-

démique. On ne dispose par exemple pas de méthodologie simple pour concevoir des fonctions de hachage sûres et efficaces. Dans les années 90, plusieurs fonctions de hachage ont été conçues et normalisées, MD5 (128 bits) et SHA1 (160 bits) notamment. Ces fonctions sont largement utilisées, notamment dans la génération des signatures RSA, et permettent de hacher plusieurs dizaines de mégaoctets par seconde sur un ordinateur. Depuis une dizaine d'années, des travaux de cryptanalyse menés par divers chercheurs tendaient à indiquer des faiblesses potentielles de MD5 et SHA1. De fait, en 2005, une chercheuse chinoise Xiaoyun Wang et ses collaborateurs ont pu montrer [352, 351] que la sécurité de MD5 et celle de SHA1 étaient notablement plus faibles qu'attendu. Pour MD5, la recherche de collisions ne prend désormais que quelques minutes : avec un temps de calcul plus élevé mais toujours inférieur au paradoxe des anniversaires, on peut même choisir librement les préfixes des collisions [332]. Pour SHA1, le niveau de sécurité ne dépasse pas 2^{63} au lieu de 2^{80}. Même si la marge demeure pour l'instant suffisante pour les applications, il est urgent que la communauté scientifique propose une nouvelle méthodologie pour concevoir des fonctions de hachage sûres et efficaces. Le NIST a justement lancé [249] le 2 novembre 2007 un appel d'offres public pour développer une nouvelle famille de fonctions de hachage cryptographique, qui sera appelée SHA-3.

1.3.3 Générateur d'aléa

Un générateur d'aléa est en quelque sorte le contraire d'une fonction de hachage. Alors qu'une fonction de hachage renvoie des condensés de taille fixe à partir d'entrées de taille arbitrairement grande, un générateur d'aléa est une fonction à sens unique qui prend en entrée des données de taille fixe et renvoie une suite infinie de bits.

L'entrée correspond à une petite quantité secrète, la *graine*, qui peut par exemple être un mot de passe, ou être produite par des mesures physiques. La sortie quant à elle est utilisée comme une source de bits aléatoires pour bon nombre de mécanismes cryptographiques, citons la génération de clefs cryptographiques ou le chiffrement asymétrique. D'un point de vue cryptographique, on souhaite que la sortie d'un générateur d'aléa soit indistinguable de bits parfaitement aléatoires : si l'on présente à un adversaire deux suites de bits, l'une pseudo-aléatoire issue d'un générateur d'aléa de graine inconnue et l'autre parfaitement aléatoire, l'adversaire ne doit pouvoir deviner laquelle provient du générateur d'aléa. En particulier, étant donné un nombre arbitrairement grand de bits issus d'un générateur d'aléa dans lequel on supprime le dernier bit, un adversaire ne peut deviner le bit supprimé mieux qu'en tirant à pile ou face.

On connaît aujourd'hui des générateurs d'aléa prouvé sûrs (sous certaines hypothèses), mais ces constructions sont malheureusement encore inefficaces. Dans la pratique, on préfère utiliser des contructions très performantes issues de la théorie des fonctions de hachage et de la cryptographie symétrique, pour lesquelles les propriétés de sécurité ne sont qu'empiriques : aucune attaque réaliste n'est pour l'instant connue. La recherche actuelle vise à améliorer l'efficacité des générateurs prouvés sûrs.

1.4 La cryptographie symétrique ou à clef secrète

Pendant très longtemps, il n'a existé qu'un seul type de cryptographie : la cryptographie symétrique où l'on suppose qu'au moins deux personnes partagent la connaissance d'un même secret, et ont donc alors des rôles symétriques. Cette cryptographie est aussi connue sous le nom de *cryptographie à clé secrète* ou *cryptographie conventionnelle*. Elle est extrêmement répandue pour des raisons historiques mais surtout à cause de ses performances remarquables, comparables à celles du hachage.

1.4.1 Chiffrement et déchiffrement

La cryptographie symétrique est principalement liée aux services de confidentialité. Elle réalise sur les données m une transformation $c = \mathsf{E}_\mathsf{k}(m)$, par l'intermédiaire d'un algorithme de *chiffrement* E. Cet algorithme prend en entrée le message clair m et un paramètre secret k, qu'on appelle la *clé*. Le message m varie dans un ensemble $\mathcal{M}$ et la clé k dans un ensemble $\mathcal{K}$. La restauration du texte clair à partir du *chiffré* ou *cryptogramme* c se fait par un algorithme de *déchiffrement* D_k, prenant en entrée le chiffré et la même clé. On doit avoir $\mathsf{D}_\mathsf{k}(\mathsf{E}_\mathsf{k}(m)) = m$. En général, le chiffré prend sa valeur dans le même espace $\mathcal{M}$ et l'on a aussi $\mathsf{E}_\mathsf{k}(\mathsf{D}_\mathsf{k}(c)) = c$, c'est à dire que les algorithmes E_k et D_k réalisent une permutation de $\mathcal{M}$.

La distinction entre l'algorithme et la clé s'est établie il y a fort longtemps, notamment dans les travaux du cryptologue Auguste Kerckhoffs [180]. Ce dernier a en effet su reconnaître que l'algorithme de chiffrement n'exigeait pas le secret, dans la mesure où il risquait de toutes façons de passer aux mains de l'ennemi. La cryptologie moderne recommande même des méthodes de chiffrement totalement explicites, de manière à ce qu'elles soient évaluées et validées par un débat ouvert entre experts. Du coup, une convention secrète entre entités qui souhaitent communiquer de façon chiffrée, se limite à l'échange d'une clé k.

Il existe aujourd'hui deux types de chiffrement symétrique : le chiffrement par bloc et le chiffrement par flot. Le chiffrement par bloc découpe le message clair en une multitude de blocs relativement grands (par exemple 128 bits) et opère des transformations bien choisies sur des blocs, tandis que le chiffrement par flot considère le message clair comme un flot de bits (ou d'octets), qu'il combine avec un autre flot de bits (ou d'octets) généré de façon pseudo-aléatoire.

1.4.2 Décryptement

L'opération qui consiste à calculer le clair m à partir du chiffré $c = \mathsf{E}_\mathsf{k}(m)$, mais sans la connaissance de la clé k est appelée *décryptement*. La confidentialité est assurée si cette opération est impossible. On distingue divers scénarios possibles d'attaque

- les attaques à chiffré seul, où l'adversaire dispose d'un certain nombre de chiffrés $\mathsf{E}_\mathsf{k}(m_i)$;
- les attaques à clair connu, où l'adversaire dispose d'un certain nombre de chiffrés $\mathsf{E}_\mathsf{k}(m_i)$ et des clairs correspondants m_i ;
- les attaques à clair choisi, où l'adversaire dispose d'un certain nombre de chiffrés $\mathsf{E}_\mathsf{k}(m_i)$ correspondant à des clairs de son choix m_i ; si de plus chaque message m_i est défini en fonction des chiffrés obtenus antérieurement, on parle d'attaque à clair choisi adaptative.

Le lecteur pourra définir d'autres variantes, comme l'attaque à chiffré choisi. Le but de l'attaque est la découverte de la clé ou le décryptement d'un chiffré c, correspondant à un clair dont on ne dispose pas. Les attaques à chiffré seul sont les plus difficiles. Néanmoins, l'adversaire dispose en général d'informations statistiques sur le clair. En d'autres termes, les messages sont créés en suivant une probabilité qui correspond à une distribution sur $\mathcal{M}$, appelée distribution *a priori*. L'interception d'un (ou plusieurs) chiffrés a pour effet de conditionner cette distribution, produisant une distribution *a posteriori* : par exemple si l'on sait qu'un message chiffré provient d'une distribution équiprobable sur les mots “tas”, “sas”, “mur” et si le chiffrement est une substitution de lettres, alors l'interception du chiffré XUV élimine sas. On dit qu'un chiffrement est parfait si les deux distributions coïncident. Le théorème de Shannon énonce que l'espace des clés $\mathcal{K}$ est alors de taille au moins égale à l'espace des messages. Il existe d'ailleurs un mécanisme appelé chiffrement de Vernam ou *one-time pad*, qui assure un tel niveau de sécurité : il consiste à chiffrer un message de b bits m_i à l'aide d'un clé k de b bits également, le chiffré étant le ou exclusif bit à bit défini par $c_i = m_i \oplus \mathsf{k}_i$. Pour autant qu'on génère la clé aléatoirement et qu'on ne l'utilise qu'une fois, cette méthode de chiffrement offre une sécurité absolue, qu'on nomme aussi *inconditionnelle*.

En général, on ne peut utiliser un chiffrement de Vernam et on conserve une même clé k pour chiffrer un certain nombre de messages. La connaissance d'un petit nombre de chiffrés produit alors une distribution conditionnelle qui définit la clé de manière unique. Pour le comprendre, il suffit d'imaginer qu'une algorithme de chiffrement opère sur des mots de huit octets mais qu'on a intercepté quelques chiffrés correspondant à des suites de huit caractères ASCII 7 bits. Pour chaque clé k et pour chaque chiffré intercepté c, la probabilité que $\mathsf{D}_\mathsf{k}(c)$ soit un message bien formé est environ 1/2000. Ce chiffre provient, par un calcul simple, du pourcentage dans chaque octet des caractères ASCII, lequel est de 38.6 %. Si donc on exploite 10 chiffrés l'espace des clés compatibles avec ces chiffrés est réduit d'un facteur environ 2^{-110}. Même pour des clés de 128 bits, on n'a plus que quelques solutions et on tombe rapidement à une seule solution avec quelques chiffrés supplémentaires. De fait la sécurité devient *algorithmique* : on ne peut que demander que, compte tenu de la puissance de calcul dont il dispose, l'adversaire ne puisse déterminer l'unique valeur de la clé. A cet égard, il existe toujours une méthode permettant de retrouver la clé à partir de quelques couples clair/chiffré, m_i, c_i, en nombre suffisant pour assurer l'unicité. Elle consiste à explorer l'espace des clés et à tester pour chaque clé si $\mathsf{E}_\mathsf{k}(m_1) = c_1$. Si le test est positif, on effectue le test analogue sur m_2 et ainsi de suite. On s'arrête quand la clé a été trouvée. En moyenne, on parcourt la moitié de l'espace des clés $\mathcal{K}$.

1.4.3 Chiffrement par bloc

Dans les algorithmes de chiffrement par *bloc*, l'espace des messages est de la forme $\{0,1\}^b$. Autrement dit le clair (comme le chiffré) est une suite de b bits. Des messages de taille supérieure à b sont chiffrés en les complétant à un multiple de b bits, par une règle de formatage convenue et en chiffrant bloc par bloc au moyen d'un mode d'opération. Le mode ECB (*electronic code book*) chiffre successivement chaque bloc. Le mode CBC (*cipher block chaining*), fait le ou exclusif de chaque bloc avec le chiffré précédent avant d'appliquer l'algorithme de chiffrement, soit $c_i = \mathsf{E}_\mathsf{k}(c_{i-1} \oplus m_i)$. On peut convenir que, pour chiffrer le premier bloc m_1, on prend c_0 nul ou ajouter un vecteur d'initialisation IV, transmis en clair, et poser $c_0 = IV$. Le déchiffrement calcule m_i par $c_{i-1} \oplus \mathsf{D}_\mathsf{k}(c_i)$.

Le plus connu des algorithmes de chiffrement est le DES (*Data Encryption Standard*, voir [231]), qui est une version remaniée par la NSA d'un algorithme conçu initialement par IBM dans les années 1970 : ses spécifications sont publiques, mais sa conception est longtemps restée secrète. Le DES opère sur des blocs de 64 bits avec des clés de 56 bits. Il est essentiellement composé d'une suite de 16 tours identiques, chaque tour réalisant une transformation dite de Feistel. Une telle tranformation génére une permutation sur $2n$ bits à partir d'une fonction f dépendant d'une clé k et dont les entrées sont sur n bits. Les $2n$ bits sont séparés en deux blocs de n bits L et R et on pose $L' = R$, $R' = L \oplus f(R)$. Cette fonction est inversible. Les clés de tour sont formés de 48 bits extraits de la clé du DES par une méthode qui dépend du tour considéré. On considère aujourd'hui le DES comme obsolète, principalement à cause de la taille trop réduite de la clé : on a pu construire des machines dédiées, coutant quelques centaines de milliers de dollars, qui retrouvent la clé en quelques heures. Pour renforcer la sécurité, on utilise souvent le triple DES avec deux clés $(\mathsf{k}_1, \mathsf{k}_2)$, la fonction de chiffrement étant dérivée de celle du DES par la formule $\mathsf{E}_{\mathsf{k}_1}(\mathsf{D}_{\mathsf{k}_2}(\mathsf{E}_{\mathsf{k}_1}(m)))$. En prenant $\mathsf{k}_1 = \mathsf{k}_2$, on retrouve le DES.

Le successeur officiel du DES est l'AES [79], choisi après une compétition ouverte aux équipes de recherche industrielles et académiques. C'est un algorithme de chiffrement par blocs dont les blocs ont 128 bits et les clés ont 128, 192 ou 256 bits. L'AES est une suite de r tours, chacun réalisant une suite de permutations et de substitutions dépendant d'une clé de tour et opérant sur une matrice 4×4 d'octets. La valeur de r est fixée à 10 pour les clés de 128 bits, à 12 pour des clés de 192 bits et à 14 pour des clés de 256 bits.

La théorie du chiffrement par bloc s'est beaucoup développée depuis l'apparition du DES. En particulier, deux techniques très puissantes d'attaques statistiques ont été découvertes au début des années 1990 : la cryptanalyse différentielle et la cryptanalyse linéaire. Elles restent les principales

attaques à l'aune desquelles on évalue la sécurité d'un nouvel algorithme de chiffrement par bloc. La notion de sécurité idéale pour le chiffrement par bloc est formalisé par le concept de permutation pseudo-aléatoire : un algorithme de chiffrement de bloc ne fait qu'associer à une clef une permutation sur un bloc, et l'on souhaite que cette permutation soit indistinguable d'une permutation choisie aléatoirement avec distribution uniforme. On ne connaît toujours pas de chiffrement par bloc efficace et prouvé sûr sous des hypothèses satisfaisantes : les meilleurs arguments de sécurité connus sont très limités et bien souvent heuristiques. En ce sens, la théorie du chiffrement par bloc est encore en friche (notamment en ce qui concerne les modes d'opération et les méthodes de formatage), même si elle est dans un état bien plus avancé que celle du chiffrement par flot ou des fonctions de hachage.

1.4.4 Chiffrement par flot

Le chiffrement par flot est en quelque sorte une version pragmatique du chiffrement de Vernam, qui utilise une clé de petite taille et non aussi grande que le message. Dans les algorithmes de chiffrement par flot, une suite d'octets ou de bits r_i est produite à partir de la clé. Cette suite est combinée aux octets ou aux bits du clair m_i pour donner les octets ou les bits du chiffré c_i, suivant la formule $c_i = m_i \oplus r_i$.

RC4 est le plus célèbre algorithme de chiffrement par flot, utilisé notamment dans le protocole SSL de Netscape et dans beaucoup de logiciels. L'algorithme est confidentiel et propriété de la société RSA Data Security Inc. mais les versions publiées n'ont pas été démenties. À partir de la clé de longueur variable, par exemple 128 bits, un tableau S de 256 octets est initialisé et deux compteurs i et j mis à zéro. Pour générer un nouvel octet aléatoire, on applique les opérations suivantes :

$i = (i + 1) \bmod 256$
$j = j + S[i] \bmod 256$
échanger $S[i]$ et $S[j]$
$t = S[i] + S[j] \bmod 256$
retourner $S[t]$

On ne connaît aucune attaque réaliste contre RC4. Toutefois, l'extrême simplicité de ces opérations a ouvert la voie à des attaques contre certains protocoles de communication qui resynchronisaient de façon maladroite le tableau S à partir d'une clé fixe et de données propres à chaque trame. Ce fut le cas notamment dans certaines versions très répandues du protocole WEP de réseaux sans fil. RC4 est cependant un algorithme à part : il ne ressemble à aucun autre algorithme de chiffrement.

Une méthode extrêmement efficace pour produire une suite de bits utilisable pour un chiffrement par flot, notamment dans les environnements matériels se fonde sur les registres à décalages. Ces dispositifs ont L registres, numérotés de 0 à $L-1$, chacun contenant un bit d'état interne. A chaque coup d'horloge, le contenu du registre numéroté 0 est retourné, le contenu s_i du i-ième registre ($i \geq 1$) avance dans le $i-1$-ième. Le dernier registre s_{L-1} reçoit une valeur calculée par une fonction de rétroaction f dépendant de $s_{L-1}, \cdots s_0$, notée $f(s_{L-1}, \cdots s_0)$. Il est clair que si le contenu initial des registres est $[s_{L-1}, \cdots, s_0]$, le bit s_j produit au j-ième coup d'horloge est donné, pour $j \geq L$, par la relation de récurrence

$$s_j = f(s_{j-1}, s_{j-2}, \cdots, s_{j-L})$$

Lorsque f est linéaire on parle de registre à décalages linéaire (LFSR pour *linear feedback shift register*). On sait qu'un unique LFSR n'assure aucune sécurité, mais en combinant plusieurs LFSR à l'aide de fonctions booléennes bien choisies, on peut renforcer notablement la sécurité. La plupart des algorithmes de chiffrement par flot ont été conçus de cette façon, par exemple l'algorithme A5/1 déployé dans les téléphones mobiles GSM, ou l'algorithme E0 du protocole Bluetooth. Toutefois, au contraire de ce qui se passe pour le chiffrement par bloc, la quasi totalité des algorithmes proposés est susceptible d'attaques "académiques" qui, sans nécessairement mettre en cause la sécurité pratique,

découvrent des invariants statistiques en temps inférieure à la sécurité nomiale 2^k censée être garantie par k bits de clé.

Enfin, on peut aussi obtenir du chiffrement par flot à partir d'un algorithme de chiffrement par bloc, et d'un mode d'opération approprié. C'est la méthode utilisée dans la norme UMTS des téléphones mobiles de nouvelle génération.

La théorie du chiffrement par flot reste largement à faire. Les connaissances dans ce domaine sont plutôt heuristiques, et il n'existe pas de norme de chiffrement par flot. Ceci s'explique sans doute par le fait que plus encore que dans d'autres champs de la cryptographie, le chiffrement par flot est longtemps resté confiné à des utilisations militaires ultra-confidentielles. Le projet eStream lancé par le réseau européen Ecrypt devrait changer la donne.

1.4.5 Codes d'authentification

La cryptographie symétrique offre des services d'intégrité et d'authenticité au moyen des codes d'authentification, plus connus par leur acronyme MAC pour *Message Authentication Code*. Un MAC est en quelque sorte une fonction de hachage paramétrée par une clé secrète : il calcule efficacement un condensé de taille fixe à partir d'un message formé d'une suite de bits quelconque et d'une clé. Un adversaire ne connaissant pas la clé ne doit pas être capable de calculer le MAC d'un message donné, même en connaissant un nombre arbitraire de condensés d'autres messages. En révélant le MAC d'un message, on peut ainsi authentifier ce message à tous ceux qui connaissent la clé secrète. Mais un MAC ne garantit pas la non répudiation, puisque la clé secrète est partagée.

On sait construire des codes d'authentification sûrs à partir d'une fonction de hachage ou d'un algorithme de chiffrement par bloc. Par exemple, le CBC-MAC consiste à appliquer un algorithme de chiffrement par bloc avec le mode d'opération CBC, et à ne renvoyer que le dernier bloc sous forme chiffrée. La construction très répandue HMAC permet elle de construire un MAC à l'aide de deux exécutions appropriées d'une fonction de hachage. Cependant, on ne connaît pratiquement aucune construction dédiée de MAC, c'est-à-dire ne reposant pas sur d'autres mécanismes cryptographiques.

1.5 La cryptographie asymétrique ou à clef publique

La cryptographie asymétrique, connue aussi sous le nom de *cryptographie à clé publique*, fut inventée il y a presque trente ans par Diffie et Hellman dans un article aujourd'hui légendaire [82]. Elle se distingue de la cryptographie symétrique à plusieurs titres. La différence est d'abord sémantique : les scénarios de la cryptographie asymétrique font toujours intervenir au moins une personne qui détient un secret que nulle autre ne possède. Mais la différence apparaît également dans la nature des outils manipulés. Alors que la cryptographie symétrique repose essentiellement sur les fonctions booléennes et les statistiques, la cryptographie asymétrique s'appuie massivement sur la théorie des nombres. Il en résulte que les performances de la cryptographie asymétrique sont sensiblement inférieures à celles de la cryptographie symétrique : c'est peut-être le prix à payer pour des fonctionnalités plus puissantes.

1.5.1 Chiffrement asymétrique

Dans le chiffrement symétrique, chiffrement et de déchiffrement utilisent la même clé. Pourtant, rien n'impose dans l'absolu que les clés de chiffrement et de déchiffrement soient identiques. Si l'on pousse à l'extrême le principe de Kerckhoffs, même la clé de chiffrement peut passer sans inconvénient aux mains de l'ennemi. On a dans ce cas une clé de chiffrement publique connue de tous, et une clé de déchiffrement distincte qui reste privée. Le chiffrement devient asymétrique, l'expéditeur et le destinataire n'ayant plus des rôles symétriques : seul le destinataire dispose de la clé privée lui permettant de déchiffrer. Cette possibilité a été découverte par Diffie et Hellman dans [82]. Le

cryptosystème asymétrique le plus répandu au monde est de loin RSA [294], du nom de ses auteurs Rivest, Shamir et Adleman.

RSA s'appuie sur le phénomène suivant : il est facile de produire des nombres premiers arbitrairement grands, mais on ne sait pas factoriser efficacement des nombres composés, surtout les produits de deux nombres premiers de même taille choisis indépendamment. Pour de tels nombres, le record actuel de factorisation est à 200 chiffres décimaux (663 bits), établi en mai 2005. Dans RSA, on commence par sélectionner aléatoirement deux nombres premiers p et q de grande taille (par exemple, 512 bits). On calcule $n = pq$ et un couple d'entiers (e, d) distincts de 1 et tels que $ed \equiv 1 \bmod (p-1)(q-1)$. Le petit théorème de Fermat et le théorème des restes chinois permettent de montrer que pour tout entier x de $\{0, \ldots, n-1\}$:

$$(x^e \bmod n)^d \bmod n = (x^d \bmod n)^e \bmod n = x$$

Ainsi, les fonctions $x \mapsto x^e \bmod n$ et $x \mapsto x^d \bmod n$ sont des permutations de $\{0, \ldots, n-1\}$ inverses l'une de l'autre. On peut donc voir $x \mapsto x^e \bmod n$ comme une fonction de chiffrement sur $\{0, \ldots, n-1\}$, la fonction de déchiffrement étant $x \mapsto x^d \bmod n$. Le couple $\mathsf{pk} = (n, e)$ prend le nom de clé publique et permet le chiffrement, tandis que l'entier $\mathsf{sk} = d$ est la clé privée, qui autorise le déchiffrement.

La description de RSA que l'on vient de donner est celle que l'on trouve dans la plupart des ouvrages de cryptologie. Cependant, on n'implémente jamais RSA tel quel, car ce RSA-là ne satisfait aucune notion forte de sécurité : on peut par exemple montrer que tout chiffré fait fuir au moins un bit d'information sur le message clair, et l'on connaît de nombreuses attaques contre de mauvaises implémentations de RSA (voir [267]). Dans la pratique, on effectue des transformations sur le texte clair avant exponentiation, à l'aide de fonctions de hachage et de générateurs d'aléa, afin de renforcer le niveau de sécurité.

1.5.2 Signature numérique

La signature numérique est en quelque sorte une notion duale au chiffrement asymétrique. Chaque utilisateur dispose toujours d'une clef publique pk et d'une clef privée sk, comme en chiffrement asymétrique. La clef privée sk permet, étant donné un message arbitraire m, d'engendrer une signature s (dépendant de m) telle que n'importe qui peut, en connaissant la signature s, le message m et la clef publique pk, vérifier que s est une signature valide de m, c'est-à-dire, que seule une personne connaissant m et sk a pu fabriquer la donnée s.

Signature numérique et chiffrement asymétrique sont deux notions bien distinctes : l'un n'implique pas forcément l'autre, et réciproquement. Pourtant, tout comme en chiffrement asymétrique, la signature numérique la plus répandue à l'heure actuelle est la signature RSA, des mêmes auteurs que le chiffrement RSA. La signature RSA est d'ailleurs la première signature numérique découverte. En reprenant les notations du cryptosystème RSA précédemment décrit, on note que la connaissance de sk permet de résoudre l'équation

$$X^e = b \bmod n$$

où b est une constante arbitraire, c'est à dire d'extraire des racines e-ièmes arbitraires, ce qui permet de signer des messages arbitraires. L'équation est publique et une solution est publiquement vérifiable. Le RSA permet donc d'offrir le service de non répudiation, hors d'atteinte de la cryptologie symétrique.

De même qu'en chiffrement asymétrique, on n'implémente jamais la signature RSA telle quelle. Afin de renforcer le niveau de sécurité, on effectue des transformations sur le message avant signature, à l'aide de fonctions de hachage et de générateurs d'aléa,

1.5.3 Preuve de connaissance

Puisque la cryptographie asymétrique requiert la conservation d'une clef privée étroitement liée à une clef publique, on peut se demander s'il est possible de convaincre autrui que l'on détient effectivement la clef privée liée à une clef publique donnée, sans pour autant compromettre cette clef privée. Aussi paradoxal que cela puisse paraître, une *preuve de connaissance zero-knowledge* permet de répondre positivement à cette question : il est possible de démontrer à autrui que l'on détient un secret, sans rien révéler sur ce secret, si ce n'est qu'on le connaît. Les preuves de connaissance zero-knowledge sont l'une des découvertes majeures de la cryptographie moderne.

Même si les preuves de connaissance zero-knowledge ont théoriquement de nombreuses applications, notamment pour assurer l'identification, elles restent encore peu utilisées dans la pratique. On leur préfère souvent des solutions plus heuristiques mais plus efficaces à base de MACs ou de signatures. Cependant, la situation va peut-être évoluer : la recherche actuelle vise en effet à améliorer les garanties de sécurité et l'efficacité des preuves de connaissance zero-knowledge.

1.5.4 Quelle sécurité ?

L'une des plus grandes avancées en matière de cryptographie asymétrique depuis son apparition est la méthodologie de la sécurité prouvée, qui complète la cryptanalyse en garantissant en quelque sorte l'absence de faille. Il s'agit dans un premier temps de modéliser la notion même de sécurité, puis de construire des cryptosystèmes prouvés sûrs dans ce modèle, sous des hypothèses mathématiques précises et plausibles. Mais il convient de relativiser la portée des résultats obtenus. Aujourd'hui, la "bonne" notion de sécurité communément admise est ce qu'on appelle l'*indistinguabilité sous des attaques adaptatives à chiffré choisi* pour le chiffrement asymétrique, et la *résistance aux contrefaçons existentielles sous des attaques à message choisi* pour la signature numérique. Pour atteindre une telle notion de sécurité, une approche réductionniste est utilisée, en traduisant une propriété de sécurité en une hypothèse sur la difficulté calculatoire d'un problème bien connu et bien défini comme la factorisation, le logarithme discret, *etc.* Si l'hypothèse calculatoire est vérifiée, le système est sûr. Le principal avantage avec cette approche est que l'on peut clairement identifier l'hypothèse de sécurité, le principal inconvénient étant que l'on n'obtient aucune preuve de sécurité absolue : on a juste remplacé une hypothèse complexe dans un monde complexe par une hypothèse plus claire dans un monde plus élémentaire. Il reste à suivre les progrès dans la résolution des problèmes supposés difficiles et à dimensionner la taille des clés en conséquence : le statut des hypothèses calculatoires évolue avec le temps, et il est souvent difficile de comparer des hypothèses différentes (par exemple, la factorisation et le logarithme discret). En outre, dans la plupart des cryptosystèmes asymétriques pratiques, notamment ceux qui sont normalisés, la traduction (dans l'analyse de sécurité) du monde complexe des cryptosystèmes au monde plus simple des problèmes calculatoires n'est pas nécessairement pertinente pour les tailles de paramètres courantes. Pour contourner ce problème, une idéalisation des fonctions de hachage, connue sous le nom de *modèle de l'oracle aléatoire*, fut introduite par Bellare et Rogaway [24]. Cependant, ce modèle très fructueux n'est qu'une idéalisation : il n'est pas très satisfaisant d'un point de vue théorique, puisqu'il existe des cryptosystèmes théoriquement sûrs dans le modèle de l'oracle aléatoire mais qui ne sont pourtant pas sûrs quel que soit le choix de la fonction de hachage. La méthode revient à faire l'hypothèse supplémentaire que l'attaquant n'exploitera pas les spécificités intrinsèques des fonctions de hachage utilisées. Dans ce modèle idéal, de nombreux systèmes cryptographiques efficaces ont pu être prouvés sûrs, sous des hypothèses calculatoires plausibles.

1.5.5 Quelles alternatives à RSA ?

Le cryptosystème RSA est de loin le cryptosystème asymétrique le plus répandu au monde, que ce soit en chiffrement ou en signature. Mais cela ne signifie pas que RSA soit le seul cryptosystème asymétrique connu, ni que l'on ne doive pas en concevoir d'autres. Trouver des alternatives

à RSA constitue d'ailleurs un domaine de recherche majeur depuis l'apparition de la cryptographie asymétrique, et ce, pour plusieurs raisons :

Sécurité : Il n'est jamais bon de mettre "tous ses œufs dans le même panier". Pour le cas où la factorisation s'avère être un problème plus facile que prévu, ce que l'on ne peut pas prédire malheureusement, on aimerait disposer d'une solution alternative prête à l'emploi. En outre, on sait déjà que la factorisation deviendra facile si des ordinateurs quantiques efficaces voient le jour.

Taille des clés : Les clés RSA deviennent de plus en plus longues. Alors qu'en 1977, on conjecturait qu'un module RSA de 428 bits serait sûr, on sait aujourd'hui factoriser des modules RSA de 663 bits, et la taille minimale recommandée est de 1024 bits (voire beaucoup plus pour une sécurité à très long terme). Comme le meilleur algorithme de factorisation connu a une complexité sous-exponentielle, doubler le niveau de sécurité requiert beaucoup plus que de rajouter un seul bit à la clé, comme en cryptographie symétrique. On connaît aujourd'hui des cryptosystèmes asymétriques avec des clefs bien plus courtes.

Efficacité : L'augmentation de la taille des clefs a une autre conséquence, elle rend le cryptosystème RSA de moins en moins efficace. Dans les implémentations usuelles de RSA, doubler la taille du module RSA ralentit le chiffrement et le déchiffrement d'un facteur multiplicatif de respectivement 4 et 8. Il se peut que les clés RSA deviennent un jour si longues qu'elles rendent le temps de calcul inacceptable.

Les alternatives connues à RSA peuvent se répartir en deux familles, selon le type de stratégie adoptée :

- Raccourcir les clés. Cela peut se faire en remplaçant le problème de la factorisation d'entier par un problème potentiellement bien plus difficile. C'est le cas du problème du logarithme discret pour des variétés algébriques sur des corps finis : si la variété est convenablement choisie, aucun algorithme sous-exponentiel n'est connu, ce qui a conduit au développement de la cryptographie à base de courbes elliptiques [241, 183] (ECC), et ses variantes comme la cryptographie à base de courbes hyper-elliptiques, *etc.* Dans ces cryptosystèmes, la clef privée peut être aussi petite que 160 bits pour une sécurité "minimale" de l'ordre de 80 bits, mais les mathématiques sous-jacentes sont bien plus complexes que pour RSA. Cependant, des mathématiques plus complexes ne signifient pas nécessairement plus de sécurité : par exemple, il a été montré que le logarithme discret pour des courbes de genre élevé ou certaines courbes elliptiques n'était pas plus difficile que le logarithme discret sur les corps finis. Une autre façon de raccourcir les clés est de trouver une représentation compacte des éléments, comme dans le cas des cryptosystèmes LUC et XTR [206]. Pour ces systèmes, la taille de la clé est une fraction de celle de RSA, mais le meilleur algorithme connu pour attaquer ces systèmes reste sous-exponentiel, ce qui limite toute amélioration asymptotique : les clés pour XTR/LUC vont s'allonger beaucoup plus vite que pour ECC. Quand la clé est plus courte que pour RSA, l'efficacité est en général meilleure, même si les opérations de base sont éventuellement plus coûteuses : dans un certain sens, on est prêt à échanger beaucoup de multiplications modulaires par un plus petit nombre d'opérations plus complexes (comme les additions sur une courbe elliptique).
- Utiliser des opérations plus efficaces que l'exponentiation modulaire. Une stratégie répandue s'appuie sur la théorie de la complexité, et plus précisément sur les problèmes dits NP-durs. Les problèmes NP-durs sont des problèmes calculatoires très particuliers dont on a établi qu'ils étaient au moins aussi difficiles que tout problème d'une certaine famille naturelle de nombreux problèmes calculatoires : si jamais un problème NP-dur peut se résoudre efficacement (asymptotiquement parlant), alors tous les problèmes de la famille peuvent aussi se résoudre efficacement, ce qui n'est pas jugé réaliste. Ainsi, il est coutume de considérer que les problèmes NP-durs sont difficiles, au moins dans le pire cas. Il s'avère que plusieurs problèmes NP-durs ne font intervenir que de l'arithmétique élémentaire. La cryptographie a donc essayé

d'utiliser de tels problèmes pour construire des cryptosystèmes, en transformant des instances faciles en des instances potentiellement difficiles. Le plus vieil exemple est le cryptosystème de Merkle-Hellman [234] à base de sac à dos. Bien que ce cryptosystème eût été cassé peu après sa découverte, beaucoup de cryptosystèmes ont ultérieurement tenté d'appliquer les mêmes principes, notamment :

- NTRU [149] et d'autres cryptosystèmes à base de réseaux, qui s'appuient sur certains problèmes algorithmiques de la géométrie des nombres (voir [267]). L'opération élémentaire est la réduction d'un vecteur modulo une base d'un réseau.
- Cryptosystèmes du type McEliece [228] à base de codes correcteurs d'erreurs, où l'opération de base est la multiplication par une matrice binaire.
- Cryptosystèmes à base de systèmes d'équations polynomiales à plusieurs variables sur un corps fini.

De manière générale, tous ces cryptosystèmes sont plus efficaces que RSA, mais le stockage des clés requiert significativement plus de mémoire.

1.5.6 La cryptographie asymétrique avec fonctionnalités spécifiques

Certaines applications nécessitent en fait un chiffrement asymétrique ou des signatures numériques ayant des propriétés supplémentaires particulières, dont voici une liste non exhaustive donnée à titre indicatif :

Chiffrement homomorphe : Lorsque le chiffrement préserve certaines relations, dans le sens où étant donné plusieurs chiffrés, on peut calculer un nouveau chiffré dont le message clair correspondant est relié autres messages clairs des autres chiffrés. Par exemple, dans un cryptosystème homomorphe additivement, étant donnés les chiffrés de deux messages, on peut facilement calculer le chiffré de la somme des deux messages. De tels cryptosystèmes sont utiles pour le vote électronique.

La cryptographie à base d'identité : En pratique, l'un des principaux problèmes de la cryptographie asymétrique est la gestion des clefs publiques, et plus précisément, la façon de garantir l'authenticité de ces dernières. Dans le cas d'Internet, ce problème est actuellement résolu à l'aide de certificats, bien connus des habitués des sites marchands. La cryptographie asymétrique fondée sur l'identité propose une solution alternative en permettant aux clefs publiques d'être directement reliées à l'identité des utilisateurs : toute chaîne de caractères, par exemple une adresse électronique, est une clef publique potentielle. Cela est rendu possible par l'intermédiaire d'une autorité en laquelle tous les utilisateurs ont confiance : l'autorité choisit des paramètres publics, et à chaque fois qu'un utilisateur souhaite enregistrer une clef publique (de valeur arbitraire), l'utilisateur l'envoie à l'autorité, qui lui retourne la clef secrète correspondante. La cryptographie à base d'identité est en plein essor, mais elle n'est pas sans inconvénient. En effet, elle a dû faire appel à des théories mathématiques complexes dont l'aspect algorithmique n'a pas encore fait l'objet de recherches approfondies, comme le couplage de Weil sur les courbes elliptiques. Par ailleurs, l'autorité a alors nécessairement connaissance de toutes les clefs secrètes, ce qui selon le contexte peut être souhaitable ou au contraire inacceptable.

Chiffrement dépistable : Dans un cryptosystème asymétrique dépistable, toute clef publique admet beaucoup de clefs privées distinctes. Ainsi, quand on diffuse un message chiffré, chaque destinataire autorisé peut déchiffrer avec sa propre clef privée. Si une coalition de destinataires se regroupe pour créer une nouvelle clef privée, il existe un algorithme qui permet de dépister les destinataires impliqués, à partir de la nouvelle clef privée. Un chiffrement dépistable permet donc dans une certaine mesure de démasquer les pirates.

Signatures en blanc : Une signature en blanc permet à quelqu'un de faire signer un document à un tiers sans que le tiers en question n'apprenne quoi que ce soit sur le document. Les signatures

en blanc ont des applications en horodatage, en monnaie électronique, en vote électronique, et de manière générale, partout où l'anonymat est requis.

1.6 Conclusion

Au terme de ce tour d'horizon des fondements et perspectives de la cryptologie, deux interprétations contradictoires sont possibles. Suivant la première, la cryptologie a essentiellement résolu les questions posées par la necessité de communiquer de manière sûre dans les réseaux ouverts et a formalisé de manière adéquate les outils qu'elle a forgés. Suivant la seconde, le développement de la cryptologie ne fait que commencer et les défis sont à venir. La réalité participe sans doute de ces deux interprétations. Dans une conférence prononcée à Paris en recevant le Doctorat *honoris causa* de l'École normale supérieure, Adi Shamir a estimé fin 2003 que neuf dixièmes des articles de recherche actuels n'auront pas de conséquences dans la pratique. *A contrario*, cela fait un dixième de l'activité de recherche qui est d'application immédiate ou différée, un pourcentage sans doute exceptionnel en informatique fondamentale.

Chapitre 2

La Cryptanalyse à Clef Publique

« Deciphering is, in my opinion, one of the most fascinating of arts, and I fear I have wasted upon it more time than it deserves. »

Charles Babbage, *Passages from the life of a philosopher* (1864)

Sommaire

2.1 Introduction 17
2.1.1 Les hypothèses calculatoires de la cryptographie à clef publique 18
2.1.2 Qu'est-ce que la cryptanalyse ? 19
2.1.3 Organisation 21
2.2 Cryptosystèmes classiques 21
2.2.1 RSA 21
2.2.2 Elgamal 23
2.3 Notions de sécurité 25
2.3.1 Sécurité des signatures numériques 26
2.3.2 Sécurité du chiffrement asymétrique 27
2.3.3 Etat de l'art 27
2.4 Attaques élémentaires 28
2.4.1 Attaques contre les signatures numériques 28
2.4.2 Attaques contre le chiffrement asymétrique 31
2.5 La cryptanalyse moderne 33
2.5.1 L'algorithmique de la cryptanalyse 33
2.5.2 Failles de conception 35
2.5.3 Failles d'implémentation 36
2.6 Conclusion 38

2.1 Introduction

Nous avons introduit dans le chapitre précédent la cryptographie à clef publique où un utilisateur U possède une paire $(\mathsf{pk}, \mathsf{sk})$ de clefs reliées entre elles : la clef pk est publique, tandis que la clef sk est gardée secrète par U. Nous avons aussi défini ses deux applications principales :

– Le chiffrement asymétrique (appelé aussi chiffrement à clef publique) : n'importe qui peut chiffrer un message à destination de U, en utilisant la clef publique pk de U. Mais seul U est capable de déchiffrer des messages chiffrés, en utilisant sa clef secrète sk.

– Signatures numériques : U peut signer n'importer quel message m, en utilisant sa clef secrète sk. N'importe qui peut vérifier si une signature donnée correspond à un message donné et à une clef publique donnée.

Ces deux fonctionnalités de base sont couramment utilisés pour sécuriser l'Internet. Par exemple, les signatures numériques sont inclus dans les certificats (utilisés tous les jours par les navigateurs), tandis que le chiffrement asymétrique est utilisé pour échanger des clefs secrètes destinés au chiffrement symétrique, comme dans le protocole SSL.

Ce chapitre va présenter la cryptanalyse à clef publique, c'est-à-dire la théorie des attaques de la cryptographie à clef publique. Une première approche en cryptanalyse consiste à étudier voire à infirmer l'hypothèse calculatoire inhérente à tout cryptosystème à clef publique.

2.1.1 Les hypothèses calculatoires de la cryptographie à clef publique

L'utilisateur U possède deux clefs pk et sk reliés entre elles, mais il devrait être calculatoirement difficile de retrouver la clef secrète sk à partir de la clef publique pk. Ainsi, la cryptographie à clef publique requiert l'existence de problèmes calculatoires difficiles. Mais existe-t-il des problèmes calculatoires prouvés difficiles ? C'est une question très difficile sous-jacente à la célèbre conjecture $\mathsf{P} \neq \mathsf{NP}$ de la théorie de la complexité. Au lieu d'essayer de résoudre cette question ouverte majeure, les cryptographes ont adopté une approche plus terre-à-terre en testant plusieurs candidats au fil du temps : si un problème calculatoire résiste aux assauts répétés de la communauté, alors peut-être peut-on raisonnablement supposer qu'il est difficile, bien qu'aucune preuve de sa difficulté ne soit connue ou même espérée. Ajoutons que la théorie de la NP-difficulté porte sur la difficulté dans le pire cas, alors que la cryptographie nécessite quant à elle de la difficulté en moyenne. Les problèmes potentiellement difficiles qui sont utilisés à l'heure actuelle en cryptographie à clef publique peuvent être classés en deux familles.

La première famille de problèmes difficiles regroupe des problèmes pour lesquels il y a très peu d'inconnues, mais dont la taille doit être relativement large pour garantir de la difficulté, ce qui rend les opérations plutôt lentes par rapport à la cryptographie symétrique. Les principaux membres de cette famille sont :

– La factorisation d'entiers, popularisée par RSA [294]. Le record actuel de factorisation pour un nombre RSA (*i.e.* un produit de deux grands nombres premiers de même taille) est la factorisation [19] d'un nombre de 200 chiffres décimaux (663 bits), obtenue avec le crible algébrique (voir [204, 77]) :

2799783391	1221327870	8294676387	2260162107	0446786955	4285375600
0992932612	8400107609	3456710529	5536085606	1822351910	9513657886
3710595448	2006576775	0985805576	1357909873	4950144178	8631789462
9518723786	9221823983	=	3532461934	4027701212	7260497819
8464368671	1974001976	2502364930	3468776121	2536794232	0005854795
6528088349	×	7925869954	4783330333	4708584148	0059687737
9758573642	1996073433	0341455767	8728181521	3538140930	4740185467

  Un problème voisin est celui de la racine e-ième, que l'on présentera avec RSA.
– Le problème du logarithme discret dans des groupes appropriés :
  – Groupes multiplicatifs de corps finis (en particulier de corps premiers), comme dans l'algorithme de signature DSA [248]. Le record actuel pour un calcul de logarithme discret dans un corps premier arbitraire est de 160 chiffres décimaux [182], obtenu avec le crible algébrique.
  – Groupes additifs de courbes elliptiques sur des corps finis. Il y a en fait deux types de courbes elliptiques utilisées aujourd'hui :
    – Des courbes elliptiques aléatoires pour lesquelles le meilleur algorithme de logarithme discret est en racine carrée du plus grand facteur premier de l'ordre de la courbe. Pour celles-ci,

le record de calcul de logarithme discret est de 109 bits [139].
- Des courbes elliptiques spéciales (*e. g.* courbes supersingulières) pour lesquelles un couplage efficace est disponible. D'une part, cela réduit la difficulté du logarithme discret à celle d'une petite extension du corps fini de base, ce qui oblige à choisir des courbes elliptiques plus grosses. Mais cela crée d'autre part de nombreuses fonctionnalités supplémentaires telles que la cryptographie à base d'identité (voir le livre [30]).

On sait depuis les célèbres travaux de Shor [318] qu'un ordinateur quantique peut résoudre ces problèmes en temps polynomial probabiliste, mais on ne sait toujours pas à l'heure actuelle si la construction d'ordinateurs quantiques suffisamment puissants est réaliste ou non : le plus grand nombre jamais factorisé par un ordinateur quantique n'est autre que $15 = 3 \times 5$.

La seconde famille de problèmes difficiles regroupe des problèmes ayant seulement de petites inconnues (par exemple des bits), mais le nombre d'inconnues doit être suffisamment grand pour garantir la difficulté. De tels problèmes sont en général reliés à des problèmes combinatoires NP-difficiles pour lesquels aucun algorithme quantique efficace n'est connu. Les principaux exemples de cette famille sont :

- Les sacs-à-dos et les problèmes de réseaux euclidiens. Dans le problème du sac-à-dos, les inconnues sont des bits. Le cryptosystème de Merkle-Hellman [234], l'une des premières alternatives au cryptosystème RSA, s'appuyait sur le problème du sac-à-dos (appelé aussi problème de la somme partielle). Bien que les cryptosystèmes à base de sac-à-dos n'aient pas vraiment été couronnés de succès (voir [274]), ils ont en quelque sorte ressuscité sous la forme de cryptosystèmes à base de réseau (voir [267]). Parmi ces derniers, le candidat le plus efficace est le cryptosystème NTRU [149], qui est en cours de normalisation à l'IEEE [164], et qui dispose de clefs bien plus petites que les autres systèmes à base de réseau ou de sac-à-dos. Les sacs-à-dos et les problèmes de réseau sont étroitement liés.
- Problèmes de la théorie des codes. Le cryptosystème de McEliece [228] est un cryptosystème relativement naturel, qui s'appuie sur la difficulté du décodage, et qui a plusieurs variantes selon le type de code utilisé. Le cryptosystème de Goldreich-Goldwasser-Halevi [118, 250] peut être vu comme un analogue à base de réseau du cryptosystème de McEliece.
- Systèmes d'équations polynomiales à plusieurs variables sur de petits corps finis. Le cryptosystème de Matsumoto-Imai [221] est considéré comme l'ancêtre de ce que l'on appelle cryptographie multivariable (voir le livre [184]). Afin d'éviter les attaques générales utilisant les bases de Gröbner, le paramètre de sécurité doit être relativement gros. Le principal inconvénient de la cryptographie multivariable concerne sa sécurité : toutes les propositions existantes utilisent des systèmes d'équations ayant une structure très particulière. A l'instar des cryptosystèmes à base de sac-à-dos, beaucoup de cryptosystèmes multivariables ont été cassés à cause de cette structure exceptionnelle. Le dernier exemple en date est la cryptanalyse spectaculaire [86] du schéma de signature SFLASH.

Au niveau de l'efficacité, le principal inconvénient avec cette seconde famille de problèmes concerne la taille globale des paramètres. En effet, mis à part NTRU [149], la taille des paramètres pour de tels problèmes est au moins quadratique en le paramètre de sécurité (voire cubique pour la cryptographie multivariable). NTRU possède une plus petite taille de clef que les autres membres de cette famille car il utilise une représentation compacte, ce qui permet de gagner un ordre de grandeur.

2.1.2 Qu'est-ce que la cryptanalyse ?

Informellement, la cryptanalyse est la science du « cassage de code ». Nous avons souligné que la cryptographie asymétrique nécessitait des problèmes calculatoires difficiles : s'il n'y a aucun problème difficile, il ne peut y avoir de cryptographie asymétrique. Si l'un des problèmes calculatoires mentionnés précédemment s'avérait finalement facile, alors les cryptosystèmes correspondants pourraient être attaqués, puisque l'on pourrait déduire la clef secrète de la clef publique en un temps réaliste. Cela

signifie qu'on peut toujours cryptanalyser en tentant de résoudre le problème calculatoire sous-jacent, tel que la factorisation d'entiers, le logarithme discret, la réduction de réseau, les bases de Gröbner, *etc.* L'auteur a ainsi proposé et étudié de nouveaux algorithmes [263, 264, 103, 270, 106] pour la réduction de réseau, que nous présenterons plus en détails dans le chapitre 3.

Alternativement, on peut essayer d'exploiter les propriétés spéciales des instances cryptographiques du problème calculatoire. C'est particulièrement vrai pour la seconde famille de problèmes difficiles : bien que le problème sous-jacent soit en général NP-difficile, il est possible que ses instances cryptographiques soient plus faciles, car les fonctionnalités cryptographiques imposent une structure inhabituelle. En particulier, cela signifie qu'il pourrait exister une attaque qui ne serve qu'à attaquer le système, mais qui soit inutile pour résoudre le problème général. Cela est d'ailleurs souvent arrivé dans la cryptographie à base de sac-à-dos ou la cryptographie multivariable. Par ailleurs, il peut aussi arriver que les outils génériques pour résoudre le problème général aient un comportement meilleur sur des instances cryptographiques (voir [97] les bases de Gröbner et [267] pour la réduction de réseau). Dans le même esprit, l'auteur a proposé des variantes [104] de l'algorithme LLL pour tirer partie de la structure exceptionnelle des réseaux NTRU.

Cependant, même si le problème calculatoire sous-jacent s'avère effectivement difficile à la fois en général et pour les instances d'intérêt cryptographique, cela ne signifiera pas forcément que le cryptosystème est sûr. Tout d'abord, il n'est pas évident de savoir ce que l'on entend précisément par le terme *sûr*. Est-ce qu'un système de chiffrement dont on pourrait retrouver le premier bit du texte clair pourrait être considéré comme sûr ? Est-ce que la clef secrète est réellement nécessaire pour déchiffrer des textes chiffrés ou signer des messages ? Si un cryptosystème est théoriquement sûr, pourrait-il y avoir d'autres faiblesses de sécurité liées à son implémentation ? Par exemple, si certaines des variables temporaires (comme les nombres pseudo-aléatoires) utilisées pendant les opérations cryptographiques sont partiellement révélées, est-ce que cela pourrait affecter la sécurité du système ? Ces quelques exemples montrent que la cryptanalyse regroupe bien plus que la simple expertise des principaux problèmes algorithmiques : elle étudie toutes les failles de sécurité possibles, qu'elles soient algorithmiques, conceptuelles ou liées à des problèmes d'implémentation. En particulier, la cryptanalyse va aussi s'intéresser à définir et à étudier des environnements réalistes pour des attaques (attaques adaptatives à chiffré choisi, attaques par canaux auxilliaires, *etc.*), ainsi qu'aux objectifs des attaques (recouvrement de la clef, information partielle, forge existentielle, différenciation, *etc.*). En ce sens, le développement de la cryptanalyse est étroitement lié à celui de la *sécurité prouvée*.

Trente ans après l'introduction de la cryptographie à clef publique, nous disposons d'une bien meilleure compréhension de ce qu'est et de ce que n'est pas la sécurité, grâce notamment aux avancées en cryptanalyse. Et c'est l'état de l'art en cryptanalyse qui détermine en grande partie le choix des tailles de clef et des algorithmes. Toute avancée en cryptanalyse peut potentiellement avoir un énorme impact sur la cryptographie mise en œuvre aujourd'hui : ce serait en particulier le cas si un nouvel algorithme de factorisation plus efficace que le crible algébrique voyait le jour.

Terminons cette introduction en remarquant que la cryptanalyse s'est aussi avérée être un excellent importateur de techniques mathématiques en cryptologie. En effet, plusieurs objets mathématiques aujourd'hui couramment utilisés pour la conception de schémas cryptographiques ont d'abord été introduits en cryptologie comme outils cryptanalytiques, par exemple :

- Les réseaux euclidiens, dont la première utilisation cryptologique fut la cryptanalyse [1, 316] du cryptosystème de Merkle-Hellman [234]. Outre la cryptanalyse, les réseaux sont aujourd'hui utilisés dans plusieurs cryptosystèmes (voir [267]), et dans plusieurs preuves de sécurité [321, 102, 59].
- Les courbes elliptiques. On peut considérer dans une certaine mesure que la première utilisation cryptologique des courbes elliptiques est l'algorithme de factorisation ECM de Lenstra [209], qui a historiquement précédé la cryptographie à base de courbes elliptiques [183, 241].
- Les couplages, dont la première utilisation fut cryptanalytique [230], pour démontrer que le

problème du logarithme discret dans certaines courbes elliptiques pouvait se réduire efficacement au problème du logarithme discret dans les corps finis (plus précisément, dans une petite extension du corps de base de la courbe elliptique). Aujourd'hui, en particulier depuis l'apparition du chiffrement à base d'identité [45, 46], les couplages sont des outils très populaires pour la conception de schémas cryptographiques : voir le livre [30] pour des applications positives des couplages en cryptographie.

On peut ainsi dire en quelque sorte que la cryptanalyse sert aussi à élargir le « terrain de jeu » des cryptographes.

2.1.3 Organisation

Tout d'abord, nous présenterons dans la section 2.2 les deux cryptosystèmes à clef publique les plus classiques (et sans doute les plus simples) : RSA [294] et Elgamal dans les corps premiers [108]. Nous ne décrirons donc pas les systèmes issus de la deuxième famille de problèmes difficiles mentionnés dans la section 2.1.1, ni ceux de la cryptographie à base de courbes elliptiques. Mais cela nous permettra d'illustrer facilement les principales notions de sécurité décrites dans la section 2.3, en présentant des attaques élémentaires dans la section 2.4. Et certains des articles joints en annexe s'attaquent justement à RSA et Elgamal, dans des cas particuliers. Enfin, dans la section 2.5, nous ferons un panorama des méthodes de cryptanalyse :

- Les méthodes algorithmiques qui se concentrent sur les hypothèses calculatoires de la cryptographie à clef publique, ou sur les problèmes utiles en cryptanalyse. On peut soit étudier directement des algorithmes, soit ramener un problème donné à un autre problème mieux connu en utilisant des réductions efficaces au sens de la théorie de la complexité.
- La recherche de failles dans les systèmes cryptographiques. Ces failles se situent soit au niveau conceptuel, soit au niveau de l'implémentation.

Tous ces aspects de la cryptanalyse à clef publique seront illustrés par de nombreux articles en annexe.

2.2 Cryptosystèmes classiques

Nous présentons ici les deux cryptosystèmes asymétriques les plus classiques : RSA [294] et Elgamal [108] sur des corps premiers. Ces deux systèmes ont l'avantage d'être très simples à décrire, du moins dans leur version historique. La cryptanalyse a joué un rôle crucial dans la façon dont ces systèmes sont aujourd'hui implémentés.

2.2.1 RSA

Le cryptosystème RSA [294] est le système asymétrique le plus utilisé au monde. Il s'appuie sur la difficulté de la factorisation d'entiers.

Génération de clefs

L'utilisateur sélectionne deux grands nombres premiers p et q (de la même taille) avec distribution uniforme, de sorte que $N = pq$ soit difficile à factoriser. Comme mentionné précédemment, le record de factorisation pour de tels nombres est actuellement de 663 bits pour N. Dans le commerce électronique, les certificats racines utilisés par les navigateurs Internet utilisent des N de 1024 ou 2048 bits.

Puis, l'utilisateur sélectionne un couple (e, d) d'entiers tels que :

$$ed \equiv 1 \pmod{\varphi(N)}, \tag{2.1}$$

où $\varphi(N) = (p-1)(q-1)$ est l'indicateur d'Euler : $\varphi(N)$ est le nombre d'entiers de $\{1, \ldots, N-1\}$ qui sont premiers avec N. Les entiers e et d sont appelés les exposants RSA : e est l'exposant public, tandis que d est l'exposant secret. La clef publique RSA est le couple (N, e), et la clef secrète RSA est d. Les nombres premiers p et q n'ont pas besoin d'être conservés.

Il y a essentiellement trois façons de sélectionner les exposants RSA :

Exposants aléatoires : l'utilisateur sélectionne un entier $d \in \{2, \ldots, \varphi(N) - 1\}$ uniformément au hasard parmi ceux qui sont premiers avec $\varphi(N)$. L'exposant public e est choisi comme l'inverse de d modulo $\varphi(N)$.

Petit exposant public : pour accélérer l'exponentiation publique, l'utilisateur sélectionne un e très petit, si possible avec un faible poids de Hamming. Si e n'est pas inversible modulo $\varphi(N)$, l'utilisateur sélectionne une nouvelle paire (p, q) de nombres premiers, sinon, l'exposant secret d est choici comme l'inverse de e modulo $\varphi(N)$. Les choix les plus populaires sont $e = 3$ et $e = 2^{16} + 1 = 65537$.

Petit exposant secret : pour accélérer l'exponentiation secrète, l'utilisateur sélectionne un d petit, de longueur suffisamment grande pour éviter la recherche exhaustive. Si d n'est pas inversible modulo $\varphi(N)$, un nouveau d est sélectionné. Sinon, l'exposant public e est l'inverse de d modulo $\varphi(N)$. Ce choix de d n'est cependant plus recommandé : on connaît des attaques polynomiales prouvées [356] lorsque $d \leq N^{1/4}$, et des attaques polynomiales heuristiques [43] lorsque $d \leq N^{1-1/\sqrt{2}} \approx N^{0.292\ldots}$, qui fonctionnent en pratique. Ces attaques retrouvent la factorisation de N, à partir uniquement de la clef publique (N, e).

Si l'on connaît la factorisation de N, alors on peut calculer l'exposant secret d à partir de l'exposant public e. En fait, il est bien connu que la connaissance de l'exposant secret d est équivalente à la celle de la factorisation de N. Plus précisément, il fut remarqué dès [294] que si l'on connaît la clef secrète d, alors on peut retrouver la factorisation de N en temps polynomial probabiliste. Il a récemment été prouvé dans [223, 73] que cela peut même se faire en temps polynomial déterministe. Ainsi, retrouver la clef RSA secrète est aussi difficile que factoriser le module RSA public, mais cela ne signifie pas nécessairement que casser RSA soit aussi difficile que la factorisation.

Permutation à trappe

Notons $\mathbb{Z}_N$ l'anneau $\mathbb{Z}/N\mathbb{Z}$, que nous représentons par $\{0, 1, \ldots, N-1\}$. La principale propriété de la génération de clef RSA est la congruence (2.1) qui implique, grâce au petit théorème de Fermat et au théorème des restes chinois, que la fonction d'exponentiation modulaire $x \mapsto x^e$ est une permutation sur $\mathbb{Z}_N$. Cette fonction est appelée la permutation RSA. Il est bien connu que son inverse est la fonction d'exponentiation modulaire $x \mapsto x^d$, d'où le nom de *permutation à trappe* : si on connaît la trappe d, on peut inverser efficacement la permutation RSA. Sans la trappe, le problème d'inversion est présumé difficile, et est connu sous le nom de problème de la racine e-ième (appelé aussi problème RSA) : étant donné un entier $y \in \mathbb{Z}_N$ choisi aléatoirement avec distribution uniforme, trouver $x \in \mathbb{Z}_N$ tel que $y \equiv x^e \bmod N$. L'hypothèse RSA affirme qu'aucun algorithme probabiliste polynomial ne peut résoudre le problème RSA avec probabilité non négligeable.

On ne sait cependant pas si la connaissance de d est réellement nécéssaire pour résoudre le problème RSA. Peut-être existe-t-il une facon alternative d'inverser la permutation RSA, autre que d'élever à la puissance d. D'ailleurs, des résultats partiels de Boneh et Venkatesan [50] suggèrent au contraire que le problème RSA avec un petit e pourrait être plus facile que la factorisation.

Une propriété importante de la permutation RSA est sa multiplicativité. Plus précisément, pour tout x et y dans $\mathbb{Z}_N$:

$$(xy)^e \equiv x^e y^e \pmod{N}. \tag{2.2}$$

Chiffrement asymétrique

Le chiffrement RSA est une simple application de la permutation RSA : on se contente d'appliquer la permutation pour chiffrer. Plus précisément, l'ensemble des messages est $\mathbb{Z}_N = \{0, 1, \dots N-1\}$. Pour chiffrer un message m, on l'élève à la puissance e modulo N, ce qui signifie que le chiffré est :

$$c = m^e \bmod N. \tag{2.3}$$

Pour déchiffrer le chiffré c, on inverse la permutation RSA :

$$m = c^d \bmod N. \tag{2.4}$$

C'est la façon dont le chiffrement asymétrique RSA fut initialement décrit dans [294] et dans de nombreux ouvrages, mais ce n'est pas la façon dont on implémente RSA aujourd'hui dans des produits commerciaux ou dans des normes, à cause de divers problèmes de sécurité, bien que les principes de base soient les mêmes. Il est maintenant communément admis qu'une permutation à trappe ne peut pas être utilisé directement comme chiffrement asymétrique : on doit transformer les messages avant d'appliquer la permutation, comme le fait par exemple OAEP [25, 287].

Signature numérique

La propriété magique de RSA est sa permutation : la plupart des systèmes asymétriques s'appuient plutôt sur une fonction à sens unique à trappe (voir [231]). Il est très facile de construire un schéma de signature numérique à partir d'une permutation à trappe.

Dans la description originale [294], l'ensemble des messages à signer est $\mathbb{Z}_N = \{0, 1, \dots, N-1\}$. La signature d'un message $m \in \mathbb{Z}_N$ est simplement son inverse via la permutation RSA :

$$s = m^d \bmod N. \tag{2.5}$$

Pour vérifier que s est une signature valide de m avec la clef publique (N, e), on vérifie que $s \in \mathbb{Z}_N$ et la congruence suivante :

$$m \equiv s^e \pmod N. \tag{2.6}$$

De même que pour le chiffrement asymétrique, ce n'est pas la façon dont on implémente aujourd'hui les signatures RSA dans les normes ou les produits commerciaux à cause de divers problèmes de sécurité, bien que le principe de base soit le même. Une permutation à trappe ne doit pas être utilisée directement comme schéma de signature : on transforme les messages avant de les signer, à l'aide d'une fonction de hachage, comme le fait FDH (*full-domain hash*) [26, 287] ou PSS (*probabilistic signature scheme*) [26, 287].

On peut remarquer que la transformation des messages pour le chiffrement asymétrique ou les signatures s'appuie sur des fonctions de hachages cryptographiques. Cependant, quand l'article [294] fut publiée, on ne disposait d'aucune fonction de hachage cryptographique. C'est pourquoi de nombreuses solutions *ad hoc* furent développées (et parfois déployées) dans les années 80, avec parfois des conséquences désastreuses. Les normes pour le RSA [197] recommandées par l'entreprise RSA sont : RSA-OAEP pour le chiffrement asymétrique et RSA-PSS pour les signatures.

2.2.2 Elgamal

Alors qu'il n'y a essentiellement qu'un seul cryptosystème RSA, on a beaucoup plus de flexibilité pour le cryptosystème d'Elgamal [108] à base de logarithme discret : il existe beaucoup de variantes selon le groupe ou sous-groupe utilisé, et selon l'encodage des textes clairs et des textes chiffrés. Ici, nous n'allons présenter que la version historique d'Elgamal, c'est-à-dire le cryptosystème d'Elgamal sur un corps premier $\mathbb{Z}_p$, comme décrit initialement dans [108]. Une autre différence significative

avec RSA est l'écart significatif entre le chiffrement Elgamal et la signature Elgamal. Dans RSA, le chiffrement asymétrique et les signatures sont les deux facettes de la permutation à trappe RSA. Comme le chiffrement Elgamal s'appuie sur une fonction à sens unique à trappe dérivée de l'échange de clef de Diffie-Hellman [82], plutôt qu'une permutation à trappe, il ne fournit pas directement de signature efficace. La signature Elgamal est donc assez différente du chiffrement Elgamal : c'est l'ancêtre de la plupart des schémas de signature à base de logarithme discret, comme DSA, ECDSA ou la signature de Schnorr (voir [231]).

Génération de clefs

On sélectionne un grand nombre premier p, de telle sorte que $p-1$ ait au moins un grand facteur premier et que sa factorisation soit connue. On estime alors que le problème du logarithme discret dans $\mathbb{Z}_p^\times$ est difficile. Grâce à la factorisation de $p-1$, l'utilisateur peut calculer un générateur g du groupe $\mathbb{Z}_p^\times$. Il y a essentiellement deux façons de sélectionner g :

Générateurs aléatoires : c'est l'option recommandée où le générateur g est choisi uniformément au hasard parmi tous les générateurs de $\mathbb{Z}_p^\times$.

Petits générateurs : on essaie de petites valeurs de g, comme $g = 2$, pour accélerer les exponentiations de base g. Si aucune ne marche, on essaie un nouveau nombre premier p. Le choix $g = 2$ est fortement déconseillé pour la signature à cause de l'attaque de Bleichenbacher [31].

Les paramètres g et p sont publics et peuvent être communs à plusieurs utilisateurs.

La clef secrète de l'utilisateur est un entier x choisi uniformément au hasard dans $\mathbb{Z}_{p-1} = \{0, 1, \ldots, p-2\}$. La clef publique correspondante est l'entier $y \in \mathbb{Z}_p^\times$ défini par :

$$y = g^x \pmod{p}. \tag{2.7}$$

Beaucoup de variantes d'Elgamal utilisent plutôt un sous-groupe d'ordre premier, plutôt que le groupe $\mathbb{Z}_p^\times$ tout entier. Plus précisément, ils choisissent $g \in \mathbb{Z}_p^\times$ d'ordre premier $q \ll p$ suffisamment grand : la clef secrète x est alors choisie dans $\mathbb{Z}_q$.

Chiffrement asymétrique

Le chiffrement Elgamal peut être vu comme une application de l'échange de clef de Diffie-Hellman [82]. Dans le protocole de base de Diffie-Hellman, Alice et Bob effectuent les opérations suivantes pour établir une clef secrète commune :

- Alice sélectionne un entier $a \in \mathbb{Z}_{p-1}$ uniformément au hasard, et envoie $A = g^a \bmod p$ à Bob.
- Bob sélectionne un entier $b \in \mathbb{Z}_{p-1}$ uniformément au hasard, et envoie $B = g^b \bmod p$ à Alice.
- La clef secrète partagée par Alice et Bob est $s = g^{ab} \bmod p$. Alice peut calculer s comme $s = B^a \bmod p$, tandis que Bob peut calculer s comme $s = A^b \bmod p$.

Pour transformer ce protocole d'échange de clef en un chiffrement asymétrique probabiliste, voyons Alice comme l'utilisateur qui possède la paire (x, y) de clefs définie par (2.7), de telle sorte que $(a, A) = (x, y)$, et voyons Bob comme celui qui souhaite lui chiffrer des messages. Bob connaît la clef publique $y = g^x \bmod p$. L'ensemble des messages est $\mathbb{Z}_p$. Pour chiffrer un message $m \in \mathbb{Z}_p$:

- Bob choisit un entier $k \in \mathbb{Z}_{p-1}$ uniformément au hasard.
- Le chiffré est le couple $(c, d) \in \mathbb{Z}_p^\times \times \mathbb{Z}_p$ defini par

$$c = g^k \pmod{p} \tag{2.8}$$

$$d = my^k \pmod{p} \tag{2.9}$$

Pour voir comment le déchiffrement fonctionne, il suffit de remarquer que grâce à l'astuce de Diffie-Hellman, Alice peut calculer le secret (virtuel) $s = g^{xk} = y^k \bmod p$ à partir de sa clef secrète x et de

la première partie c du chiffré. En effet, on a $s = c^x \bmod p$, comme si la paire (b, B) de Bob dans le protocole de Diffie-Hellman était (k, c). Une fois $y^k \bmod p$ connu, Alice peut retrouver le message m à partir de la deuxième partie d du chiffré, par division.

En d'autres termes, la première partie (2.8) du chiffré crée un secret de Diffie-Hellman $y^k = g^{kx}$ à utilisation unique. La seconde partie (2.9) du chiffré peut être vu comme un masque jetable (en utilisant la multiplication modulaire, plutôt qu'un xor) entre le message et la clef jetable. Le déchiffrement fonctionne en retrouvant la clef jetable grâce à la clef secrète de l'utilisateur et à l'astuce de Diffie-Hellman.

Comme le chiffrement Elgamal [108] est très lié à l'échange de clef de Diffie-Hellman [82], on peut se demander pourquoi il n'est pas apparu dès [82]. L'explication provient peut-être du fait que formellement, le chiffrement asymétrique défini dans [82] était associé à une permutation à trappe, de sorte qu'il soit facile d'obtenir du chiffrement et de la signature : il était supposé implicitement que l'ensemble des textes chiffrés devait être identique à l'ensemble des textes clairs. Mais le chiffrement Elgamal n'utilise pas et ne définit pas de permutation à trappe. Le seul objet utilisé par le chiffrement Elgamal qui ressemble à une permutation est la bijection suivante entre $\mathbb{Z}_p \times \mathbb{Z}_{p-1}$ et $\mathbb{Z}_p^\times \times \mathbb{Z}_p$:

$$(m, k) \mapsto (c, d) = (g^k, my^k).$$

Mais la clef secrète x sert seulement à inverser partiellement cette bijection : étant donnée une image (c, d), on sait retrouver m, mais pas la deuxième partie k, qui est un problème de logarithme discret. Ainsi, ce n'est pas tout à fait une permutation à trappe.

Signature numérique

De façon surprenante, la signature Elgamal [108] n'a rien à voir avec le chiffrement Elgamal [108]. Les seules choses en commun sont la génération de clef, et le fait que le système soit probabiliste.

L'ensemble des messages est $\mathbb{Z}_p$. Pour signer un message $m \in \mathbb{Z}_p$:

- L'utilisateur choisit uniformément au hasard un nombre $k \in \mathbb{Z}_{p-1}^\times$, c'est-à-dire un entier de $\{0, \ldots, p-2\}$ premier avec $p-1$.
- La signature de m est le couple $(a, b) \in \mathbb{Z}_p^\times \times \mathbb{Z}_{p-1}$ défini par :

$$a = g^k \pmod p \tag{2.10}$$
$$b = (m - ax)k^{-1} \pmod{p-1}. \tag{2.11}$$

Pour vérifier une signature (a, b) d'un message m, on vérifie que $(a, b) \in \mathbb{Z}_p^\times \times \mathbb{Z}_{p-1}$ et la congruence suivante :

$$g^m \equiv y^a a^b \pmod p \tag{2.12}$$

La congruence précédente peut se réécrire en :

$$m \equiv ax + b \log a \pmod{p-1}, \tag{2.13}$$

où log désigne le logarithme discret de $\mathbb{Z}_p^\times$ et de base g.

2.3 Notions de sécurité

L'une des plus grandes avancées de la cryptographie asymétrique est l'introduction de notions de sécurité rigoureuses et pertinentes pour le chiffrement et la signature. Rigoureuses, car ces notions peuvent être formellement définies à l'aide du langage de la théorie de la complexité. Pertinentes, car l'histoire relativement jeune de la cryptographie asymétrique a montré qu'elles semblent capturer la

bonne notion de sécurité, puisque des attaques réalistes ont été exhibées lorqu'on affaiblit légèrement ces notions de sécurité.

Comme ce mémoire porte sur la cryptanalyse, plutôt que la sécurité prouvée, nous n'allons pas définir formellement toutes les notions de sécurité : nous nous contenterons de définitions informelles, afin de transmettre des intuitions plus facilement. Le lecteur intéressé pourra consulter les notes de cours [287] pour un traitement plus technique.

Nous voudrions insister sur le point suivant. Certaines notions de sécurité communément acceptées aujourd'hui peuvent paraître au premier abord un peu artificielles, et peut-être aussi trop puissantes. On peut dire que c'est la découverte de certaines attaques réalistes qui a convaincu la communauté de l'importance de notions de sécurité aussi fortes. En d'autres termes, la cryptanalyse a contribué à identifier les bonnes notions de sécurité, mais elle a aussi contribué à l'acceptation de ces notions de sécurité. Par exemple, c'est l'attaque pratique de Bleichenbacher [33] qui a déclenché la migration vers OAEP pour le chiffrement RSA dans la norme PKCS [197], alors que les attaques à chiffré choisi contre RSA étaient apparues bien avant.

Il est aujourd'hui courant de définir les notions de sécurité à l'aide de jeux (voir le panorama [322]). Un schéma cryptographique est dit sûr pour une certaine notion de sécurité si un jeu spécifique entre un challenger et un attaquant ne peut être remporté par l'attaquant avec probabilité non-négligeable, l'attaquant était modélisé comme une machine de Turing polynomiale probabiliste, avec éventuellement accès à des oracles : la notion de sécurité définit précisément à quels oracles l'attaquant a accès. Informellement, une notion de sécurité consiste en deux définitions :

- Le but de l'attaquant. Cela définit les règles du jeu : ce que l'attaquant est censé faire (c'est-à-dire, à quelles conditions le jeu est-il remporté), et le déroulement du jeu.
- Les moyens de l'attaquant. C'est là que l'accès aux oracles est précisé. Par exemple, dans la sécurité à chiffré choisi, l'attaquant a accès à un oracle de déchiffrement, qui peut déchiffrer tout texte chiffré sauf celui soumis à l'attaquant.

Les oracles peuvent aussi dépendre du modèle de sécurité. Par exemple, dans le célèbre modèle de l'oracle aléatoire, une fonction de hachage est modelisée comme un oracle se comportant comme une fonction aléatoire.

2.3.1 Sécurité des signatures numériques

Nous commençons par les signatures car les notions de sécurité y sont très naturelles. De tous les objectifs possibles de l'attaquant, les plus importants sont les suivants :

Recouvrement de la clef : l'attaquant veut retrouver la clef secrète sk du signataire.

Forge universelle : l'attaquant veut pouvoir signer n'importe quel message. Cette forge est parfois aussi appelée *forge sélective*.

Forge existentielle : l'attaquant veut seulement pouvoir exhiber une nouvelle signature, sans nécessairement pouvoir choisir le message. Par une nouvelle signature, on entend habituellement une signature d'un nouveau message (dont on ne connaissait alors aucune signature valide), mais on peut aussi entendre une nouvelle signature d'un message pour lequel une signature était déjà connue, ce qui a un sens si le schéma de signature est probabiliste.

Les attaques sur les schémas de signature peuvent être classifiées selon les moyens de l'attaquant :

Attaques sans message : l'attaquant ne connaît que la clef publique pk du signataire.

Attaques à message connu : l'attaquant connaît une liste de couples (message,signature) valides.

Attaques à message choisi : l'attaquant peut demander des signatures de messages de son choix. Si les requêtes ne sont pas indépendantes, l'attaque à message choisi est dite *adaptative*. Bien sûr, selon l'objectif de l'attaquant, il y a une restriction naturelle sur les requêtes permises : par exemple, dans une forge universelle, l'attaquant ne peut pas demander une signature du message qu'il est censé devoir signer.

Les descriptions originales des principaux schémas de signature ne satisfont généralement que des notions très faibles de sécurité. Pour atteindre les plus fortes notions de sécurité sous des hypothèses appropriées, les messages doivent être transformés, généralement à l'aide de fonctions de hachage, mais il faut souligner qu'il n'est pas nécessaire que le schéma de signature soit probabiliste, contrairement au cas du chiffrement asymétrique.

2.3.2 Sécurité du chiffrement asymétrique

Historiquement, ces bonnes notions de sécurité pour le chiffrement asymétrique furent introduites bien après celles pour les signatures, ce qui suggère que la situation du chiffrement est plus complexe. De tous les objectifs possibles de l'attaquant, les plus importants sont les suivants :

Recouvrement de la clef : l'attaquant veut retrouver la clef secrète sk de l'utilisateur.

Déchiffrement : l'attaquant veut pouvoir déchiffrer n'importe quel texte chiffré. Le système de chiffrement est dit *à sens unique* si aucun attaquant ne peut efficacement déchiffrer un chiffré aléatoire avec probabilité non-négligeable. Par un chiffré aléatoire, on entend un chiffré d'un texte clair choisi uniformément au hasard dans l'ensemble des textes clairs.

Malléabilité : étant donnée une liste de textes chiffrés, l'attaquant veut construire un nouveau texte chiffré dont le texte clair correspondant soit relié aux textes clairs des textes chiffrés fournis en entrée.

Distingueur : l'attaquant veut trouver deux messages distincts m_0 et m_1 tels que si le challenger chiffre soit m_0 soit m_1 en un texte chiffré c, l'attaquant soit en mesure de dire lequel des deux messages a été chiffré, rien qu'en observant le chiffré c.

Clairement, si le schéma de chiffrement est déterministe, il y aura toujours un distingueur trivial : on peut sélectionner n'importe quelle paire de messages distincts m_0 et m_1, et en chiffrant successivement m_0 puis m_1, on pourra dire lequel correspond au texte chiffré c. Cela implique que le chiffrement doit nécessairement être probabiliste pour satisfaire de fortes notions de sécurité, ce qui n'est pas le cas en signature.

Les attaques sur les systèmes de chiffrement sont aussi classifiés selon les moyens mis à disposition de l'attaquant :

Attaques à clair connu : l'attaquant ne connaît que la clef publique pk de l'utilisateur, ce qui implique qu'il peut chiffrer n'importe quel texte clair.

Attaques à chiffré valide : l'attaquant peut vérifier si un chiffré donné est valide, c'est-à-dire s'il existe bien un texte clair qui peut être chiffré en un tel chiffré. Cela a un sens lorsque l'ensemble des textes chiffrés est plus gros que l'ensemble des textes clairs.

Attaques à clair vérifiable : l'attaquant peut vérifier si un texte chiffré donné se déchiffre en un texte clair donné.

Attaques à chiffré choisi : l'attaquant peut déchiffrer n'importe quel chiffré : si le chiffré n'est pas valide, l'attaquant le saura. Si les requêtes ne sont pas indépendantes, l'attaque à chiffré choisi est dite *adaptative*. Bien entendu, selon l'objectif de l'attaquant, il peut y avoir des restrictions naturelles sur les requêtes autorisées : par exemple, s'il s'agit d'un distingueur à chiffré choisi, l'adversaire ne peut pas demander à déchiffrer le texte chiffré c qui lui est soumis en entrée.

2.3.3 Etat de l'art

En pratique, on ne sait démontrer qu'un système satisfait telle ou telle notion de sécurité que sous des hypothèses calculatoires. Comme l'hypothèse calculatoire peut elle-même être formalisée sous forme de jeu, une « preuve de sécurité » va en fait transformer le jeu formalisant la notion de sécurité en le jeu formalisant l'hypothèse calculatoire, de sorte que s'il existait un attaquant pouvant

remporter le jeu formalisant la notion de sécurité, il existerait aussi un attaquant remportant le jeu associé à l'hypothèse calculatoire. Cette transformation de jeux ne se fait généralement pas en une seule étape : on a plutôt une série de transitions qui modifient peu à peu le jeu (voir [322, 287]). Cela permet de vérifier plus facilement les étapes d'une preuve de sécurité, puisque chaque transition ne fait que de très légères modifications, mais d'un autre côté, la multitude de transitions dans les preuves de sécurité actuelles complique la vérification.

Aujourd'hui, on a coutume de distinguer les preuves dites dans le modèle standard, par opposition à celles nécessitant le modèle de l'oracle aléatoire. On pourrait dire que dans le modèle de l'oracle aléatoire, on sait tout faire : on a des systèmes de chiffrement et de signature efficaces et sûrs pour les plus fortes notions de sécurité, sous plusieurs types d'hypothèses calculatoires. Et à partir du moment où un système de chiffrement ou de signature atteint ne serait-ce que la plus faible notion de sécurité, on sait aujourd'hui renforcer sa sécurité pour atteindre les plus fortes notions de sécurité, mais toujours dans le modèle de l'oracle aléatoire (voir [287]). Ce renforcement s'effectue à l'aide de « paddings » utilisant des fonctions de hachage. D'un point de vue pratique, le modèle de l'oracle aléatoire a donc beaucoup d'avantages. Mais on sait malheureusement [56] que ce modèle n'est qu'une idéalisation : il existe des schémas de signature qui sont sûrs dans le modèle de l'oracle aléatoire mais qui ne le sont plus dès que l'on instancie l'oracle aléatoire par une fonction de hachage concrète, quelle qu'elle soit. Toutefois, tous les contre-exemples connus à la méthodologie de l'oracle aléatoire sont assez éloignés des cryptosystèmes utilisés en pratique.

On dispose aussi dans le modèle standard de systèmes de chiffrement et de signature sûrs pour les plus fortes notions de sécurité, mais avec une efficacité un peu moins bonne, et des hypothèses calculatoires un peu différentes, voire controversées. Par exemple, en chiffrement asymétrique, toutes les hypothèses calculatoires utilisées sont en fait décisionnelles, comme par exemple le problème de Diffie-Hellman décisionnel connu sous le nom de DDH. Il est à noter que l'apparition des couplages a permis de transformer beaucoup de cryptosystèmes sûrs dans le modèle de l'oracle aléatoire en des cryptosystèmes sûrs dans le modèle standard. Cependant, cela se fait au prix des hypothèses calculatoires, qui font alors intervenir le couplage, et qui sont donc relativement nouvelles.

On est donc dans une situation assez particulière : vaut-il mieux une preuve de sécurité dans le modèle de l'oracle aléatoire avec une hypothèse calculatoire bien connue, ou une preuve de sécurité dans le modèle standard avec un problème relativement nouveau, et à première vue plus facile ?

2.4 Attaques élémentaires

Le but de cette section est d'illustrer les notions de sécurité de la section 2.3 en présentant des attaques très simples sur les descriptions historiques des cryptosystèmes classiques.

2.4.1 Attaques contre les signatures numériques

Nous commençons par des attaques élémentaires sur les signatures classiques.

Le RSA historique

Comme toute permutation à trappe utilisée directement comme schéma de signature, le RSA historique est vulnérable à une forge existentielle sans message. En effet, n'importe qui peut choisir uniformément au hasard un entier $s \in \mathbb{Z}_N$, et calculer :

$$m = s^e \bmod N. \tag{2.14}$$

L'entier s est alors une signature valide du message $m \in \mathbb{Z}_N$. Mais cette forge existentielle est loin d'être une forge universelle, puisqu'il n'y a pratiquement aucune liberté sur le choix de m.

Cependant, dans le cas particulier du RSA historique, il est facile d'obtenir une forge universelle à message choisi adaptative, grâce à la multiplicativité de la permutation RSA. En effet, supposons que l'on veuille signer un message $m \in \mathbb{Z}_N$. On choisit $m_1 \in \mathbb{Z}_N$ uniformément au hasard. Si m_1 n'est pas inversible modulo N (ce qui a une probabilité négligeable), alors nous avons trouvé un facteur non trivial de N, ce qui nous permettrait de signer m. Sinon, on peut calculer :

$$m_2 = m m_1^{-1} \pmod{N}.$$

Nous demandons à l'oracle les signatures s_1 et s_2 de respectivement m_1 et m_2. Il est alors clair par multiplicativité que $s = (s_1 s_2) \bmod N$ est une signature valide de m.

Une contremesure bien connue pour éviter les attaques précédentes est de hacher tout message avant de le signer. Plus précisément, on suppose l'existence d'une fonction de hachage cryptographique h de $\{0,1\}^*$ dans $\mathbb{Z}_N$. Plutôt que de signer un message $m \in \mathbb{Z}_N$, on signe un message binaire arbitraire $m \in \{0,1\}^*$ et l'on remplace m par $h(m)$ dans le procédé de signature (2.5) et de vérification (2.6). Le schéma de signature RSA obtenu est connu sous le nom de FDH-RSA pour *full-domain hash* RSA [26], et est prouvé sur dans le modèle de l'oracle aléatoire, sous l'hypothèse RSA. Bien entendu, pour que la fonction de hachage ne crée pas de trous de sécurité évidents, la fonction de hachage cryptographique h doit être *sans collision*, c'est-à-dire qu'il doit être calculatoirement impossible de trouver deux messages distincts m_0 et m_1 tels que $h(m_0) = h(m_1)$. D'un point de vue strictement formel, cette définition n'a pas vraiment de sens, puisqu'il existe nécessairement une infinité de collisions, et donc une machine de Turing renvoyant des collisions en temps constant. Mais cette définition a cependant une signification intuitive et pratique.

Dans le cas du RSA historique, l'utilisation d'une fonction de hachage évite les forges élémentaires et fournit même une preuve de sécurité dans le modèle de l'oracle aléatoire, mais les fonctions de hachage ne résolvent pas nécessairement tous les problèmes de sécurité, comme on va le voir avec Elgamal.

Le Elgamal historique

Tout d'abord, voyons une forge existentielle élémentaire contre le Elgamal historique. Pour forger une signature, il suffit de trouver un triplet $(m, a, b) \in \mathbb{Z}_p \times \mathbb{Z}_p^\times \times \mathbb{Z}_{p-1}$ satisfaisant (2.13) :

$$m \equiv ax + b \log a \pmod{p-1}.$$

Etant donné un m arbitraire, le signataire trouve une paire valide (a, b) car il a sélectionné un a pour lequel il connaissait déjà $\log a$ (ce logarithme est la clef jetable k) et était sûr qu'il soit inversible modulo $p-1$. Comme le signataire connait aussi l'exposant secret x, il peut résoudre (2.13) avec b pour inconnue. Mais l'attaquant ne connaît pas l'exposant secret x dans (2.13), donc il ne peut pas faire la même chose. Une façon de résoudre ce problème serait de choisir a de telle sorte que ax s'annule avec $b \log a$. Par exemple, si l'on choisit a de la forme :

$$a = g^B y^C \pmod{p},$$

où B et C sont entiers, alors

$$ax + b \log a \equiv x(a + bC) + bB \pmod{p-1}.$$

En choisissant un C premier avec $p-1$, on peut choisir b tel que :

$$a + bC \equiv 0 \pmod{p-1}.$$

Finalement, on peut choisir le message m comme :

$$m \equiv bB \pmod{p-1}.$$

Notre choix de (m, a, b) satisfait alors (2.13). Nous avons donc obtenu une forge existentielle sans message contre le Elgamal historique. Mais cette forge, initialement décrite dans [108], ne laisse aucune marge de manœuvre sur m : on peut obtenir beaucoup de forges grâce à différents choix de (B, C), mais chaque choix de (B, C) donne lieu à un unique m. Cela signifie que cette forge peut être évitée si l'on hache préalablement le message, comme dans FDH-RSA.

Nous allons maintenant décrire une autre forge existentielle, qui peut elle aussi être évitée par hachage. Mais nous verrons ensuite que cette forge existentielle peut être transformée en une forge universelle astucieuse découverte par Bleichenbacher [31], et qui ne peut donc être évitée par hachage.

Une façon alternative de trouver un triplet $(m, a, b) \in \mathbb{Z}_p \times \mathbb{Z}_p^\times \times \mathbb{Z}_{p-1}$ satisfaisant (2.13) est de résoudre cette congruence par restes chinois séparément. Plus précisément, on décompose le module $p-1$ en $p-1 = qs$ où s est friable (c'est-à-dire, sans grand facteur premier [323]). La raison pour laquelle on choisit un s friable est qu'il est facile d'extraire des logarithmes discrets dans un groupe d'ordre friable, grâce à l'algorithme élémentaire de Pohlig-Hellman (voir [231, 323]). En particulier, on ne sait pas calculer efficacement la fonction log sur $\mathbb{Z}_p^\times$, mais pour tout $z \in \mathbb{Z}_p^\times$, on peut facilement calculer $(\log z) \bmod s$. Ainsi, bien qu'on ne connaisse pas la clef secrète x, comme on dispose de la clef publique $y = g^x \bmod p$, on peut calculer la partie friable $x \bmod s$. Puisque $p-1$ est toujours pair, la partie friable s est toujours ≥ 2.

Comme $p-1 = qs$, la congruence (2.13) implique les deux congruences suivantes :

$$m \equiv ax + b \log a \pmod{q} \tag{2.15}$$

$$m \equiv ax + b \log a \pmod{s} \tag{2.16}$$

Réciproquement, si l'on trouve un triplet (m, a, b) satisfaisant à la fois (2.15) et (2.16), est-ce qu'il satisfairait (2.13) ? La réponse serait affirmative si q et s étaient premiers entre eux. Donc on va supposer que l'on a mis toute la partie friable de $p-1$ dans s, de sorte que le nombre friable s soit premier avec $q = (p-1)/s$.

Comme nous ne connaissons pas $x \bmod q$, la congruence (2.15) semble difficile à satisfaire. Toutefois, on remarque que le triplet $(m, a, b) = (m, q, 0)$ est une solution triviale de (2.15) si $m \equiv 0 \pmod{q}$. Considérons donc un message m tel que $m \equiv 0 \pmod{q}$, et posons $a = q$. Il reste à satisfaire la deuxième congruence (2.16). On peut calculer $\log a \bmod s$, et si l'on est chanceux, il sera inversible mod s, donc on pourra résoudre (2.16). Ainsi, on a obtenu une forge existentielle probabiliste, qui est faiblement universelle en ce sens où si $\log q$ est premier avec s, alors on peut forger la signature de tout message m divisible par q. Bleichenbacher [31] a trouvé une astuce remarquable pour supprimer cette limitation sur m, que nous allons à présent décrire.

Nous nous restreignons au cas le plus simple de la forge de Bleichenbacher, qui suppose que le générateur g est friable et divise $p-1$: le choix le plus naturel est $g = 2$. Posons donc $s = g$ où $p-1 = qs$ et supposons s friable comme précédemment. Cependant, nous n'allons plus supposer que q et s sont premiers entre eux, donc il ne suffira pas de travailler seulement avec (2.15) et (2.16). On va travailler directement avec la congruence (2.13) modulo $p-1$. On calcule $x_0 = x \bmod s$, de sorte que $x = x_0 + sx_1$ où x_1 est inconnu. En posant $a = q$, la congruence (2.13) devient :

$$m \equiv ax_0 + b \log a \pmod{p-1}. \tag{2.17}$$

Cette congruence semble difficile à résoudre en b puisque nous connaissons $\log a$ modulo s mais pas modulo $p-1$. L'astuce est que le choix particulier $a = q$ nous permet de calculer $\log a$. Remarquons que $\log a = \log q$ est égal à l'entier $k = (p-3)/2 = (p-1)/2 - 1$. En effet :

$$\begin{aligned} g^k &\equiv g^{(p-1)/2} g^{-1} \pmod{p} \\ &\equiv (-1) g^{-1} \text{ car } g \text{ est un générateur, donc son symbole de Legendre est -1.} \\ &\equiv qsg^{-1} \text{ car } p-1 = qs. \\ &\equiv q \text{ car } g = s. \end{aligned}$$

Il s'en suit que (2.17) peut être réécrite en :

$$m \equiv ax_0 + bk \pmod{p-1}. \tag{2.18}$$

Cette congruence linéaire peut être résolue en b si et seulement si $\text{pgcd}(k, p-1)$ divise $m - ax_0$. Pour évaluer $\text{pgcd}(k, p-1)$, remarquons que :

$$k^2 = ((p-1)/2 - 1)^2 = ((p-1)/2)^2 - (p-1) + 1 \equiv 1 + ((p-1)/2)^2 \pmod{p-1}.$$

On distingue deux cas :

- Si $p \equiv 1 \pmod 4$, alors $\text{pgcd}(k, p-1) = 1$ car la congruence précédente devient $k^2 \equiv 1 \pmod{p-1}$ car $((p-1)/2)^2$ est un multiple de $p-1$. Il s'en suit que quel que soit m, on peut toujours trouver b satisfaisant (2.18).
- Sinon, $p \equiv 3 \pmod 4$, et nous affirmons que $\text{pgcd}(k, p-1) = 2$. En effet, on a cette fois $((p-1)/2)^2 \equiv 1 \pmod{p-1}$ plutôt que 0, ce qui implique

  $$k^2 \equiv 2 \pmod{p-1}.$$

  Il s'en suit que $\text{pgcd}(k, p-1) = 2$ car nous savons déjà qu'il est ≥ 2. Par conséquent, si m est supposé uniformément distribué modulo $p-1$, alors la probabilité que $\text{pgcd}(k, p-1)$ divise $m - ax_0$ est exactement $1/2$. Cela signifie que l'on peut résoudre (2.18) une fois sur deux.

En résumé, si le générateur est friable et divise $p-1$, on peut forger des signatures de tout message si $p \equiv 1 \pmod 4$, et de la moitié des messages si $p \equiv 3 \pmod 4$. Bleichenbacher décrit dans [31] d'autres attaques similaires pour d'autres générateurs spécifiques.

De façon surprenante, Pointcheval et Stern [288] ont montré à la même conférence que [31] qu'une légère modification de la signature Elgamal est sûre dans le modèle de l'oracle aléatoire. En outre, l'attaque de Bleichenbacher peut aussi s'appliquer à cette variante, mais il n'y a heureusement pas de contradiction car la preuve de sécurité de Pointcheval-Stern suppose que le générateur g est choisi uniformément au hasard parmi tous les générateurs de $\mathbb{Z}_p^*$, auquel cas il y a très peu de chances que g soit friable et divise $p-1$. On peut en tirer la leçon suivante : il faut toujours vérifier avec précaution quelles hypothèses requiert une preuve de sécurité donnée.

2.4.2 Attaques contre le chiffrement asymétrique

Le RSA historique

Comme toute permutation à trappe utilisée directement comme chiffrement asymétrique, le RSA historique est vulnérable à une recherche exhaustive sur le texte clair. Plus précisément, un attaquant a accès à un oracle vérifiant des textes clairs : l'attaquant peut vérifier si un chiffré donné c se déchiffre en un texte clair m donné, en vérifiant si :

$$c \equiv m^e \bmod N. \tag{2.19}$$

En particulier, si l'ensemble $\mathcal{M}$ des messages (où $m \in \mathcal{M}$) est petit, on peut déchiffrer par recherche exhaustive : on énumère tous les $m' \in \mathcal{M}$ et l'on vérifie si le chiffré c correspond au message m', auquel cas $m = m'$. Ce serait par exemple le cas si l'on chiffrait lettre par lettre des textes écrits en français. En d'autres termes, si la distribution des textes clairs est très différente de la distribution uniforme sur $\mathbb{Z}_N$, (comme lorsque l'ensemble $\mathcal{M}$ est un très petit sous-ensemble de $\mathbb{Z}_N$), des attaques peuvent éventuellement en tirer parti. Un autre exemple est l'attaque à message court. Supposons que les messages soient en fait très petits : par exemple, supposons que le message m satisfait $0 \leq m \leq N^{1/e}$, (*e.g.* m est une clef AES de 128 bits, N un module de 1024 bits, et $e = 3$). Alors l'entier m satisfait : $0 \leq m^e \leq N$, ce qui signifie que la congruence (2.19) est en fait une égalité dans $\mathbb{Z}$,

$$c = m^e.$$

Mais il est bien connu que l'on peut résoudre en temps polynomial des équations polynomiales à une variable dans $\mathbb{Z}$: l'extraction de racines e-ièmes dans $\mathbb{Z}$ en est juste un cas particulier. En d'autres termes, si $0 \leq m \leq N^{1/e}$, alors on peut retrouver en temps polynomial le message m à partir de (c, N, e). En résumé, si la distribution du message m est la distribution uniforme sur $\mathbb{Z}_N$, personne ne sait aujourd'hui comment retrouver efficacement le message m à partir du chiffré $c = m^e \bmod N$: c'est exactement l'hypothèse RSA. Mais si la distribution du message m est très différente, il y a plusieurs exemples où l'on peut trouver des attaques très efficaces.

On peut aussi remarquer que la permutation RSA fait fuir de l'information. Etant donné $c = m^e \bmod N$ où m a une distribution uniforme sur $\mathbb{Z}_N$, on ne sait pas retrouver m efficacement, mais il est facile de déterminer un bit d'information sur le message m. Plus précisément, comme e doit être impair (car premier avec $\varphi(N)$ qui est pair), la congruence (2.19) implique l'égalité suivante entre symboles de Jacobi :

$$\left(\frac{c}{N}\right) = \left(\frac{m}{N}\right)^e = \left(\frac{m}{N}\right).$$

En d'autres termes, on peut calculer efficacement le symbole de Jacobi $\left(\frac{m}{N}\right)$, ce qui fournit un bit d'information sur le message m.

Nous avons vu précédemment une forge universelle à message choisi adaptative sur la signature RSA historique, fondée sur la multiplicativité de la permutation RSA. Cette attaque élémentaire a un analogue en chiffrement : elle peut être transformée en une attaque à chiffré choisi adaptative. En effet, supposons que l'on veuille déchiffrer un chiffré $c = m^e \bmod N \in \mathbb{Z}_N$: on voudrait retrouver le message $m \in \mathbb{Z}_N$. Choisissons $m_1 \in \mathbb{Z}_N$ uniformément au hasard. Si m_1 n'est pas inversible modulo N (ce qui est peu probable), alors nous avons trouvé un facteur non trivial de N, ce qui nous permet bien sûr de déchiffrer c. Sinon, on peut calculer :

$$c_2 = c m_1^{-e} \pmod{N}.$$

On demande à l'oracle de déchiffrer c_2 : cela fournit $m_2 \in \mathbb{Z}_N$ défini par $c_2 = m_2^e \bmod N$. Alors $m = (m_1 m_2) \bmod N$.

Cette attaque peut sembler artificielle. Pourtant, c'est un élément essentiel de l'attaque à chiffré choisi de Bleichenbacher [33] contre RSA-PKCS.

Le Elgamal historique

Le chiffrement Elgamal est probabiliste, contrairement au chiffrement RSA historique. En particulier, l'attaquant n'a pas d'accès à un oracle vérifiant un texte clair. Cependant, le chiffrement Elgamal fait aussi fuir un bit d'information sur le message. En effet, si g est un générateur de $\mathbb{Z}_p^*$, alors son symbole de Legendre $\left(\frac{g}{p}\right)$ vaut -1. En particulier, la congruence (2.8) implique que tout chiffré (c, d) d'un message m satisfait :

$$\left(\frac{c}{p}\right) = (-1)^k,$$

ce qui révèle la parité de la clef jetable k. De plus, la congruence (2.9) implique que :

$$\left(\frac{d}{p}\right) = \left(\frac{m}{p}\right)\left(\frac{y}{p}\right)^k.$$

Comme d, y et p sont publics, et comme la partié de k est désormais connue, on peut calculer le symbole de Legendre $\left(\frac{m}{p}\right)$, ce qui révèle un bit d'information sur le message m.

Nous avons vu en section 2.4.2.0 une attaque à chiffré choisi adaptative contre le RSA historique, fondée sur la multiplicativité de la permutation RSA. Comme le chiffrement Elgamal est lui aussi multiplicatif, cette attaque à chiffré choisi peut trivialement s'adapter au cas d'Elgamal.

Le fait que le chiffrement RSA historique soit déterministe le rend vulnérable à plusieurs attaques élémentaires, mais le transformer en un chiffrement probabiliste n'empêche pas forcément toutes les attaques, comme le montre l'exemple du chiffrement Elgamal. Il faut rendre le chiffrement probabiliste, mais il faut le faire d'une façon pertinente.

2.5 La cryptanalyse moderne

Nous allons maintenant présenter les différentes facettes de la cryptanalyse moderne des systèmes à clef publique, en précisant leurs relations avec la sécurité prouvée :

- L'algorithmique de la cryptanalyse, qui étudie les hypothèses calculatoires de la cryptographie à clef publique, mais aussi des problèmes utiles à la cryptanalyse.
- La recherche de failles dans les systèmes cryptographiques. En général, ces failles se situent soit au niveau de la conception même du système cryptographique, soit au niveau des implémentations.

2.5.1 L'algorithmique de la cryptanalyse

Une part importante de la cryptanalyse consiste à concevoir et analyser des algorithmes pour résoudre divers problèmes d'intérêt cryptanalytique. On peut classer ces algorithmes en trois catégories :

- Les algorithmes généraux qui s'attaquent directement aux hypothèses calculatoires générales de la cryptographie à clef publique.
- Les algorithmes dédiés, qui s'intéressent à des problèmes un peu moins généraux, ou à des instances particulières des problèmes généraux. Ces algorithmes dédiés peuvent bien entendu faire appel à des algorithmes généraux.
- Les algorithmes par réduction, qui ramène l'un des problèmes précédents à un problème mieux étudié, via des réductions au sens de la théorie de la complexité. On souhaite bien entendu que la réduction soit efficace. En combinant la réduction et les algorithmes connus pour le problème mieux étudié, on peut donc s'attaquer au problème initial.

Cette partie de la cryptanalyse est complémentaire de la sécurité prouvée en ce sens où elle teste concrètement les hypothèses calculatoires de la sécurité prouvée.

Algorithmes généraux

On peut regrouper sous cette appellation tous les algorithmes concernant les hypothèses calculatoires générales de la cryptographie à clef publique : factorisation, logarithme discret, réduction de réseau, problème de théorie des codes, systèmes d'équations polynomiales à plusieurs variables, *etc.* On trouve ainsi par exemple le crible algébrique [204] et l'algorithme LLL [205]. Les principaux algorithmes de réduction de réseau seront décrits dans le chapitre 3, et nos contributions dans ce domaine se trouvent en annexe à partir de la page 95.

Mais on peut aussi rajouter dans cette rubrique des principes généraux en algorithmique de la cryptanalyse, comme les algorithmes en racine carrée et les compromis temps/mémoire. Lorsqu'une recherche exhaustive sur une quantité secrète est possible, les cryptanalystes essaient de trouver des améliorations à base de compromis temps/mémoire. En général, la recherche exhaustive requiert un espace négligeable S et un temps exponentiel T. Un compromis temps/mémoire essaie de rééquilibrer ces deux coûts, souvent en transformant la consommation mémoire et le temps de calcul en la racine carrée du temps de calcul de la recherche exhaustive : ainsi, si T est le temps de calcul de la recherche exhaustive, le temps de calcul et la consommation mémoire deviennent essentiellement $\sqrt{T}$. Parfois, il est en outre possible de réduire grandement la complexité espace de ces attaques en racine carrée, ce qui a un intérêt pratique considérable. Un exemple typique est le problème du logarithme discret.

L'algorithme pas-de-bébé-pas-de-géant est un compromis temps/mémoire en la racine carrée du temps de la recherche exhaustive, mais les algorithmes ρ et kangourou de Pollard sont meilleurs car ils atteignent le même temps de calcul en la racine carrée, tout en ne consommant que très peu de mémoire (voir [77]).

Algorithmes dédiés

Les algorithmes généraux précédents permettent parfois de résoudre des classes de problèmes qui interviennent souvent en cryptanalyse. Un exemple caractéristique est la réduction de réseau, auquelle est consacré le chapitre 3, et qui est sans doute aujourd'hui l'outil le plus populaire en cryptanalyse à clef publique. Sans même connaître l'objet de la réduction de réseau, on peut comprendre son importance en voyant la liste des problèmes qui peuvent se résoudre à l'aide de la réduction de réseau. Les deux applications principales de la réduction de réseau en cryptanalyse sont les suivantes :

Systèmes d'équation linéaires : Supposons pour simplifier que l'on ait une congruence linéaire de la forme

$$\sum_{i=1}^{n} a_i x_i \equiv b \pmod{M}, \tag{2.20}$$

où seuls les entiers x_i soient inconnus, tandis que les entiers $a_i \in \mathbb{Z}$, l'entier $b \in \mathbb{Z}$ et le module M sont connus. Si l'on n'impose aucune contrainte sur la taille des x_i, il est facile de trouver une solution $(x_1, \ldots, x_n) \in \mathbb{Z}^n$ à (2.20), donc on s'intéresse plutôt à des solutions ayant des propriétés spéciales, par exemple quand les x_i sont petits. Quand n est petit (disons, inférieur à 50), nous avons les résultats suivants :

- La réduction de réseau permet en général de trouver efficacement une solution $(x_1, \ldots, x_n) \in \mathbb{Z}^n$ telle que $x_i = O(M^{1/n})$ pour tout i. Si $b \equiv 0 \pmod{M}$, cela revient à trouver un vecteur court dans un réseau. Si $b \not\equiv 0 \pmod{M}$, cela revient à trouver un vecteur proche dans un réseau.
- S'il existe une solution exceptionnelle $(x_1, \ldots, x_n) \in \mathbb{Z}^n$ telle que $\prod_{i=1}^{n} x_i$ soit beaucoup plus petit que M, alors elle peut sans doute être retrouvée en pratique, et même peut-être en théorie. Plus précisément, si $b \equiv 0 \pmod{M}$, cela signifie qu'il existe un vecteur exceptionnellement court dans un certain réseau. Et si $b \not\equiv 0 \pmod{M}$, cela signifie qu'il y a dans un certain réseau un vecteur exceptionnellement proche d'un certain vecteur cible.

De tels résultats ont été appliqués maintes fois en cryptanalyse (voir [267]), sous cette forme, ou dans sa version plus générale avec un système d'équations plutôt qu'une seule équation (2.20). Deux exemples célèbres sont l'attaque de Wiener [356] contre le RSA à petit exposant secret, et l'attaque à chiffré choisi de Bleichenbacher [33] contre RSA-PKCS#1 version 1.5, que nous verrons plus en détail dans la section 2.5.3 : bien que ces deux attaques ne furent historiquement pas présentées à l'aide de réseaux, leur description alternative à base de réseaux (voir [257]) est sans doute plus simple. Plus récemment, une congruence de la forme (2.20) a été utilisé par l'auteur pour s'attaquer [255] à l'implémentation de la signature Elgamal dans le logiciel libre GPG [128] de sécurisation du courrier électronique : voir l'annexe N page 333.

Petites racines d'équations polynomiales : Ces problèmes, devenus très populaires depuis les travaux de Coppersmith [70, 71], ont beaucoup d'applications pour la cryptanalyse de RSA (voir les panoramas [71, 267, 40]) : RSA à petit exposant secret, chiffrement RSA lorsqu'une grande partie du message clair est déjà connue, *etc.* Le résultat fondateur dans ce domaine est le suivant :

Théorème 2.1 (Coppersmith [70]) *Soit $P(x) \in \mathbb{Z}[x]$ un polynôme unitaire de degré δ, et N un entier de factorisation inconnue. Alors on peut en temps polynomial en $(\log N, \delta)$ trouver tous les entiers x_0 tels que $P(x_0) \equiv 0 \pmod{N}$ et $|x_0| \leq N^{1/\delta}$.*

Ce théorème admet une généralisation [44, 41, 157, 224] en termes de pgcd, qui permet par exemple de démontrer que l'on peut factoriser un module RSA usuel si l'on connaît la moitité des bits de poids fort d'un des facteurs premiers :

Théorème 2.2 *Soit $P(x) \in \mathbb{Z}[x]$ un polynôme unitaire de degré δ, et N un entier de factorisation inconnue. Soit $\alpha \in \mathbb{Q}$ tel que $0 \leq \alpha \leq 1$. Alors on peut en temps polynomial en $(\log N, \delta, \alpha)$ trouver tous les entiers x_0 tels que $\text{pgcd}(P(x_0), N) \geq N^{\alpha}$ et $|x_0| \leq N^{\alpha^2/\delta}$.*

Le théorème 2.1 est le cas particulier $\alpha = 1$. Il peut heuristiquement se généraliser aux congruences à deux variables, dont l'application principale est l'attaque de Boneh-Durfee [43] contre le RSA à petit exposant secret, qui améliore celle de Wiener [356]. Il admet aussi une autre généralisation [70, 38] elle prouvée aux équations polynomiales à deux variables sur $\mathbb{Z}$.

En collaboration avec Glenn Durfee, l'auteur a développé [87] (voir l'annexe K page 287) une version heuristique à trois variables du théorème 2.1, afin d'adapter l'attaque de Boneh-Durfee [43] aux modules RSA déséquilibrés, ce qui a permis d'attaquer avec succès des variantes de RSA proposées par Sun-Yang-Laih [337].

La boite à outils du cryptanalyste va sans doute continuer à s'agrandir, et il serait vain de fournir une liste exhaustive des algorithmes dédiés. Par exemple, en collaboration avec Oded Regev, l'auteur a récemment introduit [259] (voir l'annexe H page 243) en cryptanalyse des techniques issues du traitement du signal, plus précisément l'analyse de composantes indépendantes [160].

Algorithmes par réduction

Lorsqu'un problème nouveau apparaît, on peut aussi s'intéresser à des réductions efficaces entre ce nouveau problème et des problèmes mieux connus, au sens de la théorie de la complexité. Ainsi, au lieu d'utiliser des algorithmes directs, on va combiner la réduction et les algorithmes existants pour les problèmes connus pour résoudre le nouveau problème.

Bien entendu, si le nouveau problème est dans la classe NP, on pourrait toujours se ramener à des problèmes NP-difficiles bien connus. Cependant, les réductions polynomiales sous-jacentes ne sont pas forcément très intéressantes d'un point de vue pratique, et il faut aussi sélectionner un problème connu que l'on sache bien résoudre jusqu'à de grandes dimensions.

La cryptanalyse s'est ainsi particulièrement intéressé aux réductions aux problèmes de réseaux euclidiens, notamment le problème du plus court vecteur : le premier résultat de ce type est dû à Lagarias et Odlyzko [200]. Il y a des résultats de NP-difficulté pour ces problèmes, mais on sait aussi résoudre ces problèmes en pratique pour de grandes dimensions : voir le chapitre 3 pour plus de détails. L'annexe page 207 contient deux articles illustrant cette méthode par réduction.

Cette branche de l'algorithmique de la cryptanalyse se rapproche beaucoup de la théorie de la complexité et de la sécurité prouvée. Remarquons cependant que la sécurité prouvée s'intéresse à des réductions dans le sens inverse : les cryptanalystes veulent utiliser les algorithmes des problèmes connus pour résoudre des problèmes nouveaux, tandis que les cryptographes souhaitent montrer que s'il existait un adversaire efficace contre le nouveau système, alors on pourrait s'attaquer à un problème bien connu.

2.5.2 Failles de conception

On quitte maintenant les failles de nature algorithmique pour s'intéresser aux failles plus cryptographiques. On va d'abord s'intéresser aux failles liées à la conception même du système. L'annexe page 241 regroupe quatre contributions de l'auteur dans ce domaine.

Il est sans doute utile de faire le lien entre ces failles de conception et la sécurité prouvée. On utilise de plus en plus des systèmes cryptographiques ayant des propriétés de sécurité prouvée, mais tous les systèmes utilisés en pratique ne disposent pas forcément de telles garanties : l'exemple de

la norme PKCS#1 v1.5 [197] pour le cryptosystème RSA en témoigne. En l'absence de résultat de sécurité prouvée, seule la cryptanalyse permettra de réellement évaluer la sécurité du système. C'est ainsi que l'auteur a, en collaboration avec Oded Regev [259] (voir l'annexe H page 243), attaqué avec succès les versions de base des schémas de signature de Goldreich-Goldwasser-Halevi [118] (GGH) et de NTRU [148]. Ces schémas de signature n'avaient pas de sécurité prouvée : et pour cause, on savait déjà que chaque signature nouvellement générée faisait fuir de l'information sur la clef secrète, mais on ne savait pas réellement exploiter ces informations supplémentaires.

Enfin, lorsqu'il existe des résultats de sécurité prouvée, cela ne garantit pas forcément l'absence d'attaque réaliste. En effet, plusieurs cas de figure peuvent se présenter :

- La preuve de sécurité est incorrecte.
- Toutes les hypothèses requises par la preuve de sécurité ne sont pas vérifiées. Par exemple, si on utilise la signature Elgamal avec un choix spécial de générateur, la preuve de sécurité de Pointcheval et Stern [288] n'est plus valable, et on peut tomber sous le coup de l'attaque de Bleichenbacher [31] sur la signature Elgamal. Un autre exemple est le chiffrement NTRU [149]. Pendant longtemps, les choix de paramètres de NTRU étaient tels que le procédé de déchiffrement pouvait en fait parfois échouer (avec faible probabilité) : or ces erreurs de déchiffrement mettent à mal les techniques de preuves de sécurité, notamment celles utilisées pour renforcer la sécurité d'un système de chiffrement à l'aide de fonctions de hachage. Par exemple, les preuves de sécurité de [258] ne peuvent pas s'appliquer lorsque le choix de paramètres permet des erreurs de déchiffrement avec probabilité non-négligeable. Dans ce cas, on a même des attaques à chiffré choisi fonctionnant quel que soit le padding utilisé, comme l'a montré l'auteur [158] (voir l'annexe I page 259) en collaboration avec Nick Howgrave-Graham, David Pointcheval, John Proos, Joseph Silverman, Ari Singer et William Whyte. Une attaque hybride entre cette attaque et l'attaque [168] a récemment été présentée par l'auteur [105], en collaboration avec Nicolas Gama.
- La preuve de sécurité n'est pas suffisamment efficace. Une preuve de sécurité est intrinsèquement une réduction entre deux problèmes : or, si elle est trop coûteuse, elle risque de ne pas signifier grand-chose lorsqu'on sélectionne des paramètres concrets. Par exemple, quelle interprétation pourrait-on tirer d'une preuve de sécurité démontrant que si l'on pouvait casser le système pour tel choix concret de paramètres, alors on pourrait factoriser des nombres de 64 bits ?
- Le modèle de sécurité n'est pas pertinent, car il ne reflète pas suffisamment bien la réalité. C'est ainsi que l'auteur a, en collaboration avec Igor Shparlinski [262] (voir l'annexe J page 275), exhibé une attaque polynomiale prouvée contre un schéma de signature RSA assistée pour lequel était annoncée une preuve de sécurité dans le modèle dit générique. Le problème ici est que le modèle générique est trop restrictif : l'attaque n'était pas « générique », comme c'est souvent le cas en cryptanalyse. Notons cependant que Koblitz et Menezes [185] ont récemment remarqué que la preuve de sécurité en question était en outre incorrecte.

2.5.3 Failles d'implémentation

Même lorsque l'algorithme cryptographique et ses paramètres sont bien sélectionnés, il peut subsister des failles liées à des problèmes d'implémentation : ces failles sont d'importance cruciale en pratique. L'annexe page 299 regroupe trois contributions de l'auteur dans ce domaine.

Pour bien comprendre ces problèmes, décrivons le principe de l'attaque de Bleichenbacher [33] contre la norme PKCS#1 v1.5 [197] de chiffrement RSA. On a vu au début de ce chapitre qu'il fallait à tout prix transformer les messages avant de les chiffrer par RSA au moyen d'une simple exponentiation modulaire. Mais quelle transformation (*padding*) doit-on utiliser ? Dans les années 90, une solution très répandue était d'utiliser la norme PKCS#1 v1.5 [197] conçue par l'entreprise RSA : cette norme était notamment utilisée dans le protocole SSL v3.0, largement déployé par les navigateurs. La norme spécifiait donc comment transformer un message m, avant de lui appliquer le

chiffrement RSA brut (c'est-à-dire, l'exponentiation modulaire à la puissance e) :

- Le message m à chiffrer était supposé bien plus petit que le module RSA noté N : il devait avoir quelques octets de moins que N.
- Cette valeur était ensuite formatée comme décrit par la norme PKCS#1 v1.5 de type 02 (voir la figure 2.1) : un octet nul est rajouté à gauche, ainsi qu'un nombre approprié d'octets aléatoires non nuls, de façon à ce que les deux premiers octets de la valeur finale soient 00 02 suivis par autant d'octets aléatoires non nuls que nécessaire pour atteindre la taille du module. En d'autres termes, la valeur finale décrite dans la figure 2.1 doit avoir le même nombre d'octets que le module N.

00	02	Octets aléatoires non nuls	00	Message m

FIG. 2.1 – Formatage de chiffrement PKCS#1 v1.5, type 02.

Lorsqu'on déchiffre un texte RSA chiffré par la norme PKCS#1 v1.5, on retrouve d'abord une valeur de forme donnée par la figure 2.1 puis l'on procède de la fçon suivante pour retrouver le message m :

- On vérifie d'abord que les deux octets de poids fort soient 00 et 02.
- On enlève tous les octets non nuls jusqu'à tomber sur un octet 00.
- Le reste est le message m.

Mais que faire si le déchiffrement échoue ? Par exemple, que faire si les deux octets de poids fort de $c^d \pmod N$ ne sont pas 00 et 02 ? Une telle situation peut facilement arriver puisque quiconque peut envoyer des textes chiffrés, et donc, ces textes chiffrés ne sont pas forcément valides. Bleichenbacher [33] remarqua en 1998 que dans de nombreuses implémentations de SSL v3.0, la personne qui déchiffrait (dans la pratique, un serveur qui reçoit beaucoup de messages chiffrés avec sa clef publique RSA) renvoyait en fait un message d'erreur s'il y avait un problème lors du déchiffrement. En d'autres termes, pour ces implémentations, un adversaire mal intentionné a accès à un 0002-oracle : étant donné $c \in \mathbb{Z}_N$, l'oracle répond si oui ou non les deux octets de poids fort de $c^d \pmod N$ sont 00 et 02. Dans [33], Bleichenbacher a montré que l'accès à un tel oracle permettait à un attaquant de déchiffrer n'importe quel chiffré RSA $c = m^e \pmod N$, ce que l'on peut voir facilement en termes de réseaux, car c'est en fait une variante du problème dit du nombre caché [48, 261] (voir [257]).

Cette attaque est typique d'une faille d'implémentation. On implémente au départ un cryptosystème réputé sûr (en l'occurence, RSA), mais suite à une accumulation d'erreurs (même petites), il peut finalement y avoir une attaque réaliste. L'auteur a exhibé un autre exemple dans [255] (voir l'annexe N page 333) : le logiciel libre GPG [128] de sécurisation du courrier électronique, inclus dans la plupart des distributions du système d'exploitation LINUX.

De manière générale, l'attaque de Bleichenbacher [33] montre aussi qu'en pratique, selon le contexte, un adversaire peut avoir accès à quantité d'informations supplémentaires, autres que la clef publique. C'est le point de départ des attaques dites physiques, initiées par Kocher [187, 188] (voir [11, 127] pour plus d'informations). Ces attaques sont particulièrement importantes dans le cas des implémentations sur carte à puce. En effet, on peut dans ce cas imaginer que l'adversaire a temporairement un accès physique à la carte à puce, ce qui laisse entrevoir beaucoup de possibilités, comme :

- La mesure du temps de calcul des opérations cryptographiques [187].
- La mesure de la consommation de courant des opérations cryptographiques [188].

Et rien n'interdit de se contenter de faire des mesures physiques, on peut aussi agir activement sur la carte à puce, comme le font les attaques par injection de faute [42, 29, 172, 21]. En collaboration avec David Naccache, Michael Tunstall et Claire Whelan [246] (voir l'annexe M page 323), l'auteur a ainsi expérimenté avec succès une attaque par injection de faute sur une implémentation typique de DSA sur carte à puce, en combinant les injections de faute avec les attaques classiques [159, 261] contre les

schémas de signature à base de logarithme discret lorsque les nombres aléatoires sont partiellement révélés.

Heureusement, depuis l'apparition des attaques matérielles [187, 188], l'industrie de la carte à puce a développé toute une série de contre-mesures visant à contrecarrer ces attaques : voir par exemple [11, 127] pour plus d'informations.

2.6 Conclusion

Nous avons vu dans ce chapitre les principales notions de sécurité pour le chiffrement asymétrique et les signatures, que nous avons illustrées par des attaques élémentaires contre les descriptions historiques de RSA et ElGamal. Nous avons présenté les deux branches principales de la cryptanalyse moderne des systèmes à clef publique : l'algorithmique de la cryptanalyse qui regroupe les algorithmes généraux, les algorithmes dédiés, et les algorithmes par réduction ; et la recherche de failles, soit au niveau de la conception même du système cryptographique, soit au niveau des implémentations, en utilisant éventuellement les informations supplémentaires auxquelles peut avoir accès en pratique un adversaire chevronné. Cela nous a permis de préciser le contexte dans lequel s'inscrivent nos contributions en cryptanalyse regroupées en annexe. Le chapitre suivant est consacré à la géométrie des nombres algorithmique, dont on a déjà mentionné les applications majeures en cryptanalyse.

Chapitre 3

La Géométrie des Nombres Algorithmique

« Comme ce sujet est un des plus curieux de l'Arithmétique, et qu'il mérite particulièrement l'attention des Géomètres par les grandes difficultés qu'il renferme, je vais tâcher de le traiter plus à fond qu'on ne l'a encore fait. »

Joseph-Louis Lagrange (1773)

Sommaire

3.1 Introduction 40
3.1.1 Les empilements de sphères 40
3.1.2 Généralisations géométriques de l'algorithme d'Euclide 41
3.1.3 Organisation 42
3.2 La géométrie des nombres 42
3.2.1 Réseaux euclidiens 42
3.2.2 Le théorème fondamental de Minkowski 45
3.2.3 Minima successifs et constante d'Hermite 46
3.3 La réduction de réseau 47
3.3.1 Bases réduites 47
3.3.2 Réduction en dimension deux 48
3.3.3 La réduction de Minkowski 49
3.3.4 L'orthogonalisation de Gram-Schmidt 49
3.3.5 La réduction faible 50
3.3.6 La réduction de Hermite, Korkine et Zolotareff 51
3.4 Réseaux aléatoires 52
3.4.1 L'ensemble des réseaux 52
3.4.2 Mesure naturelle sur l'ensemble des réseaux 53
3.4.3 Cas de la dimension deux 53
3.4.4 Propriétés des réseaux aléatoires 54
3.4.5 Génération de réseaux aléatoires 55
3.5 Problèmes algorithmiques 55
3.5.1 Représentation 55
3.5.2 Problème du plus court vecteur 56
3.5.3 Problème du plus proche vecteur 57
3.6 Algorithmes de réduction de réseau ou de plus court vecteur 58
3.6.1 L'algorithme de Lagrange et sa généralisation gloutonne 59

3.6.2 Gram-Schmidt et la réduction faible 61
3.6.3 Les algorithmes d'Hermite et de Lenstra-Lenstra-Lovász 61
3.6.4 Applications à la recherche de plus court vecteur et à la réduction HKZ . 69
3.6.5 Généralisations par blocs de LLL 72
3.6.6 Les algorithmes de Babai 77
3.6.7 Faits expérimentaux 78
3.7 Conclusion 79

3.1 Introduction

La géométrie des nombres doit son nom à Minkowski, qui publia à la fin du dix-neuvième siècle un ouvrage en allemand intitulé *Geometrie der Zahlen* [243]. Cette branche de la théorie des nombres a pour objet les réseaux euclidiens, qui sont des arrangements réguliers et infinis de points de l'espace à n dimensions, tels que l'ensemble $\mathbb{Z}^n$ des points à coordonnées entières. Elle trouve ses origines dans deux problèmes historiques, l'un purement mathématique, l'autre ayant par contre une forte connotation algorithmique.

3.1.1 Les empilements de sphères

Le premier problème historique est l'un des problèmes ouverts présentés par Hilbert en 1900. Il s'agit du problème des empilements de sphères [68] : quelle est la densité maximale possible d'un empilement de sphères de $\mathbb{R}^n$ pour une dimension n donnée ? Par *empilement de sphères* de $\mathbb{R}^n$, on entend une collection $\mathcal{S}$ de boules de même rayon r et d'intérieurs disjoints ; et par *densité*, on entend la proportion de $\mathbb{R}^n$ qui est couverte pas les intérieurs des boules de $\mathcal{S}$, que l'on définit comme $\lim_U \text{vol}(\cup_{B\in\mathcal{S}} B \cap U)/\text{vol}(U)$ où U est une suite croissante de sous-ensembles convexes de $\mathbb{R}^n$ qui recouvrent $\mathbb{R}^n$. Selon l'empilement de sphères, il n'est pas certain que cette limite existe, mais on peut démontrer qu'il existe bien une densité maximale. Curieusement, le problème des empilements de sphères s'avère extrêmement difficile : il est ouvert dès que $n \geq 4$. Le problème est trivial en dimension un. En dimension deux, l'empilement le plus dense, à savoir l'empilement hexagonal de la Figure 3.1, remonte à l'Antiquité, mais la preuve de son optimalité ne date que du début du vingtième siècle.

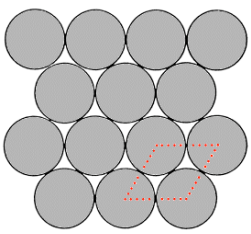


FIG. 3.1 – L'empilement le plus dense en dimension deux : l'empilement hexagonal.

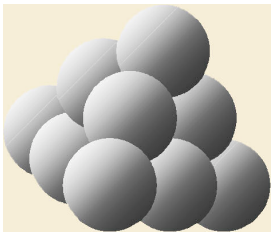


FIG. 3.2 – L'empilement le plus dense en dimension trois.

Képler conjectura en 1611 que l'empilement traditionnel des marchands de fruits (voir Figure 3.2) était le plus dense en dimension trois. Mais la première démonstration correcte de la conjecture de Képler ne fut annoncée qu'en 1998 par Hales, et finalement publiée en 2005 et 2006 dans [135, 136]. La démonstration de Hales est extrêmement complexe : elle nécessite plus de trois cent pages, et s'appuie sur bon nombre de calculs effectués par des machines. Plus précisément, les ordinateurs ont servi à résoudre 5,000 instances de problèmes d'optimisation non linéaire. La difficulté du problème des empilements de sphères suggère de s'intéresser à des classes particulières d'empilements. Les cas de la dimension deux et trois motivent l'étude des empilements de réseau, c'est-à-dire les empilements de sphères où les centres des sphères forment un réseau. C'est l'une des motivations historiques de l'étude des réseaux euclidiens, et l'on connaît d'ailleurs aujourd'hui les empilements de réseau les plus denses de la dimension 1 à la dimension 8, en passant par la dimension 24. Plus précisément, l'empilement de réseau le plus dense fut établi par :

- Lagrange [201] en dimension deux.
- Gauss [109] en dimension trois.
- Korkine et Zolotareff [191] en dimensions quatre et cinq. Ils devinèrent également les empilements optimaux de la dimension six à huit.
- Blichfeldt [37] en dimensions six à huit.
- Cohn et Kumar [67] en dimension vingt-quatre.

Connaître la densité de l'empilement de réseau le plus dense revient à déterminer la valeur exacte de la constante dite d'Hermite, qui majore de façon optimale le minimum sur $\mathbb{Z}^n \setminus \{0\}$ d'une forme quadratique définie positive. Bien que les empilements les plus denses soient jusqu'en dimension trois des empilements de réseau, il est plutôt conjecturé que cela ne devrait pas être le cas en général.

3.1.2 Généralisations géométriques de l'algorithme d'Euclide

Le second problème historique concerne le célèbre algorithme d'Euclide permettant de calculer le plus grand commun diviseur (pgcd) de deux entiers. L'élégance et l'efficacité de l'algorithme d'Euclide motive la recherche de généralisations jouissant de propriétés analogues. Cherchant à généraliser des travaux antérieurs de Fermat et Euler, Lagrange [201] s'intéressa à la fin du dix-huitième siècle aux nombres qui sont représentables par des formes quadratiques : étant donné $(a, b, c) \in \mathbb{Z}^3$, quels sont les nombres entiers de la forme $ax^2 + bxy + cy^2$ où $(x, y) \in \mathbb{Z}^2$? Fermat avait par exemple caractérisé le cas particulier des nombres qui sont sommes de deux carrés, c'est-à-dire de la forme $x^2 + y^2$ où $(x, y) \in \mathbb{Z}^2$. Pour résoudre ces questions de manière générale, Lagrange inventa une généralisation [201, pages 698–700] de l'algorithme d'Euclide aux formes quadratiques binaires définies positives. L'algorithme de Lagrange est souvent attribué (de façon incorrecte) à Gauss [109], et il fut ensuite lui-même généralisé par Hermite aux formes quadratiques définies positives de dimension quelconque. L'algorithme d'Hermite servit historiquement à établir l'existence de la constante d'Hermite [142]. Il est étroitement lié au célèbre algorithme LLL [205] de réduction de réseau.

On peut voir ces algorithmes de façon géométrique. L'algorithme d'Euclide calcule en fait la plus petite combinaison linéaire (non nulle) de deux entiers, puisque $\mathrm{pgcd}(a, b)\mathbb{Z} = a\mathbb{Z} + b\mathbb{Z}$. Si l'on

remplace les entiers a et b par un nombre quelconque de vecteurs $\mathbf{b}_1, \ldots, \mathbf{b}_n$ à coordonnées entières, on peut se demander s'il est possible de déterminer la plus petite combinaison linéaire (non nulle) de ces vecteurs, c'est-à-dire un vecteur non nul de la forme $a_1\mathbf{b}_1 + \cdots + a_n\mathbf{b}_n$ (où les a_i sont dans $\mathbb{Z}$) et dont la norme euclidienne soit minimale. Quand la dimension de l'espace engendré par ces vecteurs est petite, on peut résoudre ce problème de façon aussi efficace que l'algorithme d'Euclide. Mais lorsque la dimension est élevée, on ne sait calculer que des approximations, comme celle fournie par l'algorithme LLL.

3.1.3 Organisation

Nous présenterons dans ce chapitre les notions et les résultats essentiels de la géométrie des nombres algorithmique, de façon à pouvoir expliquer nos contributions [263, 264, 265, 104, 103] dans ce domaine, qui figurent toutes dans ce mémoire. Dans la section 3.2, nous rappellerons d'abord les bases de la géométrie des nombres (voir notamment [57, 220, 68, 95, 326, 130])). La section 3.3 sera consacrée à la théorie de la réduction, au sens mathématique : on y introduira les réductions de Minkowski et Korkine-Zolotareff. Dans la section 3.4, nous expliquerons la notion de réseau aléatoire. Nous aborderons ensuite les aspects algorithmiques (voir [65, 129, 214, 240]). Dans la section 3.5, nous introduirons les principaux problèmes algorithmiques de la géométrie des nombres. Cela nous conduira à présenter les principaux algorithmes de réduction de réseau en section 3.6. Enfin, nous terminerons par les principaux problèmes ouverts en section 3.7.

3.2 La géométrie des nombres

3.2.1 Réseaux euclidiens

Nous ne démontrerons aucun des résultats classiques mentionnés ici. On pourra trouver des démonstrations dans les ouvrages de référence [57, 220, 326, 130]. Dans tout ce qui suit, on considère $\mathbb{R}^n$ muni de sa structure naturelle d'espace euclidien, sans faire de distinction entre points et vecteurs, et l'on note μ la mesure de Lebesgue. Nous utiliserons des lettres en gras pour noter des vecteurs, en notation ligne. Le produit scalaire euclidien de deux vecteurs $\mathbf{x} = (x_i)_{i=1}^n$ et $\mathbf{y} = (y_i)_{i=1}^n$ est dénoté par :

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{n} x_i y_i.$$

La norme euclidienne correspondante est dénotée par :

$$\|\mathbf{x}\| = \sqrt{x_1^2 + \cdots + x_n^2}.$$

On a défini informellement un réseau comme un arrangement régulier de points de l'espace. Plus précisément, on appelle *réseau* de $\mathbb{R}^n$ tout sous-groupe discret de $(\mathbb{R}^n, +)$, c'est-à-dire tout ensemble L non vide stable par soustraction, pour lequel il existe une boule centrée en l'origine dont l'intersection avec L se réduise à $\{0\}$. C'est notamment le cas de $\mathbb{Z}^n$, le réseau hypercubique. On s'intéressera d'ailleurs souvent aux *réseaux entiers*, *i.e.* les réseaux inclus dans $\mathbb{Z}^n$, soit encore les sous-groupes de $\mathbb{Z}^n$. Le théorème suivant caractérise les réseaux, et justifie notre interprétation :

Théorème 3.1 *Soit L une partie non vide de $\mathbb{R}^n$. Les propriétés suivantes sont équivalentes :*

1. *L est un réseau.*
2. *Il existe des vecteurs $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_d$ linéairements indépendants de $\mathbb{R}^n$ tels que L soit exactement l'ensemble $L(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ des combinaisons linéaires à coefficients entiers des $\mathbf{b}_i$, c'est-à-dire :*

$$L = \mathbb{Z}.\mathbf{b}_1 \oplus \mathbb{Z}.\mathbf{b}_2 \oplus \cdots \oplus \mathbb{Z}.\mathbf{b}_d.$$

Ainsi, les réseaux sont en quelque sorte des analogues discrets des espaces vectoriels : tout réseau est en particulier un $\mathbb{Z}$-module libre de rang fini. On appelle *base* d'un réseau L toute $\mathbb{Z}$-base de L en tant que $\mathbb{Z}$-module, ou de façon équivalente, toute famille de vecteurs satisfaisant la condition 2 du théorème 3.1. Toutes les bases ont le même nombre d'éléments, égal à la dimension de $\mathbb{R}.L$, le sous-espace vectoriel engendré par L. Ce nombre est appelé le *rang*, ou la *dimension* de L, noté $\dim L$. Dans le cas particulier important où la dimension d du réseau est égale à la dimension de l'espace n, on dira que le réseau est *total*[1]. Par la suite, la dimension d'un réseau sera notée n si le réseau est total, et d dans le cas général.

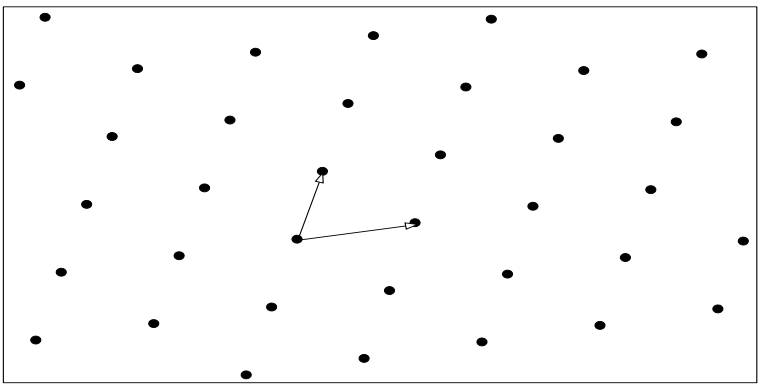


FIG. 3.3 – Un réseau de dimension 2, et une base.

Les réseaux furent historiquement étudiés en termes de formes quadratiques définies positives, à cause de la correspondance élémentaire suivante :

Théorème 3.2 *Soit* $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ *une base d'un réseau* L *de* $\mathbb{R}^n$*. Alors la fonction*

$$q(x_1, \ldots, x_d) = \|x_1\mathbf{b}_1 + \cdots + x_d\mathbf{b}_d\|^2, \tag{3.1}$$

est une forme quadratique définie positive sur $\mathbb{R}^d$*, de discriminant* $\det\left(\langle \mathbf{b}_i, \mathbf{b}_j\rangle\right)_{1\leq i,j\leq d}$*. Réciproquement, si* q *est une forme quadratique définie positive sur* $\mathbb{R}^d$*, il existe une famille libre* $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ *de* $\mathbb{R}^d$ *vérifiant* (3.1) *pour tout* $(x_1, \ldots, x_d) \in \mathbb{R}^d$*.*

On peut ainsi associer à tout réseau une classe de formes quadratiques définies positives, celles qui correspondent aux bases du réseau. Et réciproquement, à toute forme quadratique définie positive, on peut associer une famille de réseaux totaux.

Les bases diffèrent entre elles par des matrices de passage unimodulaires : si $\mathcal{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_d)$ est une base de L, alors une famille $(\mathbf{c}_1, \ldots, \mathbf{c}_d)$ de $\mathbb{R}^n$ est une base de L si et seulement si la matrice carrée $d \times d$ exprimant les $\mathbf{c}_i$ selon les $\mathbf{b}_i$ appartient à $GL_d(\mathbb{Z})$, c'est-à-dire qu'elle soit à coefficients entiers et de déterminant ± 1. En particulier, le déterminant $\Delta(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ de la matrice de Gram $(\langle \mathbf{b}_i, \mathbf{b}_j\rangle)_{1\leq i,j\leq d}$ est un réel strictement positif indépendant de la base $\mathcal{B}$: on l'appelle le *discriminant* de L, noté $\Delta(L)$. Le *déterminant* ou *volume* de L est par définition la racine carrée du discriminant :

$$\det(L) = \mathrm{vol}(L) = \sqrt{\Delta(L)}.$$

Remarquons que l'inégalité d'Hadamard nous assure que :

$$\mathrm{vol}(L) \leq \prod_{i=1}^{d} \|\mathbf{b}_i\|.$$

[1] L'adjectif « total » est issu de la terminologie employée dans l'ouvrage de Descombes [81, chapitre II].

On appelle défaut d'orthogonalité de la base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ le quotient $\prod_{i=1}^{d} \|\mathbf{b}_i\|/\mathrm{vol}(L) \geq 1$.

L'appelation de volume se justifie facilement dans le cas des réseaux totaux. En effet, le déterminant du réseau n'est alors autre que le volume au sens usuel (la mesure de Lebesgue) des *parallélepipèdes fondamentaux* du réseau, c'est-à-dire les ensembles de la forme

$$\left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i \ : \ (x_1, \ldots, x_n) \in [0,1[^n \right\},$$

où les $\mathbf{b}_i$ forment une base du réseau (voir figure 3.4).

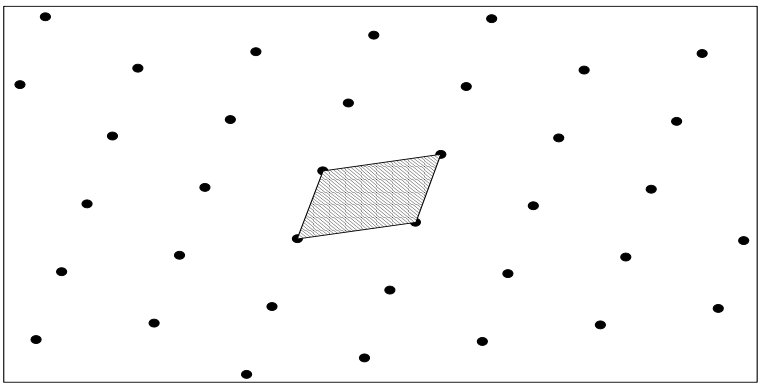


FIG. 3.4 – Interprétation géométrique du déterminant.

Un tel parallélepipède est un cas particulier de *domaine fondamental* du réseau L, *i.e.* une partie D mesurable de $\mathbb{R}^n$ telle que les ensembles $\mathbf{b} + D$ forment, pour $\mathbf{b}$ décrivant L, un recouvrement de $\mathbb{R}^n$, et sont d'intérieurs deux à deux disjoints. Tous les domaines fondamentaux ont même volume, égal au volume du réseau. Un autre exemple de domaine fondamental est la *cellule de Voronoï*, qui est l'ensemble des points de l'espace pour lesquels l'origine est le point du réseau le plus proche :

$$\mathrm{Vor}(L) = \{\mathbf{x} \in \mathrm{span}(L) : \forall \mathbf{y} \in L, \|\mathbf{x}\| \leq \|\mathbf{x} - \mathbf{y}\|\}.$$

Le déterminant ou volume d'un réseau est parfois aussi appelé *co-volume*. En effet, le quotient $\mathrm{span}(L)/L$ est compact et a la structure d'un tore, et son volume n'est autre que celui du réseau L.

Tout sous-groupe d'un réseau L est par définition encore un réseau. On dit que c'est un *sous-réseau* de L. Lorsqu'il a même dimension que L, on parlera de *sous-réseau total*. Toute famille libre d'un réseau est une base d'un sous-réseau. On peut caractériser les sous-réseaux totaux à l'aide de la notion usuelle d'indice de sous-groupe. Plus précisément, un sous-réseau M de L est total si et seulement si l'indice $[L : M]$ est fini, et dans ce cas, on a

$$\det(M) = \det(L) \times [L : M].$$

On appelle *sous-réseau primitif* de L tout sous-réseau qui est l'intersection de L avec un sous-espace vectoriel. Un sous-réseau M de L est primitif si et seulement si toute base du sous-réseau M peut se compléter en une base du réseau L (en fait, une seule base suffit). Une base d'un sous-réseau primitif est un ensemble de vecteurs *primitifs* du réseau.

Nous terminons cette sous-section en adaptant aux réseaux une notion classique de l'algèbre linéaire : la dualité. Pour toute famille libre $(\mathbf{e}_1, \ldots, \mathbf{e}_d)$ de $\mathbb{R}^n$, il existe une unique famille libre $(\mathbf{e}_1^*, \ldots, \mathbf{e}_d^*)$ du sous-espace engendré par les $\mathbf{e}_i$, vérifiant les relations :

$$\langle \mathbf{e}_i, \mathbf{e}_j^* \rangle = \begin{cases} 0 & \text{si } i \neq j, \\ 1 & \text{sinon.} \end{cases}.$$

La famille $(\mathbf{e}_1^*, \ldots, \mathbf{e}_d^*)$ est appelée la famille duale de $(\mathbf{e}_1, \ldots, \mathbf{e}_d)$. Soit Λ un réseau de $\mathbb{R}^n$. Le *réseau dual* de Λ est :

$$\Lambda^* = \{\mathbf{x} \in \mathrm{span}(L) \ : \forall \mathbf{y} \in \Lambda, \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}\}.$$

Le lien est naturel : Si $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ est une base de Λ, alors $(\mathbf{b}_1^*, \ldots, \mathbf{b}_d^*)$ est une base de Λ^*. Ainsi, le bidual n'est autre que le réseau lui-même :

$$(\Lambda^*)^* = \Lambda.$$

Notons que $\mathbb{Z}^n$ est autodual, c'est-à-dire qu'il est son propre dual : $(\mathbb{Z}^n)^* = \mathbb{Z}^n$. Par ailleurs, lorsque $(\mathbf{e}_1, \ldots, \mathbf{e}_n)$ est une base de $\mathbb{R}^n$, représentée par les vecteurs colonnes d'une matrice M selon la base canonique, alors la base duale est formée des vecteurs lignes de M^{-1}. On en déduit pour tout réseau L :

$$\det(L) \times \det(L^*) = 1.$$

3.2.2 Le théorème fondamental de Minkowski

On peut remarquer que l'intersection d'un réseau avec un ensemble borné de l'espace est nécessairement finie. Le théorème fondamental de Minkowski donne une condition suffisante sur l'ensemble pour que cette intersection soit non triviale. Il peut s'interpréter comme une généralisation du célèbre lemme combinatoire des tiroirs (parfois appelé principe de Dirichlet), qui assure que lorsqu'on range strictement plus que n objets dans n tiroirs, au moins deux objets sont dans le même tiroir. Cette analogie est apparente dans le résultat suivant, qui se démontre par un simple argument de volume :

Lemme 3.1 (Blichfeldt) *Soient L un réseau total de $\mathbb{R}^n$, et F une partie mesurable de $\mathbb{R}^n$ telle que $\mu(F) > \det(L)$. Alors F contient deux vecteurs distincts dont la différence est dans L.*

Ce lemme est au cœur du théorème de Minkowski :

Théorème 3.3 (Minkowski) *Soit L un réseau total de $\mathbb{R}^n$. Soit C une partie mesurable de $\mathbb{R}^n$, convexe, symétrique par rapport à 0, et telle que*

$$\mu(C) > 2^n \det(L).$$

Alors C contient au moins un point non nul de L.

On remarque que la borne sur les volumes est la meilleure possible, en considérant

$$C = \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i \mid (x_1, \ldots, x_n) \in]-1, 1[^n \right\},$$

où les $\mathbf{b}_i$ forment une base quelconque du réseau. Le théorème s'étend au cas d'égalité $\mu(C) \geq 2^n \mathrm{vol}(L)$, lorsque C est supposé de plus compact. En l'appliquant à des boules fermées, on obtient ainsi :

Corollaire 3.1 *Tout réseau L de $\mathbb{R}^n$ de dimension d contient un élément non nul $\mathbf{x}$ tel que*

$$\|\mathbf{x}\| \leq 2 \left(\frac{\det(L)}{v_d} \right)^{\frac{1}{d}},$$

v_d désignant le volume de la boule unité fermée de $\mathbb{R}^d$.

En utilisant les formules usuelles sur le volume de la boule unité fermée de $\mathbb{R}^d$, à savoir $v_d = \pi^{d/2}/\Gamma(1 + d/2)$, on peut montrer que la borne supérieure ci-dessus est inférieure à :

$$\sqrt{1 + \frac{d}{4}} \det(L)^{1/d}.$$

3.2.3 Minima successifs et constante d'Hermite

Le théorème de Minkowski nous amène naturellement à nous intéresser à la plus petite boule contenant un vecteur non nul d'un réseau, ou plus généralement, à la plus petite boule contenant un nombre donné de vecteurs linéairement indépendants d'un réseau. On définit ainsi les *minima successifs* d'un réseau d-dimensionnel L de $\mathbb{R}^n$ comme étant les nombres réels positifs $\lambda_1(L), \ldots, \lambda_d(L)$ où $\lambda_r(L)$ est la borne inférieure des nombres réels λ tels qu'il existe r vecteurs linéairement indépendants de L de norme inférieure ou égale à λ. Cette borne inférieure est atteinte, puisqu'un réseau est discret. Et clairement : $\lambda_1(L) \leq \lambda_2(L) \leq \cdots \leq \lambda_d(L)$. On appellera *écart* d'un réseau le rapport entre le second et le premier minimum, c'est-à-dire, $g(L) = \lambda_2(L)/\lambda_1(L) \geq 1$.

Le premier minimum $\lambda_1(L)$ est aussi la distance minimale entre deux points distincts de L. On appelle *invariant d'Hermite*[2] de L la quantité :

$$\gamma(L) = \left(\frac{\lambda_1(L)}{\det(L)^{1/\dim L}}\right)^2.$$

Une conséquence du théorème de Minkowski est qu'à dimension n fixée, l'invariant $\gamma(L)$ est borné lorsque L parcourt tous les réseaux de dimension n. On peut ainsi définir la *constante d'Hermite de rang* n, notée γ_n, comme étant la borne supérieure de $\gamma(L)$ pour L parcourant l'ensemble des réseaux de dimension n.[3]

$$\gamma_n = \max_{\dim(L)=n} \gamma(L).$$

Cette constante est nécessairement atteinte, et les réseaux totaux qui l'atteignent font l'objet de nombreuses recherches : ce sont les réseaux dits *critiques* (voir par exemple [220]).

La détermination exacte de la constante d'Hermite est l'un des problèmes fondamentaux de la géométrie des nombres : voir [68], qui est la « bible » dans ce domaine. En effet, comme chaque réseau L induit un empilement de sphères (de rayon $\lambda_1(L)/2$ et de centres les points du réseau), on remarque que connaître la valeur exacte de la constante d'Hermite revient à déterminer la densité de l'empilement de réseau le plus dense.

On sait que γ_n^n est un rationnel, parce qu'il existe toujours un réseau critique dont les matrices de Gram sont à coefficients entiers, mais on ne sait pas si γ_n est une suite croissante. Les valeurs exactes de γ_n ne sont connues que pour $1 \leq n \leq 8$ et $n = 24$ (voir Figure 3.5) : pour ces dimensions, les réseaux critiques sont connus et sont uniques. Le résultat pour $n = 24$ ne date que de 2004 : il a été établi par Cohn et Kumar [67], qui ont prouvé que le réseau de Leech était critique, et qu'il était l'unique réseau critique de dimension 24.

n	2	3	4	5	6	7	8	24
γ_n	$2/\sqrt{3}$	$2^{1/3}$	$\sqrt{2}$	$8^{1/5}$	$(64/3)^{1/6}$	$64^{1/7}$	2	4
Valeur approchée	1.1547	1.2599	1.4142	1.5157	1.6654	1.8114	2	4

FIG. 3.5 – Valeurs connues de la constante d'Hermite.

Si la valeur exacte de γ_n n'est en général pas connue, on dispose cependant de bonnes estimations. Tout d'abord, comme on l'a déjà mentionné dans la section précédente, le théorème de Minkowski permet de montrer :

$$\forall n,\ \gamma_n \leq 1 + \frac{n}{4}.$$

[2] L'invariant d'Hermite est élevé au carré pour des raisons historiques.

[3] Hermite fut le premier à démontrer l'existence d'une telle constante, dans le langage des formes quadratiques. Les techniques d'Hermite fournissent une majoration exponentielle en n, tandis que celles de Minkowski donnent une majoration linéaire en n.

Le meilleur encadrement asymptotique connu de la constante d'Hermite (voir [242, chapitre II] pour la minoration qui est une conséquence du théorème de Minkowski-Hlawka, et [68, chapitre 9] pour la majoration, qui raffine la constante donnée par le théorème de Minkowski) est le suivant :

$$\frac{n}{2\pi e} + \frac{\log(\pi n)}{2\pi e} + o(1) \leq \gamma_n \leq \frac{1.744n}{2\pi e}(1 + o(1)).$$

Pour de petites valeurs de n, les meilleures majorations connues de la constante d'Hermite sont celles de Cohn et Elkies [66].

Toute majoration de la constante d'Hermite fournit une borne supérieure sur le premier minimum d'un réseau quelconque de dimension d, et plus précisément sur $\lambda_1(L)/\det(L)^{1/d}$. Cependant, il faut garder à l'esprit que la constante d'Hermite correspond au pire cas. On peut facilement construire des réseaux L pour lesquels $\lambda_1(L)/\det(L)^{1/d}$ est arbitrairement petit. Toutefois, le cas moyen n'est pas très éloigné du pire cas, comme on le verra dans la section sur les réseaux aléatoires.

On a donné les principales estimations concernant le premier minimum d'un réseau. Pour ce qui est des autres minima, Minkowski a montré plus généralement que la constante d'Hermite s'appliquait aussi pour la moyenne géométrique des minima :

Théorème 3.4 (Deuxième théorème de Minkowski) *Pour tout réseau L de dimension d, les minima successifs vérifient pour tout $r \leq d$:*

$$\left(\prod_{i=1}^{r} \lambda_i(L)\right)^{1/r} \leq \sqrt{\gamma_d}\det(L)^{1/d}.$$

Il est normal de considérer la moyenne géométrique, plutôt que les minima séparément. En effet, il est facile de construire des réseaux L pour lesquels $\lambda_1(L)/\det(L)^{1/d}$ est arbitrairement petit, tandis que les $\lambda_i(L)/\det(L)^{1/d}$ pour $i \geq 2$ sont arbitrairement grands.

3.3 La réduction de réseau

L'un des résultats fondamentaux de l'algèbre bilinéaire est que tout espace euclidien admet une base orthogonale. Comme les réseaux sont des analogues discrets des espaces euclidiens, on peut se demander ce qu'il en est pour les réseaux. Or il n'est pas difficile de voir qu'un réseau n'admet en général pas de base orthogonale, contrairement au cas de $\mathbb{Z}^n$. C'est l'absence de base orthogonale dans le cas général qui motive la réduction de réseau, c'est-à-dire la formalisation de bases dites *réduites*, qui soient formées par des vecteurs relativement courts et quasi orthogonaux. Historiquement, la réduction de réseau fut introduite pour démontrer des propriétés universelles des réseaux, mais aussi pour obtenir des bornes supérieures sur la constante d'Hermite. En petite dimension, ces bornes supérieures sont tellement fines qu'elles permettent en fait de trouver la valeur exacte de la constante d'Hermite.

Toutes les notions de réduction que nous allons voir s'adaptent naturellement aux formes quadratiques définies positives. Les réductions de Minkowski et Korkine-Zolotareff ont d'ailleurs été introduites en termes de formes quadratiques définies positives. On verra dans la section 3.6 qu'il existe nécessairement des bases réduites pour toutes les notions de réduction présentées ici.

3.3.1 Bases réduites

Dans un premier temps, on peut se demander s'il existe une notion naturelle de base réduite, reliée aux minima successifs. On peut facilement montrer par récurrence qu'il existe une famille libre du réseau qui réalise les minima successifs : il existe des vecteurs linéairement indépendants $\mathbf{a}_1, \ldots, \mathbf{a}_d$ de L telle que pour tout i, $\|\mathbf{a}_i\| = \lambda_i(L)$. Curieusement, de telles familles de vecteurs ne forment pas

nécessairement une base, dès que la dimension du réseau est ≥ 4. Cela peut se voir en considérant le réseau formé par les quadruplets $(x_1, \ldots, x_4) \in \mathbb{Z}^4$ tels que $x_1 + \cdots + x_4$ est pair. Tous les minima de ce réseau valent $\sqrt{2}$. Et l'on peut facilement exhiber une famille libre de ce réseau qui atteigne tous ces minima, mais qui n'engendre qu'un sous-réseau d'indice deux.

Plus curieusement encore, il n'existe en général pas de base réalisant tous les minima successifs d'un réseau, dès que la dimension est supérieure ou égale à 5. Ce résultat non intuitif fut formulé pour la première fois par Korkine et Zolotareff [190] dans le langage des formes quadratiques. Considérons le réseau L engendré par les vecteurs $\mathbf{e}_1, \ldots, \mathbf{e}_n$ de la base canonique de $\mathbb{R}^n$, et le vecteur $\mathbf{h}$ dont toutes les coordonnées sont égales à $\frac{1}{2}$. On remarque que $\{\mathbf{h}, \mathbf{e}_2, \ldots, \mathbf{e}_n\}$ est une base de L, puisque $\mathbf{e}_1 = 2\mathbf{h} - \sum_{i=2}^{n} \mathbf{e}_i$. Les $\mathbf{e}_i$ ne forment cependant pas une base de L, puisqu'ils engendrent $\mathbb{Z}^n$, qui ne contient pas $\mathbf{h}$. Or $\|\mathbf{h}\| = \sqrt{n/4}$. Donc, si $n > 4$, tous les minima successifs de L sont égaux à 1, et ils sont atteints par les $\mathbf{e}_i$, mais ces n vecteurs linéairement indépendants ne forment pas une base. Pour $n \geq 11$, on peut même construire des réseaux engendrés par leurs vecteurs minimaux mais dont aucun ensemble de n vecteurs minimaux ne constitue une base (voir [69]).

Ainsi, il n'existe pas de notion canonique de base réduite. Cela signifie que plusieurs notions de réductions vont cohabiter. De façon intuitive, une base réduite sera formée de vecteurs dont les normes ne sont pas trop éloignées des minima successifs. D'un point de vue mathématique, les deux notions principales de réduction sont celles de Minkowski et Hermite-Korkine-Zolotareff (HKZ). Comme ces deux notions de réduction coïncident en dimension deux, nous allons d'abord étudier le cas de la dimension deux. Puis nous verrons la réduction de Minkowski, qui est peut-être la plus naturelle de toutes les notions de réduction. Enfin, nous présenterons pour terminer celle de Hermite-Korkine-Zolotareff.

3.3.2 Réduction en dimension deux

C'est Lagrange [201] qui formalisa le premier une notion de réduction en dimension deux, dans le langage des formes quadratiques. Cette notion de réduction est tellement naturelle que toutes les autres notions de réduction coïncident généralement avec elle en dimension deux.

Soit L un réseau de $\mathbb{R}^n$ de dimension deux. Une base $(\mathbf{b}_1, \mathbf{b}_2)$ de L est dite *Lagrange-réduite* (ou simplement *réduite*) si et seulement si $\|\mathbf{b}_1\| \leq \|\mathbf{b}_2\|$ et $|\langle \mathbf{b}_1, \mathbf{b}_2 \rangle| \leq \|\mathbf{b}_1\|^2/2$. Géométriquement, cela implique que $\mathbf{b}_2$ est à l'extérieur du disque de rayon $\|\mathbf{b}_1\|$ centré en l'origine, et que l'angle $(\mathbf{b}_1, \mathbf{b}_2)$ modulo π varie entre $\pi/3$ et $2\pi/3$.

Ainsi, il est très facile de vérifier si une base donnée est réduite ou pas. Le résultat suivant montre que cette notion de réduction est optimale :

Théorème 3.5 *Soit $(\mathbf{b}_1, \mathbf{b}_2)$ une base d'un réseau L de $\mathbb{R}^n$. La base $(\mathbf{b}_1, \mathbf{b}_2)$ est Lagrange-réduite si et seulement si $\|\mathbf{b}_1\| = \lambda_1(L)$ et $\|\mathbf{b}_2\| = \lambda_2(L)$.*

Si l'on admet ce résultat, il est alors clair qu'il existe toujours des bases L-réduites. Or par définition, le premier vecteur d'une telle base vérifie :

$$\|\mathbf{b}_1\| \leq (4/3)^{1/4} \mathrm{vol}(L)^{1/2}.$$

En particulier, on en déduit l'inégalité $\gamma_2 \leq \sqrt{4/3}$. Mais on a aussi clairement $\gamma_2 \geq \sqrt{4/3}$, en considérant le réseau hexagonal engendré par $(\mathbf{b}_1, \mathbf{b}_2)$ tel que $\|\mathbf{b}_1\| = \|\mathbf{b}_2\|$ et $\langle \mathbf{b}_1, \mathbf{b}_2 \rangle = \|\mathbf{b}_1\|^2/2$, c'est-à-dire le cas d'égalité de la réduction de Lagrange.

En d'autres termes, on pourrait résumer la réduction de Lagrange en une seule égalité :

$$\gamma_2 = \sqrt{4/3}.$$

3.3.3 La réduction de Minkowski

Soit L un réseau de $\mathbb{R}^n$ de dimension d. Une base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ de L est dite *Minkowski-réduite* (M-réduite) si $\|\mathbf{b}_1\| = \lambda_1(L)$ et si plus généralement, pour tout i allant de 1 à d, $\mathbf{b}_i$ est de norme minimale parmi tous ceux tels que $(\mathbf{b}_1, \ldots, \mathbf{b}_i)$ est un sous-ensemble primitif de L. De manière équivalente, pour tout $1 \leq i \leq d$, et pour tout d-uplet $(x_1, \ldots, x_d) \in \mathbb{Z}^d$ tel que $x_i, \ldots, x_d$ soient premiers entre eux :

$$\|x_1\mathbf{b}_1 + \cdots + x_d\mathbf{b}_d\| \geq \|\mathbf{b}_i\|. \tag{3.2}$$

Cette définition appelle plusieurs remarques. Tout d'abord, la définition est gloutonne : le choix de $\mathbf{b}_i$ dépend a priori des $\mathbf{b}_j$ choisis précédemment pour $j \leq i$. Ainsi, il est possible que pour un réseau donné, le vecteur des normes $(\|\mathbf{b}_1\|, \ldots, \|\mathbf{b}_d\|)$ varie selon la base Minkowski-réduite $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ choisie. Ce n'est pas le cas en petite dimension (≤ 4), mais Ryskov [295, 297] a exhibé un tel exemple en dimension 7. Pour autant, les quatre premières coordonnées de $(\|\mathbf{b}_1\|, \ldots, \|\mathbf{b}_d\|)$ sont toujours les mêmes, car les quatre premiers vecteurs d'une base M-réduite atteignent les quatre premiers minima :

Théorème 3.6 (van der Waerden [350]) *On définit $\delta_1 = \delta_2 = \delta_3 = \delta_4 = 1$ et par récurrence, $\delta_k = 5/4 + \sum_{5 \leq j < k} \delta_j$ pour tout $k \geq 5$. Soit $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ une base Minkowski-réduite de L. Alors pour tout $1 \leq i \leq d$:*

$$\|\mathbf{b}_i\| \leq \sqrt{\delta_i}\lambda_i(L).$$

En particulier, la réduction de Minkowski atteint simultanément tous les minima pour toute dimension ≤ 4. Cela montre aussi que la réduction de Minkowski coïncide avec celle de Lagrange en dimension deux.

Par ailleurs, la définition de la réduction de Minkowski semble nécessiter un nombre infini d'inégalités (3.2). Le premier théorème de finitude de Minkowski montre qu'à dimension d fixée, on peut se restreindre à un nombre fini d'inégalités (3.2), qu'on appelle conditions de Minkowski. En d'autres termes, pour toute dimension d, il existe un ensemble fini de d-uplets $(x_1, \ldots, x_d) \in \mathbb{Z}^d$ tel que toute base de d vecteurs soit M-réduite si et seulement si les inégalités (3.2) sont vérifiées pour cet ensemble de conditions. On connaît de tels ensembles jusqu'en dimension huit, mais leur cardinal explose très rapidement. C'est ce théorème de finitude qui permet d'interpréter l'ensemble des bases M-réduites comme un cône défini par un nombre fini de faces.

3.3.4 L'orthogonalisation de Gram-Schmidt

Afin d'évaluer la qualité d'une base, il est utile de trianguliariser la représentation de la base, ce que permettent de faire plusieurs factorisations usuelles. Pour des raisons historiques, c'est la méthode de Gram-Schmidt qui est la plus populaire dans la littérature des algorithmes de réduction, mais on pourrait aussi bien utiliser d'autres factorisations.

Soit $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ une famille libre de $\mathbb{R}^n$. On rappelle que le procédé d'orthogonalisation de Gram-Schmidt construit par récurrence une famille orthogonale $(\mathbf{b}_1^\star, \ldots, \mathbf{b}_d^\star)$, en prenant $\mathbf{b}_i^\star = \pi_i(\mathbf{b}_i)$, où π_i est la projection orthogonale sur le supplémentaire orthogonal du sous-espace $\sum_{j=1}^{i-1} \mathbb{R}\mathbf{b}_j = \sum_{j=1}^{i-1} \mathbb{R}\mathbf{b}_j^\star$. Cela revient à la formulation suivante :

$$\mathbf{b}_i^\star = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j}\mathbf{b}_j^\star, \text{ avec } \mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^\star \rangle}{\|\mathbf{b}_j^\star\|^2}.$$

La matrice dont les lignes expriment les $\mathbf{b}_i^\star$ selon les $\mathbf{b}_i$ est donc triangulaire inférieure, et sa diagonale n'est composée que de 1. Ainsi,

$$\mathrm{vol}(L) = \prod_{i=1}^{d} \|\mathbf{b}_i^\star\|. \tag{3.3}$$

On a en outre :

$$\|\mathbf{b}_i^\star\|^2 \leq \|\mathbf{b}_i\|^2 = \|\mathbf{b}_i^\star\|^2 + \sum_{j=1}^{i-1} |\mu_{i,j}|^2 \|\mathbf{b}_j^\star\|^2. \tag{3.4}$$

Cela signifie que les $\mathbf{b}_i^\star$ encadrent en quelque sorte les $\mathbf{b}_i$.

L'intérêt principal de l'orthogonalisation de Gram-Schmidt est qu'elle permet de trianguliariser la représentation matricielle d'un réseau. En effet, la famille $(\mathbf{b}_1^\star/\|\mathbf{b}_1^\star\|, \ldots, \mathbf{b}_d^\star/\|\mathbf{b}_d^\star\|)$ est une base orthonormale de $\mathbb{R}^n$. Or si l'on exprime la base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ selon cette base orthonormale (plutôt que la base canonique), on obtient en représentation ligne une matrice triangulaire inférieure, dont les coefficients diagonaux sont les $\|\mathbf{b}_i^*\|$:

$$\begin{pmatrix} \|\mathbf{b}_1^\star\| & 0 & \ldots & 0 \\ \mu_{2,1}\|\mathbf{b}_1^\star\| & \|\mathbf{b}_2^\star\| & \ddots & \\ & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \mu_{d,1}\|\mathbf{b}_1^\star\| & \ldots & \mu_{d,d-1}\|\mathbf{b}_{d-1}^\star\| & \|\mathbf{b}_d^\star\| \end{pmatrix}$$

On a déjà signalé que l'on aurait pu obtenir des résultats similaires avec d'autres factorisations. Par exemple, si l'on avait utilisé la décomposition d'Iwasawa de la matrice ligne M associée à $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$, on aurait écrit M sous la forme $M = UDO$ où U est une matrice triangulaire inférieure avec une diagonale remplie de 1, D est une matrice diagonale, et O est une matrice orthogonale. En d'autres termes, U serait la matrice des $\mu_{i,j}$, D la matrice des $\|\mathbf{b}_i^*\|$, et O la représentation ligne de $(\mathbf{b}_1^\star/\|\mathbf{b}_1^\star\|, \ldots, \mathbf{b}_d^\star/\|\mathbf{b}_d^\star\|)$.

On peut déduire de la représentation triangulaire que la famille de Gram-Schmidt d'une base d'un réseau permet de minorer les minima successifs :

Lemme 3.2 *Si $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ est une base d'un réseau L, alors pour tout $1 \leq i \leq d$:*

$$\lambda_i(L) \geq \min_{i \leq j \leq d} \|\mathbf{b}_j^\star\|.$$

Pour approcher le premier minimum d'un réseau, il suffit donc de déterminer une base telle que les $\mathbf{b}_i^\star$ ne soient pas trop éloignés les uns des autres. Ainsi, aucun des $\mathbf{b}_i^\star$ ne pourra être très éloigné de $\mathbf{b}_1^\star = \mathbf{b}_1$, et donc $\|\mathbf{b}_1\|$ sera relativement proche de $\lambda_1(L)$. Dans ce cas, on a même chacun des $\|\mathbf{b}_i\|$ qui n'est pas trop éloigné de $\lambda_i(L)$, si les $|\mu_{i,j}|$ sont petits, d'après (3.4). On va à présent introduire la notion de réduction faible, qui permet justement de majorer les $|\mu_{i,j}|$.

3.3.5 La réduction faible

On dit qu'une base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ d'un réseau est *faiblement réduite* si son orthogonalisation de Gram-Schmidt vérifie : pour tout $1 \leq j < i \leq d$,

$$|\mu_{i,j}| \leq \frac{1}{2}.$$

Géométriquement, cela signifie que la projection de $\mathbf{b}_i$ dans le sous-espace engendré par $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$, se situe à l'intérieur du parallélotope engendré par $\mathbf{b}_1^\star, \ldots, \mathbf{b}_{i-1}^\star$ avec coefficients $\leq 1/2$ en valeur absolue : on tente de réduire la composante de $\mathbf{b}_i$ sur le sous-espace engendré par $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$. Intuitivement, la base est faiblement orthogonale. On a alors d'après (3.4) :

$$\|\mathbf{b}_i^\star\|^2 \leq \|\mathbf{b}_i\|^2 \leq \|\mathbf{b}_i^\star\|^2 + \frac{1}{4} \sum_{j=1}^{i-1} \|\mathbf{b}_j^\star\|^2. \tag{3.5}$$

Cette notion fut introduite pour la première fois par Hermite [142], dans le langage des formes quadratiques. Elle est d'autant plus intéressante qu'il existe un algorithme très simple permettant de réduire faiblement une base, tout en conservant la même famille de Gram-Schmidt. Nous verrons cet algorithme dans la section 3.6 dédiée aux algorithmes de réduction.

La réduction faible est déjà apparente dans la réduction de Lagrange. En effet, une base $(\mathbf{b}_1, \mathbf{b}_2)$ est L-réduite si et seulement si elle est faiblement réduite et si $\|\mathbf{b}_1\| \leq \|\mathbf{b}_2\|$.

3.3.6 La réduction de Hermite, Korkine et Zolotareff

On dit qu'une base numérotée $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ d'un réseau L est *réduite au sens de Hermite, Korkine et Zolotareff* (HKZ) si elle est faiblement réduite, et si pour tout i, $\|\mathbf{b}_i^\star\| = \lambda_1(L_i)$ où L_i est la projection orthogonale $\pi_i(L)$ du réseau L sur le supplémentaire orthogonal du sous-espace engendré par $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$: l'ensemble L_i est un réseau contenant $\mathbf{b}_i^\star$, bien que la projection d'un réseau ne soit pas nécessairement un réseau en général. Cette notion est naturelle car la réduction faible permet de réduire la composante de $\mathbf{b}_i$ sur le sous-espace engendré par $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$, tandis que la deuxième condition minimise l'autre composante de $\mathbf{b}_i$, celle sur le supplémentaire orthogonal du sous-espace précédent. En particulier, comme pour la réduction de Minkowski, le premier vecteur $\mathbf{b}_1$ d'une base HKZ-réduite est un plus court vecteur de L.

Cette notion de réduction fut introduite dans le langage des formes quadratiques par Korkine et Zolotareff [191]. C'est une version renforcée de la notion de réduction suivante, introduite par Hermite [142] dans sa deuxième lettre à Jacobi, et qui correspond à la réduction LLL optimale avec insertion profonde de Schnorr et Euchner [307] : la base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ doit être faiblement réduite, et telle que pour tout i, $\mathbf{b}_i^\star$ ait une norme minimale parmi les vecteurs $\pi_i(\mathbf{b}_i), \pi_i(\mathbf{b}_{i+1}), \ldots, \pi_i(\mathbf{b}_d)$ (qui forment une base du réseau projeté L_i). La réduction HKZ coïncide avec la réduction de Lagrange et celle de Minkowski en dimension deux, mais la HKZ-réduction et la M-réduction peuvent différer à partir de la dimension trois. En particulier, lorsqu'une base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ est M-réduite, on a nécessairement $\|\mathbf{b}_1\| \leq \|\mathbf{b}_2\| \leq \cdots \leq \|\mathbf{b}_d\|$, ce qui n'est pas nécessairement vrai pour une base HKZ-réduite.

Les bases HKZ-réduites ont deux propriétés particulièrement intéressantes. La première est qu'une base HKZ-réduite fournit une très bonne approximation des minima successifs du réseau :

Théorème 3.7 *Si $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ est une base HKZ-réduite, alors : pour tout indice i tel que $1 \leq i \leq d$,*

$$\frac{4}{i+3} \leq \left(\frac{\|\mathbf{b}_i\|}{\lambda_i(L)}\right)^2 \leq \frac{i+3}{4}$$

La borne supérieure se démontre très facilement et est essentiellement due à Mahler [217] : il suffit de remarquer que $\|\mathbf{b}_i^\star\| = \lambda_1(L_i) \leq \lambda_i(L)$ (l'égalité découle de la HKZ-réduction, tandis que l'inégalité peut se montrer en considérant une famille libre atteignant simultanément tous les minima), et d'utiliser l'inégalité de droite de (3.5). La borne inférieure est elle démontrée dans [199] : on remarque d'abord que la HKZ-réduction implique pour tout $j \leq i$, $\|\mathbf{b}_j^\star\| \leq \|\mathbf{b}_i\|$, donc $\|\mathbf{b}_j\|^2/\|\mathbf{b}_i\|^2 \leq (j+3)/4$ d'après (3.5). Il est à noter que l'on n'a pas forcément $\|\mathbf{b}_i\| \geq \lambda_i(L)$ car l'on n'a pas nécessairement $\|\mathbf{b}_2\| \leq \|\mathbf{b}_3\| \leq \cdots \leq \|\mathbf{b}_d\|$. Ainsi, l'écart entre une base HKZ-réduite et les minima successifs du réseau est au plus polynomial, plus exactement inférieur à $\sqrt{(i+3)/4}$, tandis que la meilleure majoration connue pour une base M-réduite est par contre exponentielle. L'article [199] montre que les bornes du théorème 3.7 ne sont pas loin d'être optimales dans le pire cas.

La deuxième propriété intéressante des bases HKZ-réduites est leur adaptation aux raisonnements locaux. En effet, si $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ est HKZ-réduite, alors $(\pi_i(\mathbf{b}_i), \pi_i(\mathbf{b}_{i+1}), \ldots, \pi_i(\mathbf{b}_j))$ est HKZ-réduite pour tout $1 \leq i \leq j \leq d$. Ainsi, en étudiant les bases HKZ-réduites en petite dimension, on peut déduire des propriétés valables pour n'importe quelle dimension. Par exemple, le cas de la dimension deux montre que toute base HKZ-réduite $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ satisfait : $\|\mathbf{b}_i^\star\|/\|\mathbf{b}_{i+1}^\star\| \leq \sqrt{4/3}$ pour tout

$1 \leq i \leq d$. C'est avec ce genre de raisonnement que Korkine et Zolotareff ont trouvé de meilleures majorations sur la constante d'Hermite que l'inégalité d'Hermite.

3.4 Réseaux aléatoires

La plupart des résultats mathématiques sur les réseaux ne sont optimaux que dans le pire cas. Naturellement, on peut se demander ce qui se passe en moyenne. Curieusement, il existe une notion rigoureuse et naturelle de réseaux aléatoires, pour laquelle on sait prouver des propriétés en moyenne, et que nous allons résumer dans cette section. Cette notion de réseau aléatoire provient de mesures de Haar de groupes classiques. Elle fut historiquement introduite par Siegel en 1945 [325], pour fournir une démonstration alternative au théorème de Minkowski-Hlawka, qui permet de minorer la constante d'Hermite. Toutes les minorations connues de la constante d'Hermite font plus ou moins intervenir la méthode probabiliste : pour démontrer qu'il existe des réseaux suffisamment denses, on prouve qu'une certaine classe de réseaux doit en contenir par des arguments probabilistes. Ainsi, on sait construire de façon probabiliste des réseaux qui ont de grandes chances d'être denses, mais on ne sait pas construire de façon déterministe des réseaux aussi denses.

3.4.1 L'ensemble des réseaux

Pour pouvoir parler de réseaux aléatoires, il nous faut d'abord identifier l'ensemble des réseaux.

Un réseau total de $\mathbb{R}^n$ est défini par une base, qui peut se représenter par les lignes d'une matrice inversible $n \times n$ à coefficients réels. Dans un premier temps, on pourrait donc identifier l'ensemble des réseaux au groupe linéaire $GL_n(\mathbb{R})$ des matrices inversibles $n \times n$ à coefficients réels. Mais comme deux bases reliées par une matrice unimodulaire engendrent le même réseau, on préférerait plutôt identifier l'ensemble des réseaux au quotient $GL_n(\mathbb{Z})\backslash GL_n(\mathbb{R})$, où $GL_n(\mathbb{Z})$ est le sous-groupe des matrices $n \times n$ de déterminant ± 1, à coefficients dans $\mathbb{Z}$. Ici, la notation $B\backslash A$ désigne le quotient de A par B par la gauche : nous prenons cette notation car nous utilisons une représentation ligne des matrices, donc si $M \in GL_n(\mathbb{R})$ et $U \in GL_n(\mathbb{Z})$, les lignes de M et les lignes de UM engendrent le même réseau.

Cette identification n'est cependant pas parfaite. En effet, elle distingue les réseaux homothétiques, c'est-à-dire ceux qui sont reliés par multiplication par un scalaire. Pour toute base d'un réseau donné, on peut multiplier cette base par un scalaire de façon à obtenir une matrice de déterminant ± 1. Quitte à changer le signe de l'un des vecteurs de la base, on peut même garantir que ce déterminant vale $+1$, c'est-à-dire que la matrice appartient au groupe spécial linéaire $SL_n(\mathbb{R})$ des matrices $n \times n$ de déterminant $+1$, à coefficients dans $\mathbb{R}$. Et on peut garder le signe de ce déterminant lorsqu'on passe d'une base à une autre si l'on se restreint au sous-groupe $SL_n(\mathbb{Z})$ des matrices $n \times n$ de déterminant $+1$, à coefficients dans $\mathbb{Z}$. Ainsi, l'ensemble X_n des réseaux (totaux de $\mathbb{R}^n$) à homothétie près peut s'identifier au quotient $\Gamma\backslash G$ où $G = SL_n(\mathbb{R})$ et $\Gamma = SL_n(\mathbb{Z})$.

Il reste un inconvénient à cette nouvelle identification : elle distingue les réseaux isométriques, au sens suivant. Deux bases appartenant à G peuvent être distinctes modulo Γ et définir pourtant la même forme quadratique, c'est-à-dire qu'elles ont la même matrice de Gram. Ainsi, on dira que deux réseaux sont *isométriques* s'ils sont images l'un de l'autre par une transformation linéaire orthogonale de $\mathbb{R}^n$. Deux matrices lignes A et A' (de dimension $n \times n$) engendrent des réseaux isométriques si et seulement si $A' = MAU$ où $M \in GL_n(\mathbb{Z})$ et $U \in O_n(\mathbb{R})$, la notation $O_n(\mathbb{R})$ désignant le sous-groupe des matrices $n \times n$ orthogonales à coefficients dans $\mathbb{R}$. Par conséquent, on peut identifier l'ensemble Λ_n des réseaux (totaux de $\mathbb{R}^n$) à similitude près au double quotient $SL_n(\mathbb{Z})\backslash SL_n(\mathbb{R})/SO_n(\mathbb{R}) = X_n/SO_n(\mathbb{R})$, connu sous le nom d'espace des réseaux isométriques de densité 1, appelés aussi *réseaux unimodulaires*. Ici, $SO_n(\mathbb{R})$ désigne le sous-groupe des matrices de $O_n(\mathbb{R})$ de déterminant $+1$. En particulier, on en déduit que Λ_n a pour dimension $(n-1)(n+2)/2$. En effet, les groupes de Lie

$SL_n(\mathbb{R})$ et $SO_n(\mathbb{R})$ ont pour dimension respective n^2-1 et $n(n-1)/2$, tandis que $SL_n(\mathbb{Z})$ est discret.

3.4.2 Mesure naturelle sur l'ensemble des réseaux

La mesure de Lebesgue admet plusieurs généralisations intéressantes. Celle qui va nous intéresser ici est celle introduite par Haar [134] en 1933 pour les groupes topologiques localement compacts (voir par exemple [166]). Le groupe topologique $(\mathbb{R}^n,+)$ est localement compact, et on peut voir la mesure de Lebesgue sur $\mathbb{R}^n$ comme l'unique mesure (à constante multiplicative près) sur $\mathbb{R}^n$ qui soit invariante par translation. Plus généralement, pour n'importe quel groupe topologique localement compact G, la mesure de Haar est l'unique mesure (à constante multiplicative près) qui soit invariante par l'opération de groupe : comme le groupe G n'est pas nécessairement commutatif, on peut avoir une mesure de Haar à gauche et une autre à droite, et on dit que le groupe G est *unimodulaire* lorsque les deux mesures coïncident.

Le groupe de Lie $GL_n(\mathbb{R})$ est unimodulaire, et sa mesure de Haar est la suivante : un borélien S de $GL_n(\mathbb{R})$ a pour mesure

$$\mu(S)=\int_S \frac{dX}{|det(X)|^n},$$

où dX désigne la mesure de Lebesgue sur $\mathbb{R}^{n^2}$. En d'autres termes, la mesure est donnée par $dg=\det(g)^{-n}\prod_{i,j} dg_{i,j}$ où les $g_{i,j}$ désignent les coefficients de g. La propriété d'invariance signifie que pour tout borélien $\mathcal{S}$ de $GL_n(\mathbb{R})$, et pour tout $M\in GL_n(\mathbb{R})$, on a $\mu(M\mathcal{S})=\mu(\mathcal{S}M)=\mu(\mathcal{S})$.

Le groupe $SL_n(\mathbb{R})$ est lui aussi unimodulaire, donc il possède une mesure de Haar invariante à gauche et à droite, mais qui n'est pas nécessairement la même que celle de $GL_n(\mathbb{R})$. Lorsque l'on projette cette mesure sur le quotient $X_n=SL_n(\mathbb{Z})\backslash SL_n(\mathbb{R})$, on obtient une mesure finie pour X_n. Plus précisément, la valeur de cette mesure est (voir [299]) :

$$\mu\left(SL_n(\mathbb{Z})\backslash SL_n(\mathbb{R})\right)=\zeta(2)\dots\zeta(n)\sqrt{n}.$$

On peut donc normaliser cette mesure de façon à ce que X_n soit de mesure 1, et on obtient ainsi un espace de probabilité, d'où une notion naturelle de réseau aléatoire pour les réseaux à homothétie près.

Dans le cas des réseaux à similitude près, l'ensemble $\Lambda_n=SL_n(\mathbb{Z})\backslash SL_n(\mathbb{R})/SO_n(\mathbb{R})=X_n/SO_n(\mathbb{R})$ a lui aussi une mesure naturelle qui découle de la mesure précédente sur X_n et de la mesure invariante de Haar sur $SO_n(\mathbb{R})$. La mesure de Λ_n est aussi finie, ce que Siegel [325] démontra d'ailleurs à l'aide de la réduction de Minkowski. La valeur exacte est calculée par exemple dans [300, 340]. Si l'on plonge X_n dans $\mathbb{R}^{n^2}$ en associant à toute matrice $M\in GL_n(\mathbb{R})$ le cône $\{\lambda M : 0<\lambda\leq 1\}\subseteq\mathbb{R}^{n^2}$, le plongement de X_n a pour mesure de Lebesgue

$$\frac{\zeta(2)\zeta(3)\cdots\zeta(n)}{n}.$$

Là encore, on a un espace de probabilité sur Λ_n, d'où une notion naturelle de réseau aléatoire.

3.4.3 Cas de la dimension deux

Pour mieux comprendre la notion de réseau aléatoire, on peut regarder en détail le cas de la dimension deux. D'après ce qui précède, $\Lambda_2=SL_2(\mathbb{Z})\backslash\mathcal{H}$ où $\mathcal{H}=SL_2(\mathbb{R})/SO_2(\mathbb{R})$. Et $\mathcal{H}$ est isomorphe au demi-plan de Poincaré $\{x+iy\in\mathbb{C} : y>0\}$, car chaque classe d'équivalence de $\mathcal{H}$ a un représentant unique de la forme $y^{-1/2}\begin{pmatrix}1 & x\\ 0 & y\end{pmatrix}$, obtenu par rotation du premier vecteur colonne en un multiple positif de $\begin{pmatrix}1\\0\end{pmatrix}$.

Ainsi, Λ_2 est étroitement lié à l'action usuelle de $SL_2(\mathbb{Z})$ sur le demi-plan de Poincaré par les homographies $\tau \mapsto (a\tau + b)/(c\tau + d)$. La réduction de Lagrange montre que $\mathcal{D} = \{(x, y) : x^2 + y^2 \geq 1, |x| \leq 1/2, y > 0\}$ est un domaine fondamental pour l'action de $SL_2(\mathbb{Z})$ sur $\mathcal{H}$. En rajoutant la condition $x \geq 0$, on obtient un domaine fondamental pour $GL_2(\mathbb{Z})$. Par exemple, si $x + iy$ est un point de ce domaine fondamental, le réseau unimodulaire associé a pour premier minimum $y^{-1/2}$.

La mesure naturelle sur Λ_2 provient de la métrique hyperbolique $dxdy/y^2$ invariante par toute homographie $\tau \mapsto (a\tau + b)/(c\tau + d)$ où $ad - bc > 0$. Le domaine fondamental associé $\mathcal{D}$ n'est pas compact, mais il a une mesure finie car l'intégrale $\int_{y_0}^{\infty} dy/y^2$ converge. Pour normaliser la mesure de façon à ce que $\mu(\Lambda_2) = 1$, il suffit de prendre :

$$d\mu(x + iy) = \frac{3}{\pi} \frac{dxdy}{y^2}.$$

On peut ainsi calculer des espérances sur des réseaux aléatoires, en intégrant $d\mu(x + iy)$ sur $\{(x, y) : x^2 + y^2 \geq 1, |x| \leq 1/2, y > 0\}$. Par exemple, la valeur moyenne du premier minimum d'un réseau aléatoire unimodulaire de dimension deux vaut :

$$\frac{3}{\pi} \int_{-1/2}^{1/2} \left(\int_{\sqrt{1-x^2}}^{\infty} \frac{dy}{y^{2+1/2}} \right) dx = \frac{2}{\pi} \int_{-1/2}^{1/2} \frac{dx}{(1 - x^2)^{3/4}} \approx 0.6826,$$

tandis que la valeur maximale pour un réseau unimodulaire est $\sqrt{\gamma_2} = (4/3)^{1/4} \approx 1.0746$.

De même, on trouve la valeur moyenne de plusieurs invariants de réseaux :

- Pour λ_1^2, on obtient $(3 \ln 3)/(2\pi) \approx 0.5245$.
- Pour λ_2, on obtient ≈ 1.97314 par intégration numérique.
- Pour λ_2^2, on obtient l'infini.

On peut également calculer des valeurs moyennes concernant des bases réduites de réseaux aléatoires unimodulaires. Par exemple, la valeur moyenne de $|\mu_{2,1}|$ est $3(2 - \sqrt{3})/\pi \approx 0.2559$.

3.4.4 Propriétés des réseaux aléatoires

Il est utile pour l'intuition de connaître les propriétés des réseaux aléatoires. L'article [3] cite les suivantes :

- Pour tout borélien $X \subseteq \mathbb{R}^n \backslash \{0\}$, si L est un réseau aléatoire unimodulaire alors $E(|L \cap X|) = \text{vol}(X)$ et $E(|L \cap X|^2)$ est finie, où $|A|$ désigne le cardinal d'un ensemble A.
- Un réseau aléatoire n'admet avec probabilité 1 qu'une seule base de Korkine-Zolotareff (au signe près des vecteurs).

L'article [6] en cite d'autres. Il existe des constantes c et c' telles qu'un réseau aléatoire unimodulaire L de dimension n vérifie avec probabilité supérieure à $1 - n^{-c}$:

- $\lambda_1(L)$ vaut r_n à une erreur multiplicative près d'au plus $1 \pm c' n^{-1} \log n$, où r_n est le rayon de la boule n-dimensionnelle de volume 1. En particulier, on a asymptotiquement :

$$r_n = \frac{\Gamma(1 + n/2)^{1/n}}{\sqrt{\pi}} \approx \sqrt{\frac{n}{2\pi e}}.$$

- Il existe une base de L dont tous les vecteurs ont une norme $\leq r_n(1 + c' n^{-1} \log n)$. En particulier, tous les minima sont inférieurs à $r_n(1 + c' n^{-1} \log n)$, et ils sont donc asymptotiquement équivalents.

Ces propriétés permettent de facilement distinguer des réseaux spécifiques de réseaux aléatoires. Par exemple, en cryptologie, on rencontre fréquemment des réseaux dont le premier minimum est bien plus faible que l'estimation provenant de la constante d'Hermite, donc ces réseaux ne peuvent être aléatoires, et possèdent des propriétés exceptionnelles qui peuvent peut-être être exploitées. Lorsqu'un réseau est très loin d'être aléatoire, certains problèmes difficiles qui sont difficiles en général peuvent devenir faciles.

3.4.5 Génération de réseaux aléatoires

Récemment, plusieurs articles [121, 3] ont montré qu'il était possible de simuler facilement des réseaux aléatoires. Plus précisément, Ajtai [3] montre comment générer un réseau aléatoire avec erreur polynomialement faible en temps polynomial. D'un point de vue pratique, le résultat de Goldstein et Mayer [121] est sans doute plus intéressant. Ils montrent que lorsque N tend vers l'infini, l'ensemble fini $\mathcal{L}_N$ des réseaux entiers (de $\mathbb{Z}^n$) de déterminant N est uniformément distribué dans l'ensemble X_n des réseaux à homothétie près, au sens suivant : pour tout borélien A de X_n dont la frontière est de mesure nulle, la fraction des réseaux de $\mathcal{L}_N/N^{1/n}$ qui appartiennent à A tend vers $\mu(A)$ lorsque N tend vers l'infini. Ce résultat permet de simuler des réseaux aléatoires de la façon suivante :

- Choisir un grand entier N.
- Parmi tous les réseaux entiers totaux de dimension n et de volume N (qui sont en nombre fini), en prendre un aléatoirement avec distribution uniforme.
- Multiplier par $N^{-1/n}$ pour obtenir un réseau de volume 1.

Lorsque $N = p$ est premier, il est particulièrement facile de tirer aléatoirement dans $\mathcal{L}_N$, car lorsque p est suffisamment grand, la très grande majorité des réseaux de $\mathcal{L}_p$ sont engendrés par les lignes de matrices de la forme

$$\begin{pmatrix} p & 0 & 0 & \dots & 0 \\ x_1 & 1 & 0 & \dots & 0 \\ x_2 & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ x_{n-1} & 0 & \dots & 0 & 1 \end{pmatrix}$$

où les entiers x_i sont choisis avec distribution uniforme dans $\{0, \dots, p-1\}$. En effet, quand on regarde la forme normale d'Hermite des réseaux de $\mathcal{L}_p$, la diagonale est forcément composée d'un coefficient p et de $n-1$ coefficients égaux à 1, et il y a beaucoup plus de telles matrices si l'on fait figurer le coefficient p en haut de la diagonale.

3.5 Problèmes algorithmiques

On s'intéresse maintenant aux problèmes algorithmiques de la géométrie des nombres. On va d'abord clarifier les questions de représentation, et rappeler les problèmes simples bien résolus. Puis on énoncera les principaux problèmes difficiles de la géométrie des nombres. On donnera sans démonstration les résultats connus concernant la complexité algorithmique de ces problèmes. Les idées générales des algorithmes ne seront exposées que dans la section suivante.

3.5.1 Représentation

En algorithmique, on ne manipule que des *réseaux rationnels*, c'est-à-dire les réseaux inclus dans $\mathbb{Q}^n$. Dans ce cas, on peut toujours se ramener aux réseaux entiers, ceux qui sont inclus dans $\mathbb{Z}^n$. On représente généralement ces réseaux par une base, c'est-à-dire une matrice à coefficients entiers. Quand on donnera explicitement une matrice, on adoptera une représentation en lignes : les vecteurs lignes de la matrice seront les vecteurs de la base. La taille du réseau est mesurée par les dimensions de la matrice (le nombre de lignes d correspondant à la dimension du réseau et le nombre de colonnes n), et le nombre de bits maximal $\log B$ nécessaires à la représentation de chaque coefficient de la matrice.

Les problèmes algorithmiques de la géométrie des nombres sont souvent relatifs à des normes. On se limitera au cas le plus répandu de la norme euclidienne.

Avant d'aborder les problèmes difficiles, rappelons deux problèmes faciles qui peuvent se résoudre en temps polynomial déterministe :

– A partir d'une famille génératrice d'un réseau entier L, trouver une base du réseau L.
– Etant donnée une base d'un réseau entier L, décider si un vecteur donné de l'espace appartient à L, et si oui, trouver sa décomposition selon la base.

3.5.2 Problème du plus court vecteur

Le problème le plus célèbre de la géométrie algorithmique des nombres est le suivant :

Problème 3.1 (Plus court vecteur (SVP pour *Shortest Vector Problem*)) *Étant donnée une base d'un réseau entier L de dimension d, trouver $\mathbf{u} \in L$ tel que $\|\mathbf{u}\| = \lambda_1(L)$.*

Il y a deux problèmes d'approximation associés :
– Le problème d'approximation usuel consiste à trouver un point non nul $\mathbf{v} \in L$ tel que $\|\mathbf{v}\| \leq f\lambda_1(L)$, où f est un facteur multiplicatif donné.
– Le problème d'approximation d'Hermite (HSVP) consiste à trouver un point non nul $\mathbf{v} \in L$ tel que $\|\mathbf{v}\| \leq f\mathrm{vol}(L)^{1/d}$, où le facteur f est donné. La principale différence avec le problème d'approximation précédent est que l'on peut vérifier si une solution donnée est valide.

La constante d'Hermite montre si l'on peut approcher SVP à un facteur k, on peut résoudre HSVP à un facteur $f\sqrt{\gamma_d}$. Réciproquement, il est montré dans [214] que si l'on dispose d'un oracle résolvant HSVP à un facteur f, alors en temps polynomial et avec un nombre linéaire (en d) d'appels à l'oracle, on peut approcher SVP à un facteur f^2.

SVP a été conjecturé NP-dur dès 1981 [94] (voir également [214]). Ajtai a presque résolu cette conjecture en 1998 [5], en démontrant que SVP était NP-dur sous des réductions randomisées. La conjecture initiale avec des réductions déterministes reste ouverte, mais le meilleur résultat connu à l'heure actuelle [140] montre qu'on ne peut pas espérer approcher efficacement SVP à des facteurs quasi-polynomiaux. Mais ces résultats de NP-difficulté ont des limites : essentiellement, approcher SVP à un facteur $\sqrt{d/\log d}$ n'est sans doute pas NP-dur. Plus précisément, Aharonov et Regev [2] ont montré qu'il existe une constante c telle qu'approcher SVP à un facteur $c\sqrt{d}$ est dans l'intersection NP∩coNP, tandis que Goldreich et Goldwasser [116] ont montré que pour toute constante c, approcher SVP à un facteur $c\sqrt{d/\log d}$ est dans NP∩coAM.

Nous verrons en détail dans la section 3.6 les principaux algorithmes de réduction de réseau, mais nous pouvons déjà résumer la situation. L'algorithme LLL [205] (section 3.6.3.0) approche SVP à un facteur $(4/3+\varepsilon)^{(d-1)/2}$, et HSVP à un facteur $(4/3+\varepsilon)^{(d-1)/4}$, le tout en temps polynomial en $1/\varepsilon$ et la taille du réseau. Cet algorithme est la clef de voute de la plupart des algorithmes de réseau. Il est notamment utilisé dans les meilleurs algorithmes connus pour résoudre SVP de façon exacte :
– L'algorithme déterministe de Kannan [176] de complexité super-exponentielle $2^{O(d\log d)}$ opérations polynomiales (voir [138] pour une analyse de la constante).
– L'algorithme probabiliste d'Ajtai, Kumar et Sivakumar [8] de complexité simplement exponentielle $2^{O(d)}$ opérations polynomiales.

Les meilleurs algorithmes polynomiaux connus pour approcher SVP (mieux que LLL) utilisent ces algorithmes en petite dimension : en effet, on dimension d, on peut utiliser comme sous-routine un algorithme exact pour SVP en dimension $k = f(d)$, si la fonction $f(d)$ est suffisamment faible pour que le coût de la sous-routine reste polynomial en d. Par exemple, le temps super-exponentiel $2^{O(k\log k)}$ de l'algorithme de Kannan [176] reste polynomial en d si l'on choisit $k = \log d/\log\log d$. Avec un nombre polynomial d'appels à l'oracle SVP de dimension $\leq k$, Schnorr [302] a démontré que l'on pouvait approcher SVP à un facteur $(2k)^{2d/k}$, et HSVP à un facteur $(2k)^{d/k}$. Gama *et al.*[103] ont montré que l'analyse de Schnorr [302] n'était pas optimale : on peut élever à la puissance $\ln 2 \approx 0.69 < 1$ ces deux facteurs d'approximation. Gama *et al.*[103] ont aussi trouvé un algorithme légèrement meilleur : il approche SVP à un facteur $O(k)^{d/k}$ et HSVP à un facteur $O(k)^{d/(2k)}$, toujours avec un nombre polynomial d'appels à un oracle SVP de dimension $\leq k$. Le meilleur algorithme de ce type à l'heure actuelle est l'algorithme [106] développé par l'auteur en collaboration avec Nicolas Gama, qui

approche SVP à un facteur $((1+\varepsilon)\gamma_d)^{(d-k)/(k-1)}$ et HSVP à un facteur $\sqrt{(1+\varepsilon)\gamma_d}^{(d-1)/(k-1)}$, avec un nombre d'appels (polynomial en $1/\varepsilon$ et la taille du réseau) à un oracle SVP de dimension $\leq k$. A k fixé, les facteurs d'approximations de tous ces algorithmes restent donc exponentiels en d, comme pour LLL. Mais en prenant $k = \log d/\log\log d$, on obtient ainsi des algorithmes polynomiaux probabilistes qui approchent SVP et HSVP à des facteurs légèrement sous-exponentiels : $2^{O(d\log\log d/\log d)}$ si l'on utilise l'algorithme AKS [8] comme sous-routine.

3.5.3 Problème du plus proche vecteur

Le problème du plus court vecteur peut être vu comme un problème homogène : on cherche le rayon de la plus petite boule centrée en l'origine qui intersecte un réseau donné en un point non nul. On obtient une version non homogène en considérant des boules centrées cette fois en un point quelconque de l'espace donné. Pour un point $\mathbf{x}$ de $\mathbb{R}^n$, et un réseau L de $\mathbb{R}^n$, on notera ainsi $\operatorname{dist}(\mathbf{x}, L)$ la distance minimale entre $\mathbf{x}$ et un point de L. Le problème correspondant est le suivant :

Problème 3.2 (Plus proche vecteur (CVP pour *Closest Vector Problem*)) *Étant donnés une base d'un réseau entier $L \subseteq \mathbb{Z}^n$ de dimension d, et un point $\mathbf{x} \in \mathbb{Z}^n$, trouver $\mathbf{y} \in L$ tel que $\|\mathbf{x}-\mathbf{y}\| = \operatorname{dist}(\mathbf{x}, L)$. On obtient un problème d'approximation de façon analogue à SVP.*

Il a été démontré dès 1981 [94] que CVP était NP-dur (la preuve a depuis été simplifiée [178]). Approcher CVP à un facteur quasi-polynomial $2^{\log^{1-\varepsilon} d}$ est NP-dur [14, 83]. Comme pour SVP, l'approximation de CVP à un facteur $\sqrt{d/\log d}$ [116] n'est pas NP-dur, à moins que la hiérarchie polynomiale ne s'effondre.

Ces résultats suggèrent que CVP est plus dur que SVP. En tout état de cause, il n'est pas plus facile, puisque le résultat suivant a récemment été montré dans [119] : Étant donné un oracle d'approximation de CVP à un facteur $f(d)$ pour un réseau de dimension d, on peut approcher SVP en temps polynomial pout tout réseau de dimension d, avec le même facteur $f(d)$. Réciproquement, Kannan a démontré dans un manuscrit non publié que tout algorithme approchant SVP à un facteur $f(d)$ pouvait être transformé en un algorithme approchant CVP à un facteur $d^{3/2}f(d)$. On en déduit que l'on a essentiellement les mêmes résultats que pour SVP. En particulier, si l'on utilise l'algorithme de Schnorr [302] ou ses variantes [103], on obtient des algorithmes polynomiaux probabilistes qui approchent CVP à des facteurs légèrement sous-exponentiels : $2^{O(d\log\log d/\log d)}$ si l'on utilise AKS [8] comme oracle SVP de petite dimension. Moralement, ces facteurs sont de l'ordre de $(1+\varepsilon)^d$. Babai [17] (section 3.6.6) avait auparavant montré en 1986 que l'on pouvait approcher CVP à un facteur $(4/3+\varepsilon)^{d/2}$ en temps polynomial à l'aide de l'algorithme LLL. Et l'algorithme super-exponentiel de Kannan [176, 178] (section 3.6.4) peut aussi s'adapter à CVP pour fournir une solution exacte.

En pratique, on utilise souvent une technique heuristique (appelée *plongement*) pour réduire CVP en dimension d à SVP en dimension $d+1$, qui fut initiée par Kannan [178]. Considérons une base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ d'un réseau L de $\mathbb{Z}^n$, et un point $\mathbf{c}$ de l'espace dont on recherche le plus proche vecteur. La technique de plongement construit le réseau de $\mathbb{Z}^{n+1}$ engendré par les lignes de la matrice suivante :

$$L' = \begin{pmatrix} - & \mathbf{b}_1 & - & 0 \\ - & \vdots & - & 0 \\ - & \mathbf{b}_d & - & 0 \\ - & \mathbf{c} & - & 1 \end{pmatrix},$$

où le dernier coefficient 1 en bas à droite doit éventuellement être remplacé par une meilleure constante.

Les réseaux L' et L ont pratiquement la même dimension, et lorsque L est total, ils ont même déterminant. On pourrait donc croire dans un premier temps que $\lambda_1(L') \approx \lambda_1(L)$. Maintenant, supposons qu'un point $\mathbf{v} \in L$ minimise la distance à $\mathbf{c}$. On peut voir que $(\mathbf{c}-\mathbf{v}, 1) \in \mathbb{Z}^{n+1}$ est court

et appartient à L'. Ainsi, si la distance de $\mathbf{c}$ au réseau L est bien plus petite que $\lambda_1(L)$, on pourrait espérer que ce vecteur $(\mathbf{c} - \mathbf{v}, 1)$ est un plus court vecteur de L'. Si tel était le cas, on résoudrait notre instance de CVP au moyen de la SVP-instance définie par L'. Cette méthode est clairement heuristique (et on peut même trouver des contre-exemples [236]), mais elle est souvent utilisée en pratique pour de grandes dimensions.

3.6 Algorithmes de réduction de réseau ou de plus court vecteur

On va dresser dans cette section un inventaire des principaux algorithmes de réduction de réseau et de plus court vecteur (voir [214, 129, 240]) :

- Les algorithmes de réduction de réseau cherchent à trouver des bases réduites, en faisant des compromis entre qualité de réduction et temps de calcul, d'où plusieurs notions de réduction. Ces algorithmes résolvent souvent en même temps HSVP et l'approximation de SVP : dans ce cas, l'analyse de l'algorithme fournit une borne supérieure absolue sur la constante d'Hermite, et il n'est alors pas surprenant que ces algorithmes soient en fait des versions algorithmiques d'inégalités mathématiques bien connues portant sur la constante d'Hermite.
- Les algorithmes de plus court vecteur cherchent eux à trouver directement un plus court vecteur, mais tous les algorithmes connus font appel au préalable à un algorithme de réduction de réseau : il est donc logique de les regrouper dans la même section.

Notre inventaire commencera naturellement par l'algorithme de Lagrange [201] (section 3.6.1) qui résout complètement le cas de la dimension 2 en temps polynomial, et même quadratique. Cet algorithme effectue deux types d'opération, comme l'algorithme d'Euclide : des échanges et des translations (en soustrayant à un vecteur une combinaison linéaire des autres). Nous verrons ensuite plusieurs façons de généraliser cet algorithme en grande dimension.

La première et sans doute la plus naturelle consiste à conserver la philosophie de l'algorithme d'Euclide, en ordonnant les vecteurs de la base par norme croissante (quitte à faire des échanges), et à remplacer la réduction modulaire par une recherche de plus proche vecteur (CVP), c'est-à-dire à prendre la translation optimale. On obtient ainsi la réduction gloutonne [263] : jusqu'en dimension 4, elle est aussi efficace que l'algorithme de Lagrange et elle renvoie un résultat optimal, mais de grosses difficultés surviennent en dimension supérieure. En particulier, la phase de CVP paraît bien trop coûteuse pour être réaliste en grande dimension.

La deuxième façon procède ainsi :

- On assouplit les opérations de translation : au lieu de choisir la translation optimale, on va se contenter d'une translation convenable, dont la qualité n'est pas trop éloignée de la translation optimale. Cela revient à faire du CVP approché.
- Les échanges ne servent plus à ordonner les vecteurs de la base par norme croissante, mais plutôt les coefficients diagonaux dans la représentation triangulaire de la base.

Dans cette deuxième approche, l'idée générale est d'appliquer localement l'algorithme de Lagrange à des réseaux projetés de dimension deux bien choisis : ces réseaux ne sont autres que les matrices 2×2 qui se trouvent sur la diagonale, lorsqu'on représente la base de façon triangulaire, ce qui implique que les coefficients diagonaux de la représentation triangulaire ne décroissent pas trop vite. Cette approche remonte historiquement à Hermite [142], qui avait introduit les premiers algorithmes de réduction de réseau en dimension quelconque, afin de démontrer l'inégalité suivante sur la constante d'Hermite, connue aujourd'hui sous le nom d'inégalité d'Hermite :

$$\gamma_d \leq \gamma_2^{d-1}.$$

Hermite avait démontré que ces algorithmes terminaient, mais on ne sait toujours pas s'ils sont polynomiaux à dimension variable. L'algorithme LLL [205] (section 3.6.3.0) est une variante polynomiale des algorithmes d'Hermite : la qualité des bases renvoyées par l'algorithme LLL est très proche de

celles renvoyées par l'algorithme d'Hermite, mais l'algorithme LLL est lui prouvé polynomial, et c'est le premier algorithme de ce type.

L'algorithme LLL est à la base de tous les autres algorithmes de réduction efficaces connus. Tout d'abord, il constitue la première étape des deux seuls algorithmes efficaces de plus court vecteur, dont nous décrirons les idées de base : l'algorithme d'énumération déterministe de Kannan [176] qui est super-exponentiel, et l'algorithme probabiliste AKS [8] qui est simplement exponentiel.

Nous aborderons ensuite les généralisations [302, 103] de LLL par bloc, qui améliorent le facteur d'approximation de LLL en utilisant un algorithme exact pour SVP en petite dimension. De même que l'on pouvait voir l'algorithme LLL comme la version algorithmique de l'inégalité d'Hermite, on peut voir ces généralisations comme des versions algorithmiques approchées d'une autre inégalité portant sur la constante d'Hermite : $\gamma_d \leq \gamma_k^{(d-1)/(k-1)}$, appelée inégalité de Mordell.

Enfin, nous présenterons les algorithmes de Babai pour approcher CVP.

3.6.1 L'algorithme de Lagrange et sa généralisation gloutonne

L'algorithme de Lagrange [201] résout totalement le problème de la réduction de réseau en dimension 2 : il détermine une base réalisant les deux premiers minima, en un temps équivalent à celui de l'algorithme d'Euclide. Dans la littérature, il est souvent attribué (de façon incorrecte) à Gauss [109].

L'algorithme de Lagrange peut être vu comme une généralisation en dimension 2 de la variante centrée de l'algorithme d'Euclide (Algorithme 1).

Algorithme 1 L'algorithme d'Euclide centré.

Entrée : $(n, m) \in \mathbb{Z}^2$.
Sortie : $\text{pgcd}(n, m)$.

1:
2: **if** $|n| \leq |m|$ **then**
3: échanger n et m.
4: **end if**
5:
6: **while** $m \neq 0$ **do**
7: $r \longleftarrow n - qm$ où $q = \left\lfloor \frac{n}{m} \right\rceil$.
8: $n \longleftarrow m$
9: $m \longleftarrow r$
10: **end while**
11: Renvoyer $|n|$.

Cet algorithme correspond à une réduction dans le cas unidimensionnel. En effet, le pgcd n'est autre que le premier minimum du réseau $n\mathbb{Z} + m\mathbb{Z}$ engendré par n et m. La seule différence avec l'algorithme d'Euclide classique se situe dans l'étape 3, où on prend pour q l'entier le plus proche de $\frac{n}{m}$, et non sa partie entière. Cela revient à choisir l'entier q de façon à ce que $n - qm$ soit le plus petit possible en valeur absolue, ce qui garantit : $|n - qm| \leq \frac{|m|}{2}$. On montre facilement que l'algorithme d'Euclide centré a une complexité quadratique.

L'algorithme de Lagrange (Algorithme 2) est une généralisation naturelle en dimension 2.

Algorithme 2 L'algorithme de réduction de Lagrange.

Entrée : une base $(\mathbf{u}, \mathbf{v})$ d'un réseau L.
Sortie : une base réduite de L, réalisant $\lambda_1(L)$ et $\lambda_2(L)$.

1: **if** $\|\mathbf{u}\| < \|\mathbf{v}\|$ **then**
2: échanger $\mathbf{u}$ et $\mathbf{v}$
3: **end if**
4: **repeat**
5: $\mathbf{r} \longleftarrow \mathbf{u} - q\mathbf{v}$ où $q = \left\lfloor \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\|\mathbf{v}\|^2} \right\rceil$.
6: $\mathbf{u} \longleftarrow \mathbf{v}$
7: $\mathbf{v} \longleftarrow \mathbf{r}$
8: **until** $\|\mathbf{u}\| \leq \|\mathbf{v}\|$
9: Renvoyer $(\mathbf{u}, \mathbf{v})$.

L'analogie est claire : l'étape 3 cherche un entier q tel que $\mathbf{r} = \mathbf{u} - q\mathbf{v}$ soit le plus court possible. C'est précisément le cas lorsque la projection orthogonale de $\mathbf{r}$ sur $\mathbf{v}$ est aussi petite que possible, et cette projection peut être rendue $\leq \|\mathbf{v}\|/2$. On peut le voir géométriquement, et un calcul élémentaire assure que $q = \left\lfloor \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\|\mathbf{v}\|^2} \right\rceil$ convient.

On peut montrer que l'algorithme de Lagrange a une complexité quadratique (en la taille maximale des coefficients de la base d'entrée), mais la démonstration est plus difficile que pour l'algorithme d'Euclide centré : voir [314].

Semaev [314] a proposé une généralisation naturelle de l'algorithme de Lagrange en dimension trois, qui fut elle-même généralisée en dimension arbitraire par l'auteur, en collaboration avec Stehlé [263]. L'algorithme obtenu est appelé l'algorithme glouton (Algorithme 3).

Algorithme 3 L'algorithme de réduction gloutonne.

Entrée : une base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ d'un réseau L, et sa matrice de Gram
Sortie : une base G-réduite de L et sa matrice de Gram.

1: **if** $d = 1$ **then**
2: renvoyer $\mathbf{b}_1$
3: **end if**
4: **repeat**
5: Ordonner $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ par norme croissante, et mettre à jour la matrice de Gram.
6: Appeler récursivement l'algorithme pour G-réduire $(\mathbf{b}_1, \ldots, \mathbf{b}_{d-1})$.
7: Calculer un vecteur $\mathbf{c} \in L(\mathbf{b}_1, \ldots, \mathbf{b}_{d-1})$ le plus proche possible de $\mathbf{b}_d$.
8: $\mathbf{b}_d := \mathbf{b}_d - \mathbf{c}$ et mettre à jour la matrice de Gram.
9: **until** $\|\mathbf{b}_d\| \geq \|\mathbf{b}_{d-1}\|$
10: Renvoyer $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ et sa matrice de Gram.

On ordonne les vecteurs de la base par norme croissante, et on tente de réduire la norme du vecteur le plus long, en lui soustrayant une combinaison linéaire des premiers vecteurs : trouver la combinaison linéaire optimale revient exactement à résoudre un problème de plus proche vecteur. Une fois que le vecteur est réduit, on réordonne la base, et l'on continue jusqu'à ce qu'on ne puisse plus rien faire. En dimension deux, cela correspond bien à l'algorithme de Lagrange.

En collaboration avec Damien Stehlé [263], l'auteur a simplifié l'analyse de complexité (très technique) de Semaev [314] en dimension trois, et l'a généralisée jusqu'en dimension cinq. On sait désormais que l'algorithme glouton est quadratique jusqu'en dimension cinq : pour ces dimensions, il est donc aussi efficace que l'algorithme d'Euclide. En outre, les conditions de Minkowski montrent que jusqu'en dimension quatre, les bases renvoyées par l'algorithme glouton sont M-réduites, mais

[263] a exhibé des contre-exemples en dimension supérieures : à partir de la dimension cinq, les bases renvoyées par l'algorithme glouton peuvent être arbitrairement éloignées du premier minimum. Tous ces résultats se trouvent dans l'appendice A.

On peut résumer les résultats obtenus de la façon suivante : l'algorithme d'Euclide admet une généralisation naturelle en termes de réduction de réseau. Cet algorithme est en un sens optimal jusqu'en dimension cinq, mais pas au-delà. A partir de la dimension cinq, on ne sait plus si l'algorithme est encore polynomial, mais on sait de toute façon que le résultat envoyé peut être de qualité médiocre.

3.6.2 Gram-Schmidt et la réduction faible

Si les vecteurs $\mathbf{b}_1, \ldots, \mathbf{b}_d$ sont à coordonnées entières bornées par B, le calcul des coefficients de Gram-Schmidt (c'est-à-dire, de tous les rationnels $\mu_{i,j}$ et $\|\mathbf{b}_i^\star\|^2$) peut se faire en $O(d^5 \log^2 B)$ sans arithmétique rapide.

A partir de la représentation triangulaire d'une base, il est très facile de voir comment réduire faiblement une base (voir l'algorithme 4) : les vecteur $\mathbf{b}_i$ sont modifiés, mais pas leurs projections $\mathbf{b}_i^\star$.

Algorithme 4 Un algorithme de réduction faible.

Entrée : une base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ d'un réseau L.
Sortie : la base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ est faiblement réduite.

```
Calculer les coefficients de Gram-Schmidt μ_{i,j}.
for i allant de 2 à d do
    for j allant de i − 1 à 1 do
        b_i ⟵ b_i − ⌈μ_{i,j}⌋ b_j
        for k allant de 1 à j do
            μ_{i,k} ⟵ μ_{i,k} − ⌈μ_{i,j}⌋ μ_{j,k}
        end for
    end for
end for
```

3.6.3 Les algorithmes d'Hermite et de Lenstra-Lenstra-Lovász

Tous les algorithmes de cette section peuvent être vus comme des versions algorithmiques du résultat élémentaire suivant :

Théorème 3.8 (Inégalité d'Hermite [142]) *Pour tout entier $d \geq 2$:*

$$\gamma_d \leq \gamma_2^{d-1}. \tag{3.6}$$

Preuve. On va faire une démonstration par récurrence, un peu différente de celle donnée historiquement par Hermite. L'inégalité étant triviale pour $d = 2$, on la suppose vraie pour $d-1$. Considérons alors un plus court vecteur $\mathbf{b}_1$ d'un réseau L de dimension d. Notons $L' = \pi_1(L)$ le réseau de dimension $d-1$ obtenu par projection de L sur $\mathbf{b}_1^\perp$. Son volume vaut $\mathrm{vol}(L') = \mathrm{vol}(L)/\|\mathbf{b}_1\|$. Soit $\mathbf{b}_2^\star$ un plus court vecteur de L'. Par hypothèse, on a :

$$\|\mathbf{b}_2^\star\| \leq (4/3)^{(d-2)/4} \mathrm{vol}(L')^{1/(d-1)}.$$

Or on peut relever $\mathbf{b}_2^\star$ (par réduction faible) en un vecteur $\mathbf{b}_2 \in L$ non nul tel que $\|\mathbf{b}_2\|^2 \leq \|\mathbf{b}_2^\star\|^2 + \|\mathbf{b}_1\|^2/4$. Comme $\mathbf{b}_1$ ne peut être plus long que $\mathbf{b}_2$, on en déduit :

$$\|\mathbf{b}_1\| \leq \sqrt{4/3}\|\mathbf{b}_2^\star\| \leq (4/3)^{d/4} \mathrm{vol}(L')^{1/(d-1)},$$

ce que l'on peut réécrire en :

$$\|\mathbf{b}_1\| \leq (4/3)^{(d-1)/4} \mathrm{vol}(L)^{1/d},$$

ce qui achève la démonstration. Rétrospectivement, on peut remarquer qu'avec l'inégalité $\|\mathbf{b}_1\| \leq \sqrt{4/3}\|\mathbf{b}_2^\star\|$, on a en fait démontré l'inégalité :

$$\gamma_d \leq (4\gamma_{d-1}/3)^{(d-1)/d}.$$

En composant toutes ces inégalités, on retrouve bien l'inégalité d'Hermite :

$$\gamma_d \leq (4/3)^{(d-1)/d+(d-2)/d+\cdots+1/d} = (4/3)^{(d-1)/2}.$$

La démonstration historique donnée par Hermite dans sa première lettre [142] à Jacobi datée du 6 août 1845, procédait elle aussi par récurrence, mais de façon légèrement différente. On part d'un vecteur primitif $\mathbf{b}_1$ quelconque du réseau L. Si $\mathbf{b}_1$ satisfait l'inégalité d'Hermite, c'est-à-dire si $\|\mathbf{b}_1\| \leq (4/3)^{(d-1)/4}\mathrm{vol}(L)^{1/d}$, il n'y a rien à démontrer. Sinon, on applique l'hypothèse de récurrence au réseau projeté $L' = \pi_1(L)$: on sait donc qu'il existe un vecteur primitif non nul $\mathbf{b}_2^\star \in L'$ satisfaisant l'inégalité d'Hermite : $\|\mathbf{b}_2^\star\| \leq (4/3)^{(d-2)/4}\mathrm{vol}(L')^{1/(d-1)}$. On peut relever ce vecteur $\mathbf{b}_2^\star \in L'$ en un vecteur primitif $\mathbf{b}_2 \in L$ non nul tel que $\|\mathbf{b}_2\|^2 \leq \|\mathbf{b}_2^\star\|^2 + \|\mathbf{b}_1\|^2/4$. Comme $\mathbf{b}_1$ ne satisfait pas l'inégalité d'Hermite, on remarque que $\|\mathbf{b}_2\| < \|\mathbf{b}_1\|$: on peut donc remplacer $\mathbf{b}_1$ par $\mathbf{b}_2$, et recommencer depuis le début. Mais ce procédé ne peut continuer indéfiniment : en effet, il n'y a qu'un nombre fini de vecteurs de L qui soient de norme $\leq \|\mathbf{b}_1\|$. Par conséquent, il existe un vecteur non nul $\mathbf{b}_1 \in L$ satisfaisant l'inégalité d'Hermite. □

L'inégalité (3.6) suggère que si l'on utilise intelligemment un algorithme pour réduire des réseaux en dimension deux, on peut trouver dans n'importe quel réseau L de dimension d un vecteur non nul de norme inférieure à :

$$\sqrt{\gamma_2^{d-1}}\mathrm{vol}(L)^{1/d} = (4/3)^{(d-1)/4}\mathrm{vol}(L)^{1/d}.$$

C'est en quelque sorte l'idée sous-jacente à tous les algorithmes de cette section : les algorithmes d'Hermite et l'algorithme de Lenstra-Lenstra-Lovász (LLL). En fait, la preuve de (3.6) que l'on a donnée fournit un tel algorithme de façon implicite. Cet algorithme fait en sorte que la base soit faiblement réduite et que toutes les sections locales $(\pi_i(\mathbf{b}_i), \pi_i(\mathbf{b}_{i+1})) = (\mathbf{b}_i^\star, \mathbf{b}_{i+1}^\star + \mu_{i+1,i}\mathbf{b}_i^\star)$ soient L-réduites : ces sections locales correspondent aux matrices 2×2 se trouvant sur la diagonale lorsque l'on représente la base sous forme triangulaire. En d'autres termes, les bases réduites obtenues seraient faiblement réduites et telles que pour tout $1 \leq i \leq d$:

$$\|\mathbf{b}_{i+1}^\star\|^2 \geq \frac{3}{4}\|\mathbf{b}_i^\star\|^2, \tag{3.7}$$

c'est-à-dire que la décroissance des normes des vecteurs de Gram-Schmidt (ou encore, les coefficients diagonaux dans la représentation triangulaire) est au pire géométrique, ce qu'on appelle parfois condition de Siegel [326]. Il est alors facile de voir que le premier vecteur d'une telle base satisfait :

$$\|\mathbf{b}_1\| \leq (4/3)^{(d-1)/4}\mathrm{vol}(L)^{1/d},$$

comme annoncé. Mais on ne sait pas si cet algorithme et ceux d'Hermite sont polynomiaux : l'algorithme LLL va parvenir à garantir un temps polynomial en relâchant les inégalités (3.7).

Les algorithmes d'Hermite

Nous allons maintenant décrire les premiers algorithmes de réduction en dimension arbitraire, présentés par Hermite dans ses célèbre lettres [142] à Jacobi, dans le langage des formes quadratiques. Ils sont très proches de l'algorithme implicite de la preuve de (3.6), mais ils ne font pas appel de façon explicite à l'algorithme de Lagrange, bien qu'ils tentent de le généraliser. Ils ont été historiquement été présentés de façon récursive, mais on peut aisément les dérécursiver à la manière de LLL.

Algorithme 5 Le premier algorithme de réduction d'Hermite, décrit dans la première lettre à Jacobi [142].

Entrée : une base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ d'un réseau L.
Sortie :
1: **if** $d = 1$ **then**
2: renvoyer $\mathbf{b}_1$
3: **end if**
4: Appliquer récursivement l'algorithme à la base $(\pi_1(\mathbf{b}_2), \ldots, \pi_1(\mathbf{b}_d))$ du réseau projeté $\pi_1(L)$.
5: Relever les vecteurs $(\pi_1(\mathbf{b}_2), \ldots, \pi_1(\mathbf{b}_d))$ en $\mathbf{b}_2, \ldots, \mathbf{b}_d$ de telle sorte qu'ils soient faiblement réduits par rapport à $\mathbf{b}_1$.
6: **if** $\mathbf{b}_1$ satisfait l'inégalité d'Hermite, c'est-à-dire $\|\mathbf{b}_1\| \leq (4/3)^{(d-1)/4}\mathrm{vol}(L)^{1/d}$ **then**
7: Renvoyer $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$
8: **end if**
9: Echanger $\mathbf{b}_1$ et $\mathbf{b}_2$ car $\|\mathbf{b}_2\| < \|\mathbf{b}_1\|$, et recommencer depuis le début.

Le premier algorithme d'Hermite (Algorithme 5) date du 6 août 1845 et est décrit dans la première lettre [142] à Jacobi. Il est facile de voir que cet algorithme termine, et que la base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ renvoyée vérifie la notion de réduction suivante (que l'on appellera H1) :

- La base est faiblement réduite.
- Pour tout i, $\mathbf{b}_i^\star$ vérifie l'inégalité d'Hermite dans le réseau projeté $\pi_i(L)$:

$$\|\mathbf{b}_i^\star\| \leq (4/3)^{(d-i)/4}\mathrm{vol}(\pi_i(L))^{1/(d-i+1)}$$

On remarque que cette notion de réduction n'est pas très forte : par exemple, le défaut d'orthogonalité d'une base H1-réduite peut être arbitrairement grand, dès que l'on dépasse la dimension trois, comme l'illustre la base triangulaire suivante (où l'on fait tendre $\varepsilon > 0$ vers 0) :

$$\begin{pmatrix} 1 & 0 & 0 \\ 1/2 & \varepsilon & 0 \\ 1/2 & \varepsilon/2 & 1/\varepsilon \end{pmatrix}.$$

Par ailleurs, Hermite remarque lui-même que son premier algorithme ne coïncide pas avec l'algorithme de Lagrange en dimension deux. C'est semble-t-il l'une des raisons pour lesquelles il présente un deuxième algorithme (voir Algorithme 6) dans sa deuxième lettre [142] à Jacobi.

Algorithme 6 Le deuxième algorithme de réduction d'Hermite, décrit dans la deuxième lettre à Jacobi [142].

Entrée : une base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ d'un réseau L.
Sortie : une base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ faiblement réduite telle que pour tout i, $\|\mathbf{b}_i^\star\|/\|\mathbf{b}_{i+1}^\star\| \leq \gamma_2 = \sqrt{4/3}$.
En particulier, chaque $\mathbf{b}_i^\star$ vérifie l'inégalité d'Hermite dans le réseau projeté $\pi_i(L)$.

if $d = 1$ **then**
  renvoyer $\mathbf{b}_1$
end if
Quitte à faire un échange, s'assurer que $\|\mathbf{b}_1\| \leq \|\mathbf{b}_i\|$ pour tout $i \geq 2$.
Appliquer récursivement l'algorithme à la base $(\pi_1(\mathbf{b}_2), \ldots, \pi_1(\mathbf{b}_d))$ du réseau projeté $\pi_1(L)$.
Relever les vecteurs $(\pi_1(\mathbf{b}_2), \ldots, \pi_1(\mathbf{b}_d))$ en $\mathbf{b}_2, \ldots, \mathbf{b}_d$ de telle sorte qu'ils soient faiblement réduits par rapport à $\mathbf{b}_1$.
if $\|\mathbf{b}_1\| \leq \|\mathbf{b}_i\|$ pour tout $i \geq 2$ **then**
  renvoyer $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$
else
  recommencer depuis le début.
end if

Il est facile de voir que cet algorithme termine, et que la base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ renvoyée vérifie la notion de réduction suivante (que l'on appellera H2) :

- La base est faiblement réduite.
- Pour tout i, $\mathbf{b}_i^\star$ a une norme minimale parmi tous les vecteurs de la base $(\pi_i(\mathbf{b}_i), \pi_i(\mathbf{b}_{i+1}) \ldots, \pi_i(\mathbf{b}_d))$ du réseau projeté $\pi_i(L)$, c'est-à-dire que $\|\mathbf{b}_i^\star\| \leq \|\pi_i(\mathbf{b}_j)\|$ pour tout $1 \leq i \leq j \leq d$. En particulier, cela implique que

On remarque qu'une base H2-réduite vérifie nécessairement (3.7), c'est-à-dire pour tout i :

$$\|\mathbf{b}_i^\star\|/\|\mathbf{b}_{i+1}^\star\| \leq \gamma_2 = \sqrt{4/3}.$$

Cela implique que son défaut d'orthogonalité est borné :

$$\prod_{i=1}^{d} \|\mathbf{b}_i^\star\| \leq (4/3)^{d(d-1)/4} \mathrm{vol}(L(\mathbf{b}_1, \ldots, \mathbf{b}_d)).$$

Et cela montre aussi qu'une base H2-réduite est nécessairement H1-réduite.

Le deuxième algorithme d'Hermite est très proche de l'algorithme LLL avec insertion profonde de Schnorr-Euchner [307] : ils veulent atteindre la même notion de réduction.

L'algorithme LLL

Curieusement, on ne sait pas si les algorithmes d'Hermite sont polynomiaux lorsque la dimension est variable. C'est aussi le cas pour l'algorithme de Lenstra [207], qui est une version relachée du deuxième algorithme d'Hermite, où l'on remplace les inégalités $\|\mathbf{b}_i^\star\| \leq \|\pi_i(\mathbf{b}_j)\|$ par $c\|\mathbf{b}_i^\star\| \leq \|\pi_i(\mathbf{b}_j)\|$ où c est une constante telle que $1/4 < c < 1$. Cependant, Lenstra a démontré que son algorithme était polynomial à dimension fixée, ce qui était suffisant pour son célèbre résultat sur la programmation entière [207].

Il revient à Lenstra, Lenstra et Lovász [205] d'avoir inventé en 1982 le premier algorithme de réduction polynomial renvoyant des bases de qualité très proches de celles d'Hermite. Cet algorithme, connu sous le nom de LLL ou L^3, est essentiellemment une version relâchée du deuxième algorithme d'Hermite : László Lovász découvrit qu'une modification cruciale pouvait garantir un temps polynomial ; plus précisément, par rapport à la réduction H2, on remplace pour tout i toutes les inégalités

$\|\mathbf{b}_i^\star\| \leq \|\pi_i(\mathbf{b}_j)\|$ par une unique inégalité $c\|\mathbf{b}_i^\star\| \leq \|\pi_i(\mathbf{b}_{i+1})\|$ où c est une constante telle que $1/4 < c < 1$. L'algorithme final (en collaboration avec Arjen Lenstra) fut publié dans [205].

Soit un réel δ dans $]\frac{1}{4}, 1]$. Une base numérotée $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ de L est dite LLL-*réduite* à un facteur δ si elle est faiblement réduite, et si elle satisfait la *condition de Lovász* : pour tout $1 < i \leq d$,

$$\left\|\mathbf{b}_{i+1}^\star + \mu_{i+1,i}\mathbf{b}_i^\star\right\|^2 \geq \delta\|\mathbf{b}_i^\star\|^2.$$

Expliquons cette mystérieuse condition. Comme l'orthogonalisation de Gram-Schmidt dépend de l'ordre des éléments, ses vecteurs changent si $\mathbf{b}_i$ et $\mathbf{b}_{i+1}$ sont interchangés ; en fait, seuls $\mathbf{b}_i^\star$ et $\mathbf{b}_{i+1}^\star$ peuvent éventuellement changer. Et le nouveau $\mathbf{b}_i^\star$ n'est autre que $\mathbf{b}_{i+1}^\star + \mu_{i+1,i}\mathbf{b}_i^\star$, donc la condition de Lovász signifie qu'en interchangeant $\mathbf{b}_i$ et $\mathbf{b}_{i+1}$, la longueur de $\mathbf{b}_i^\star$ ne diminue pas trop, la perte étant quantifiée par δ : on ne peut pas gagner beaucoup sur $\|\mathbf{b}_i^\star\|$ en les échangeant. De manière équivalente :

$$\delta\|\mathbf{b}_i^\star\|^2 \leq \|\pi_i(\mathbf{b}_{i+1})\|^2,$$

ce qui illustre bien la relation avec la notion de réduction H2. La valeur la plus naturelle de la constante δ est donc $\delta = 1$ (en dimension 2, on retombe sur la réduction de Lagrange), mais on ne connaît alors pas d'algorithme prouvé polynomial pour obtenir une base LLL-réduite. La réduction LLL fut initialement[4] présentée [205] avec le facteur $\delta = \frac{3}{4}$, si bien que par réduction LLL dans la littérature, on entend souvent réduction LLL avec ce facteur-là.

La condition de Lovász peut aussi s'écrire de façon équivalente : pour tout i,

$$\|\mathbf{b}_{i+1}^\star\|^2 \geq \left(\delta - \mu_{i+1,i}^2\right)\|\mathbf{b}_i^\star\|^2,$$

ce qui est un relâchement de (3.7). La réduction LLL garantit ainsi que chaque $\mathbf{b}_{i+1}^\star$ ne peut être beaucoup plus court que $\mathbf{b}_i^\star$: la décroissance est au pire géométrique. Le résultat suivant en découle :

Théorème 3.9 *On suppose $\frac{1}{4} < \delta \leq 1$, et on note $\alpha = 1/(\delta - \frac{1}{4})$. Soit $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ une base LLL-réduite à un facteur δ d'un réseau de L de $\mathbb{R}^n$. Alors :*

1. $\|\mathbf{b}_1\| \leq \alpha^{(d-1)/4}(\det L)^{1/d}$.
2. *Pour tout i :* $\|\mathbf{b}_i\| \leq \alpha^{(d-1)/2}\lambda_i(L)$.
3. $\|\mathbf{b}_1\| \times \cdots \times \|\mathbf{b}_d\| \leq \alpha^{d(d-1)/4} \det L$.

Ainsi, une base LLL-réduite fournit une solution approchée au problème de la réduction de réseau. En prenant δ très proche de 1, on retombe sur l'inégalité d'Hermite de façon approchée, où la constante $4/3$ serait remplacée par $4/3 + \varepsilon$.

L'autre intérêt de cette notion de réduction provient du fait qu'il existe un algorithme simple qui permet de réduire une base au sens de LLL, et qui est relativement proche du deuxième algorithme d'Hermite (Algorithme 6). Dans sa version la plus simple, l'algorithme LLL correspond à l'algorithme 7.

Algorithme 7 L'algorithme de réduction LLL.

Entrée : une base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ d'un réseau L.
Sortie : la base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ est LLL-réduite à un facteur δ.

1: Réduire faiblement $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ (voir Algorithme 4).
2: **if** il existe un indice j qui ne satisfait pas la condition de Lovász **then**
3: échanger $\mathbf{b}_j$ et $\mathbf{b}_{j+1}$, puis retourner à l'étape 1.
4: **end if**

Par rapport à cette version simple, les versions dites itératives de l'algorithme LLL considèrent plutôt le plus petit indice j ne satisfaisant pas la condition de Lovász : à l'inverse, le deuxième algorithme d'Hermite s'intéressait lui au plus grand indice j violant la définition de la réduction H2.

[4] Cela simplifie l'exposition, et qualitativement, les résultats sont identiques.

Théorème 3.10 *On suppose ici $\frac{1}{4} < \delta < 1$. Si les $\mathbf{b}_i$ sont rationnels, l'algorithme 7 fournit une base LLL-réduite en temps polynomial en la taille maximale des coefficients des $\mathbf{b}_i$, la dimension du réseau et la dimension de l'espace.*

On va esquisser une preuve de ce résultat fondamental, en supposant pour simplifier les $\mathbf{b}_i$ entiers. Tout d'abord, il est clair que si l'algorithme termine, la base obtenue est LLL-réduite pour le facteur δ. Pour voir que l'algorithme termine, analysons chaque échange (étape 3). Lorsqu'on échange $\mathbf{b}_j$ et $\mathbf{b}_{j+1}$, seuls $\mathbf{b}_j^\star$ et $\mathbf{b}_{j+1}^\star$ peuvent être modifiés parmi tous les vecteurs de Gram-Schmidt. Notons donc $\mathbf{c}_j^\star$ et $\mathbf{c}_{j+1}^\star$ les nouveaux vecteurs de Gram-Schmidt après échange. Comme le produit des normes des vecteurs de Gram-Schmidt vaut $\det L$, on a :

$$\|\mathbf{c}_j^\star\| \times \|\mathbf{c}_{j+1}^\star\| = \|\mathbf{b}_j^\star\| \times \|\mathbf{b}_{j+1}^\star\|.$$

Le fait que la condition de Lovász ne soit pas satisfaite s'écrit : $\|\mathbf{c}_j^\star\|^2 < \delta\|\mathbf{b}_j^\star\|^2$. On en déduit :

$$\|\mathbf{c}_j^\star\|^{2(d-j+1)}\|\mathbf{c}_{j+1}^\star\|^{2(d-j)} < \delta\|\mathbf{b}_j^\star\|^{2(d-j+1)}\|\mathbf{b}_{j+1}^\star\|^{2(d-j)}.$$

Cela nous invite à considérer la quantité :

$$D = \|\mathbf{b}_1^\star\|^{2d}\|\mathbf{b}_2^\star\|^{2(d-1)} \times \cdots \times \|\mathbf{b}_d^\star\|^2.$$

À chaque échange, D diminue d'un facteur $\delta < 1$. On remarque que D peut se décomposer comme un produit de d déterminants de Gram $D_i = \Delta(\mathbf{b}_1, \ldots, \mathbf{b}_i)$ pour i variant de 1 à d. D est donc en fait un entier, puisque les $\mathbf{b}_i$ le sont. Il s'en suit que le nombre d'échanges est au plus logarithmique en la valeur initiale de D, que l'on peut majorer par B^{2d} où B est le maximum des normes initiales $\|\mathbf{b}_i\|$. Pour borner la complexité de l'algorithme, il faut aussi majorer la taille des coefficients rationnels $\mu_{i,j}$. Une analyse minutieuse fondée sur les D_i permet de montrer que les $\mu_{i,j}$ restent polynomiaux (voir [205, 214, 65, 88]).

Comme on ne sait pas si l'algorithme reste polynomial pour $\delta = 1$, on choisit généralement en pratique $\delta = 0.99$.

L'algorithme 7 est loin d'être optimal. Toutes les implémentations réécrivent l'algorithme de façon incrémentale : on garde en mémoire un indice k qui nous assure que les k premiers vecteurs sont déjà réduits, et on cherche à augmenter k jusqu'à atteindre d. Pour des versions « optimisées » de l'algorithme LLL (voir [65, 307, 170]), la complexité (sans arithmétique rapide) est $\mathcal{O}(nd^5 \log^3 B)$, B étant un majorant des valeurs absolues des coordonnées des $\mathbf{b}_i$, n la dimension de l'espace, et d la dimension du réseau.

Variantes de LLL

Il existe de nombreuses variantes de l'algorithme LLL (voir [65]). On peut notamment modifier simplement l'algorithme de façon à ce qu'il accepte en entrée une famille génératrice, tout en renvoyant une base LLL-réduite (voir [65]). La modification la plus importante en pratique concerne le calcul de l'orthogonalisation de Gram-Schmidt. L'algorithme LLL décrit précédemment opère sur des rationnels, et tous les calculs doivent être réalisés de manière exacte. Sa complexité est polynomiale, mais elle n'est pas négligeable : elle est en particulier cubique en $\log B$. En grande dimension, la taille des coefficients $\mu_{i,j}$ (qui est en $O(d \log B)$) rend l'algorithme impraticable, si bien qu'on est tenté de garder seulement une approximation de $\mu_{i,j}$ en utilisant une représentation en virgule flottante. Mais il peut alors se poser des problèmes de stabilité numérique : la base n'est plus assurée d'être faiblement réduite, et les échanges peuvent conduire à des boucles infinies. Schnorr [303] a proposé une version flottante de l'algorithme, qui est prouvée stable. En pratique, on utilise des variantes plus simples dont la stabilité n'est pas assurée, par exemple celle de Schnorr-Euchner [307, 319].

En collaboration avec Damien Stehlé, l'auteur a revisité l'algorithme LLL en virgule flottante [264] (voir l'appendice B). Jusqu'à [264], toutes les versions connues de l'algorithme LLL avaient une complexité cubique en $\log B$: même en virgule flottante, la précision requise était telle que la complexité pouvait devenir cubique. En un sens, ceci n'était pas naturel, car l'algorithme de Lagrange et sa généralisation gloutonne sont quadratiques en petite dimension, tout en fournissant une meilleure qualité de réduction que LLL. L'article [264] a introduit l'algorithme L^2, qui est la première variante de l'algorithme LLL qui soit prouvée quadratique en $\log B$ (comme l'algorithme de Lagrange), tout en restant polynomiale à dimension variable. Plus précisément, l'algorithme L^2 a une complexité d'essentiellement $O(nd^4 \log^2 B)$. Pour le cas usuel $d = n$, on a donc une complexité $O(d^5 \log^2 B)$ qui est similaire à celle du calcul exact des coefficients de Gram-Schmidt.

Ainsi, on peut voir l'algorithme L^2 comme une version algorithmique de l'inégalité d'Hermite, et dont la complexité généralise naturellement celle de l'algorithme d'Euclide, contrairement à l'algorithme initial LLL.

Les différences entre l'algorithme L^2 et l'algorithme LLL initial sont les suivantes. Tout d'abord, au niveau du fonctionnement de l'algorithme, on ne garde qu'une approximation des coefficients de Gram-Schmidt, et pour des raisons de stabilité numérique, la réduction faible est décomposée en plus d'étapes, en surveillant que les approximations ne dégénèrent pas. En outre, l'analyse de l'algorithme L^2 est bien plus complexe :

- Il faut pouvoir garantir que les approximations des coefficients de Gram-Schmidt sont suffisamment bonnes à chaque instant. En particulier, on utilise le fait que dans la version incrémentale de LLL, les k premiers vecteurs sont déjà LLL-réduits, donc ils ont certaines propriétés d'orthogonalité, ce qui permet de contrôler les erreurs lorsqu'on calcule de nouveaux coefficients de Gram-Schmidt de façon approchée. L'analyse d'erreur a d'ailleurs permis d'exhiber des pires cas pour d'autres implémentations en virgule flottante de l'algorithme LLL.
- Pour garantir que la complexité est quadratique comme l'algorithme d'Euclide, on utilise une analyse amortie, qui s'avère bien plus technique que pour l'algorithme d'Euclide ou l'algorithme de Lagrange. Dans l'algorithme d'Euclide, on a un nombre linéaire d'itérations qui ont chacune un coût au plus quadratique. On pourrait donc croire que le coût global est cubique : pourtant ce coût est en fait quadratique, car il y a un phénomène d'amortissement dû au fait que les divisions euclidiennes opèrent sur des nombres de plus en plus petits. Un phénomène similaire se produit pour l'algorithme L^2.

Efficacité pratique de LLL

Il est facile de voir que la première borne du théorème 3.9 peut être atteinte : il suffit de construire une matrice triangulaire pour laquelle les blocs 2×2 diagonaux aient une forme convenable. (par exemple, le réseau hexagonal pour $\alpha = 1$). En rajoutant des contraintes à l'exemple précédent, on peut voir que la deuxième borne du théorème 3.9 est elle presque atteinte, c'est-à-dire qu'on peut construire une base LLL-réduite telle que $\|\mathbf{b}_1\|/\lambda_1(L)$ soit aussi gros que $\alpha^{(d-1)/2}$ à une constante près. Mais dans ces pires cas, les bases LLL-réduites ont une forme très particulière, et l'on peut raisonnablement penser que la situation puisse être meilleure en pratique.

En fait, dès l'apparition de l'algorithme LLL, la communauté scientifique s'est rendu compte que l'algorithme se comportait bien mieux que ne le laissait supposer le théorème 3.9, tant du point de vue du temps de calcul, que de la qualité de la base renvoyée. Cela a largement contribué à la popularité de LLL. En petite dimension, les implémentations de LLL trouvaient toujours un vecteur extrêmement court, voire le plus court vecteur (lorsque l'écart du réseau était suffisamment important). Plus généralement, lorsque les premiers minima étaient beaucoup plus faibles que les autres, il n'était pas rare de retrouver le « sous-espace » engendré par ces premiers minima. Cependant, ces phénomènes n'avaient jamais vraiment été quantifiés.

Afin d'y voir un peu plus clair, l'auteur a récemment mené en collaboration avec Damien Stehlé [265]

(voir l'appendice C) de nombreuses expérimentations sur LLL avec des réseaux aléatoires, jusqu'en dimension 150. Avec le recul, le phénomène semble être le suivant :

– Jusqu'à des dimensions de l'ordre de 50, LLL se comporte extrêmement bien, souvent comme un oracle de plus court vecteur en petite dimension. Mais les choses empirent à mesure que la dimension augmente : voir la figure 3.6.

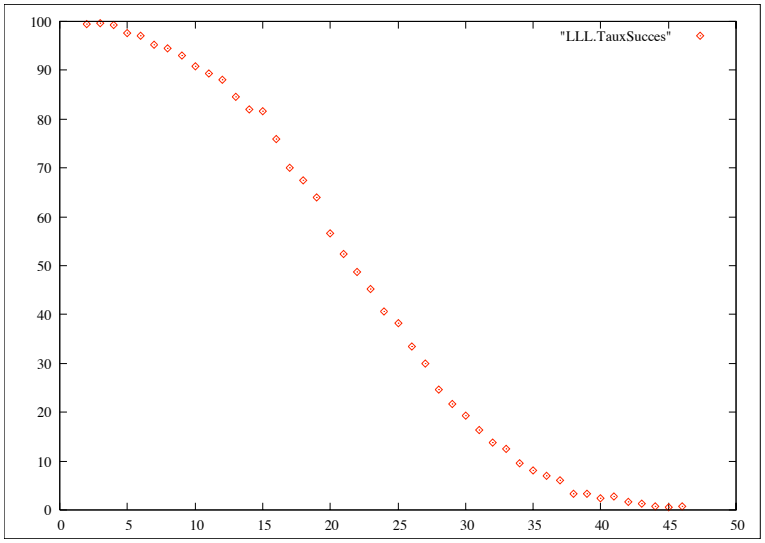


FIG. 3.6 – Probabilité expérimentale (en pourcentage) que l'algorithme LLL (avec facteur optimal) renvoie un plus court vecteur d'un réseau aléatoire, en fonction de la dimension.

– Au-delà, le premier vecteur renvoyé par LLL a une norme de l'ordre de $1.02^d \mathrm{vol}(L)^{1/d}$ (contre $1.07^d \mathrm{vol}(L)^{1/d}$ dans le pire cas), comme en témoigne la figure 3.7. Cela signifie qu'en pratique, les bornes du théorème 3.9 semblent être atteintes, mais avec des constantes différentes. L'approximation renvoyée est encore exponentielle, mais avec une constante se rapprochant encore plus de 1. La figure 3.7 montre que cette constante expérimentale croit très lentement avec la dimension. Ainsi, entre les dimensions 100 à 300, la constante se situe entre 1.020 et 1.022.

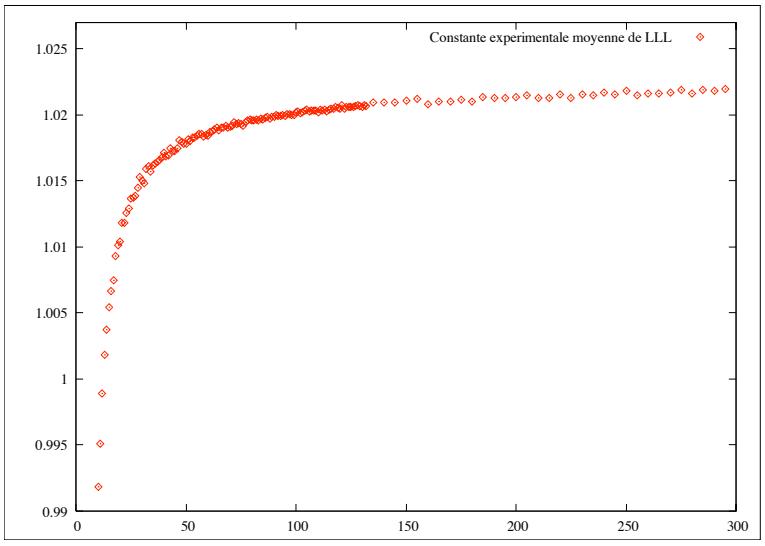


FIG. 3.7 – Exponentielle de la valeur moyenne de $\log(\|\mathbf{b}_1\|/\mathrm{vol}(L)^{1/d})/d$ où $\mathbf{b}_1$ est le premier vecteur renvoyé par l'algorithme LLL (avec facteur optimal) sur un réseau aléatoire, en fonction de la dimension.

De nombreuses questions restent en suspens. Tout d'abord, les expériences de [265] portaient uniquement sur des réseaux aléatoires, et il serait intéressant de voir si la situation évolue avec les réseaux non-aléatoires utilisés en cryptographie : une réponse plus complète est fournie dans [107]. Par ailleurs, il serait utile de caractériser (au moins expérimentalement) les réseaux pour lesquels LLL trouve le plus court vecteur avec grande probabilité. Enfin, peut-on prouver des résultats probabilistes sur le comportement de LLL sur des réseaux aléatoires ?

3.6.4 Applications à la recherche de plus court vecteur et à la réduction HKZ

L'algorithme LLL a tout de suite eu de nombreuses applications. L'une des plus importantes est la recherche de plus court vecteur en petite dimension, notamment parce qu'elle permet de trouver des bases HKZ-réduites. Théoriquement, le premier vecteur renvoyé par LLL n'est pas forcément le plus court vecteur : on sait seulement que sa norme est au pire exponentiellement éloignée du premier minimum. Même si les constantes expérimentales sont très proches de 1, LLL ne trouve pas toujours le plus court vecteur, et ce, même en petite dimension.

On va maintenant présenter les idées principales des deux méthodes connues pour trouver un plus court vecteur d'un réseau. Comme ces algorithmes sont au moins exponentiels, ils ne sont pratiques qu'en petite dimension, au maximum de l'ordre de 60.

Algorithmes d'énumération

La méthode la plus naturelle consiste à énumérer les coordonnées d'un plus court vecteur. Plus précisément, l'algorithme LLL renvoie une base qui est quasi-orthogonale, ce qui impose des restrictions sur les coordonnées des plus courts vecteurs, rendant possible leur énumération en petite dimension. Cette idée naturelle semble être apparue au début des années 80 avec Pohst [285], Kannan [176], et Fincke-Pohst [98]. La même approche fonctionne pour CVP, mais on va se limiter au cas de SVP.

Considérons une base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ d'un réseau L. Il existe un plus court vecteur de L de la forme $\mathbf{x} = x_1\mathbf{b}_1 + \cdots + x_d\mathbf{b}_d$, où les x_i sont entiers et $x_d \geq 0$. On a :

$$\mathbf{x} = \sum_{i=1}^{d} x_i \mathbf{b}_i = \sum_{i=1}^{d} x_i \left(\mathbf{b}_i^\star + \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^\star \right) = \sum_{j=1}^{d} \left(x_j + \sum_{i=j+1}^{d} \mu_{i,j} x_i \right) \mathbf{b}_j^\star.$$

Comme les $\mathbf{b}_j^\star$ sont deux à deux orthogonaux, on obtient :

$$\|\mathbf{x}\|^2 = \sum_{j=1}^{d} \left(x_j + \sum_{i=j+1}^{d} \mu_{i,j} x_i \right)^2 \|\mathbf{b}_j^\star\|^2.$$

Or $\mathbf{x}$ est plus court que tous les $\mathbf{b}_i$, en particulier $\mathbf{b}_1$ (lorsque la base est réduite, c'est généralement le plus court), donc nécessairement :

$$0 \leq x_d < \frac{\|\mathbf{b}_1\|}{\|\mathbf{b}_d^\star\|},$$

et pour tout $j = 1, \ldots, d-1$,

$$\left(x_j + \sum_{i=j+1}^{d} \mu_{i,j} x_i \right)^2 \|\mathbf{b}_j^\star\|^2 < \|\mathbf{b}_1\|^2 - \sum_{k=j+1}^{d} \left(x_k + \sum_{i=k+1}^{d} \mu_{i,k} x_i \right)^2 \|\mathbf{b}_k^\star\|^2.$$

Par conséquent, on peut faire une recherche exhaustive sur $(x_1, \ldots, x_d)$ en commençant par x_d. Et on remarque que lorsque $x_{j+1}, \ldots, x_d$ sont choisis et satisfont les contraintes précédentes, on a au

plus $\lfloor 2\|\mathbf{b}_1\|/\|\mathbf{b}_j^\star\| \rfloor + 1$ valeurs possibles pour l'entier x_j, puisqu'on a alors une contrainte de la forme $|x_j + A| \times \|\mathbf{b}_j^\star\| \leq B$ avec A et B qui sont déjà calculés, et où $B < \|\mathbf{b}_1\|$. La recherche exhaustive parcourt donc un ensemble de cardinal inférieur à

$$\frac{\|\mathbf{b}_1\|}{\|\mathbf{b}_d^\star\|} \prod_{j=2}^{d-1} \left(\left\lfloor \frac{2\|\mathbf{b}_1\|}{\|\mathbf{b}_j^\star\|} \right\rfloor + 1 \right) = 2^{O(d)} \frac{\|\mathbf{b}_1\|^d}{\mathrm{vol}(L)} \tag{3.8}$$

Lorsque la base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ est préalablement LLL-réduite, on voit facilement que (3.8) est majoré par $2^{O(d^2)}$, soit une complexité totale de $2^{O(d^2)}$ opérations polynomiales. Ce temps super-exponentiel peut sembler à première vue exhorbitant, mais comme les constantes expérimentales de l'algorithme LLL sont très proches de 1, il est en fait réaliste en petite dimension, comme en témoigne la figure 3.8.

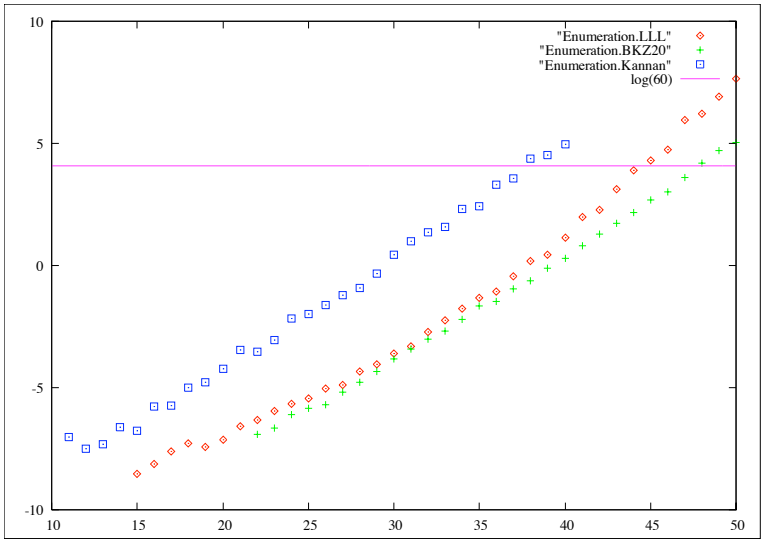


FIG. 3.8 – Temps en secondes (échelle logarithmique) de l'énumération de Schnorr-Euchner, en fonction de la dimension et de la qualité de la base de départ (LLL-réduite ou BKZ-20 réduite). Comparaison avec l'énumération récursive de Kannan. La ligne horizontale indique 1 minute.

On voit ainsi expérimentalement que le logarithme du temps de calcul de l'énumération est convexe, et semble correspondre à une fonction quadratique.

On peut considérablement réduire ce temps en améliorant la qualité de la base : en pratique, on utilise la BKZ-réduction de Schnorr-Euchner [307], mais en théorie, on a intérêt à utiliser l'algorithme de Kannan [176, 178, 138]. Si la base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ est quasi-HKZ-réduite (ce qui signifie que $(\pi_2(\mathbf{b}_2), \ldots, \pi_2(\mathbf{b}_d))$ est HKZ-réduite, et que $\|\mathbf{b}_1\|$ n'est pas beaucoup plus long que $\|\mathbf{b}_2^\star\|$), il est facile de voir que (3.8) devient $2^{O(d \log d)}$. Or Kannan [176] a remarqué qu'en appliquant récursivement l'algorithme d'énumération, on pouvait justement transformer une base LLL-réduite en une base quasi-HKZ-réduite en seulement $2^{O(d \log d)}$ opérations polynomiales. L'algorithme d'énumération récursif de Kannan [176] a donc une complexité totale de $2^{O(d \log d)}$ opérations polynomiales (voir [138] pour une analyse précise de la constante). Cependant, il n'est pas clair que l'algorithme de Kannan ait un intérêt pratique, à cause du coût des appels récursifs, ce que l'on peut voir sur la figure 3.8. Plus précisément, il semble que pour les dimensions d'intérêt pratique, les $2^{O(d \log d)}$ opérations polynomiales de Kannan [176] sont plus coûteuses que les $2^{O(d^2)}$ opérations polynomiales de l'énumération simple à partir d'une base LLL-réduite, car les opérations polynomiales en question sont bien plus coûteuses dans le premier cas.

Enfin, il est à noter que d'un point de vue pratique, il est important d'énumérer les d-uplets $(x_1, \ldots, x_d)$ dans un ordre approprié. Plus précisément, une fois que l'on a isolé la coordonnée x_i

dans un certain intervalle, on a plutôt intérêt à commencer par le milieu de l'intervalle, et non pas par les bords. Cela permet de tester en priorité le candidat qui a le plus de chances d'être court, et donc de mettre plus rapidement à jour la borne supérieure temporaire sur le plus court vecteur, ce qui influence le temps de calcul. C'est cette méthode d'énumération qui est la plus populaire en pratique : elle fut décrite par Schnorr et Euchner [307].

Algorithmes de crible

En 2001, Ajtai, Kumar et Sivakumar [8] ont découvert un algorithme probabiliste, qui est asymptotiquement bien meilleur que l'algorithme déterministe super-exponentiel de Kannan [176]. En effet, il renvoie avec très forte probabilité un plus court vecteur du réseau L en temps $2^{O(d)}$. Mis à part le temps de calcul, cet algorithme est intéressant car son fonctionnement est radicalement différent : il est fondé sur une méthode de crible.

Nous nous contenterons ici de donner l'idée principale, en faisant des simplifications significatives : pour plus de détails, voir un travail joint [270] de l'auteur avec Thomas Vidick, qui présente la variante la plus pratique connue de l'algorithme AKS. Cette variante a une complexité de $(4/3)^d$ opérations polynomiales, mais son résultat n'est pas garanti d'être le plus court vecteur. Cependant, des expérimentations jusqu'en dimension 48 ont montré que cette variante se comportait en pratique comme théoriquement prévu.

Les algorithmes décrits précédemment énumèrent tous les vecteurs de $L \cap S$ pour un choix de S bien adapté à l'énumération, et renvoient un vecteur de norme minimale $\lambda_1(L)$. Avec un choix approprié pour S [176, 141], on a $|L \cap S| = 2^{O(d \log d)}$, d'où une complexité de $2^{O(d \log d)}$ opérations polynomiales.

L'idée principale des algorithmes de crible est de faire une énumération probabiliste d'un ensemble plus petit. Intuitivement, l'algorithme va échantillonner $N = 2^{\Omega(d)}$ vecteurs dans $L \cap S$ pour un ensemble S bien choisi tel que $|L \cap S| = 2^{O(d)}$. On va prendre pour S une boule centrée en l'origine et de rayon $O(\lambda_1(L))$.

Si l'échantillonnage était tel que chaque point de $L \cap S$ ait une probabilité d'être renvoyé d'ordre $|L \cap S|^{-1}$, et si $N \gg |L \cap S|$, alors l'un des N échantillons serait un plus court vecteur avec probabilité proche de 1. Malheureusement, il n'est pas certain que cette propriété soit satisfaite par l'échantillonnage AKS. Par contre, on montre qu'il existe $\mathbf{w} \in L \cap S$ tel que $\mathbf{w}$ et $\mathbf{w} + \mathbf{s}$, où $\mathbf{s}$ est un plus court vecteur, aient tous les deux une probabilité non nulle d'être renvoyé. Ainsi, en calculant la plus petite différence entre les N vecteurs échantillonnés dans $L \cap S$ où $N \gg |L \cap S|$, on obtient un plus court vecteur de L avec probabilité proche de 1.

Cependant, échantillonner directement à l'intérieur d'une boule centrée en l'origine et de rayon $O(\lambda_1(L))$ est difficile. Mais en partant d'une base LLL-réduite, il est facile d'échantillonner avec un rayon $2^{O(d)}\lambda_1(L)$. Pour réduire le facteur $2^{O(d)}$ en $O(1)$, on fait appel à un crible, qui est la phase la plus coûteuse de l'algorithme.

Le crible raccourcit de façon itérative les vecteurs de S, au moins d'un facteur géométrique γ (tel que $0 < \gamma < 1$) à chaque itération : ainsi, un nombre linéaire d'itérations du crible suffit à réduire le facteur $2^{O(d)}$ en $O(1)$. A chaque itération, chaque vecteur renvoyé par le crible est une soustraction de deux vecteurs d'entrée. En d'autres termes, le crible va sélectionner un sous-ensemble C de l'ensemble de départ S, et l'ensemble de sortie sera obtenu en soustrayant un vecteur de C à chaque vecteur de $S \setminus C$. Par des arguments de volume, on peut choisir un ensemble C qui ne soit jamais trop gros, de façon à ce que le nombre d'échantillons ne diminue pas trop. Intuitivement, on utilise le fait que pour tout $0 < \gamma < 1$, une boule de rayon R peut être recouverte par un nombre au plus exponentiel de boules de rayon γR.

Nous venons de décrire les principes de l'algorithme AKS [8], mais l'algorithme prouvé est un peu plus complexe, et son analyse est non triviale.

La réduction HKZ

Il est facile de voir que les algorithmes de plus court vecteur permettent de trouver des bases HKZ-réduites, et ce, avec les mêmes complexités asymptotiques. Par exemple, on peut procéder de la façon suivante :

- On détermine d'abord un plus court vecteur $\mathbf{b}_1$ du réseau L.
- On complète $\mathbf{b}_1$ en une base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ de L de façon à connaître une base du réseau projeté $\pi_1(L)$, grâce aux coefficients de Gram-Schmidt.
- On détermine un plus court vecteur $\mathbf{b}'_2$ du réseau projeté $\pi_1(L)$.
- On relève $\mathbf{b}'_2$ en un vecteur $\mathbf{b}_2$ de L, en rajoutant un multiple approprié de $\mathbf{b}_1$ pour que $(\mathbf{b}_1, \mathbf{b}_2)$ soit faiblement réduit.
- On complète $(\mathbf{b}_1, \mathbf{b}_2)$ en une base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ de L de façon à connaître une base du réseau projeté $\pi_2(L)$. Et ainsi de suite.

3.6.5 Généralisations par blocs de LLL

Dans cette section, on va présenter des algorithmes qui permettent d'améliorer les facteurs d'approximation de LLL, mais en augmentant bien entendu le temps de calcul. Ils correspondent à des compromis entre la réduction LLL et la réduction HKZ.

Algorithmes prouvés

On a vu l'algorithme LLL comme une version algorithmique de l'inégalité d'Hermite (3.6). Il résout en temps polynomial HSVP à un facteur $(4/3 + \varepsilon)^{(d-1)/4}$ et approche SVP à un facteur $(4/3 + \varepsilon)^{(d-1)/2}$. Or l'inégalité d'Hermite (3.6) est un cas particulier de l'inégalité dite de Mordell, qui repose sur la dualité :

Théorème 3.11 (Inégalité de Mordell) *Pour tout* $2 \leq k \leq d$,

$$\gamma_d \leq \gamma_k^{(d-1)/(k-1)}. \tag{3.9}$$

Preuve. On remarque qu'il suffit de démontrer l'inégalité pour $k = d - 1$, ce qui peut se faire par dualité (voir [220, Théorème 3.1]). Soit L un réseau de dimension d. Soit $\mathbf{x}$ un plus court vecteur du réseau dual L^*, et soit H l'hyperplan $\mathbf{x}^\perp$. Notons M le réseau $L \cap H$ de dimension $d - 1$. On a : $\mathrm{vol}(M) = \mathrm{vol}(L)\|\mathbf{x}\|$ et $\|\mathbf{x}\| \leq \sqrt{\gamma_d}\mathrm{vol}(L^*)^{1/d} = \sqrt{\gamma_d}\mathrm{vol}(L)^{-1/d}$, donc :

$$\mathrm{vol}(M) \leq \sqrt{\gamma_d}\mathrm{vol}(L)^{1-1/d}.$$

En particulier :

$$\lambda_1(M) \leq \sqrt{\gamma_{d-1}} \left(\sqrt{\gamma_d}\mathrm{vol}(L)^{1-1/d}\right)^{1/(d-1)} = \sqrt{\gamma_{d-1}}\sqrt{\gamma_d}^{1/(d-1)}\mathrm{vol}(L)^{1/d}.$$

Or on a aussi $\lambda_1(L) \leq \lambda_1(M)$. On en déduit par définition de γ_d :

$$\sqrt{\gamma_d} \leq \sqrt{\gamma_{d-1}}\sqrt{\gamma_d}^{1/(d-1)}.$$

La démonstration est terminée puisque l'on peut réécrire cette dernière inégalité en :

$$\gamma_d \leq \gamma_{d-1}^{(d-1)/(d-2)}.$$

□

Cette inégalité (3.9) est atteinte pour $(k, d) = (3, 4)$ et $(7, 8)$. L'inégalité de Mordell suggère l'existence d'un algorithme qui utiliserait un algorithme de plus court vecteur en dimension $\leq k$ pour résoudre

HSVP à un facteur $\sqrt{\gamma_k}^{(d-1)/(k-1)}$, et approcher SVP à un facteur $\gamma_k^{(d-1)/(k-1)}$ grâce au résultat de Lovász [214]).

Nous verrons que cela est fondamentalement vrai : l'auteur, en collaboration avec Nicolas Gama, Nick Howgrave-Graham et Henrik Koy, a présenté [103] (voir l'appendice E) une variante de l'algorithme de Schnorr [302] qui, en appelant un nombre polynomial de fois un oracle de plus court vecteur en dimension $\leq k+1$, approche à des facteurs d'essentiellement $O(\gamma_k^{d/(2k)})$ pour HSVP et $O(\gamma_k^{d/k})$ pour SVP. Ainsi, cet algorithme (appelé algorithme de réduction par transférence) est en quelque sorte une version algorithmique approchée de l'inégalité de Mordell. En choisissant comme taille de bloc $k = \log d / \log\log d$ et en utilisant l'algorithme de crible [8] pour trouver des plus courts vecteurs en dimension $\leq k$, on obtient ainsi (avec [302] ou [103]) un algorithme polynomial probabiliste qui approche SVP et HSVP à des facteurs légèrement sous-exponentiels : $2^{O(d \log\log d / \log d)}$. C'est le meilleur facteur asymptotique connu pour approcher SVP et HSVP en temps polynomial.

Nous allons seulement décrire les principes de l'algorithme de transférence [103], et de l'algorithme de Schnorr [302], dont la philosophie est essentiellement celle de LLL. Il faut souligner que la façon dont on analyse la complexité et les facteurs d'approximation obtenus par ces algorithmes par bloc est identique à LLL. L'algorithme LLL considère les vecteurs deux par deux. A chaque étape, le réseau de dimension deux $L_i = [\pi_i(\mathbf{b}_i), \pi_i(\mathbf{b}_{i+1})]$, est partiellement réduit (grâce à un échange) afin de diminuer $\|\mathbf{b}_i^\star\|$ d'au moins un facteur géométrique. Quand tous les réseaux L_i sont quasi réduits, chaque quotient $\|\mathbf{b}_i^\star\|/\|\mathbf{b}_{i+1}^\star\|$ est approximativement inférieur à $\gamma_2 = \sqrt{\frac{4}{3}}$, ce qui est parfois appelé condition de Siegel [58].

Schnorr [302] a proposé une généralisation par bloc de LLL, qui est paramétrée par un entier k tel que $1 \leq 2k \leq d$. Dans cet algorithme, les vecteurs $\mathbf{b}_i^\star$ sont remplacés par de petits blocs k-dimensionnels $S_i = [\pi_{ik-k+1}(\mathbf{b}_{ik-k+1}), \ldots, \pi_{ik-k+1}(\mathbf{b}_{ik})]$ où $1 \leq i \leq \frac{d}{k}$. L'analogue du réseau de dimension deux L_i dans LLL est le gros bloc $2k$-dimensionnel $L_i = [\pi_{ik-k+1}(\mathbf{b}_{ik-k+1}), \ldots, \pi_{ik-k+1}(\mathbf{b}_{ik+k})]$ où $1 \leq i \leq \frac{d}{k} - 1$. Il y a des relations entre les petits blocs $S_1, \ldots, S_{d/k}$ et les gros blocs $L_1, \ldots, L_{d/k-1}$: le bloc S_i consiste en les k premiers vecteurs de L_i, tandis que S_{i+1} est la projection des k derniers vecteurs de L_i sur $\mathrm{span}(S_i)^\perp$. Ainsi, $\mathrm{vol}(L_i) = \mathrm{vol}(S_i) \times \mathrm{vol}(S_{i+1})$.

Comme LLL, l'algorithme de Schnorr réduit le gros bloc L_i à chaque itération, si cela diminue $\mathrm{vol}(S_i)$ d'un facteur géométrique $1/(1+\varepsilon)$. Dans ce cas, $\mathrm{vol}(S_i)/\mathrm{vol}(S_{i+1}) = \mathrm{vol}(S_i)^2/\mathrm{vol}(L_i)$ diminuerait d'un facteur $1/(1+\varepsilon)^2$. Mais quel type de réduction faudrait-il utiliser sur le gros bloc L_i ? Idéalement, on aimerait résoudre le problème algorithmique suivant, appelé problème de demi-volume dans [103] : étant donné un réseau de dimension $2k$, trouver une base qui minimise le volume de ses k premiers vecteurs. Lorsque $k = 1$, on retombe sur SVP, mais les choses se compliquent lorsque k augmente. Schnorr [302] a remarqué que la HKZ-réduction fournissait toujours une solution approchée à ce problème. Plus précisément, il a montré l'existence de la constante suivante :

$$\beta_k = \max_{\substack{L \text{ réseau de dimension } 2k \\ H \text{ base HKZ-réduite de L}}} \left(\frac{\|\mathbf{h}_1^\star\| \times \cdots \times \|\mathbf{h}_k^\star\|}{\|\mathbf{h}_{k+1}^\star\| \times \cdots \times \|\mathbf{h}_{2k}^\star\|} \right)^{\frac{2}{k}}$$

que l'on peut réécrire plus géométriquement comme

$$\beta_k = \max_{\substack{L \text{ réseau de dimension } 2k \\ H \text{ base HKZ-réduite de L}}} \left(\frac{\mathrm{vol}(\mathbf{h}_1, \ldots, \mathbf{h}_k)}{\mathrm{vol}(\pi_{k+1}(\mathbf{h}_{k+1}), \ldots, \pi_{k+1}(\mathbf{h}_{2k}))} \right)^{\frac{2}{k}}$$

L'algorithme de Schnorr applique donc une HKZ-réduction au réseau L_i de dimension $2k$, ce qui coûte essentiellement une recherche de plus court vecteur en dimension $2k$. Et on en déduit à l'instar du théorème 3.9 que l'algorithme de Schnorr approche SVP à un facteur $\beta_k^{d/k}$, et HSVP à un facteur

$\beta_k^{d/(2k)}$. Pour pouvoir borner concrètement les facteurs d'approximation, il nous reste à majorer la constante β_k. Schnorr [302] a d'abord démontré que $\beta_k \leq 4k^2$. L'auteur en collaboration avec Nicolas Gama *et al.* [103] a montré que la majoration de Schnorr [302] sur β_k n'était pas optimale : on a en fait $\beta_k \lessapprox 0.38 \times k^{2 \ln 2} \approx 0.38 \times k^{1.39}$, donc les facteurs d'approximation donnés dans [302] peuvent être élevés à la puissance $\ln 2 \approx 0.69 < 1$. En outre, on peut même vérifier numériquement que $\beta_k \leq k^{1.1}$ pour $k \leq 100$. Cela étant, comme l'algorithme fait appel à des recherches de plus court vecteur en dimension $2k$, on se rend compte que les facteurs d'approximations ne sont pas tout à fait conformes à ce que l'inégalité de Mordell laisse espérer. En effet, l'inégalité de Mordell suggère qu'on pourrait trouver un vecteur non nul du réseau dont la norme divisée par $\mathrm{vol}(L)^{1/d}$ soit inférieure à :

$$\sqrt{\gamma_{2k}}^{(d-1)/(2k)} \approx k^{d/(2k)},$$

contre $\approx \beta_k^{d/(2k)} \leq k^{(2 \ln 2)d/(2k)} \approx k^{0.69d/k}$ pour l'algorithme de Schnorr.

A l'inverse, Ajtai a prouvé [7] qu'il existait $\varepsilon > 0$ tel que $\beta_k \geq k^{\varepsilon}$, mais aucune valeur de ε n'était connue. L'article [103] montre une borne inférieure effective : $\beta_k \geq k/12$, ce qui assure que la borne supérieure précédente ne peut pas être considérablement améliorée. Cette borne inférieure repose sur un lien entre β_k et une constante introduite par Rankin [292] il y a plus de cinquante ans pour généraliser la constante d'Hermite. En fait, le problème de demi-volume est à la constante de Rankin ce que SVP est à la constante d'Hermite. Donc la solution approchée au problème du demi-volume fournie par la HKZ-réduction ne peut pas être meilleure que la solution optimale.

En outre, ce rapprochement suggère de remplacer la HKZ-réduction dans l'algorithme de Schnorr par une meilleure solution au problème du demi-volume, afin d'obtenir de meilleurs facteurs d'approximation à effort constant. C'est justement ce que fait [103], qui remplace la HKZ-réduction du gros bloc L_i par une réduction moins coûteuse dite de transférence, car celle-ci fait des allers-retours entre le primal et le dual. L'algorithme obtenu garantit des facteurs d'approximations d'essentiellement $\gamma_k^{d/(2k)}$ pour HSVP et $\gamma_k^{d/k}$ pour SVP, en utilisant seulement un nombre polynomial de recherches de plus court vecteur en dimension $\leq k+1$ (et non $2k$ pour l'algorithme de Schnorr).

On obtient ainsi des facteurs d'approximation qui sont équivalents (asymptotiquement) à ceux de l'inégalité de Mordell, tout en étant légèrement moins bons à k fixé. Récemment, en collaboration avec Nicolas Gama, l'auteur a développé un nouvel algorithme polynomial de réduction par bloc [106], qui est à l'inégalité de Mordell ce que LLL est à l'inégalité d'Hermite. Plus précisément, cet algorithme utilise un oracle de plus court vecteur en dimension $\leq k$, et le premier vecteur de la base réduite renvoyée vérifie, lorsque $d \equiv 0 \pmod k$:

$$\|\mathbf{b}_1\| \leq \sqrt{(1+\varepsilon)\gamma_k}^{(d-1)/(k-1)} \mathrm{vol}(L)^{1/d},$$

et le nombre d'appels à l'oracle est polynomial en $1/\varepsilon$ et la taille du réseau, tout comme le reste du temps de calcul de l'algorithme. En outre :

$$\|\mathbf{b}_1\| \leq ((1+\varepsilon)\gamma_k)^{(d-k)/(k-1)} \lambda_1(L).$$

Algorithmes heuristiques

Curieusement, lorsqu'on souhaite améliorer la qualité d'une base LLL-réduite, on ne fait pas appel en pratique aux algorithmes prouvés précédents, qui ne sont d'ailleurs pas implémentés dans les bibliothèques usuelles. A la place, on utilise l'algorithme heuristique BKZ [307, 308], qui fournit de très bons résultats expérimentalement, mais dont on ne sait même pas s'il est polynomial. Comme pour les algorithmes prouvés par bloc, on utilise un paramètre β, qui détermine une taille de bloc. Le fonctionnement est essentiellement le suivant :

– On applique d'abord une réduction LLL pour réduire la taille des vecteurs.

- On parcourt successivement tous les blocs de la forme $(\mathbf{b}_i, \ldots, \mathbf{b}_{\min(i+\beta-1,d)})$, et pour chaque bloc :
  - On détermine un plus court vecteur $\mathbf{u}$ du réseau projeté.
  - Si le vecteur de Gram-Schmidt associé au premier vecteur du bloc n'est pas aussi court (ou à un facteur δ près), on « remplace »[5] le premier vecteur du bloc par un relèvement de $\mathbf{u}$ mais sans pour autant modifier le réseau formé par le bloc.
- On réitère l'étape précédente tant que toutes les conditions n'ont pas été satisfaites.

La recherche de plus court vecteur s'effectue par énumération, en utilisant des calculs en virgule flottante. Elle est en général irréaliste pour des blocs de taille supérieure à 25, à la fois à cause du grand nombre d'appels à la routine de plus court vecteur, mais aussi du coût de cette routine. Si l'on veut utiliser de plus grands blocs, il faut élaguer l'arbre de recherche en appliquant des heuristiques (voir [307, 308]). La meilleure heuristique à l'heure actuelle est celle de Schnorr-Hörner [308]. Elle s'appuie sur une heuristique due à Gauss pour estimer le nombre de points d'un réseau dans une boule. Lorsque l'estimation est inférieure à un certain seuil (associé à un facteur d'élagage), on coupe la branche correspondante de la recherche en profondeur. Il y a naturellement un compromis à faire entre la réduction du temps de calcul et la qualité de la base obtenue. L'algorithme de réduction BKZ utilisant l'heuristique de Schnorr-Hörner est en pratique le meilleur algorithme de réduction de réseaux connu à ce jour. Mais on ne l'utilise que lorsque l'algorithme LLL et les algorithmes BKZ traditionnels se sont révélés insuffisants.

Bien qu'on ne sache pas borner de façon convenable la complexité de l'algorithme BKZ, on peut facilement voir que les bases renvoyées fournissent de très bons facteurs d'approximation. En effet, si l'on néglige le facteur d'erreur δ, une base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ renvoyée par l'algorithme BKZ est *HKZ-réduite par bloc de taille β* en ce sens où elle est faiblement réduite et pour tout $i = 1, \ldots, d$,

$$\|\mathbf{b}_i^\star\| = \lambda_1(\pi_i(L(\mathbf{b}_i, \ldots, \mathbf{b}_{\min(i+\beta-1,d)}))).$$

Pour $\beta = d$, on retombe sur la réduction HKZ. Pour $\beta = 2$, la définition est équivalente à celle de la réduction LLL pour un facteur $\delta = 1$.

Schnorr a démontré dans [305] le résultat suivant :

Théorème 3.12 *Toute base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ BKZ-réduite par bloc de taille β d'un réseau L vérifie pour tout $i = 1, \ldots, d$:*

$$\frac{4}{i+3}\gamma_\beta^{-2\frac{i-1}{\beta-1}} \leq \left(\frac{\|\mathbf{b}_i\|}{\lambda_i(L)}\right)^2 \leq \gamma_\beta^{2\frac{d-i}{\beta-1}}\frac{i+3}{4}.$$

On peut en donner l'idée principale. On sait que pour tout i, $\mathbf{b}_i = \mathbf{b}_i^\star + \sum_{j=1}^{i-1}\mu_{i,j}\mathbf{b}_j^\star$, d'où

$$\pi_i(\mathbf{b}_k) = \mathbf{b}_k^\star + \sum_{j=i}^{k-1}\mu_{i,j}\mathbf{b}_j^\star.$$

On en déduit, pour tout $i \leq d - \beta + 1$:

$$\det(\pi_i(L(\mathbf{b}_i, \ldots, \mathbf{b}_{i+\beta-1}))) = \prod_{k=i}^{i+\beta-1}\|\mathbf{b}_k^\star\|.$$

La propriété BKZ implique donc par définition de la constante d'Hermite :

$$\|\mathbf{b}_i^\star\| \leq \sqrt{\gamma_\beta}\left(\prod_{k=i}^{\min(i+\beta-1,d)}\|\mathbf{b}_k^\star\|\right)^{1/\beta}.$$

[5] En fait, on rajoute un relèvement de $\mathbf{u}$ au bloc, et on réduit ce bloc par une version de LLL acceptant les familles génératrices : la propriété BKZ est conservée, et le réseau formé par le bloc aussi.

Dans la réduction LLL, on avait une inégalité géométrique entre $\|\mathbf{b}_i^\star\|$ et $\|\mathbf{b}_{i+1}^\star\|$. Ici, on obtient une inégalité entre $\|\mathbf{b}_i^\star\|$ et la moyenne géométrique de $\|\mathbf{b}_i^\star\|, \ldots, \|\mathbf{b}_{i+\beta-1}^\star\|$. La décroissance des normes est donc plus faible. Il est alors possible d'établir une meilleure majoration de $\|\mathbf{b}_1\| = \|\mathbf{b}_1^\star\|$ vis-à-vis de $\min_{j=1}^{d} \|\mathbf{b}_j^\star\|$, et ainsi de se rapprocher du premier minimum. Le théorème 3.12 est un raffinement de ce raisonnement.

Quand on regarde la borne supérieure sur $\|\mathbf{b}_1\|/\lambda_1(L)$ fournie par le théorème 3.12, on se rend compte qu'elle correspond exactement au carré du facteur d'Hermite associé à l'inégalité de Mordell : en effet, cette borne vaut $\gamma_\beta^{(d-1)/(\beta-1)}$.

D'un point de vue pratique, les bases renvoyées par BKZ ressemblent beaucoup à des bases LLL-réduites : elles semblent vérifier le théorème 3.9 mais avec de bien meilleures constantes, comme en témoignent les figures 3.9 et 3.10. Cependant, à taille de bloc fixée, les facteurs d'approximation semblent asymptotiquement rester exponentiels. Et le coût de l'algorithme BKZ devient très élevé lorsqu'on dépasse la taille de bloc 20 : voir la figure 3.11.

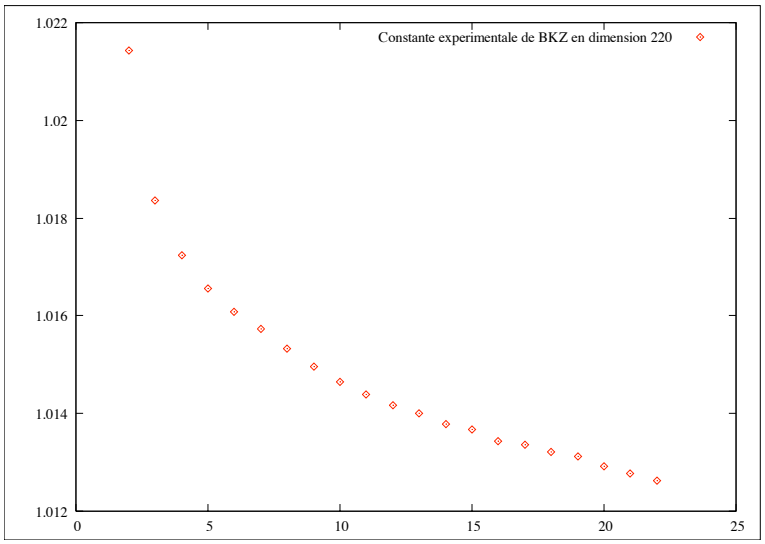


FIG. 3.9 – Exponentielle de la valeur moyenne de $\log(\|\mathbf{b}_1\|/\mathrm{vol}(L)^{1/d})/d$ où $\mathbf{b}_1$ est le premier vecteur renvoyé par l'algorithme BKZ sur un réseau aléatoire de dimension 220, en fonction de la taille de bloc.

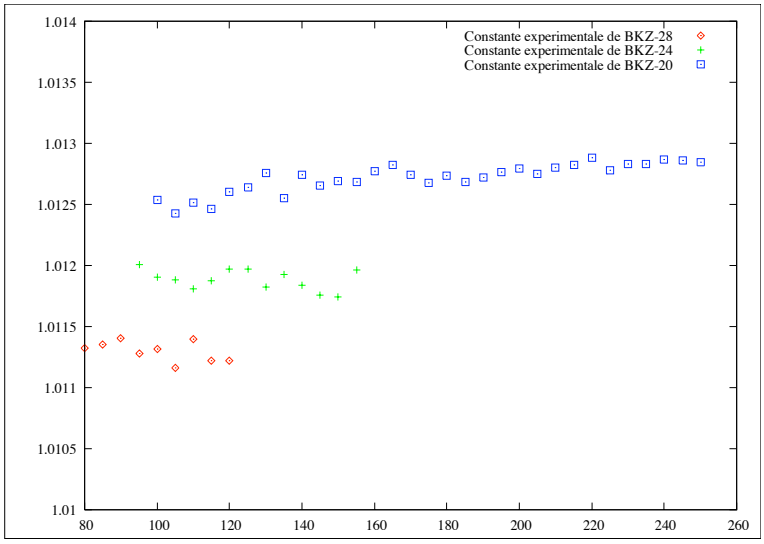


FIG. 3.10 – Exponentielle de la valeur moyenne de $\log(\|\mathbf{b}_1\|/\mathrm{vol}(L)^{1/d})/d$ où $\mathbf{b}_1$ est le premier vecteur renvoyé par l'algorithme BKZ sur un réseau aléatoire, en fonction de la dimension, pour des tailles de bloc de 20, 24 et 28.

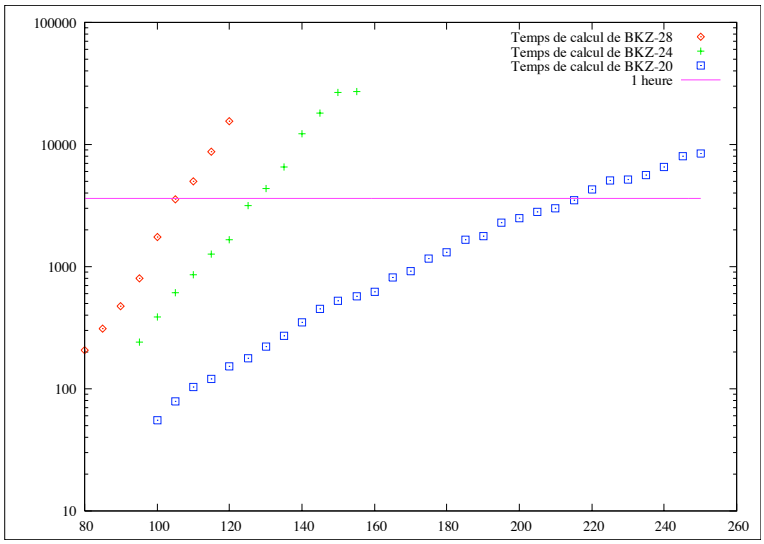


FIG. 3.11 – Temps en secondes (échelle logarithmique) de l'algorithme BKZ sur un réseau aléatoire, en fonction de la dimension, pour des tailles de bloc de 20, 24 et 28.

3.6.6 Les algorithmes de Babai

Il est facile de résoudre CVP dans le cas du réseau hypercubique $\mathbb{Z}^n$. En 1986, Babai [17] a montré plus généralement que lorsqu'on connaissait des bases raisonnablement orthogonales (par exemple, une base LLL-réduite), on pouvait approcher CVP de façon satisfaisante. Il a en fait présenté deux algorithmes assez intuitifs (voir les algorithmes 8 et 9), dont on peut garantir la qualité du résultat lorsque les bases fournies en entrée sont LLL-réduites.

Algorithme 8 L'algorithme d'arrondi de Babai.

Entrée : une base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ d'un réseau L de $\mathbb{Z}^n$, et un point $\mathbf{x}$ de $\mathbb{Z}^n$.
Sortie : Un point $\mathbf{y}$ du réseau L, approchant $\mathbf{x}$.

1: Par algèbre linéaire, décomposer la projection orthogonale de $\mathbf{x}$ sur l'espace engendré par $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ en une combinaison linéaire de $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ à coefficients réels.
2: En déduire $\mathbf{y} \in L$ tel que la projection orthogonale de $\mathbf{x}-\mathbf{y}$ sur l'espace engendré par $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ soit une combinaison linéaire de $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ à coefficients $\leq 1/2$ en valeurs absolues.

Algorithme 9 L'algorithme « hyperplan le plus proche » de Babai.

Entrée : une base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ d'un réseau L de $\mathbb{Z}^n$, et un point $\mathbf{x}$ de $\mathbb{Z}^n$.
Sortie : Un point $\mathbf{y}$ du réseau L, approchant $\mathbf{x}$.

1: A l'instar de l'algorithme de réduction faible de la figure 4, trouver $\mathbf{y} \in L$ tel que la projection orthogonale de $\mathbf{x} - \mathbf{y}$ sur l'espace engendré par $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ soit une combinaison linéaire de $(\mathbf{b}_1^\star, \ldots, \mathbf{b}_d^\star)$ à coefficients $\leq 1/2$ en valeurs absolues.

Les algorithmes des figures 8 et 9 sont polynomiaux. Et on peut montrer que lorsqu'on fournit des bases LLL-réduites avec un facteur $\frac{3}{4}$ en entrée, ces algorithmes renvoient un point du réseau approchant CVP à des facteurs respectifs de $1 + 2d(9/2)^{d/2}$ et $2^{d/2}$, ce dernier facteur devenant $(4/3 + \varepsilon)^{d/2}$ si l'on utilise LLL avec facteur proche de 1.

L'algorithme de la figure 9 est en fait déjà présent dans la version incrémentale de l'algorithme LLL : en effet, lorsque les k premiers vecteurs de la base sont déjà réduits, et que l'on réduit faiblement le $(k+1)$-ième vecteur de la base par rapport aux k premiers, on soustrait justement à ce $(k+1)$-ième vecteur son approximation par l'algorithme de la figure 9. Remarquons l'analogie avec l'algorithme glouton, qui lui soustrait un plus proche vecteur, et non une approximation du plus proche vecteur.

Schnorr [305] a montré que l'on pouvait améliorer les bornes fournies par Babai en utilisant des bases BKZ-réduites, avec l'algorithme « hyperplan le plus proche ». Plus précisément, toute base BKZ-réduite de taille β permet de fournir en temps polynomial un point du réseau approchant CVP à un facteur $\sqrt{d}\gamma_\beta^{(d-1)/(\beta-1)}$.

3.6.7 Faits expérimentaux

A ceux qui sont intéressés par faire des expériences, on peut conseiller la bibliothèque NTL [319] qui contient des implémentations efficaces des principaux algorithmes de réduction de réseau. En petite dimension, on peut aussi utiliser GP/PARI [22].

Dans cette section, nous résumons ce que l'on peut espérer en pratique concernant la solvabilité des problèmes de réseau : plus d'informations sont fournis dans l'article [107]. Il faut souligner qu'il n'y a malheureusement pas de recette de cuisine pour prédire à l'avance ce qu'il est possible de résoudre en pratique. En petite dimension, disons ≤ 70, les problèmes de réseau les plus importants deviennent faciles : par exemple, SVP et CVP peuvent se résoudre par énumération (voir la figure 3.8). Mais lorsqu'on dépasse ces dimensions, il est difficile de prédire à l'avance les résultats expérimentaux. Plusieurs facteurs semblent influencer le résultat : la dimension du réseau, la base d'entrée, la structure du réseau, et dans le cas de CVP, la distance du vecteur cible au réseau. Ce qui est toujours vrai est que l'on peut approcher SVP et CVP à des facteurs exponentiels où la constante est très proche de 1 (voir les figures 3.9 et 3.10), mais ces facteurs exponentiels peuvent s'avérer insuffisants dans un contexte cryptanalytique, selon l'application. Par exemple, l'algorithme BKZ-20 renvoie en un temps raisonnable un vecteur de norme de l'ordre au plus $1.013^d \mathrm{vol}(L)^{1/d}$ en grande dimension. Si de meilleurs facteurs d'approximation sont requis, on doit faire des expérimentations pour voir si les algorithmes existants sont suffisants. Si le réseau et la base d'entrée n'ont pas de propriétés

exceptionnelles, il n'y a pas de raison de croire que SVP puisse être résolu en très grande dimension (disons ≥ 300), bien qu'on puisse toujours essayer. De même, si la distance du vecteur cible au réseau n'est pas exceptionnellement faible, il n'y a aussi aucune raison de croire que CVP puisse être résolu en très grande dimension (disons ≥ 300).

Un exemple de structure de réseau inhabituelle est lorsque l'on connaît un vecteur non nul du réseau qui soit largement plus court que la borne d'Hermite : on compare la norme du vecteur avec $\sqrt{d}\text{vol}(L)^{1/d}$. Dans ce cas, il faut essayer les algorithmes existants : par exemple, l'auteur [250] a ainsi pu résoudre plusieurs instances de SVP/CVP pour des dimensions allant jusqu'à 350.

3.7 Conclusion

Nous avons vu les notions et les résultats essentiels de la géométrie des nombres algorithmique, en mettant en évidence une connection entre le problème mathématique de borner la constante d'Hermite et le problème algorithmique d'approcher le plus court vecteur dans un réseau. Nous terminons ce chapitre en décrivant les principaux problèmes ouverts dans ce domaine.

Tout d'abord, on peut remarquer qu'il n'existe finalement que très peu d'algorithmes de réduction de réseau. Il serait donc intéressant de trouver de nouveaux algorithmes, en particulier des algorithmes s'appuyant sur des principes différents. On a vu que l'algorithme LLL était une version algorithmique de l'inégalité d'Hermite, tandis que ses généralisations par bloc [302, 103] pouvaient être vues comme des versions algorithmiques approchées de l'inégalité de Mordell : on peut naturellement se demander si d'autres bornes supérieures sur la constante d'Hermite ont des pendants algorithmiques. A l'inverse, les rares algorithmes exacts pour SVP (méthodes par énumération ou par crible) n'ont pour l'instant aucun rapport avec des inégalités sur la constante d'Hermite.

D'un point de vue théorique, on a vu la réduction de réseau comme une généralisation géométrique du problème du pgcd. Cette interprétation n'est pas du tout artificielle, puisque l'algorithme d'Euclide possède un véritable équivalent en termes de réduction de réseau : l'algorithme L^2 [264] trouve une base quasi Hermite-réduite en temps polynomial, quadratique en fonction de la taille des coefficients, sans arithmétique rapide. Or justement, on sait qu'il existe des algorithmes quasi-linéaires pour le calcul du pgcd, en utilisant de l'arithmétique rapide. Mais à l'heure actuelle, on ne connaît pas d'algorithme de réduction de réseau qui trouve une base quasi Hermite-réduite en temps polynomial à dimension variable, et qui soit quasi-linéaire en fonction de la taille des coefficients. Il n'est par contre pas difficile de montrer qu'à dimension fixée, on peut trouver une base quasi Hermite-réduite en temps quasi-linéaire [90].

Peut-être le principal problème ouvert en réduction de réseau concerne l'existence d'un algorithme polynomial ou sous-exponentiel pouvant approcher HSVP ou SVP à un facteur polynomial.

Un autre problème important concerne l'optimisation des algorithmes de réduction de réseau pour des classes particulières de réseau. Par exemple, l'auteur, en collaboration avec Nicolas Gama et Nick Howgrave-Graham, a présenté des versions symplectiques [104] (voir l'appendice D) de l'algorithme LLL, afin de tirer partir de la structure particulière des réseaux utilisés par le cryptosystème NTRU. On ne sait pas encore si des gains substantiels peuvent être obtenus pour des algorithmes de réduction supérieurs à LLL.

Par ailleurs, il y a encore un écart considérable en réduction de réseau entre la théorie et la pratique. Ainsi, il est curieux de constater que les meilleurs algorithmes théoriques [302, 103] d'approximation pour SVP ne sont pas utilisés en pratique : il semble qu'ils soient moins performants (à puissance de calcul égale) que l'algorithmique heuristique BKZ [307]. De même, les meilleurs algorithmes théoriques pour résoudre exactement SVP, à savoir l'algorithme de Kannan [176] et l'algorithme AKS [8], ne sont eux non plus pas utilisés en pratique : on leur préfère l'énumération simple de Schnorr-Euchner [307] avec LLL ou BKZ comme précalculs. Cela semble indiquer que les analyses existantes des algorithmes ne sont pas très satisfaisantes. En outre, l'analyse sur la qualité du facteur

d'approximation de BKZ n'est pas non plus satisfaisante : en effet, la borne théorique indiquée par le théorème 3.12 ne réflète pas bien les valeurs expérimentales des figures 3.9 et 3.10, comme en témoigne la figure 3.12. On ne dispose pas aujourd'hui d'une bonne modélisation du comportement des algorithmes LLL et BKZ, et encore moins de preuves sur le comportement moyen de ces algorithmes.

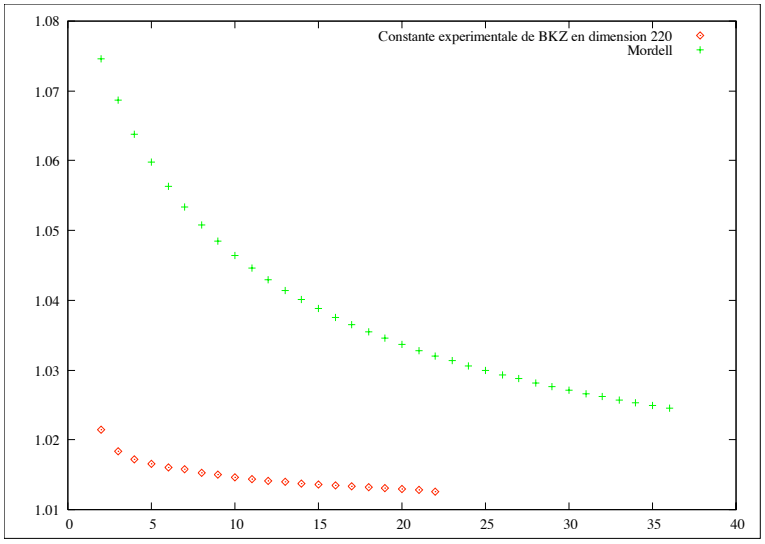


FIG. 3.12 – Comparaison de la courbe de la figure 3.9 avec la courbe correspondant à l'inégalité de Mordell. La courbe Mordell est générée à partir des valeurs exactes connues de $\gamma_\beta^{1/(\beta-1)}$, et des bornes supérieures de [66].

Deuxième partie

Curriculum vitæ et publications

Chapitre 1

Curriculum vitæ

Sommaire

1.1 Formation **83**

1.2 Distinctions **84**

1.3 Valorisation de la recherche **84**

1.3.1 Activités éditoriales et organisation de conférences 84

1.3.2 Conférences invitées 84

1.4 Enseignement **85**

1.5 Encadrement de la recherche **85**

1.6 Projets de recherche **86**

1.1 Formation

1996-1999 Thèse de doctorat en informatique, Université Paris 7
Recherches effectuées à l'École normale supérieure.
Directeur : Jacques Stern.
Titre : La géométrie des nombres en cryptologie.
Rapporteurs : Arjen K. Lenstra (Citibank) et Brigitte Vallée (Univ. Caen).

1993-1997 Élève-fonctionnaire-stagiaire de l'École Normale Supérieure de Lyon
Agrégation de Mathématiques (1997).
DEA Algorithmique (1996).
Maîtrise de mathématiques discrètes (1995).
Licence d'informatique (1994).

1.2 Distinctions

- *Best paper award* au congrès EUROCRYPT 2006, pour un article en collaboration avec Oded Regev (Univ. Tel-Aviv).
- Prix Cor Baayen 2001 du *European Research Consortium for Informatics and Mathematics* (ERCIM).
- Prix de thèse 2000 de *l'Association française d'informatique théorique* (AFIT).
- Troisième prix d'Alembert des *des lycéens* décerné en juin 2000 par la *Société mathématique de France* (SMF) pour la vulgarisation des mathématiques.

1.3 Valorisation de la recherche

1.3.1 Activités éditoriales et organisation de conférences

- Editeur associé des revues suivantes depuis 2006 :
  - **Journal of Cryptology**, publié par Springer.
  - **Journal of Mathematical Cryptology**, publié par de Gruyter.
- Participation à plus d'une vingtaine de comités de programme de congrès internationaux, dont les trois principaux congrès en cryptologie :
  - **CRYPTO** en 2006 et 2009 ;
  - **EUROCRYPT** en 2002, 2005, 2007 et 2008 ;
  - **ASIACRYPT** en 2002, 2003 et 2005.
- Président du comité de programme de :
  - congrès **VietCrypt** (2006) dont les actes furent publiés par Springer comme volume 4341 des *Lecture Notes in Computer Science*.
  - congrès **LLL+25** (2007) commémorant le 25ième anniversaire de l'algorithme LLL. Livre à paraître chez Springer.
- Organisateur des évènements suivants :
  - Congrès *Financial Cryptography* 2003 en Guadeloupe ;
  - École d'été ECRYPT en cryptanalyse (2005) à Samos, Grèce ;
  - *rump session* du congrès CRYPTO 2005 à Santa Barbara, Etats-Unis.

1.3.2 Conférences invitées

- Conférencier invité aux congrès et colloques internationaux suivants :
  - 2007 Workshop on **Practical Aspects of Cryptography** – *Espagne*
  - 2006 **Lattice day** – *Pays-Bas*
    Workshop on **Number Theory and Cryptography - Open Problems** – *Etats-Unis*
    Conference on **Theories and Applications of Computer Science** – *Vietnam*
  - 2005 Journées nationales de **Calcul Formel** – *France*
    Workshop on **Mathematical Problems and Techniques in Cryptology** – *Espagne*
    Journée **Cryptographie** – *France*
    Oberwolfach Meeting on **Lattices and their applications** – *Allemagne*
  - 2004 Colloque de **Cryptographie** – *France*
  - 2003 Workshop on **Mathematics of Cryptology** – *Pays-Bas*
    Workshop on **Lattice reduction and applications to cryptology** – *France*
    One day meeting on **Cryptographic Number Theory** – *Royaume-Uni*
    Workshop on **Algebra and Communications** – *Suisse*
  - 2002 The 6th Workshop on Elliptic Curve Cryptography (**ECC 2002**) – *Allemagne*

2001 Workshop on **Algebraic methods in cryptography** – *Allemagne*
8th Workshop on Selected Areas in Cryptography (**SAC 2001**) – *Canada*
Mathematics of Public Key Cryptography workshop – *Corée du sud*
Cryptography and Lattices Conference (**CaLC 2001**) – *Etats-Unis*
Oberwolfach Meeting on **Finite fields** – *Allemagne*
2000 **Cryptography** workshop – *Corée du sud*
MSRI **Number-theoretic cryptography** workshop – *Etats-Unis*

– Conférencier invité par les institutions étrangères suivantes : IBM Research, Lucent Bell Labs, Microsoft Research, Stanford, MSRI (Etats-Unis), Univ. of MacQuarie, Univ. of Sydney, Univ. of Wollongong (Australie), Univ. de Francfort, Univ. de Bochum (Allemagne), EPFL (Suisse), Lorentz Center (Pays Bas), Korea Institute for Advanced Study (Corée du Sud), NTT, Univ. of Electro-Communications (Japon), Univ. of Bristol, Royal Holloway (Royaume Uni), Univ. Madrid (Espagne).

1.4 Enseignement

– Cours/TD pour un nombre total d'heures compris entre 50 et 80 heures par an :
  – Depuis 2004 : cours *réseaux euclidiens et applications en cryptologie* pour le M2 du MPRI.
  – Depuis 2002 : cours *Introduction à la cryptologie* à l'EPITA, dans les sections *réseaux et sécurité* et *télécommunications*.
  – 2005 : cours *Introduction à la cryptologie* à l'INSIA.
  – 2001–03 : cours *Vingt-cinq ans d'attaques de RSA* pour les élèves de première année de l'ENS.
  – 2001–03 : cours de mastère *cryptographie à clef publique* à l'ENSI–Bourges ;
  – 2001–03 : séminaires de cryptographie à clef publique à l'ENPC.
– Professeur invité à l'Université de Francfort, Allemagne (2001).
– Enseignant invité dans les écoles d'été suivantes : Bonn (Allemagne, 2006), Santander (Espagne, 2005), Bordeaux (France, 2005), Grenoble (France, 2002), Leiden (Pays-Bas, 2001).
– Moniteur en informatique, université Paris 7 (1997–1999).
– Examinateur en mathématiques, dans les classes préparatoires du lycée Charlemagne, Paris (1995-1996) et du lycée Fauriel, Saint-Étienne (1994-1995).
– Responsable de l'épreuve écrite d'informatique pour le concours d'entrée international de l'ENS, depuis 2006.

1.5 Encadrement de la recherche

– Doctorants :
  2003–05 **Damien Stehlé** (encadrement partiel) – *algorithmique des réseaux euclidiens*
  2005– **Nicolas Gama** – *algorithmique des réseaux euclidiens et applications en cryptanalyse*
  2006– **Gaëtan Leurent** (encadrement partiel) – *fonctions de hachage*
– Etudiants en stage de M2 ou DEA :
  2007 **Thomas Vidick** (ENS) – *l'algorithme AKS de plus court vecteur*
  2006 **Gaëtan Leurent** (ENS) – *collisions sur les fonctions de hachage MD4-MD5*
  Aurore Bernard (Paris 7) – *nouvelles fonctions de hachage à sécurité prouvée*
  2005 **Nicolas Gama** (ENS) – *réduction de réseau en grande dimension*
  Thai Hoang Le (Polytechnique) – *couplages et applications en cryptographie*
  2004 **Antoine Scemama** (ENSICAEN) – *expérimentations autour de la réduction de réseau*
  2002 **Damien Stehlé** (ENS) – *réduction de réseau en petite dimension*
– Etudiants en stage de licence :

2001 **Bessem Hmidi** (Polytechnique) – *attaques contre NTRU*
1998 **Christophe Coupé** (ENS-Lyon) – *attaques contre RSA*

1.6 Projets de recherche

– Coordinateur du laboratoire virtuel AZTEC (2004–08). AZTEC est l'un des cinq laboratoires virtuels du réseau d'excellence européen ECRYPT (IST-2002-507932) en cryptologie, qui réunit trente-deux partenaires (250 chercheurs) représentant quatorze pays européens. AZTEC est dédié à la cryptologie à clef publique, et regroupe vingt équipes de recherche.

– Responsable pour l'ENS de projets de recherche : projet européen STORK (2002–03) et projet RNRT sur les turbo-signatures (2001).

Chapitre 2

Publications

Sommaire

2.1 Édition d'actes de conférence 87
2.2 Chapitres de livre 87
2.3 Articles de revue 87
2.4 Conférences internationales avec comité de lecture 88
2.4.1 Articles invités 88
2.4.2 Congrès FOCS et STOC 88
2.4.3 Congrès CRYPTO et EUROCRYPT 88
2.4.4 Congrès ASIACRYPT 89
2.4.5 Autres congrès de cryptologie 89
2.4.6 Congrès de théorie des nombres algorithmique 90
2.5 Vulgarisation 91
2.6 Brevets 91
2.7 Thèse 91

Ce chapitre contient la liste de toutes mes publications entre 1997 et 2007. Les publications les plus prestigieuses en cryptologie sont celles des conférences CRYPTO et EUROCRYPT, qui ont été créées au début des années 1980.

2.1 Édition d'actes de conférence

1. P. Q. Nguyen (Ed.). Progress in Cryptology – VIETCRYPT 2006 – Proceedings of the First International Conference on Cryptology in Vietnam. Volume 4341 of *Lecture Notes in Computer Science*, Springer, 2006.

2.2 Chapitres de livre

1. P. Q. Nguyen and J. Stern. La Cryptologie : enjeux et perspectives. Chapitre 6 de *Paradigmes et enjeux de l'informatique*, Lavoisier, 2005.
2. P. Q. Nguyen. *Encyclopedia of Cryptography and Security*, Springer, 2005. Entrées *Lattice* et *Lattice Reduction*.

2.3 Articles de revue

1. P. Q. Nguyen and I. E. Shparlinski. The insecurity of the Digital Signature Algorithm with partially known nonces. *Journal of Cryptology*, 15(3) :151–176, 2002.

2. P. Q. Nguyen and I. E. Shparlinski. The insecurity of the elliptic curve Digital Signature Algorithm with partially known nonces. *Design, Codes and Cryptography*, 30 :201–217, 2003.
3. N. A. Howgrave-Graham, P. Q. Nguyen, and I. E. Shparlinski. Hidden number problem with hidden multipliers, timed-release crypto and noisy exponentiation. *Mathematics of Computation*, 72(243) :1473–1485, 2003.
4. D. Coppersmith, N. A. Howgrave-Graham, P. Q. Nguyen, and I. E. Shparlinski. Testing set proportionality and the Ádám isomorphism of circulant graphs. *Journal of Discrete Algorithms* 4(2) : 324–335, 2006.
5. P. Q. Nguyen and T. Vidick. Sieve Algorithms for the Shortest Vector Problem Are Practical. A paraître au *J. of Mathematical Cryptology*.

2.4 Conférences internationales avec comité de lecture

2.4.1 Articles invités

1. P. Q. Nguyen and J. Stern. Lattice reduction in cryptology : An update. In *Algorithmic Number Theory – Proc. of ANTS-IV*, volume 1838 of *LNCS*, pages 85–112. Springer, 2000.
2. P. Q. Nguyen and J. Stern. The two faces of lattices in cryptology. In *Cryptography and Lattices – Proc. CALC '01)*, volume 2146 of *LNCS*, pages 146–180. Springer, 2001.
3. P. Q. Nguyen. The two faces of lattices in cryptology. In *Selected Areas in Cryptography – Proc. SAC '01*, volume 2259 of *LNCS*, page 313. Springer, 2001. Résumé de l'entrée précédente.

2.4.2 Congrès FOCS et STOC

1. N. Gama and P. Q. Nguyen. Finding Short Lattice Vectors Within Mordell's Inequality. A paraître dans *STOC '08 : 40th ACM Symposium on Theory of Computing*, ACM, 2008.

2.4.3 Congrès CRYPTO et EUROCRYPT

1. P. Nguyen and J. Stern. Merkle-Hellman Revisited : a Cryptanalysis of the Qu-Vanstone Cryptosystem Based on Group Factorizations. In *Proc. of the 17th Cryptology Conference (Crypto '97)*, volume 1294 of *Lecture Notes in Computer Science*, pages 198–212. IACR, Springer, 1997.
2. P. Nguyen and J. Stern. Cryptanalysis of the Ajtai-Dwork Cryptosystem. In *Proc. of the 18th Cryptology Conference (Crypto '98)*, volume 1462 of *Lecture Notes in Computer Science*, pages 223–242. IACR, Springer, 1998.
3. P. Nguyen and J. Stern. The Hardness of the Hidden Subset Sum Problem and its Cryptographic Implications. In *Proc. of the 19th Cryptology Conference (Crypto '99)*, volume 1666 of *Lecture Notes in Computer Science*, pages 31–46. IACR, Springer, 1999.
4. P. Nguyen. Cryptanalysis of the Goldreich-Goldwasser-Halevi Cryptosystem from Crypto '97. In *Proc. of the 19th Cryptology Conference (Crypto '99)*, volume 1666 of *Lecture Notes in Computer Science*, pages 288–304. IACR, Springer, 1999.
5. D. Bleichenbacher and P. Q. Nguyen. Noisy Polynomial Interpolation and Noisy Chinese Remaindering. In *Proc. of the 18th Eurocrypt Conference (Eurocrypt '00)*, volume 1807 of *LNCS*, pages 53–69. IACR, Springer, 2000.
6. P. Q. Nguyen and D. Pointcheval. Analysis and improvements of NTRU encryption paddings. In *Proc. of the 22nd Cryptology Conference (Crypto '02)*, volume 2442 of *LNCS*, pages 210–225. IACR, Springer, 2002.

7. N. A. Howgrave Graham, P. Q. Nguyen, D. Pointcheval, J. Proos., J. H. Silverman, A. Singer, and W. Whyte. The impact of decryption failures on the security of NTRU encryption. In *Proc. of the 23rd Cryptology Conference (Crypto '03)*, volume 2729 of *LNCS*, pages 226–246. IACR, Springer, 2003.
8. P. Q. Nguyen. Can we trust cryptographic software ? cryptographic flaws in GNU privacy guard v1.2.3. In *Proc. of the 22nd Eurocrypt Conference*, volume 3027 of *LNCS*, pages 555–570. IACR, Springer, 2004.
9. P. Q. Nguyen and D. Stehlé. Floating-Point LLL Revisited. In *Proc. of the 23rd Eurocrypt Conference*, volume 3494 of *LNCS*, pages 215–233. IACR, Springer, 2005.
10. N. Gama and N. A. Howgrave-Graham and P. Q. Nguyen. Symplectic Lattice Reduction and NTRU. In *Proc. of the 24th Eurocrypt Conference*, volume 4004 *LNCS*, pages 233–253. IACR, Springer, 2006.
11. P. Q. Nguyen and O. Regev. Learning a Parallelepiped : Cryptanalysis of GGH and NTRU Signatures. In *Proc. of the 24th Eurocrypt Conference*, volume 4004 of *LNCS*, pages 271–288. IACR, Springer, 2006. **Best Paper Award**.
12. N. Gama and N. A. Howgrave-Graham and H. Koy and P. Q. Nguyen. Rankin's Constant and Blockwise Lattice Reduction. In *Proc. of the 26th Crypto Conference*, volume 4117 of *LNCS*, pages 112–130. IACR, Springer, 2006.
13. P.-A. Fouque and G. Leurent and P. Q. Nguyen. Full Key-Recovery Attacks on HMAC/NMAC-MD4 and NMAC-MD5. In *Proc. of the 27th Crypto Conference*, volume 4622 of *LNCS*, pages 13–30. IACR, Springer, 2007.
14. N. Gama and P. Q. Nguyen. Predicting Lattice Reduction. In *Proc. of the 26th Eurocrypt Conference*, A paraître dans *LNCS*, IACR, Springer, 2008.

2.4.4 Congrès ASIACRYPT

1. P. Nguyen and J. Stern. The Béguin-Quisquater Server-Aided RSA Protocol from Crypto '95 is not Secure. In *Advances in Cryptology – Proceedings of Asiacrypt '98*, volume 1514 of *Lecture Notes in Computer Science*, pages 372–379. Springer, 1998.
2. G. Durfee and P. Q. Nguyen. Cryptanalysis of the RSA schemes with short secret exponent from Asiacrypt '99. In *Proc. of Asiacrypt '00*, volume 1976 of *LNCS*, pages 14–29. IACR, Springer, 2000.
3. D. Boneh, A. Joux, and P. Q. Nguyen. Why textbook ElGamal and RSA encryption are insecure. In *Proc. of Asiacrypt '00*, volume 1976 of *LNCS*, pages 30–43. IACR, Springer, 2000.
4. P. Q. Nguyen and I. E. Shparlinski. On the insecurity of a server-aided RSA protocol. In *Proc. of Asiacrypt '01*, volume 2248 of *LNCS*, pages 21–35. IACR, Springer, 2001.
5. D. Catalano, P. Q. Nguyen, and J. Stern. The hardness of Hensel lifting : the case of RSA and discrete logarithm. In *Proc. of Asiacrypt '02*, volume 2501 of *LNCS*, pages 299–310. IACR, Springer, 2002.
6. P. Q. Nguyen, and J. Stern. Adapting Density Attacks to Low-Weight Knapsacks. In *Proc. of Asiacrypt '05*, volume 3788 of *LNCS*, pages 41–58. IACR, Springer, 2005.

2.4.5 Autres congrès de cryptologie

1. P. Nguyen and J. Stern. Cryptanalysis of a fast public key cryptosystem presented at SAC '97. In *Selected Areas in Cryptography – Proc. of SAC '98*, volume 1556 of *LNCS*. Springer, 1998.

2. C. Coupé, P. Nguyen, and J. Stern. The Effectiveness of Lattice Attacks Against Low-Exponent RSA. In *Proceedings of PKC '98*, volume 1560 of *Lecture Notes in Computer Science*, pages 204–218. Springer, 1999.

3. O. Baudron, H. Gilbert, L. Granboulan, H. Handschuh, A. Joux, P. Nguyen, F. Noilhan, D. Pointcheval, T. Pornin, G. Poupard, J. Stern, and S. Vaudenay. DFC update. In *Proc. of the 2nd Advanced Encryption Standard (AES) Candidate Conference*. NIST, 1999.

4. O. Baudron, H. Gilbert, L. Granboulan, H. Handschuh, A. Joux, P. Nguyen, F. Noilhan, D. Pointcheval, T. Pornin, G. Poupard, J. Stern, and S. Vaudenay. Report on the AES candidates. In *Proc. of the 2nd Advanced Encryption Standard (AES) Candidate Conference*. NIST, 1999.

5. L. Granboulan, P. Q. Nguyen, F. Noilhan, and S. Vaudenay. Dfcv2. In *Selected Areas in Cryptography – Proc. of SAC '00*, volume 2012 of *LNCS*. Springer, 2001.

6. P. Q. Nguyen. The dark side of the hidden number problem : Lattice attacks on DSA. In K.-Y. Lam, I. E. Shparlinski, H. Wang, and C. Xing, editors, *Proc. Workshop on Cryptography and Comp. Number Theory*, pages 321–330. Birkhauser, 2001.

7. P. Q. Nguyen, I. E. Shparlinski, and J. Stern. Distribution of Modular Sums and Security of Server-Aided Exponentiation. In K.-Y. Lam, I. E. Shparlinski, H. Wang, and C. Xing, editors, *Proc. Workshop on Cryptography and Comp. Number Theory*, pages 331–342. Birkhäuser, 2001.

8. E. El Mahassni, P. Q. Nguyen, and I. E. Shparlinski. The insecurity of Nyberg–Rueppel and other DSA-like signature schemes with partially known nonce. In *Cryptography and Lattices – Proc. CALC '01)*, volume 2146 of *LNCS*, pages 97–109. Springer, 2001.

9. D. Catalano, R. Gennaro, N. Howgrave-Graham, and P. Q. Nguyen. Paillier's cryptosystem revisited. In P. Samarati, editor, *8th ACM Conference on Computer and Communications Security*. ACM Press, 2001.

10. M. Jakobsson, A. Juels, and P. Q. Nguyen. Proprietary certificates. In *Proc. RSA '02*, volume 2271 of *LNCS*. Springer, 2002.

11. D. Naccache, P. Q. Nguyen, M. Tunstall and C. Whelan. Experimenting with Faults, Lattices and the DSA. In *Proceedings of PKC '05*, volume 3386 of *Lecture Notes in Computer Science* Springer, 2005.

12. E. Biham, L. Granboulan and P. Q. Nguyen. Impossible Fault Analysis of RC4 and Differential Fault Analysis of RC4. In *Proceedings of FSE '05*, volume 3557 of *Lecture Notes in Computer Science* Springer, 2005.

13. N. Gama and P. Q. Nguyen. New Chosen-Ciphertext Attacks on NTRU. In *Proceedings of PKC '07*, volume 4450 of *Lecture Notes in Computer Science* Springer, 2007.

2.4.6 Congrès de théorie des nombres algorithmique

1. P. Nguyen. A Montgomery-like Square Root for the Number Field Sieve. In *Algorithmic Number Theory – Proceedings of ANTS-III*, volume 1423 of *Lecture Notes in Computer Science*, pages 151–168. Springer, 1998.

2. P. Q. Nguyen and D. Stehlé. Low-dimensional lattice basis reduction revisited. In *Algorithmic Number Theory – Proc. of ANTS-VI*, volume 3076 of LNCS, pages 338–357. Springer, 2004.

3. P. Q. Nguyen and D. Stehlé. LLL on the Average. In *Algorithmic Number Theory – Proc. of ANTS-VII*, volume 4076 of LNCS, pages 238–256. Springer, 2006.

2.5 Vulgarisation

1. P. Nguyen. La géométrie des nombres : de Gauss aux codes secrets. *Pour la Science*, (36) :104–106, Juillet 2002. Hors-Série.
2. P. Q. Nguyen, editor. *New Trends in Cryptology*. European Union, 2003. European project STORK – Strategic Roadmap for Crypto – IST-2002-38273.

2.6 Brevets

1. M. Jakobsson and P. Q. Nguyen. Methods of apparatus for private certificates in public key cryptography. USA patent 1200-1013, June 2002.

2.7 Thèse

1. P. Q. Nguyen. *La Géométrie des Nombres en Cryptologie*. Thèse de doctorat, Université Paris 7, 1999.

Troisième partie

Articles joints

Algorithmique des réseaux euclidiens

Cette partie se compose de cinq articles :

Page 97 : version complète de [263].

Low-Dimensional Lattice Basis Reduction Revisited, ANTS-VI (2004)
Phong Q. Nguyen et Damien Stehlé

Cet article étudie une généralisation naturelle (dite gloutonne) de l'algorithme de Lagrange. Il montre que jusqu'en dimension quatre, cette généralisation possède les mêmes propriétés essentielles que l'algorithme d'Euclide : le résultat renvoyé est optimal, et le temps de calcul est quadratique en la taille des entrées sans arithmétique rapide.

Page 135 : version complète de [264].

An LLL Algorithm with Quadratic Complexity, EUROCRYPT 2005
Phong Q. Nguyen et Damien Stehlé

Cet article introduit l'algorithme L^2, la variante prouvée la plus efficace connue du célèbre algorithme LLL. C'est la seule qui soit prouvée quadratique en la taille des coefficients des vecteurs de base, sans arithmétique rapide, tout comme l'algorithme d'Euclide pour le calcul du pgcd.

Page 161 : référence [265].

LLL on the Average, ANTS-VII (2006)
Phong Q. Nguyen et Damien Stehlé

Cet article présente des expérimentations intensives avec l'algorithme LLL, de façon à comparer précisément les performances expérimentales « moyennes » de LLL avec les bornes théoriques prouvées. On dit souvent que LLL se comporte bien mieux en pratique : cette affirmation est à la fois vraie et fausse.

Page 175 : référence [104].

Symplectic Lattice Reduction and NTRU, EUROCRYPT 2006
Nicolas Gama, Nick Howgrave-Graham et Phong Q. Nguyen

Cet article exhibe une propriété mathématique spéciale des réseaux utilisés par le cryptosystème NTRU, et en tire parti pour accélérer l'algorithme LLL.

Page 191 : référence [103].

Rankin's Constant and Blockwise Lattice Reduction, CRYPTO 2006
Nicolas Gama, Nick Howgrave-Graham, Henrik Koy et Phong Q. Nguyen

Cet article améliore l'analyse de l'algorithme de Schnorr. Il propose aussi une légère amélioration de cet algorithme, qui est en théorie le meilleur algorithme d'approximation du plus court vecteur (SVP) connu.

Annexe A

Low-Dimensional Lattice Basis Reduction Revisited

ANTS-VI (2004)
Version complète de [263], avec Damien Stehlé (LORIA)

Abstract: *Lattice reduction is a geometric generalization of the problem of computing greatest common divisors. Most of the interesting algorithmic problems related to lattice reduction are NP-hard as the lattice dimension increases. This article deals with the low-dimensional case. We study a greedy lattice basis reduction algorithm for the Euclidean norm, which is arguably the most natural lattice basis reduction algorithm, because it is a straightforward generalization of an old two-dimensional algorithm of Lagrange, usually known as Gauss' algorithm, and which is very similar to Euclid's gcd algorithm. Our results are two-fold. From a mathematical point of view, we show that up to dimension four, the output of the greedy algorithm is optimal : the output basis reaches all the successive minima of the lattice. However, as soon as the lattice dimension is strictly higher than four, the output basis may be arbitrarily bad as it may not even reach the first minimum. More importantly, from a computational point of view, we show that up to dimension four, the bit-complexity of the greedy algorithm is quadratic without fast integer arithmetic, just like Euclid's gcd algorithm. This was already proved by Semaev up to dimension three using rather technical means, but it was previously unknown whether or not the algorithm was still polynomial in dimension four. We propose two different analyses : a global approach based on the study of the geometry of the current basis when the length decrease stalls, and a local approach based on the fact that a significant length decrease is reached every $O(1)$ consecutive steps. Our analyses, which strongly rely upon geometric properties of low-dimensional lattices, arguably simplify Semaev's analysis in dimensions two and three and unify the cases of dimensions two to four. Although the global approach seems more powerful, we also present the local approach because it gives a more precise understanding of the behaviour of the algorithm.*

A.1 Introduction

A *lattice* is a discrete subgroup of $\mathbb{R}^n$. Any lattice L has a *lattice basis*, i.e., a set $\{\mathbf{b}_1, \ldots, \mathbf{b}_d\}$ of linearly independent vectors such that the lattice is the set of all integer linear combinations of the $\mathbf{b}_i$'s : $L[\mathbf{b}_1, \ldots, \mathbf{b}_d] = \left\{\sum_{i=1}^{d} x_i \mathbf{b}_i, x_i \in \mathbb{Z}\right\}$. A lattice basis is usually not unique, but all the bases have the same number of elements, called the *dimension* or *rank* of the lattice. In dimension higher

than one, there are infinitely many bases, but some are more interesting than others : they are called *reduced*. Roughly speaking, a reduced basis is a basis made of reasonably short vectors that are almost orthogonal. Finding good reduced bases has proved invaluable in many fields of computer science and mathematics, particularly in cryptology (see for instance the survey [267]) ; and the computational complexity of lattice problems has attracted considerable attention in the past few years (see for instance the book [240]), following Ajtai's discovery [4] of a connection between the worst-case and average-case complexities of certain lattice problems. Lattice reduction can be viewed as a geometric generalization of gcd computations.

There exist many different notions of reduction, such as those of Hermite [142], Minkowski [243], Korkine-Zolotarev (KZ) [143, 191], Venkov [296], Lenstra-Lenstra-Lovász [205], etc. Among these, the most intuitive one is perhaps Minkowski's, and up to dimension four it is arguably optimal compared to all other known reductions, because it reaches all the so-called successive minima of a lattice. However, finding a Minkowski-reduced basis or a HKZ-reduced basis is NP-hard under randomised reductions as the dimension increases, because such bases contain a shortest lattice vector and the shortest vector problem is NP-hard under randomised reductions [5, 237]. In order to better understand lattice reduction, it is tempting to study the low-dimensional case. Improvements in low-dimensional lattice reduction may lead to significant running-time improvements in high-dimensional lattice reduction, as the best lattice reduction algorithms known in theory and in practice for high-dimensional lattices, namely Schnorr's block-wise HKZ-reduction [302] and its heuristic variants [307, 308], are based on a repeated use of low-dimensional HKZ-reduction.

Lagrange's algorithm [201], usually called Gauss' algorithm, computes in quadratic time (without fast integer arithmetic [309]) a Minkowski-reduced basis of any two-dimensional lattice. This algorithm, which is a natural generalization of Euclid's gcd algorithm, was also described later by Gauss [109], and is often erroneously called Gauss' algorithm. It was extended to dimension three by Vallée [343] in 1986 and Semaev [314] in 2001 : Semaev's algorithm is quadratic without fast integer arithmetic, whereas Vallée's has cubic complexity. More generally, Helfrich [141] showed in 1985 by means of the LLL algorithm [205] how to compute in cubic time a Minkowski-reduced basis of any lattice of fixed (arbitrary) dimension, but the hidden complexity constant grows very fast with the dimension. Finally, Eisenbrand and Rote described in [90] a lattice basis reduction algorithm with a quasi-linear time complexity in any fixed dimension, but it is based on exhaustive enumerations and the complexity seems to blow up very quickly when the dimension increases.

In this paper, we generalize Lagrange's algorithm to arbitrary dimension. Although the obtained greedy algorithm is arguably the simplest lattice basis reduction algorithm known, its analysis becomes remarkably more and more complex as the dimension increases. Semaev [314] was the first to prove that the algorithm was still polynomial time in dimension three, but the polynomial-time complexity remained open for higher dimension. We show that up to dimension four, the greedy algorithm computes a Minkowski-reduced basis in quadratic time without fast arithmetic. This implies that a shortest vector and a HKZ-reduced basis can be computed in quadratic time up to dimension four. Independently of the running time improvement, we hope our analysis may help to design new lattice reduction algorithms.

We propose two different approaches, both based on geometric properties of low-dimensional lattices, which generalize two different analyses of Lagrange's algorithm. The global approach generalizes up to dimension four the two-dimensional analysis of Vallée [344] and Akhavi [9]. This approach has been used in [9] to bound the number of loop iterations of the so-called optimal LLL algorithm in any dimension. Our generalization is different from this one. Roughly speaking, the global approach considers the following question : what happens when the algorithm stops making the lengths of the basis vectors decrease much ? The local approach considers the dual question : can we bound the number of consecutive steps necessary to significantly decrease the lengths of the basis vectors ? In dimension two, this method is very close to the argument given by Semaev in [314], which is itself very different

from previous analyses of Lagrange's algorithm [174, 198, 344]. In dimension three, Semaev's analysis [314] is based on a rather exhaustive analysis of all the possible behaviours of the algorithm, which involves quite a few computations and makes it difficult to extend to higher dimension. We replace the main technical arguments by geometrical considerations on two-dimensional lattices. This makes it possible to extend the analysis to dimension four, by carefully studying geometrical properties of three-dimensional lattices, although a few additional technical difficulties appear.

Road-map of the paper : In Section A.2, we recall useful facts about lattices. In Section A.3, we recall Lagrange's algorithm and give two different complexity analyses. In Section A.4 we describe its natural greedy generalization. Section A.5 provides an efficient low-dimensional closest vector algorithm, which is the core of the greedy algorithm. In Section A.6, we give our global approach to bound the number of loop iterations of the algorithm, and in Section A.7 we prove the claimed quadratic complexity bound. In Section A.8 we give an alternative proof for the bound of the number of loop iterations, the so-called local approach. In dimension below four, the quadratic complexity bound can also be derived from Sections A.8 and A.7 independently of Section A.6. In Section A.9, we prove geometrical results on low-dimensional lattices that are useful to prove the so-called Gap Lemma, an essential ingredient of the local approach of Section A.8. Finally, in Section A.10, we explain the difficulties arising in dimension 5.

Preliminary remark : The present article is an extended and improved version of the conference paper [263]. Interestingly, the cancellation technique of Section A.7 has been slightly modified to give a precise analysis of a provable floating-point LLL algorithm, in [264], leading to the first LLL algorithm with quadratic complexity. In the conference version [263], we claimed that all variants of the LLL algorithm were at least cubic in any fixed dimension : this is no longer true since [264], which was motivated by the present low-dimensional work.

Notations : Let $\|\cdot\|$ and $\langle\cdot,\cdot\rangle$ denote respectively the Euclidean norm and inner product of $\mathbb{R}^n$; variables in bold are vectors ; whenever the notation $[\mathbf{b}_1,\ldots,\mathbf{b}_d]_{\leq}$ is used, we have $\|\mathbf{b}_1\| \leq \ldots \leq \|\mathbf{b}_n\|$ and in such a case, we say that the $\mathbf{b}_i$'s are *ordered*. Besides, the complexity model we use is the RAM model and the computational cost is measured in elementary operations on bits. In any complexity statement, we assume that the underlying lattice L is integral ($L \subseteq \mathbb{Z}^n$). If $x \in \mathbb{R}$, then $\lfloor x \rceil$ denotes a nearest integer to x. For any $n \in \mathbb{N}$, $\mathcal{S}_n$ denotes the group of the permutations of $[\![1, n]\!]$.

A.2 Preliminaries

We assume the reader is familiar with geometry of numbers (see [57, 220, 326]).

A.2.1 Some Basic Definitions

We first recall some basic definitions related to Gram-Schmidt orthogonalisation and the first minima.

Gram-Schmidt orthogonalisation. Let $\mathbf{b}_1,\ldots,\mathbf{b}_d$ be vectors. The *Gram matrix* $G(\mathbf{b}_1,\ldots,\mathbf{b}_d)$ of $\mathbf{b}_1,\ldots,\mathbf{b}_d$ is the $d \times d$ symmetric matrix $(\langle \mathbf{b}_i, \mathbf{b}_j\rangle)_{1\leq i,j\leq d}$ formed by all the inner products. The vectors $\mathbf{b}_1,\ldots,\mathbf{b}_d$ are linearly independent if and only if the determinant of $G(\mathbf{b}_1,\ldots,\mathbf{b}_d)$ is not zero. The volume vol$L$ of a lattice L is the square root of the determinant of the Gram matrix of any basis of L. The orthogonality-defect of a basis $(\mathbf{b}_1,\ldots,\mathbf{b}_d)$ of L is defined as $(\prod_{i=1}^{d} \|\mathbf{b}_i\|)/\mathrm{vol}L$: it is always greater than 1, with equality if and only if the basis is orthogonal. Let $(\mathbf{b}_1,\ldots,\mathbf{b}_d)$ be linearly independent vectors. The *Gram-Schmidt orthogonalisation* $(\mathbf{b}_1^*,\ldots,\mathbf{b}_d^*)$ is defined as follows : $\mathbf{b}_i^*$ is the component of $\mathbf{b}_i$ that is orthogonal to the subspace spanned by the vectors $\mathbf{b}_1,\ldots,\mathbf{b}_{i-1}$.

Gram-Schmidt coefficients. Any basis vector $\mathbf{b}_i$ can be expressed as a (real) linear combination of the previous $\mathbf{b}_j^*$'s. We define the *Gram-Schmidt coefficients* $\mu_{i,j}$'s as the coefficients of these linear

combinations, namely, for any $i \in [\![1, d]\!]$:

$$\mathbf{b}_i = \mathbf{b}_i^* + \sum_{j<i} \mu_{i,j} \mathbf{b}_j^*.$$

We have the equality $\mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\|\mathbf{b}_j^*\|^2}$, for any $j < i$.

Successive minima and Minkowski reduction. Let L be a d-dimensional lattice in $\mathbb{R}^n$. For $1 \leq i \leq d$, the *i-th minimum* $\lambda_i(L)$ is the radius of the smallest closed ball centred at the origin containing at least i linearly independent lattice vectors. The most famous lattice problem is the *shortest vector problem* (SVP) : given a basis of a lattice L, find a lattice vector whose norm is exactly $\lambda_1(L)$. There always exist linearly independent lattice vectors $\mathbf{v}_i$'s such that $\|\mathbf{v}_i\| = \lambda_i(L)$ for all i. Amazingly, as soon as $d \geq 4$ such vectors do not necessarily form a lattice basis, and when $d \geq 5$ there may not even exist a lattice basis reaching all the minima.

A.2.2 Different Types of Reduction

Several types of strong lattice basis reductions are often considered in the literature. The following list is not exhaustive but includes the most frequent ones.

Minkowski reduction. A basis $[\mathbf{b}_1, \ldots, \mathbf{b}_d]_{\leq}$ of a lattice L is *Minkowski-reduced* if for all $1 \leq i \leq d$, the vector $\mathbf{b}_i$ has minimal norm among all lattice vectors $\mathbf{b}_i$ such that $[\mathbf{b}_1, \ldots, \mathbf{b}_i]_{\leq}$ can be extended to a basis of L. Equivalently :

Lemma A.1 *A basis $[\mathbf{b}_1, \ldots, \mathbf{b}_d]_{\leq}$ of a lattice L is Minkowski-reduced if and only if for any $i \in [\![1, d]\!]$ and for any integers $x_1, \ldots, x_d$ such that $x_i, \ldots, x_d$ are altogether coprime, we have :*

$$\|x_1\mathbf{b}_1 + \ldots + x_d\mathbf{b}_d\| \geq \|\mathbf{b}_i\|.$$

With the above statement, one could think that to ensure that a given basis is Minkowski reduced, there is an infinity of conditions to be checked. A classical result states that in any fixed dimension, it is sufficient to check a finite subset of them. This result is described as the second finiteness theorem in [326]. Several sets of conditions are possible. We call *Minkowski conditions* such a subset with minimal cardinality. Minkowski conditions have been obtained by Tammela [339] up to dimension 6. As a consequence, in small dimension one can check very quickly if a basis is Minkowski-reduced by checking these conditions.

Theorem A.1 (Minkowski conditions) *Let $d \leq 6$. A basis $[\mathbf{b}_1, \ldots, \mathbf{b}_d]_{\leq}$ of L is Minkowski-reduced if and only if for any $i \leq d$ and for any integers $x_1, \ldots, x_d$ that satisfy both conditions below, we have the inequality :*

$$\|x_1\mathbf{b}_1 + \ldots + x_d\mathbf{b}_d\| \geq \|\mathbf{b}_i\|.$$

1. *The integers $x_i, \ldots, x_d$ are altogether coprime,*
2. *For some permutation σ of $[\![1, d]\!]$, $(|x_{\sigma(1)}|, \ldots, |x_{\sigma(d)}|)$ appears in the list below (where blanks eventually count as zeros).*

1	*1*				
1	*1*	*1*			
1	*1*	*1*	*1*		
1	*1*	*1*	*1*	*1*	
1	*1*	*1*	*1*	*2*	
1	*1*	*1*	*1*	*1*	*1*
1	*1*	*1*	*1*	*1*	*2*
1	*1*	*1*	*1*	*2*	*2*
1	*1*	*1*	*1*	*2*	*3*

Moreover this list is minimal, which means that if any condition is disregarded, then a basis can satisfy all the others without being Minkowski-reduced.

A basis of a d-dimensional lattice that reaches the d minima must be Minkowski-reduced, but a Minkowski-reduced basis may not reach all the minima, except the first four ones (see [350]) : if $[\mathbf{b}_1, \ldots, \mathbf{b}_d]_{\leq}$ is a Minkowski-reduced basis of L, then for all $1 \leq i \leq \min(d, 4)$ we have $\|\mathbf{b}_i\| = \lambda_i(L)$. Therefore, a Minkowski-reduced basis is optimal in a natural sense up to dimension four. Another classical result (see [350]) states that the orthogonality-defect of a Minkowski-reduced basis can be upper-bounded by a constant that only depends on the lattice dimension.

Hermite reduction. Hermite [143] defined different types of reduction, which sometimes creates confusions. In particular, Hermite reduction is different from what we call below Hermite-Korkine-Zolotarev reduction. An ordered basis $[\mathbf{b}_1, \ldots, \mathbf{b}_d]_{\leq}$ is *Hermite-reduced* if it is the smallest basis of L for the lexicographic order : for any other basis $[\mathbf{b}'_1, \ldots, \mathbf{b}'_d]_{\leq}$ of L, we must have $\|\mathbf{b}_1\| = \|\mathbf{b}'_1\|, \ldots, \|\mathbf{b}_{i-1}\| = \|\mathbf{b}'_{i-1}\|$ and $\|\mathbf{b}'_i\| > \|\mathbf{b}_i\|$ for some $i \in [\![1, d]\!]$. In particular a Hermite-reduced basis is always Minkowski-reduced. The converse is true as long as $d \leq 6$ (see [296]).

Hermite-Korkine-Zolotarev reduction. This reduction is often called more simply Korkine-Zolotarev reduction. A basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ of a lattice L is *Hermite-Korkine-Zolotarev-reduced* (HKZ-reduced for short) if $\|\mathbf{b}_1\| = \lambda_1(L)$ and for any $i \geq 2$, the vector $\mathbf{b}_i$ is a lattice vector having minimal nonzero distance to the linear span of $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$, and the basis is *weakly reduced*, i.e., its Gram-Schmidt coefficients $\mu_{i,j}$ have absolute values $\leq 1/2$. In high dimension, the reduction of Hermite-Korkine-Zolotarev [143, 191] seems to be stronger than Minkowski's, as all the elements of a HKZ-reduced basis are known to be close to the successive minima (see [199]), while the best bound known for Minkowski-reduced bases [350] is exponential. HKZ-reduction is equivalent to Minkowski's in dimension two. But in dimension three, there are lattices such that no Minkowski-reduced basis is HKZ-reduced :

Lemma A.2 *Let* $\mathbf{b}_1 = [100, 0, 0], \mathbf{b}_2 = [49, 100, 0]$ *and* $\mathbf{b}_3 = [0, 62, 100]$. *Then* $L[\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3]_{\leq}$ *has no basis that is simultaneously Minkowski-reduced and HKZ-reduced.*

Proof. First, the basis $[\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3]_{\leq}$ is not HKZ-reduced because $|\mu_{3,2}| = \frac{62}{100} > 1/2$. But it is Minkowski-reduced (it suffices to check the Minkowski conditions to prove it). Therefore $[\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3]_{\leq}$ reaches the first three minima. More precisely, the vectors $\pm\mathbf{b}_1$ are the only two vectors reaching the first minimum, the vectors $\pm\mathbf{b}_2$ are the only two vectors reaching the second minimum and the vectors $\pm\mathbf{b}_3$ are the only two vectors reaching the third minimum. If the lattice $L[\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3]_{\leq}$ had a basis which was both Minkowski-reduced and HKZ-reduced, this basis would reach the first three minima and would be of the kind $[\pm\mathbf{b}_1, \pm\mathbf{b}_2, \pm\mathbf{b}_3]_{\leq}$, which contradicts the fact that $[\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3]_{\leq}$ is not HKZ-reduced. □

A.2.3 Voronoï Cell and Voronoï Vectors

This subsection is useful for the local approach of Sections A.8 and A.9 and can be disregarded by the reader interested only in the global approach.

The *Voronoï cell* [349] of a lattice $L = L[\mathbf{b}_1, \ldots, \mathbf{b}_d]_{\leq}$, denoted by $\mathrm{Vor}(\mathbf{b}_1, \ldots, \mathbf{b}_d)$, is the set of vectors $\mathbf{x}$ in the linear span of L that are closer to $\mathbf{0}$ than to any other lattice vector : for all $\mathbf{v} \in L$, we have $\|\mathbf{x} - \mathbf{v}\| \geq \|\mathbf{x}\|$, that is $\|\mathbf{v}\|^2 \geq 2\langle \mathbf{v}, \mathbf{x} \rangle$. The Voronoï cell is a finite polytope that tiles the linear span of L by translations by lattice vectors. We extend the notation $\mathrm{Vor}(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ to the case where the first vectors may be zero (the remaining vectors being linearly independent) : $\mathrm{Vor}(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ denotes the Voronoï cell of the lattice spanned by the non-zero $\mathbf{b}_i$'s. A vector $\mathbf{v} \in L$ is called a *Voronoï vector* if the vector $\mathbf{v}/2$ belongs to the Voronoï cell (in which case the vector $\mathbf{v}/2$ will be on the boundary of the Voronoï cell). A vector $\mathbf{v} \in L$ is a *strict Voronoï vector* if $\mathbf{v}/2$ is contained in the interior of a $(d-1)$-dimensional facet of the Voronoï cell. A classical result states that Voronoï vectors correspond to the minima of the cosets of $L/2L$. We say that $(x_1, \ldots, x_d) \in \mathbb{Z}^d$ is a *possible Voronoï coord* (respectively *possible strict Voronoï coord*) if there exists a Minkowski-reduced basis $[\mathbf{b}_1, \ldots, \mathbf{b}_d]_{\leq}$ such that $x_1\mathbf{b}_1 + \cdots + x_d\mathbf{b}_d$ is a Voronoï vector (respectively strict Voronoï vector). In his PhD thesis, Tammela [339] listed the possible strict Voronoï coords up to dimension 6. We will notice later that the set of possible Voronoï coords is strictly larger than the set of possible strict Voronoï coords.

Theorem A.2 *Let $d \leq 6$. The possible strict Voronoï coords in dimension d are the possible Voronoï coordinates in dimension $d-1$ and the d-tuples $(x_1, \ldots, x_d)$ such that there exists a permutation σ of $[\![1, d]\!]$ such that $(|x_{\sigma(1)}|, \ldots, |x_{\sigma(d)}|)$ appears in the d-th block of the table below :*

1				
1	1			
1	1	1		
1	1	1	1	
1	1	1	2	
1	1	1	1	1
1	1	1	1	2
1	1	1	2	2
1	1	1	2	3

1	1	1	1	1	1
1	1	1	1	1	2
1	1	1	1	1	3
1	1	1	1	2	2
1	1	1	1	2	3
1	1	1	2	2	2
1	1	1	2	2	3
1	1	1	2	2	4
1	1	2	2	2	3
1	1	1	2	3	3
1	1	1	2	3	4
1	1	2	2	3	4
1	2	2	2	3	3

In some parts of the article, we will deal with Voronoï coordinates with respect to other types of reduced bases : the kind of reduction considered will be clear from the context. The *covering radius* $\rho(L)$ of a lattice L is half of the diameter of the Voronoï cell. The *closest vector problem* (CVP) is a non-homogeneous version of the SVP : given a basis of a lattice and an arbitrary vector $\mathbf{x}$ of $\mathbb{R}^n$, find a lattice vector $\mathbf{v}$ minimising the distance $\|\mathbf{v} - \mathbf{x}\|$. In other words, if $\mathbf{y}$ denotes the orthogonal projection of the vector $\mathbf{x}$ over the linear span of L, the goal is to find $\mathbf{v} \in L$ such that $\mathbf{y} - \mathbf{v}$ belongs to the Voronoï cell of L.

We have seen that if $[\mathbf{b}_1, \ldots, \mathbf{b}_d]_{\leq}$ is Minkowski-reduced, then the Voronoï coordinates are confined. There is a result due to Delone and Sandakova (see [80, 335]) claiming a stronger statement : if the basis is reduced, then the (real) coordinates towards the basis vectors of any point of the Voronoï

cell are bounded. This result holds in any dimension and for different types of basis reductions, but the following is sufficient for our needs :

Theorem A.3 *The following statements hold :*

1. *Let* $[\mathbf{b}_1, \mathbf{b}_2]_\leq$ *be a Minkowski-reduced basis and* $\mathbf{u} \in \mathrm{Vor}(\mathbf{b}_1, \mathbf{b}_2)$. *Write* $\mathbf{u} = x\mathbf{b}_1 + y\mathbf{b}_2$. *Then* $|x| < 3/4$ *and* $|y| \leq 2/3$.
2. *Let* $[\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3]_\leq$ *be a Minkowski-reduced basis and* $\mathbf{u} \in \mathrm{Vor}(\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$. *Write* $\mathbf{u} = x\mathbf{b}_1 + y\mathbf{b}_2 + z\mathbf{b}_3$. *Then* $|x| < 3/2, |y| \leq 4/3$ *and* $|z| \leq 1$.

A.3 Two Classical Analyses for Lagrange's Algorithm

Lagrange's algorithm – described in Figure A.1 – can be seen as a two-dimensional generalisation of the centred Euclidean algorithm.

Input : A basis $[\mathbf{u}, \mathbf{v}]_\leq$ with its Gram matrix.
Output : A reduced basis of $L[\mathbf{u}, \mathbf{v}]$ with its Gram matrix.

1. Repeat
2. $\quad \mathbf{r} \longleftarrow \mathbf{v} - x\mathbf{u}$ where $x \longleftarrow \left\lfloor \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\|\mathbf{u}\|^2} \right\rceil$,
3. $\quad \mathbf{v} \longleftarrow \mathbf{u}$,
4. $\quad \mathbf{u} \longleftarrow \mathbf{r}$,
5. $\quad$ Update the Gram matrix of $(\mathbf{u}, \mathbf{v})$,
6. Until $\|\mathbf{u}\| \geq \|\mathbf{v}\|$.
7. Return $[\mathbf{v}, \mathbf{u}]_\leq$ and its Gram matrix.

FIG. A.1 – Lagrange's algorithm.

At Step 2 of each loop iteration, the vector $\mathbf{u}$ is shorter than the vector $\mathbf{v}$, and one would like to shorten $\mathbf{v}$, while preserving the fact that $[\mathbf{u}, \mathbf{v}]_\leq$ is a lattice basis. This can be achieved by subtracting to $\mathbf{v}$ a multiple $x\mathbf{u}$ of $\mathbf{u}$, because such a transformation is unimodular. The optimal choice (to make the norm of the vector $\mathbf{v}$ decrease as much as possible for this loop iteration) is when $x\mathbf{u}$ is the closest vector to $\mathbf{v}$, in the one-dimensional lattice spanned by $\mathbf{u}$. This gives rise to $x \longleftarrow \left\lfloor \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\|\mathbf{u}\|^2} \right\rceil$. The values $\langle \mathbf{u}, \mathbf{v} \rangle$ and $\|\mathbf{u}\|^2$ are extracted from $G(\mathbf{u}, \mathbf{v})$, which is updated at Step 5 of each loop iteration. The following is a classical result :

Theorem A.4 *Given as input any basis* $[\mathbf{u}, \mathbf{v}]_\leq$, *Lagrange's algorithm outputs a Minkowski-reduced basis of the lattice* $L[\mathbf{u}, \mathbf{v}]$ *in time* $O(\log \|\mathbf{v}\| \cdot [1 + \log \|\mathbf{v}\| - \log \lambda_1(L)])$.

Notice that this result is not trivial to prove. It is not even clear *a priori* why Lagrange's algorithm would output a Minkowski-reduced basis. The correctness statement of the theorem comes from Minkowski's conditions in dimension 2, that is Theorem A.1. Here we give two different approaches showing that Lagrange's algorithm has a quadratic bit complexity, a global analysis and a local analysis. In the remainder of the paper we will generalise both analyses to higher dimension.

A.3.1 The Global Analysis of Lagrange's Algorithm

The global analysis of Lagrange's algorithm can be found in [344] and [9]. In this approach, we distinguish two phases in the execution of Lagrange's algorithm : in the first phase, any created vector is far shorter than the one it replaces, namely it is at least $\sqrt{1+\eta}$ times shorter for some $\eta > 0$ to be made explicit later, and the second phase contains only $O(1)$ loop iterations. For this analysis, we need the definition of the η-Lagrange algorithm of Figure A.2. The first phase of an execution of

Lagrange's algorithm is exactly the execution of the η-Lagrange algorithm, while the second phase is the execution of Lagrange's algorithm given as input an η-Lagrange reduced basis, i.e., a basis that is invariant for the η-Lagrange algorithm.

Input : A basis $[\mathbf{u}, \mathbf{v}]_{\leq}$.
Output : A basis of $L[\mathbf{u}, \mathbf{v}]$.

1. Repeat
2. $\quad \mathbf{r} \longleftarrow \mathbf{v} - x\mathbf{u}$ where $x \longleftarrow \left\lfloor \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\|\mathbf{u}\|^2} \right\rceil$,
3. $\quad$ If $(1+\eta) \cdot \|\mathbf{r}\|^2 \leq \|\mathbf{v}\|^2$ then
4. $\quad\quad \mathbf{v} \longleftarrow \mathbf{u}$,
5. $\quad\quad \mathbf{u} \longleftarrow \mathbf{r}$,
6. $\quad$ Else goto Step 7.
7. Return $[\mathbf{u}, \mathbf{v}]_{\leq}$.

FIG. A.2 – The η-Lagrange algorithm.

The number of loop iterations within the first phase (that is the number of loop iterations within the η-Lagrange algorithm) is bounded by $O(1 + \log \|\mathbf{v}\|)$, because the product of the lengths of the basis vectors decreases at least by a factor $\sqrt{1+\eta}$ at each loop iteration. We now bound the length of the second phase. Let $[\mathbf{u}, \mathbf{v}]_{\leq}$ be the current basis at the end of the first phase and let $\mathbf{v}'$ be $\mathbf{v} - x\mathbf{u}$ where $x \in \mathbb{Z}$ is chosen to shorten $\mathbf{v}'$ as much as possible. We have :

$$\|\mathbf{v}'\| \leq \|\mathbf{v}\| \leq (1+\eta) \cdot \|\mathbf{v}'\|.$$

We claim that for a sufficiently small $\eta > 0$ Lagrange's algorithm terminates in $O(1)$ loop iterations. Write $\mathbf{v} = \mathbf{v}^* + \mathbf{v}^-$, where $\mathbf{v}^- = \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\|\mathbf{v}\|^2}\mathbf{u}$. The Pythagorean theorem gives that $\|\mathbf{v}'\|^2 \leq \|\mathbf{v}^*\|^2 + \|\mathbf{u}\|^2/4$, which implies that :

$$\|\mathbf{v}^*\|^2 \geq \left(\frac{1}{1+\eta} - \frac{1}{4}\right) \cdot \|\mathbf{v}\|^2.$$

Let $\mathbf{b} = x_1\mathbf{u} + x_2\mathbf{v}$ be a vector appearing during the second phase of the algorithm. We have $\|\mathbf{v}\|^2 \geq \|\mathbf{b}\|^2 \geq x_2^2\|\mathbf{v}^*\|^2$, thus $|x_2| \leq 1$ if $\eta > 0$ is chosen sufficiently small. Because of the greedy choice of x, there can be at most four loop iterations in the second phase : all but one iterations create a vector shorter than all the previous ones. In any of these iterations, a shortest element of an x-class for some $x \in \{-1, 0, 1\}$ is found, where the x-class is the set of vectors $x\mathbf{v} + y\mathbf{u}$ with $y \in \mathbb{Z}$.

It remains to estimate the cost of each Step 2, i.e., the cost of computing x. Since $\frac{|\langle \mathbf{u}, \mathbf{v} \rangle|}{\|\mathbf{u}\|^2} \leq \frac{\|\mathbf{v}\|}{\|\mathbf{u}\|}$, one can see that the bit complexity of Step 2 is $O(\log \|\mathbf{v}\| \cdot [1 + \log \|\mathbf{v}\| - \log \|\mathbf{u}\|])$. If we denote by $\mathbf{u}_i$ and $\mathbf{v}_i$ the values of $\mathbf{u}$ and $\mathbf{v}$ at the i-th iteration, then $\mathbf{v}_{i+1} = \mathbf{u}_i$ and we obtain that the bit complexity of Lagrange's algorithm is bounded by :

$$\begin{aligned} O\left(\sum_{i=1}^{\tau} \log \|\mathbf{v}_i\| \cdot [1 + \log \|\mathbf{v}_i\| - \log \|\mathbf{u}_i\|]\right) &= O\left(\log \|\mathbf{v}\| \cdot \sum_{i=1}^{\tau} [1 + \log \|\mathbf{v}_i\| - \log \|\mathbf{v}_{i+1}\|]\right) \\ &= O\left(\log \|\mathbf{v}\| \cdot [\tau + \log \|\mathbf{v}\| - \log \lambda_1(L)]\right), \end{aligned}$$

where $\tau = O(\log \|\mathbf{v}\| - \log \lambda_1)$ is the number of loop iterations. This completes the proof of Theorem A.4.

A.3.2 The Local Analysis of Lagrange's Algorithm

We provide another proof of the classical result that Lagrange's algorithm has quadratic complexity. Compared to other proofs, our local method closely resembles the recent one of Semaev [314],

itself relatively different from [10, 174, 198, 344]. The analysis is not optimal (as opposed to [344]) but its basic strategy can be extended up to dimension four. This strategy gives more information on the behaviour of the algorithm. Consider the value of x at Step 2 :

- If $x = 0$, this must be the last iteration of the loop.
- If $|x| = 1$, there are two cases :
  - If $\|\mathbf{v} - x\mathbf{u}\| \geq \|\mathbf{u}\|$, then this is the last loop iteration.
  - Otherwise we have $\|\mathbf{u} - x\mathbf{v}\| < \|\mathbf{u}\|$, which means that $\mathbf{u}$ can be shortened with the help of $\mathbf{v}$. This can only happen if this is the first loop iteration, because of the greedy strategy : the vector $\mathbf{u}$ is the former vector $\mathbf{v}$.
- Otherwise $|x| \geq 2$, which implies that $x\mathbf{u}$ is not a Voronoï vector of the lattice spanned by $\mathbf{u}$. Intuitively, this means that $x\mathbf{u}$ is far from $\mathrm{Vor}(\mathbf{u})$, so that $\mathbf{v} - x\mathbf{u}$ is considerably shorter than $\mathbf{v}$. More precisely, if $\mathbf{v}^*$ is the component of the vector $\mathbf{v}$ that is orthogonal to $\mathbf{u}$, we have :

$$\|\mathbf{v}\|^2 \geq \left(\frac{3}{2}\right)^2 \cdot \|\mathbf{u}\|^2 + \|\mathbf{v}^*\|^2 \geq 2\|\mathbf{u}\|^2 + \|\mathbf{v} - x\mathbf{u}\|^2 > 3\|\mathbf{v} - x\mathbf{u}\|^2,$$

  if this is not the last loop iteration.

This shows that the product of the norms of the basis vectors decreases by a factor at least $\sqrt{3}$ every loop iteration except possibly the first and last ones. Thus, the number τ of loop iterations is bounded by : $\tau = O(1 + \log \|\mathbf{v}\| - \log \lambda_1(L))$.

The end of the complexity analysis is the same as for the local approach : the linear bound on the number of loop iterations suffices to make the remainder of the bit complexity analysis work.

A.4 A Greedy Generalisation of Lagrange's Algorithm

In the previous section, we viewed Lagrange's algorithm as a greedy algorithm based on the one-dimensional CVP. It suggests a natural generalisation to arbitrary dimension that we call the greedy reduction algorithm. We study properties of the bases output by the greedy algorithm by defining a new type of reduction and comparing it to Minkowski's reduction.

A.4.1 The Greedy Reduction Algorithm

Lagrange's algorithm suggests the general greedy algorithm described in Figure A.3, which uses reduction and closest vectors in dimension $d-1$ to reduce bases in dimension d. We make a few simple

Name : $\mathsf{Greedy}(\mathbf{b}_1, \ldots, \mathbf{b}_d)$.
Input : A basis $[\mathbf{b}_1, \ldots, \mathbf{b}_d]_{\leq}$ with its Gram matrix.
Output : An ordered basis of $L[\mathbf{b}_1, \ldots, \mathbf{b}_d]$ with its Gram matrix.

1. If $d = 1$, return $\mathbf{b}_1$.
2. Repeat
3. Sort $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ by increasing lengths and update the Gram matrix,
4. $[\mathbf{b}_1, \ldots, \mathbf{b}_{d-1}]_{\leq} \longleftarrow \mathsf{Greedy}(\mathbf{b}_1, \ldots, \mathbf{b}_{d-1})$,
5. Compute a vector $\mathbf{c} \in L[\mathbf{b}_1, \ldots, \mathbf{b}_{d-1}]$ closest to $\mathbf{b}_d$,
6. $\mathbf{b}_d \longleftarrow \mathbf{b}_d - \mathbf{c}$ and update the Gram matrix,
7. Until $\|\mathbf{b}_d\| \geq \|\mathbf{b}_{d-1}\|$.
8. Return $[\mathbf{b}_1, \ldots, \mathbf{b}_d]_{\leq}$ and its Gram matrix.

FIG. A.3 – The greedy lattice basis reduction algorithm in dimension d.

remarks on the algorithm. If the Gram matrix is not given, we may compute it in time $O(\log^2 \|\mathbf{b}_d\|)$ for a fixed dimension d. The algorithm updates the Gram matrix each time the basis changes. Step 3

is easy : if this is the first iteration of the loop, the basis is already ordered ; otherwise, $[\mathbf{b}_1, \ldots, \mathbf{b}_{d-1}]$ is already ordered, and only $\mathbf{b}_d$ has to be inserted among $\mathbf{b}_1, \ldots, \mathbf{b}_{d-1}$. At Step 4, the greedy algorithm calls itself recursively in dimension $d-1$: the Gram matrix $G(\mathbf{b}_1, \ldots, \mathbf{b}_{d-1})$ does not need to be computed before calling the algorithm, since $G(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ is already known. At this point, we do not explain how Step 5 (the computation of closest vectors) is performed : of course, Kannan's closest vector algorithms [176] can be used but they do not suffice in order to prove the quadratic bit complexity of the greedy algorithm. We will describe a closest vector algorithm in Section A.5 that allows us to prove this complexity bound. Notice that for $d = 2$, the greedy algorithm is exactly Lagrange's algorithm. From a geometrical point of view, the goal of Steps 5 and 6 is to make sure that the orthogonal projection of the vector $\mathbf{b}_d$ over the lattice spanned by $[\mathbf{b}_1, \ldots, \mathbf{b}_{d-1}]_{\leq}$ lies in the Voronoï cell of that sublattice.

An easy proof by induction on d shows that the algorithm terminates. Indeed, the new vector $\mathbf{b}_d$ of Step 6 is strictly shorter than $\mathbf{b}_{d-1}$ if the loop does not end at Step 7. Thus the product of the norms of the $\mathbf{b}_i$'s decreases strictly at each iteration of the loop that is not the last one. But for all B, the number of lattice vectors of norm less than B is finite, which completes the proof.

Although the description of the greedy algorithm is fairly simple, analysing its complexity seems very difficult. Even the two-dimensional case of Lagrange's algorithm is not trivial.

A.4.2 An Iterative Description of the Greedy Algorithm

It is also possible to give an iterative version of the greedy algorithm, see Figure A.4. This algorithm is exactly the same as the recursive greedy algorithm of the previous subsection. With this alternative description, the resemblance with the LLL algorithm is clearer : the closest vector of Step 2 is replaced in the LLL algorithm by an approximate closest vector and the length condition of Step 4 is replaced by the so-called Lovász condition. We will use this iterative description in the bit-complexity analysis, namely in Section A.7, while the recursive description will be used in Section A.6 and A.8 as a tool to show that the number of loop iterations of the iterative algorithm is at most linear in the bit-size of the input.

Name : $\mathsf{Iterative-Greedy}(\mathbf{b}_1, \ldots, \mathbf{b}_d)$.
Input : A basis $[\mathbf{b}_1, \ldots, \mathbf{b}_d]_{\leq}$ with its Gram matrix.
Output : An ordered basis of $L[\mathbf{b}_1, \ldots, \mathbf{b}_d]$ with its Gram matrix.

1. $k \longleftarrow 2$. While $k \leq d$, do :
2. Compute a vector $\mathbf{c} \in L[\mathbf{b}_1, \ldots, \mathbf{b}_{k-1}]$ closest to $\mathbf{b}_k$,
3. $\mathbf{b}_k \longleftarrow \mathbf{b}_k - \mathbf{c}$ and update the Gram matrix,
4. If $\|\mathbf{b}_k\| \geq \|\mathbf{b}_{k-1}\|$, then $k \longleftarrow k+1$
5. Else insert $\mathbf{b}_k$ at his length rank k' (the vectors are sorted by increasing lengths), update the Gram matrix, $k \longleftarrow k'+1$.

FIG. A.4 – The iterative description of the greedy algorithm.

The main result of the paper is the following :

Theorem A.5 *Let $d \leq 4$. Given as input an ordered basis $[\mathbf{b}_1, \ldots, \mathbf{b}_d]_{\leq}$, the greedy algorithm of Figures A.3 and A.4 based on the closest vector algorithm of Section A.5 outputs a Minkowski-reduced basis of $L[\mathbf{b}_1, \ldots, \mathbf{b}_d]$, using a number of bit operations bounded by $O(\log \|\mathbf{b}_d\| \cdot [1 + \log \|\mathbf{b}_d\| - \log \lambda_1(L)])$. Moreover, in dimension five, the output basis may not be Minkowski-reduced.*

A.4.3 Greedy Reduction

Here we study some properties of the bases output by the greedy algorithm. As previously mentioned, it is not clear why Lagrange's algorithm outputs a Minkowski-reduced basis. But it is obvious

that the output basis $[\mathbf{u}, \mathbf{v}]_{\leq}$ satisfies $\|\mathbf{u}\| \leq \|\mathbf{v}\| \leq \|\mathbf{v} - x\mathbf{u}\|$ for all $x \in \mathbb{Z}$. This suggests the following definition :

Definition A.1 *An ordered basis* $[\mathbf{b}_1, \ldots, \mathbf{b}_d]_{\leq}$ *is* greedy-reduced *if for all* $2 \leq i \leq d$ *and for all* $x_1, \ldots, x_{i-1} \in \mathbb{Z}$ *:*

$$\|\mathbf{b}_i\| \leq \|\mathbf{b}_i + x_1\mathbf{b}_1 + \cdots + x_{i-1}\mathbf{b}_{i-1}\|.$$

In other words, we have the following recursive definition : a one-dimensional basis is always greedy-reduced, and an ordered basis $[\mathbf{b}_1, \ldots, \mathbf{b}_d]_{\leq}$ is greedy-reduced if and only if $[\mathbf{b}_1, \ldots, \mathbf{b}_{d-1}]_{\leq}$ is greedy-reduced and the projection of $\mathbf{b}_d$ onto the linear span of $\mathbf{b}_1, \ldots, \mathbf{b}_{d-1}$ lies in the Voronoï cell $\mathrm{Vor}(\mathbf{b}_1, \ldots, \mathbf{b}_{d-1})$. The greedy algorithm always outputs a greedy-reduced basis and if the input basis is greedy-reduced, then the output basis will be equal to the input basis. The fact that Lagrange's algorithm outputs Minkowski-reduced bases is a particular case of the following result, which compares greedy and Minkowski reductions :

Lemma A.3 *The following statements hold :*

1. *Any Minkowski-reduced basis is greedy-reduced.*
2. *A basis of* $d \leq 4$ *vectors is Minkowski-reduced if and only if it is greedy-reduced.*
3. *If* $d \geq 5$*, there exists a basis of* d *vectors that is greedy-reduced but not Minkowski-reduced.*

Proof. The first statement follows directly from the definitions of the Minkowski and greedy reductions. The second one is obvious if one considers Theorem A.1 : up to dimension four the conditions involve only zeros and ones. It now remains to give a counterexample in dimension five. We consider the lattice spanned by the columns of the following basis :

$$\begin{bmatrix} 2 & 0 & 0 & 0 & 1 \\ 0 & 2 & 0 & 0 & 1 \\ 0 & 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 2+\varepsilon & 1+\frac{\varepsilon}{2} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

where $\varepsilon \in \left(0, -2 + 4\frac{\sqrt{3}}{3}\right)$.

The upper bound on ε implies that $\|\mathbf{b}_1\| \leq \|\mathbf{b}_2\| \leq \|\mathbf{b}_3\| < \|\mathbf{b}_4\| < \|\mathbf{b}_5\|$. The given basis is not Minkowski-reduced because it does not reach the first four minima of the lattice : $2\mathbf{b}_5 - \mathbf{b}_4 - \mathbf{b}_3 - \mathbf{b}_2 - \mathbf{b}_1 =^t (0, 0, 0, 0, 2)$ is linearly independent with the vectors $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$ and is strictly shorter than the vectors $\mathbf{b}_4$ and $\mathbf{b}_5$. However, the basis is greedy-reduced : $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \mathbf{b}_4$ are pairwise orthogonal, and the vector $\mathbf{b}_5$ cannot be longer than any linear integer combination of those four vectors added to $\mathbf{b}_5$. □

As a consequence the greedy algorithm outputs a Minkowski-reduced basis up to dimension four, thus reaching all the successive minima of the lattice. Furthermore, beyond dimension four, the greedy algorithm outputs a greedy-reduced basis that may not be Minkowski-reduced. The following lemma shows that greedy-reduced bases may considerably differ from Minkowski-reduced bases beyond dimension four :

Lemma A.4 *Let* $d \geq 5$*. For any* $\varepsilon > 0$*, there exists a lattice* L *and a greedy-reduced basis* $[\mathbf{b}_1, \ldots, \mathbf{b}_d]_{\leq}$ *of* L *such that* $\frac{\lambda_1(L)}{\|\mathbf{b}_1\|} \leq \varepsilon$ *and* $\frac{\mathrm{vol}L}{\prod_{i=1}^{d} \|\mathbf{b}_i\|} \leq \varepsilon$*.*

Proof. Consider the greedy-reduced basis spanned by the columns of the following matrix :

$$\begin{bmatrix} 2 & 0 & 0 & 0 & 1 \\ 0 & 2 & 0 & 0 & 1 \\ 0 & 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & \varepsilon \end{bmatrix},$$

where $\varepsilon > 0$ is small. Then $2\mathbf{b}_5 - \mathbf{b}_1 - \mathbf{b}_2 - \mathbf{b}_3 - \mathbf{b}_4$ is a lattice vector of length 2ε. Moreover, the vector $\mathbf{b}_1$ is of length 2, which proves the first fact. Finally, we have $\mathrm{vol}L = 16\varepsilon$ and the product of the $\|\mathbf{b}_i\|$'s is larger than 16, which proves the second statement of the lemma. □

Such properties do not hold for Minkowski-reduced bases. The first phenomenon shows that greedy-reduced bases may be arbitrarily far from the first minimum while the second one shows that a greedy-reduced basis may be far from being orthogonal.

A.5 The Closest Vector Problem in Low Dimensions

We now explain how Steps 5 of the recursive greedy algorithm and 2 of the iterative variant can be implemented efficiently up to $d = 5$. Step 5 is trivial only when $d \leq 2$. Otherwise, notice that the $(d-1)$-dimensional basis $[\mathbf{b}_1, \ldots, \mathbf{b}_{d-1}]_{\leq}$ is greedy-reduced, and therefore Minkowski-reduced as long as $d \leq 5$. And we know the Gram matrix of $[\mathbf{b}_1, \ldots, \mathbf{b}_{d-1}, \mathbf{b}_d]_{\leq}$.

Theorem A.6 *Let $d \geq 1$ be an integer. There exists an algorithm that, given as input a Minkowski-reduced basis $[\mathbf{b}_1, \ldots, \mathbf{b}_{d-1}]_{\leq}$, a target vector $\mathbf{t}$ longer than all the $\mathbf{b}_i$'s, together with the Gram matrix of $[\mathbf{b}_1, \ldots, \mathbf{b}_{d-1}, \mathbf{t}]_{\leq}$, outputs a closest lattice vector $\mathbf{c}$ to $\mathbf{t}$ (in the lattice spanned by the vectors $\mathbf{b}_1, \ldots, \mathbf{b}_{d-1}$), and the Gram matrix of $(\mathbf{b}_1, \ldots, \mathbf{b}_{d-1}, \mathbf{t} - \mathbf{c})$, in time :*

$$O\left(\log \|\mathbf{t}\| \cdot [1 + \log \|\mathbf{t}\| - \log \|\mathbf{b}_\alpha\|]\right),$$

where $\alpha \in [\![1, d]\!]$ is any integer such that $[\mathbf{b}_1, \ldots, \mathbf{b}_{\alpha-1}, \mathbf{t}]_{\leq}$ is Minkowski-reduced.

The index α appearing in the statement of the theorem requires an explanation. Consider the iterative version of the greedy algorithm. The index k can increase and decrease rather arbitrarily. For a given loop iteration, the index α tells us which vectors have not changed since the last loop iteration for which the index k had the same value. Intuitively, the reason why we can make the index α appear in the statement of the theorem is that we are working in the orthogonal of the vectors $\mathbf{b}_1, \ldots, \mathbf{b}_{\alpha-1}$. The use of the index α is crucial. If we were using the weaker bound $O(\log \|\mathbf{t}\| \cdot [1 + \log \|\mathbf{t}\| - \log \|\mathbf{b}_1\|])$, we would not be able to prove the quadratic bit complexity of the greedy algorithm. Rather, we would obtain a cubic complexity bound. This index α is also crucial for the quadratic bit complexity of the floating-point LLL described in [264].

Intuitively, the algorithm works as follows : an approximation of the coordinates (with respect to the $\mathbf{b}_i$'s) of the closest vector is computed using linear algebra and the approximation is then corrected by a suitable exhaustive search.

Proof. Let $\mathbf{h}$ be the component of the vector $\mathbf{t}$ that is orthogonal to the linear span of $\mathbf{b}_1, \ldots, \mathbf{b}_{d-1}$. We do not compute $\mathbf{h}$ but introduce it to simplify the description of the algorithm. There exist $y_1, \ldots, y_{d-1} \in \mathbb{R}$ such that $\mathbf{h} = \sum_{i=1}^{d-1} y_i \mathbf{b}_i$. If $\mathbf{c} = \sum_{i=1}^{d-1} x_i \mathbf{b}_i$ is a closest vector to $\mathbf{t}$, then $\mathbf{h} - \mathbf{c}$ belongs to $\mathrm{Vor}(\mathbf{b}_1, \ldots, \mathbf{b}_{d-1})$. However, for any $C > 0$, the coordinates (with respect to any basis of orthogonality-defect $\leq C$) of any point inside the Voronoï cell can be bounded independently from the lattice (see [335]). It follows that if we know an approximation of the y_i's with sufficient precision, then $\mathbf{c}$ can be derived from a $O(1)$ exhaustive search, since the coordinates $y_i - x_i$ of $\mathbf{h} - \mathbf{c}$ are bounded (the orthogonality-defect

of the Minkowski-reduced basis $[\mathbf{b}_1, \ldots, \mathbf{b}_{d-1}]_{\leq}$ is bounded). This exhaustive search can be performed in time $O(\log \|\mathbf{t}\|)$ since the Gram matrix of $[\mathbf{b}_1, \ldots, \mathbf{b}_{d-1}, \mathbf{t}]_{\leq}$ is known.

To approximate the y_i's, we use basic linear algebra. Let $G = G(\mathbf{b}_1, \ldots, \mathbf{b}_{d-1})$ and $H = \left(\frac{\langle \mathbf{b}_i, \mathbf{b}_j \rangle}{\|\mathbf{b}_i\|^2}\right)_{i,j<d}$. The matrix H is exactly G, where the i-th row has been divided by $\|\mathbf{b}_i\|^2$. We have :

$$G \cdot \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = - \begin{bmatrix} \langle \mathbf{b}_1, \mathbf{t} \rangle \\ \vdots \\ \langle \mathbf{b}_n, \mathbf{t} \rangle \end{bmatrix}, \text{ and therefore } \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = -H^{-1} \cdot \begin{bmatrix} \frac{\langle \mathbf{b}_1, \mathbf{t} \rangle}{\|\mathbf{b}_1\|^2} \\ \vdots \\ \frac{\langle \mathbf{b}_n, \mathbf{t} \rangle}{\|\mathbf{b}_n\|^2} \end{bmatrix}.$$

We use the latter formula to compute the y_i's with an absolute error $\leq 1/2$, within the expected time. Let $r = \max_i \lceil \log \frac{|\langle \mathbf{b}_i, \mathbf{t} \rangle|}{\|\mathbf{b}_i\|^2} \rceil$. Notice that $r = O(1 + \log \|\mathbf{t}\| - \log \|\mathbf{b}_{\alpha-1}\|)$, which can be obtained by bounding $\langle \mathbf{b}_i, \mathbf{t} \rangle$ depending on whether $i \geq \alpha$. Notice also that the entries of H are all ≤ 1 in absolute value (because $[\mathbf{b}_1, \ldots, \mathbf{b}_{d-1}]_{\leq}$ is Minkowski-reduced and therefore pairwise Lagrange-reduced), and that $\det(H) = \frac{\det(G)}{\|\mathbf{b}_1\|^2 \ldots \|\mathbf{b}_d\|^2}$ is lower bounded by some universal constant (because the orthogonality-defect of the Minkowski-reduced basis $[\mathbf{b}_1, \ldots, \mathbf{b}_{d-1}]_{\leq}$ is bounded). It follows that one can compute the entries of the matrix H^{-1} with an absolute precision of $\Theta(r)$ bits, within $O(r^2)$ binary operations. One eventually derives the y_i's with an absolute error $\leq 1/2$, by a matrix-vector multiplication. □

It can be proved that this result remains valid when replacing Minkowski reduction by any kind of basis reduction that ensures a bounded orthogonality-defect, for example LLL-reduction. Besides, notice that Theorem A.3 can be used to make Theorem A.6 more practical : the bounds given in Theorem A.3 help decreasing drastically the cost of the exhaustive search following the linear algebra step.

A.6 The Global Approach

In this section we describe the global approach to prove that there is a linear number of loop iterations during the execution of the iterative version of the greedy algorithm (as described in Figure A.4). The goal of this global approach is to prove Theorem A.7. The proof of this last theorem can be replaced by another one that we describe in Sections A.8 and A.9. We call this alternative proof the local approach. In both cases, the complexity analysis of the greedy algorithm finishes with Section A.7. This last section makes use of the local and global approaches only through Theorem A.7.

In the present section, we describe a global analysis proving that the number of loop iterations of the iterative greedy algorithm is at most linear in $\log \|\mathbf{b}_d\|$, as long as $d \leq 5$. More precisely, we show that :

Theorem A.7 *Let $d \leq 5$. Let $\mathbf{b}_1, \ldots, \mathbf{b}_d$ be linearly independent vectors. The number of loop iterations performed during the execution of the greedy algorithm of Figure A.4 given as input $\mathbf{b}_1, \ldots, \mathbf{b}_d$ is bounded by $O(\log \|\mathbf{b}_d\| - \log \lambda_1)$, where $\lambda_1 = \lambda_1(L[\mathbf{b}_1, \ldots, \mathbf{b}_d])$.*

To obtain this result, we show that in any dimension $d \leq 5$, there are at most $O(1)$ consecutive loop iterations of the recursive algorithm described in Figure A.3 without a significant length decrease, i.e., without a decrease of the product of the lengths of the basis vectors by a factor higher than C for some constant $C > 1$. This fact implies that there cannot be more than $O(1)$ consecutive loop iterations of the iterative algorithm without a decrease of the product of the lengths of the basis vectors by a factor higher than C. This immediately implies Theorem A.7.

Our proof is as follows. We first define two different phases in the execution of the recursive d-dimensional greedy algorithm. In the first phase, when a vector is shortened, its length decreases

by at least a factor of $1+\eta$ for some $\eta > 0$ to be fixed later. All these steps are good steps since they make the product of the lengths of the basis vectors decrease significantly. In the second phase, the lengths of the vectors are not decreasing much, but we will show that once we enter this phase, the basis is already almost reduced and there remain very few loop iterations.

A.6.1 Two Phases in the Recursive Greedy Algorithm

We divide the successive loop iterations of the recursive greedy algorithm into two phases : the η-phase and the remaining phase. The execution of the algorithm starts with the η-phase. The loop iterations are in the η-phase as long as the newly created vector $\mathbf{b}'_d$ is far shorter than its original vector $\mathbf{b}_d$. As soon as such a large length decrease is not reached, all the remaining loop iterations are in the remaining phase. More precisely, the η-phase is exactly made of the loop iterations of the η-greedy algorithm of Figure A.5, which simulates the beginning of the execution of the greedy algorithm. The remaining phase corresponds to the execution of the recursive greedy algorithm of Figure A.3 given as input the output basis of the η-greedy algorithm.

Input : A basis $[\mathbf{b}_1, \ldots, \mathbf{b}_d]_{\leq}$ with its Gram matrix.
Output : An ordered basis of $L[\mathbf{b}_1, \ldots, \mathbf{b}_d]$ with its Gram matrix.

1. If $d = 1$, return $\mathbf{b}_1$.
2. Sort $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ by increasing lengths and update the Gram matrix,
3. $[\mathbf{b}_1, \ldots, \mathbf{b}_{d-1}]_{\leq} \longleftarrow \mathsf{Greedy}(\mathbf{b}_1, \ldots, \mathbf{b}_{d-1})$,
4. Compute a vector $\mathbf{c} \in L[\mathbf{b}_1, \ldots, \mathbf{b}_{d-1}]$ closest to $\mathbf{b}_d$,
5. $\mathbf{b}'_d \longleftarrow \mathbf{b}_d - \mathbf{c}$,
6. If $(1+\eta) \cdot \|\mathbf{b}'_d\| \leq \|\mathbf{b}_d\|$, goto Step 8.
7. Else, $\mathbf{b}_d \longleftarrow \mathbf{b}'_d$, update the Gram matrix and goto Step 2.
8. Return $[\mathbf{b}_1, \ldots, \mathbf{b}_d]_{\leq}$ and its Gram matrix.

FIG. A.5 – The η-greedy lattice basis reduction algorithm in dimension d.

It is clear that all loop iterations in the η-phase are good loop iterations : the product of the lengths of the basis vectors decreases by a factor higher than $1+\eta$. Moreover, if $d \leq 4$, when the execution of the algorithm enters the remaining phase, the basis has a bounded orthogonality defect :

Lemma A.5 *Let $d \leq 4$ and $\eta \in \left(0, \sqrt{43} - 1\right)$. Suppose that the basis $[\mathbf{b}_1, \ldots, \mathbf{b}_d]_{\leq}$ is invariant by the η-greedy algorithm of Figure A.5. Then for any $k \leq d$, we have :*

$$\|\mathbf{b}_k^*\|^2 \geq \left(\frac{1}{(1+\eta)^2} + \frac{1-k}{4}\right) \cdot \|\mathbf{b}_k\|^2.$$

Notice that if $k \leq 4$, then $\frac{1}{(1+\eta)^2} + \frac{1-k}{4} > 0$.

Proof. Let $k \leq d$ and $\mathbf{b}'_k = \mathbf{b}_k - \mathbf{c}$ where $\mathbf{c}$ is a vector closest to $\mathbf{b}_k$ in the lattice $L[\mathbf{b}_1, \ldots, \mathbf{b}_{k-1}]$. We write $\mathbf{b}'_k = \mathbf{b}_k^* + \mathbf{b}_k^-$, where $\mathbf{b}_k^-$ is in the linear span of $(\mathbf{b}_1, \ldots, \mathbf{b}_{k-1})$. Since the vector $\mathbf{b}'_k$ cannot be shortened by adding to it an integral linear combination of the vectors $\mathbf{b}_1, \ldots, \mathbf{b}_{k-1}$, we know that :

$$\|\mathbf{b}_k^-\|^2 \leq \rho(\mathbf{b}_1, \ldots, \mathbf{b}_{k-1})^2 \leq \frac{k-1}{4} \max_{i<k} \left(\|\mathbf{b}_i^*\|^2\right) \leq \frac{k-1}{4}\|\mathbf{b}_{k-1}\|^2 \leq \frac{k-1}{4}\|\mathbf{b}_k\|^2.$$

From the Pythagorean theorem, we derive that :

$$\begin{aligned} \|\mathbf{b}_k\|^2 \leq (1+\eta)^2 \cdot \|\mathbf{b}'_k\|^2 &\leq (1+\eta)^2 \cdot (\|\mathbf{b}_k^*\|^2 + \|\mathbf{b}_k^-\|^2) \\ &\leq (1+\eta)^2 \cdot \left(\|\mathbf{b}_k^*\|^2 + \frac{k-1}{4}\|\mathbf{b}_k\|^2\right), \end{aligned}$$

which gives the result. □

A.6.2 The Greedy Algorithm with a Rather Orthogonal Basis

To conclude in dimension below four, it now suffices to prove that when the $\|\mathbf{b}_i^*\|/\|\mathbf{b}_i\|$'s are all lower-bounded (i.e., the orthogonality defect is bounded), then the number of loop iterations of the greedy algorithm is $O(1)$.

Lemma A.6 *Let $d \leq 5$ and $C > 0$. There exists a constant K such that for any basis $[\mathbf{b}_1, \ldots, \mathbf{b}_d]_{\leq}$ satisfying $\prod_{i=1}^{d} \frac{\|\mathbf{b}_i^*\|}{\|\mathbf{b}_i\|} \geq C$, the recursive greedy algorithm, when given as input $\mathbf{b}_1, \ldots, \mathbf{b}_d$, terminates in at most K loop iterations.*

Proof. Since for any $i \leq d$, we have $\|\mathbf{b}_i^*\| \leq \|\mathbf{b}_i\|$, we also have $\frac{\|\mathbf{b}_i^*\|}{\|\mathbf{b}_i\|} \geq C$ for any $i \leq d$. As a consequence, if the initial basis satisfies $\prod_{i=1}^{d} \frac{\|\mathbf{b}_i^*\|}{\|\mathbf{b}_i\|} \geq C$, since the numerator is constant (it is the determinant of the lattice) and the denominator decreases, then all the bases appearing in the execution of the greedy algorithm satisfy this condition. Any vector $\mathbf{b}_i$ appearing during the execution of the algorithm satisfies $\frac{\|\mathbf{b}_i^*\|}{\|\mathbf{b}_i\|} \geq C$.

We define $X_i = \{(x_i, \ldots, x_d), \exists (x_1, \ldots, x_{i-1}) \in \mathbb{Z}^{i-1}, \|\sum_j x_j \mathbf{b}_j\| \leq \|\mathbf{b}_i\|\}$ for $i \leq d$. We prove that $|X_i| \leq (1+2/C)^{d-i+1}$ by decreasing induction on i. Let $\mathbf{b} = x_1\mathbf{b}_1 + \ldots + x_d\mathbf{b}_d$ with $\|\mathbf{b}\| \leq \|\mathbf{b}_d\|$. By considering the component of the vector $\mathbf{b}$ on $\mathbf{b}_d^*$, we have that $|x_d| \leq 1/C$. Suppose now that we want to prove the fact for some $i < d$. Let $\mathbf{b} = x_1\mathbf{b}_1 + \ldots + x_d\mathbf{b}_d$ with $\|\mathbf{b}\| \leq \|\mathbf{b}_i\|$. Since the basis is ordered, we have $\|\mathbf{b}\| \leq \|\mathbf{b}_{i+1}\|$, which gives, by using the induction hypothesis, that $(x_{i+1}, \ldots, x_d)$ belongs to a finite set. Moreover, by taking the component of $\mathbf{b}$ on $\mathbf{b}_i^*$, we obtain that :

$$\left| x_i + \sum_{j=i+1}^{d} x_j \frac{\langle \mathbf{b}_j, \mathbf{b}_i^* \rangle}{\|\mathbf{b}_i^*\|^2} \right| \leq 1/C.$$

This gives that for any choice of $(x_{i+1}, \ldots, x_d)$, there are at most $2/C + 1$ possible values for x_i.

Consider now the execution of the greedy algorithm on such a basis : the number of times a vector shorter than $\mathbf{b}_1$ is created is bounded by $|X_1| \leq (1+2/C)^d$. Therefore we can subdivide the execution of the algorithm into phases in which $\mathbf{b}_1$ remains constant. Consider such a phase. Let $\mathbf{b}_1, \ldots, \mathbf{b}_d$ be the initial basis of this phase. It satisfies the condition $\prod_{i=1}^{d} \frac{\|\mathbf{b}_i^*\|}{\|\mathbf{b}_i\|} \geq C$. At most $|X_2| \leq (1+2/C)^{d-1}$ times in this phase a vector shorter than $\mathbf{b}_2$ can be created : for any $(x_2, \ldots, x_d) \in X_2$, there are at most two possibilities for x_1, because of the greedy choice in Steps 2 of the iterative greedy algorithm and 5 of the recursive greedy algorithm. This shows that we can subdivide the execution of the algorithm into $\leq 2(1+2/C)^{2d-1}$ phases in which both $\mathbf{b}_1$ and $\mathbf{b}_2$ are constant. By using the bound on $|X_i|$ and the finiteness of the number of solutions for a closest vector problem instantiation (this is the so-called kissing number, see [68]), this reasoning can be extended to subdivide the execution of the algorithm into phases in which $\mathbf{b}_1, \ldots, \mathbf{b}_i$ do not change, and we can bound the number of phases independently of the basis. The case $i = d$ gives the result. □

A.7 Quadratic Bit Complexity

In this subsection we use Theorems A.6 and A.7 of the two previous sections to prove the quadratic bit complexity claimed in Theorem A.5. To do this, we generalise the cancellation phenomenon used in the analysis of Lagrange's algorithm in Section A.3.

Suppose that $d \leq 5$. We consider the iterative version of the d-dimensional greedy algorithm of Figure A.4 and denote by $[\mathbf{b}_1^{(t)}, \ldots, \mathbf{b}_d^{(t)}]_{\leq}$ the current ordered basis at the beginning of the t-th loop iteration. Initially we have : $[\mathbf{b}_1^{(t)}, \ldots, \mathbf{b}_d^{(t)}]_{\leq} = [\mathbf{b}_1, \ldots, \mathbf{b}_d]_{\leq}$. Theorem A.6 gives that the cost of the t-th loop iteration is bounded by $O\left(\log \|\mathbf{b}_d\| \cdot \left(1 + \log \left\|\mathbf{b}_{\kappa(t)}^{(t)}\right\| - \log \left\|\mathbf{b}_{\alpha(t)}^{(t)}\right\|\right)\right)$. Theorem A.7 gives

that the number of loop iterations τ is bounded by $O(\log \|\mathbf{b}_d\|)$. We have two indices of interest for the cost analysis : $k(t)$ because at the t-th loop iteration, we are trying to decrease the length of $\mathbf{b}_{k(t)}^{(t)}$, and $\alpha(t)$ that we define precisely in the following lemma and that corresponds to the largest i such that none of $\mathbf{b}_1, \ldots, \mathbf{b}_i$ has been modified since the last time the index k had value $k(t)$.

Lemma A.7 *Let t be a loop iteration. Let $\varphi(t) = \max(t' < t, k(t') \geq k(t))$ if it exists and 1 otherwise, and $\alpha(t) = \min(k(t'), t' \in [\![\varphi(t), t-1]\!]) - 1$. The cost of the t-th loop iteration is bounded by :*

$$O\left(\log \|\mathbf{b}_d\| \cdot \left[1 + \log \left\|\mathbf{b}_{k(t)}^{(t)}\right\| - \log \left\|\mathbf{b}_{\alpha(t)}^{(t)}\right\|\right]\right).$$

Proof. Between loop iterations $\varphi(t)$ and t, the vectors $\mathbf{b}_1, \ldots, \mathbf{b}_{\alpha(t)-1}$ do not change and because of the greedy choices of the successive Steps 2 and 3, each vector $\mathbf{b}$ created during these loop iterations is such that the basis $[\mathbf{b}_1, \ldots, \mathbf{b}_{\alpha(t)-1}, \mathbf{b}]_{\leq}$ is greedy-reduced, and therefore Minkowski-reduced if $\alpha(t) \leq 4$ (because of the equivalence of greedy and Minkowski reductions up to dimension four). This includes the vector $\mathbf{b}_{\kappa(t)}^{(t)}$. Theorem A.6 gives the result because all vectors appearing during the execution of the algorithm are shorter than $\mathbf{b}_d$. □

We are to subdivide the sum of the costs of the successive loop iterations into $O(d)$ subsums according to the value of $k(t)$:

$$\sum_{t \leq \tau} \left[1 + \log \left\|\mathbf{b}_{k(t)}^{(t)}\right\| - \log \left\|\mathbf{b}_{\alpha(t)}^{(t)}\right\|\right] \leq \tau + \sum_{k=2}^{d} \sum_{t, k(t)=k} \left(\log \left\|\mathbf{b}_{k}^{(t)}\right\| - \log \left\|\mathbf{b}_{a(t)}^{(t)}\right\|\right).$$

For each of these subsums, we keep $k-1$ positive terms and $k-1$ negative terms, and make the others vanish in a progressive cancellation. The crucial point to do this is the following :

Lemma A.8 *Let $k \in [\![2, d]\!]$ and $t_1 < t_2 < \ldots < t_k$ be loop iterations of the iterative greedy algorithm such that for any $j < k$, we have $k(t_j) = k$. Then there exists $j < k$ with $\left\|\mathbf{b}_{\alpha(t_j)}^{(t_j)}\right\| \geq \left\|\mathbf{b}_{k}^{(t_k)}\right\|$.*

Proof. We choose $j = \max(i \leq k, \alpha(t_i) \geq i)$. This definition is valid because the maximised is not empty (it contains 1). Since $\alpha(t_k) < k$ and $k(t_k) = k(t_{k-1}) = k$, there exists a first loop iteration $T_k \in [\![t_{k-1}, t_k - 1]\!]$ such that $k(T_k) \geq k \geq k(T_k + 1)$. Because for a given index the lengths of the vectors are always decreasing, we have :

$$\left\|\mathbf{b}_{k}^{(t_k)}\right\| \leq \left\|\mathbf{b}_{k}^{(T_k+1)}\right\| \leq \left\|\mathbf{b}_{k-1}^{(T_k)}\right\|.$$

By definition of T_k the vectors $\mathbf{b}_1, \ldots, \mathbf{b}_{k-1}$ do not change between loop iterations t_{k-1} and T_k. Therefore :

$$\left\|\mathbf{b}_{k}^{(t_k)}\right\| \leq \left\|\mathbf{b}_{k-1}^{(t_{k-1})}\right\|.$$

If $j = k-1$, we have the result. Otherwise there exists a first loop iteration $T_{k-1} \in [\![t_{k-2}, t_{k-1} - 1]\!]$ such that $k(T_{k-1}) \geq k-1 \geq k(T_{k-1} + 1)$. We have :

$$\left\|\mathbf{b}_{k-1}^{(t_{k-1})}\right\| \leq \left\|\mathbf{b}_{k-1}^{(T_{k-1}+1)}\right\| \leq \left\|\mathbf{b}_{k-2}^{(T_{k-1})}\right\| \leq \left\|\mathbf{b}_{k-2}^{(t_{k-2})}\right\|.$$

If $j = k-2$ we have the result, otherwise we go on constructing such loop iterations T_i's to obtain the result. □

We can now finish the complexity analysis. Let $k \in [\![2, d]\!]$ and $t_1 < t_2 < \ldots < t_{\tau_k} = \{t \leq \tau, k(t) = k\}$. We have :

$$\sum_{i=1}^{\tau_k} \left(\log \left\|\mathbf{b}_{k}^{(t_i)}\right\| - \log \left\|\mathbf{b}_{\alpha(t_i)}^{(t_i)}\right\|\right) \leq k\left(\log \|\mathbf{b}_d\| - \log \lambda_1\right) + \sum_{i=k}^{\tau_k} \log \left\|\mathbf{b}_{k}^{(t_i)}\right\| - \sum_{i=1}^{\tau_k - k + 1} \log \left\|\mathbf{b}_{\alpha(t_i)}^{(t_i)}\right\|,$$

where λ_1 is the first minimum of the lattice we are reducing. Lemma A.8 helps bounding the right hand-side of the above bound. First, we apply it with $t_1, \ldots, t_k$. Thus there exists $j < k$ such that $\|\mathbf{b}_k^{(t_k)}\| \leq \|\mathbf{b}_{\alpha(t_j)}^{(t_j)}\|$. The indices "$i = k$" in the positive sum and "$i = j$" in the negative sum cancel out. Then we apply Lemma A.8 to t_{k+1} and the $k-1$ first t_i's that remain in the negative sum. It is easy to see that t_{k+1} is larger than any of them, so that we can have another "positive-negative" pair that cancels out. We perform this operation $\tau_k - k + 1$ times, to obtain :

$$\sum_{i=k}^{\tau_k} \log \left\| \mathbf{b}_k^{(t_i)} \right\| - \sum_{i=1}^{\tau_k - k + 1} \log \left\| \mathbf{b}_{\alpha(t_i)}^{(t_i)} \right\| \leq 0.$$

The fact that $\sum_k \tau_k = \tau = O\left(\log \|\mathbf{b}_d\|\right)$ completes the proof of Theorem A.5.

A.8 The Local Approach

This section and the following give another proof for Theorem A.7. They give a more precise comprehension of the behaviour of the algorithm but may be skipped since they are not necessary to prove the main results of the paper.

In this section we give an alternative proof of Theorem A.7 for $d \leq 4$, that generalises the local analysis of Lagrange's algorithm. We prove that the number of loop iterations of the iterative greedy algorithm of Figure A.4 is $O(\log \|\mathbf{b}_d\|)$ by showing that there cannot be more than $O(1)$ consecutive loop iterations of the recursive greedy algorithm of Figure A.3 without a significant length decrease. More precisely, we show the following theorem, from which Theorem A.7 can be easily derived.

Theorem A.8 *Let $d \leq 4$. There exist three constants $C > 1, I, F$ such that in any d consecutive loop iterations of the d-dimensional recursive greedy algorithm of Figure A.3, some of the iterations are in the I initial loop iterations or in the F final loop iterations, or the product of the lengths of the current basis vectors decreases by at least a factor C.*

This result does imply Theorem A.7 for $d \leq 4$. Indeed, the maximum number of successive loop iterations (of the iterative algorithm) without a significant length decrease is bounded by $(I + F + C)^d = O(1)$. From now on we only refer to the recursive greedy algorithm of Figure A.3.

A.8.1 A Unified Geometric Analysis Up To Dimension Four

The local analysis of Lagrange's algorithm (Section A.3) was based on the fact that if $|x| \geq 2$, the vector $x\mathbf{u}$ is far from the Voronoï cell of the lattice spanned by $\mathbf{u}$. The analysis of the number of loop iterations of the greedy algorithm in dimensions three and four relies on a similar phenomenon in dimensions two and three. However, the situation is more complex, as the following basic remarks hint :

- For $d = 2$, we considered the value of x, but if $d \geq 3$, there will be several coefficients x_i instead of a single one, and it is not clear which one will be useful in the analysis.
- For $d = 2$, Step 4 cannot change the basis, as there are only two bases in dimension one. If $d \geq 3$, Step 4 may completely change the vectors, and it could be hard to keep track of what is going on.

To prove Theorem A.8, we introduce a few notations. Consider the i-th loop iteration. Denote by $\left[\mathbf{a}_1^{(i)}, \ldots, \mathbf{a}_d^{(i)}\right]_{\leq}$ the basis $[\mathbf{b}_1, \ldots, \mathbf{b}_d]_{\leq}$ at the beginning of the i-th loop iteration. The basis $\left[\mathbf{a}_1^{(i)}, \ldots, \mathbf{a}_d^{(i)}\right]_{\leq}$ becomes $\left[\mathbf{b}_1^{(i)}, \ldots, \mathbf{b}_{d-1}^{(i)}, \mathbf{a}_d^{(i)}\right]_{\leq}$ with $\left\|\mathbf{b}_1^{(i)}\right\| \leq \cdots \leq \left\|\mathbf{b}_{d-1}^{(i)}\right\|$ after Step 4, and $\left(\mathbf{b}_1^{(i)}, \ldots, \mathbf{b}_d^{(i)}\right)$ after Step 6, where $\mathbf{b}_d^{(i)} = \mathbf{a}_d^{(i)} - \mathbf{c}^{(i)}$ and $\mathbf{c}^{(i)}$ is the closest vector found at

Step 5. Let p_i be the number of integers $1 \leq j \leq d$ such that $\left\|\mathbf{b}_j^{(i)}\right\| \leq \left\|\mathbf{b}_d^{(i)}\right\|$. Let π_i be the rank of $\mathbf{b}_d^{(i)}$ once $\left(\mathbf{b}_1^{(i)}, \ldots, \mathbf{b}_d^{(i)}\right)$ is sorted by length : for example, we have $\pi_i = 1$ if $\left\|\mathbf{b}_d^{(i)}\right\| < \left\|\mathbf{b}_1^{(i)}\right\|$. Notice that π_i may not be equal to p_i because there may be several choices when sorting the vectors by length in case of length equalities. Clearly $1 \leq \pi_i \leq p_i \leq d$, if $p_i = d$ then the loop terminates, and $\left\|\mathbf{a}_{\pi_i}^{(i+1)}\right\| = \left\|\mathbf{a}_{p_i}^{(i+1)}\right\|$.

Now consider the $(i+1)$-th loop iteration for some $i \geq 1$. Notice that by definition of π_i, we have $\mathbf{a}_{\pi_i}^{(i+1)} = \mathbf{b}_d^{(i)} = \mathbf{a}_d^{(i)} - \mathbf{c}^{(i)}$, while $\left\{\mathbf{a}_j^{(i+1)}\right\}_{j \neq \pi_i} = \left\{\mathbf{b}_j^{(i)}\right\}_{j<d}$. The vector $\mathbf{c}^{(i+1)}$ belongs to $L\left[\mathbf{b}_1^{(i+1)}, \ldots, \mathbf{b}_{d-1}^{(i+1)}\right] = L\left[\mathbf{a}_1^{(i+1)}, \ldots, \mathbf{a}_{d-1}^{(i+1)}\right]$: there exist integers $x_1^{(i+1)}, \ldots, x_{d-1}^{(i+1)}$ such that $\mathbf{c}^{(i+1)} = \sum_{j=1}^{d-1} x_j^{(i+1)} \mathbf{a}_j^{(i+1)}$.

We are to prove that there exists a universal constant $K > 1$ such that for any execution of the d-dimensional greedy algorithm with $d \leq 4$, in any d consecutive iterations of the loop (except eventually the first ones and the last ones), the product of the lengths of the current basis vectors decreases by some factor higher than K :

$$\frac{\left\|\mathbf{a}_1^{(i)}\right\| \ldots \left\|\mathbf{a}_d^{(i)}\right\|}{\left\|\mathbf{a}_1^{(i+d)}\right\| \ldots \left\|\mathbf{a}_d^{(i+d)}\right\|} \geq K \tag{A.1}$$

This will automatically ensure that the number of loop iterations is at most proportional to $\log \left\|\mathbf{a}_d^{(1)}\right\|$.

We deal with the first difficulty mentioned above : which one will be the useful coefficient ? The trick is to consider the value of $x_{\pi_i}^{(i+1)}$, i.e., the coefficient of $\mathbf{a}_{\pi_i}^{(i+1)} = \mathbf{a}_d^{(i)} - \mathbf{c}^{(i)}$ in $\mathbf{c}^{(i+1)}$, and to use the greedy properties of the algorithm. This coefficient corresponds to the vector that has been created at the previous loop iteration. Since this vector has been created so that it cannot be shortened by adding to it a combination of the others, there are only two possibilities at the current iteration : either the new vector is longer than $\mathbf{a}_{\pi_i}^{(i+1)}$, in which case p_i increases (this can happen at most d times in a row), either it is shorter et we must have $|x_{\pi_i}| \neq 1$.

Lemma A.9 *Among d consecutive iterations of the loop of the greedy algorithm of Figure A.3, there is at least one iteration of index $i+1$ such that $p_{i+1} \leq p_i$. Moreover, for such a loop iteration, we have $\left|x_{\pi_i}^{(i+1)}\right| \geq 2$, or this is the last loop iteration.*

Proof. The first statement is obvious. Consider one such loop iteration $i+1$. Suppose we have a small $|x_{\pi_i}^{(i+1)}|$, that is $x_{\pi_i}^{(i+1)} = 0$ or $\left|x_{\pi_i}^{(i+1)}\right| = 1$.

- If $x_{\pi_i}^{(i+1)} = 0$, then $\mathbf{c}^{(i+1)} \in L\left[\mathbf{a}_j^{(i+1)}\right]_{j \neq \pi_i, j<d} = L\left[\mathbf{b}_1^{(i)}, \ldots, \mathbf{b}_{d-2}^{(i)}\right]$. We claim that the $(i+1)$-th iteration must be the last one. Since the i-th loop iteration was not terminal, we have $\mathbf{a}_d^{(i+1)} = \mathbf{b}_{d-1}^{(i)}$. Moreover, the basis $\left[\mathbf{b}_1^{(i)}, \ldots, \mathbf{b}_{d-1}^{(i)}\right]_{\leq}$ is greedy-reduced because of Step 4 of the i-th loop iteration. These two facts imply that $\mathbf{c}^{(i+1)}$ must be zero (or at least it does not make the length of $\mathbf{a}_d^{(i)}$ decrease if there are several closest lattice vectors), and the $(i+1)$-th loop iteration is the last one.
- If $|x_{\pi_i}^{(i+1)}| = 1$, we claim that $p_{i+1} > p_i$. We have $\mathbf{c}^{(i+1)} = \sum_{j=1}^{d-1} x_j^{(i+1)} \mathbf{a}_j^{(i+1)}$ where $\mathbf{a}_{\pi_i}^{(i+1)} = \mathbf{a}_d^{(i)} - \mathbf{c}^{(i)}$ and $\left\{\mathbf{a}_j^{(i+1)}\right\}_{j \neq \pi_i} = \left\{\mathbf{b}_j^{(i)}\right\}_{j<d}$. Thus, the vector $\mathbf{c}^{(i+1)}$ can be written as $\mathbf{c}^{(i+1)} = \mp\left(\mathbf{a}_d^{(i)} - \mathbf{c}^{(i)}\right) - \mathbf{e}$ where $\mathbf{e} \in L\left[\mathbf{b}_1^{(i)}, \ldots, \mathbf{b}_{d-1}^{(i)}\right]$. Therefore $\mathbf{a}_d^{(i+1)} - \mathbf{c}^{(i+1)} = \mathbf{b}_{d-1}^{(i)} \pm \left(\mathbf{a}_d^{(i)} - \mathbf{c}^{(i)}\right) +$

$\mathbf{e}$. In other words, we have $\left\|\mathbf{a}_d^{(i+1)} - \mathbf{c}^{(i+1)}\right\| = \left\|\mathbf{a}_d^{(i)} - \mathbf{f}\right\|$ for some $\mathbf{f} \in L\left[\mathbf{b}_1^{(i)}, \ldots, \mathbf{b}_{d-1}^{(i)}\right]$. The greedy choice of $\mathbf{b}_d^{(i)}$ at the i-th loop iteration implies that $p_{i+1} \geq 1 + p_i$, which completes the proof of the claim.

□

We will see that in dimension three, any such loop iteration $i+1$ implies that at least one of the basis vectors significantly decreases in the $(i+1)$-th loop iteration, or had significantly decreased in the i-th loop iteration. This is only "almost" true in dimension four : fortunately, we will be able to isolate the bad cases and to show that when a bad case occurs, the number of remaining loop iterations can be bounded by some constant.

We now deal with the second difficulty mentioned above, that is the possible change of the vectors during the recursive call in dimension $d-1$. Recall that $\mathbf{c}^{(i+1)} = \sum_{j=1}^{d-1} x_j^{(i+1)} \mathbf{a}_j^{(i+1)}$ but the basis $\left[\mathbf{a}_1^{(i+1)}, \ldots, \mathbf{a}_{d-1}^{(i+1)}\right]_{\leq}$ is not necessarily greedy-reduced. We distinguish two cases :

1. The basis $\left[\mathbf{a}_1^{(i+1)}, \ldots, \mathbf{a}_{d-1}^{(i+1)}\right]_{\leq}$ is somehow far from being greedy-reduced. Then the vector $\mathbf{b}_d^{(i)}$ was significantly shorter than the replaced vector $\mathbf{a}_d^{(i)}$. Notice that this length decrease concerns the i-th loop iteration and not the $(i+1)$-th.
2. Otherwise, the basis $\left[\mathbf{a}_1^{(i+1)}, \ldots, \mathbf{a}_{d-1}^{(i+1)}\right]_{\leq}$ is almost greedy-reduced. The fact that $\left|x_{\pi_i}^{(i+1)}\right| \geq 2$ roughly implies that $\mathbf{c}^{(i+1)}$ is somewhat far away from the Voronoï cell $\mathrm{Vor}\left(\mathbf{a}_1^{(i+1)}, \ldots, \mathbf{a}_{d-1}^{(i+1)}\right)$: this phenomenon will be precisely captured by the so-called Gap Lemma. When this is the case, the new vector $\mathbf{b}_d^{(i+1)}$ is significantly shorter than $\mathbf{a}_d^{(i+1)}$.

To capture the property that a set of vectors is almost greedy-reduced, we introduce the so-called *ε-greedy-reduction*, which is defined as follows :

Definition A.2 *Let $\varepsilon \geq 0$. A single vector $[\mathbf{b}_1]$ is always ε-greedy-reduced; for $d \geq 2$, we say that a d-tuple $[\mathbf{b}_1, \ldots, \mathbf{b}_d]_{\leq}$ is ε-greedy-reduced if $[\mathbf{b}_1, \ldots, \mathbf{b}_{d-1}]_{\leq}$ is ε-greedy-reduced and the orthogonal projection of the vector $\mathbf{b}_d$ onto the span of $[\mathbf{b}_1, \ldots, \mathbf{b}_{d-1}]_{\leq}$ belongs to $(1+\varepsilon) \cdot \mathrm{Vor}(\mathbf{b}_1, \ldots, \mathbf{b}_{d-1})$.*

With this definition, a greedy-reduced basis is ε-greedy-reduced for any $\varepsilon \geq 0$. In the definition of ε-greedy-reduction, we did not assume that the $\mathbf{b}_i$'s were nonzero nor linearly independent. This is because the Gap Lemma is essentially based on compactness properties : the set of ε-greedy-reduced d-tuples needs being closed (from a topological point of view), while a limit of bases may not be a basis.

We can now give the precise statements of the two cases described above. Lemma A.10 corresponds to case 1, and Lemma A.11 to case 2.

Lemma A.10 *Let $2 \leq d \leq 4$. There exists a constant $\varepsilon_1 > 0$ such that for any $\varepsilon \in (0, \varepsilon_1]$ there exists $C_\varepsilon > 1$ such that the following statement holds. Consider the $(i+1)$-th loop iteration of an execution of the d-dimensional greedy algorithm. If $\left[\mathbf{a}_1^{(i+1)}, \ldots, \mathbf{a}_{d-1}^{(i+1)}\right]_{\leq}$ is not ε-greedy-reduced, then $\left\|\mathbf{a}_d^{(i)}\right\| \geq C_\varepsilon \left\|\mathbf{b}_d^{(i)}\right\|$.*

Proof. The statement is obvious for $d = 2$ since a single vector is always ε-greedy-reduced. Suppose that $d = 3$ and that $\left[\mathbf{a}_1^{(i+1)}, \mathbf{a}_2^{(i+1)}\right]_{\leq}$ is not ε-greedy-reduced. We have $\left|\langle \mathbf{a}_2^{(i+1)}, \mathbf{a}_1^{(i+1)} \rangle\right| \geq \frac{1+\varepsilon}{2} \left\|\mathbf{a}_1^{(i+1)}\right\|^2$. Along with this, we must have $\pi_i = 1$ (otherwise $\left[\mathbf{a}_1^{(i+1)}, \mathbf{a}_2^{(i+1)}\right]_{\leq} = \left[\mathbf{b}_1^{(i)}, \mathbf{b}_2^{(i)}\right]_{\leq}$ would be Minkowski-reduced), which implies that the vector $\mathbf{a}_1^{(i+1)} = \mathbf{b}_3^{(i)}$ cannot be shortened by

adding to it multiples of $\mathbf{a}_2^{(i+1)} = \mathbf{b}_1^{(i)}$. We thus have $\left|\langle \mathbf{a}_1^{(i+1)}, \mathbf{a}_2^{(i+1)} \rangle\right| \leq \left\|\mathbf{a}_2^{(i+1)}\right\|^2/2$. These two inequalities give :

$$(1+\varepsilon) \cdot \left\|\mathbf{a}_1^{(i+1)}\right\|^2 \leq \left\|\mathbf{a}_2^{(i+1)}\right\|^2.$$

The facts that $\mathbf{a}_1^{(i+1)} = \mathbf{b}_3^{(i)}$ and that $\left\|\mathbf{a}_2^{(i+1)}\right\| \leq \left\|\mathbf{a}_3^{(i)}\right\|$ end the proof.

Suppose now that $d = 4$ and that $\left[\mathbf{a}_1^{(i+1)}, \mathbf{a}_2^{(i+1)}, \mathbf{a}_3^{(i+1)}\right]_{\leq}$ is not ε-greedy-reduced. Therefore the orthogonal projection of the vector $\mathbf{a}_2^{(i+1)}$ onto the linear span of $\mathbf{a}_1^{(i+1)}$ is not in $(1+\varepsilon) \cdot \mathrm{Vor}\left(\mathbf{a}_1^{(i+1)}\right)$, or the orthogonal projection of the vector $\mathbf{a}_3^{(i+1)}$ onto the linear span of $\left[\mathbf{a}_1^{(i+1)}, \mathbf{a}_2^{(i+1)}\right]$ is not in $(1+\varepsilon) \cdot \mathrm{Vor}\left(\mathbf{a}_1^{(i+1)}, \mathbf{a}_2^{(i+1)}\right)$. We consider these two cases separately. Suppose first that the orthogonal projection of the vector $\mathbf{a}_2^{(i+1)}$ onto the linear span of $\mathbf{a}_1^{(i+1)}$ is not in $\mathrm{Vor}\left(\mathbf{a}_1^{(i+1)}\right)$. Then $\left|\langle \mathbf{a}_1^{(i+1)}, \mathbf{a}_2^{(i+1)} \rangle\right| \geq \frac{1+\varepsilon}{2}\left\|\mathbf{a}_1^{(i+1)}\right\|^2$. Moreover, we must have $\pi_i = 1$ (otherwise $\left[\mathbf{a}_1^{(i+1)}, \mathbf{a}_2^{(i+1)}\right]_{\leq} = \left[\mathbf{b}_1^{(i)}, \mathbf{b}_2^{(i)}\right]_{\leq}$ is Minkowski-reduced), therefore the vector $\mathbf{a}_1^{(i+1)}$ cannot be shortened by adding to it multiples of the vector $\mathbf{a}_2^{(i+1)}$, which gives that $\left|\langle \mathbf{a}_1^{(i+1)}, \mathbf{a}_2^{(i+1)} \rangle\right| \leq \left\|\mathbf{a}_2^{(i+1)}\right\|^2/2$. Then :

$$(1+\varepsilon) \cdot \left\|\mathbf{b}_4^{(i)}\right\|^2 = (1+\varepsilon) \cdot \left\|\mathbf{a}_1^{(i+1)}\right\|^2 \leq \left\|\mathbf{a}_2^{(i+1)}\right\|^2 \leq \left\|\mathbf{a}_4^{(i)}\right\|^2.$$

We suppose now that the vectors $\left[\mathbf{a}_1^{(i+1)}, \mathbf{a}_2^{(i+1)}\right]_{\leq}$ are ε-greedy-reduced and that the orthogonal projection of the vector $\mathbf{a}_3^{(i+1)}$ onto the linear span of $\left[\mathbf{a}_1^{(i+1)}, \mathbf{a}_2^{(i+1)}\right]_{\leq}$ belongs to $(1+\varepsilon) \cdot \mathrm{Vor}\left(\mathbf{a}_1^{(i+1)}, \mathbf{a}_2^{(i+1)}\right)$. We distinguish two subcases : $\pi_i = 1$ and $\pi_i = 2$. Suppose first that $\pi_i = 2$. In this case $\left[\mathbf{a}_1^{(i+1)}, \mathbf{a}_2^{(i+1)}\right]_{\leq}$ is Minkowski-reduced and the possible Voronoï coordinates of $L\left[\mathbf{a}_1^{(i+1)}, \mathbf{a}_2^{(i+1)}\right]$ are the pairs $(x_1, x_2) \in \{-1, 0, 1\}^2$ (see Lemma A.15). Thus a vector $\mathbf{u}$ has its orthogonal projection onto the span of $\left[\mathbf{a}_1^{(i+1)}, \mathbf{a}_2^{(i+1)}\right]_{\leq}$ in $\mathrm{Vor}\left(\mathbf{a}_1^{(i+1)}, \mathbf{a}_2^{(i+1)}\right)$ if and only if :

$$\forall (x_1, x_2) \in \{-1, 0, 1\}^2,\ \|\mathbf{u}\| \leq \left\|\mathbf{u} + x_1\mathbf{a}_1^{(i+1)} + x_2\mathbf{a}_2^{(i+1)}\right\|.$$

This implies that there exists a pair $(x_1, x_2) \in \{-1, 0, 1\}^2$ such that $\left|\langle \mathbf{a}_3^{(i+1)}, x_1\mathbf{a}_1^{(i+1)} + x_2\mathbf{a}_2^{(i+1)} \rangle\right| \geq \frac{1+\varepsilon}{2}\left\|x_1\mathbf{a}_1^{(i+1)} + x_2\mathbf{a}_2^{(i+1)}\right\|^2$. In the case where $\pi_i = 1$, we can suppose that $\left\|\mathbf{a}_1^{(i+1)}\right\| \geq (1-\varepsilon) \cdot \left\|\mathbf{a}_2^{(i+1)}\right\|$, since otherwise $\left\|\mathbf{b}_4^{(i)}\right\| = \left\|\mathbf{a}_1^{(i+1)}\right\| \leq (1-\varepsilon) \cdot \left\|\mathbf{a}_4^{(i)}\right\|$. We will see in Subsection A.9.2 (Lemma A.22) that for a small enough $\varepsilon > 0$, the possible Voronoï coords of such an ε-greedy-reduced basis are the same as for a Minkowski-reduced basis. Therefore, as in the previous subcase, there exists a pair $(x_1, x_2) \in \{-1, 0, 1\}^2$ such that $\left|\langle \mathbf{a}_3^{(i+1)}, x_1\mathbf{a}_1^{(i+1)} + x_2\mathbf{a}_2^{(i+1)} \rangle\right| \geq \frac{1+\varepsilon}{2}\left\|x_1\mathbf{a}_1^{(i+1)} + x_2\mathbf{a}_2^{(i+1)}\right\|^2$. We now consider the two subcases simultaneously. Suppose first that $x_2 = 0$, then necessarily $|x_1| = 1$. As in the case $d = 3$, the fact that the vector $\mathbf{a}_1^{(i+1)}$ cannot be shortened by adding to it multiples of the vector $\mathbf{a}_3^{(i+1)}$ gives the result (this is obvious if $\pi_i = 1$ and, if $\pi_i = 2$, the basis $\left[\mathbf{a}_1^{(i+1)}, \mathbf{a}_3^{(i+1)}\right]_{\leq} = \left[\mathbf{b}_1^{(i)}, \mathbf{b}_2^{(i)}\right]_{\leq}$ is Minkowski-reduced). The case $x_1 = 0$ can be dealt in the same way. Therefore, it remains to consider the case $|x_1| = |x_2| = 1$. Wlog we suppose $x_1 = x_2 = 1$. We have :

$$\left|\langle \mathbf{a}_3^{(i+1)}, \mathbf{a}_1^{(i+1)} + \mathbf{a}_2^{(i+1)} \rangle\right| \geq \frac{1+\varepsilon}{2}\left\|\mathbf{a}_1^{(i+1)} + \mathbf{a}_2^{(i+1)}\right\|^2.$$

Since the vector $\mathbf{a}_{\pi_i}^{(i+1)}$ cannot be shortened by adding to it integer linear combinations of the two other vectors, we have $\left\|\mathbf{a}_3^{(i+1)} \pm \left(\mathbf{a}_1^{(i+1)} + \mathbf{a}_2^{(i+1)}\right)\right\| \geq \left\|\mathbf{a}_{\pi_i}^{(i+1)}\right\| = \left\|\mathbf{b}_4^{(i)}\right\|$. By considering the right choice for the "plus or minus" and using the fact that $\left\|\mathbf{a}_3^{i+1}\right\| \leq \left\|\mathbf{a}_4^{(i)}\right\|$, we obtain :

$$\begin{aligned}\left\|\mathbf{b}_4^{(i)}\right\|^2 &\leq \left\|\mathbf{a}_4^{(i)}\right\|^2 - 2\left|\langle \mathbf{a}_3^{(i+1)}, \mathbf{a}_1^{(i+1)} + \mathbf{a}_2^{(i+1)}\rangle\right| + \left\|\mathbf{a}_1^{(i+1)} + \mathbf{a}_2^{(i+1)}\right\|^2 \\ &\leq \left\|\mathbf{a}_4^{(i)}\right\|^2 - \varepsilon \cdot \left\|\mathbf{a}_1^{(i+1)} + \mathbf{a}_2^{(i+1)}\right\|^2.\end{aligned}$$

Since the basis $\left[\mathbf{a}_1^{(i+1)}, \mathbf{a}_2^{(i+1)}\right]_{\leq}$ is ε-greedy-reduced, we also have :

$$\left\|\mathbf{a}_1^{(i+1)} + \mathbf{a}_2^{(i+1)}\right\|^2 \geq \left\|\mathbf{a}_1^{(i+1)}\right\|^2 + \left\|\mathbf{a}_2^{(i+1)}\right\|^2 - (1+\varepsilon) \cdot \left\|\mathbf{a}_1^{(i+1)}\right\|^2 \geq (1-\varepsilon) \cdot \left\|\mathbf{a}_2^{(i+1)}\right\|^2,$$

from which we get $(1 + \varepsilon(1-\varepsilon)) \cdot \left\|\mathbf{b}_4^{(i)}\right\|^2 \leq \left\|\mathbf{a}_4^{(i)}\right\|^2$. □

Lemma A.11 *Let $2 \leq d \leq 4$. There exist two constants $\varepsilon_2 > 0$ and $D > 0$ such that the following statement holds. Consider the $(i+1)$-th loop iteration of an execution of the d-dimensional greedy algorithm. Suppose that $\left[\mathbf{a}_1^{i+1}, \ldots, \mathbf{a}_{d-1}^{i+1}\right]_{\leq}$ is ε_2-greedy-reduced, and that $\left\|\mathbf{a}_k^{i+1}\right\| \geq (1-\varepsilon_2) \cdot \left\|\mathbf{a}_d^{i+1}\right\|$ for some $k \in [\![1, d-1]\!]$. Then, if $|x_k| \geq 2$ and if we are not in the 211-case, we have :*

$$\left\|\mathbf{b}_d^{i+1}\right\|^2 + D\left\|\mathbf{b}_k^{i+1}\right\|^2 \leq \left\|\mathbf{a}_d^{i+1}\right\|^2,$$

where the 211-case is : $d = 4, |x_k| = 2$ and the other $|x_j|$'s are both equal to 1.

This last lemma is a direct consequence of the Pythagorean theorem and of the Gap Lemma. This result is crucial to our analysis, and Section A.9 is devoted to prove it. Figure A.6 illustrates the Gap Lemma : when a vector from the outer non-hashed area that is mapped to a vector within the inner non-hashed area, its length decreases significantly.

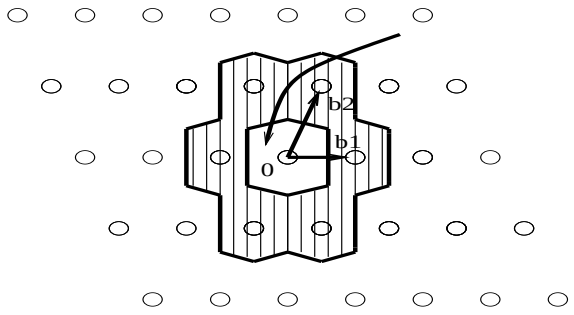


FIG. A.6 – The Gap Lemma in dimension 2.

Theorem A.9 (Gap Lemma) *Let $2 \leq d \leq 4$. There exist two constants $\varepsilon > 0$ and $D > 0$ such that the following statement holds. Let $[\mathbf{a}_1, \ldots, \mathbf{a}_{d-1}]_{\leq}$ be ε-greedy-reduced vectors, $\mathbf{u}$ be a vector of $\mathrm{Vor}(\mathbf{a}_1, \ldots, \mathbf{a}_{d-1})$ and $x_1, \ldots, x_{d-1}$ be integers. If $\|\mathbf{a}_k\| \geq (1-\varepsilon) \cdot \|\mathbf{a}_{d-1}\|$ for some $k < d$, then :*

$$\|\mathbf{u}\|^2 + D\|\mathbf{a}_k\|^2 \leq \left\|\mathbf{u} + \sum_{j=1}^{d-1} x_j \mathbf{a}_j\right\|^2,$$

where $|x_k| \geq 2$, and if $d = 4$ the two other $|x_j|$'s are not both equal to 1.

This completes the overall description of the proof of Theorem A.5. Indeed, choose three constants $\varepsilon, D > 0$ and $C > 1$ such that we can apply Lemmata A.10 and A.11. We prove that Equation A.1 holds for $K = \min\left(C, \sqrt{1+D}, \frac{1}{1-\varepsilon}\right) > 1$. Consider a loop iteration $i+1$ such that $p_{i+1} \leq p_i$. Recall that among any d consecutive iterations of the loop, there is at least one such iteration. For such an iteration we have $\left|x_{\pi_i}^{i+1}\right| \geq 2$. We distinguish four cases :

- The basis $\left[\mathbf{a}_1^{i+1}, \ldots, \mathbf{a}_{d-1}^{i+1}\right]_{\leq}$ is not ε-greedy-reduced : then Lemma A.10 gives the result *via* the i-th loop iteration.
- W have $\left\|\mathbf{a}_{\pi_i}^{i+1}\right\| < (1-\varepsilon) \cdot \left\|\mathbf{a}_d^{i+1}\right\|$: because $p_{i+1} \leq p_i$, we have the inequalities $\left\|\mathbf{b}_d^{i+1}\right\| < \left\|\mathbf{a}_{p_i}^{i+1}\right\| = \left\|\mathbf{a}_{\pi_i}^{i+1}\right\| < (1-\varepsilon) \cdot \left\|\mathbf{a}_d^{i+1}\right\|$.
- We are in the 211-case, i.e., $d = 4$ with $|x_{\pi_i}| = 2$ and the other $|x_j|$'s are all equal to 1. Then we refer to Subsection A.8.2.
- Otherwise we apply Lemma A.11, which gives the expected result *via* the $(i+1)$-th loop iteration.

A.8.2 Concluding in Dimension Four

In the previous subsections, we showed that there is at most a linear number of loop iterations in the iterative greedy algorithm in dimensions two and three, but we noticed that a new difficulty arose in dimension four : the Gap Lemma is useless in the so-called 211-case. This is because there are three-dimensional Minkowski-reduced bases $[\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3]_{\leq}$ for which $2\mathbf{b}_i + s_1\mathbf{b}_j + s_2\mathbf{b}_k$ — with $\{i,j,k\} = \{1,2,3\}$ and $|s_1| = |s_2| = 1$ — is a Voronoï vector. Indeed consider the lattice spanned by the columns $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$ of the following matrix :

$$M = \begin{bmatrix} 1 & 1 & -1 \\ 1 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

This basis is Minkowski-reduced and $\|\mathbf{b}_1+\mathbf{b}_2+2\mathbf{b}_3\| = \|\mathbf{b}_1+\mathbf{b}_2\| \leq \|(2k_1+1)\mathbf{b}_1+(2k_2+1)\mathbf{b}_1+2k_3\mathbf{b}_3\|$ for any $k_1, k_2, k_3 \in \mathbb{Z}$. Therefore, a vector in the translated Voronoï cell centred in $\mathbf{b}_1+\mathbf{b}_2+2\mathbf{b}_3$ can avoid being significantly shortened when translated inside the Voronoï cell centred in $\mathbf{0}$.

The Gap Lemma cannot tackle this problem. However, we notice that $(1,1,2)$ is rarely a Voronoï coordinate (with respect to a Minkowski-reduced basis), and when it is the case it cannot be a strict Voronoï coord : it can be proved easily that if $(1,1,2)$ is a Voronoï coord, then $\|\mathbf{b}_1+\mathbf{b}_2\| = \|\mathbf{b}_1+\mathbf{b}_2+2\mathbf{b}_3\|$, which tells us that $\mathbf{b}_1+\mathbf{b}_2+2\mathbf{b}_3$ is not the only vector in its coset of $L/2L$ reaching the length minimum. It turns out that the lattice spanned by the columns of M is essentially the only one for which $(1,1,2)$ — modulo any change of sign and permutation of coordinates — can be a Voronoï coord. More precisely, if $(1,1,2)$ — modulo any change of sign and permutation of coordinates — is a Voronoï coord for a lattice basis, then the basis matrix can be written as rUM where r is any non-zero real number and U is any orthogonal matrix. Since a basis can be arbitrarily close to one of them without actually being one of them, we need to consider a small compact set of normalised bases around the annoying ones. More precisely, this compact set is :

$$\left\{ [\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3]_{\leq} \ \varepsilon\text{-greedy-reduced}, \exists \sigma \in \mathcal{S}_3, \left\| \frac{1}{\|\mathbf{b}_3\|^2} \left| G\left(\mathbf{b}_{\sigma(1)}, \mathbf{b}_{\sigma(2)}, \mathbf{b}_{\sigma(3)}\right)\right| - \left|M^t M\right| \right\|_\infty \leq \varepsilon \right\},$$

for some sufficiently small $\varepsilon > 0$, where $\|M\|_\infty$ is the maximum of the absolute values of the matrix M and $|M|$ is the matrix made of the absolute values of the entries of M.

Now, consider we are in the 211-case at some loop iteration $i+1$. We distinguish three cases :

- The basis $\left[\mathbf{a}_1^{(i+1)}, \mathbf{a}_2^{(i+1)}, \mathbf{a}_3^{(i+1)}\right]_{\leq}$ is outside the compact. In this case, a variant of the Gap Lemma (Lemma A.28) proved in Section A.9 is valid and can be used to show that the vector $\mathbf{b}_4^{(i+1)}$ is significantly shorter than the vector $\mathbf{a}_4^{(i+1)}$.

– The basis $\left[\mathbf{a}_1^{(i+1)}, \mathbf{a}_2^{(i+1)}, \mathbf{a}_3^{(i+1)}\right]_{\leq}$ is inside the compact, but the orthogonal projection of $\mathbf{a}_4^{(i+1)}$ onto the linear span of $\left[\mathbf{a}_1^{(i+1)}, \mathbf{a}_2^{(i+1)}, \mathbf{a}_3^{(i+1)}\right]_{\leq}$ is far from the Voronoï cell $\mathrm{Vor}\left(\mathbf{a}_1^{(i+1)}, \mathbf{a}_2^{(i+1)}, \mathbf{a}_3^{(i+1)}\right)$. In this case, we can use Lemma A.28 to show that the vector $\mathbf{b}_4^{(i+1)}$ is significantly shorter than the vector $\mathbf{a}_4^{(i+1)}$.
– Otherwise the geometry of the basis $\left[\mathbf{a}_1^{(i+1)}, \mathbf{a}_2^{(i+1)}, \mathbf{a}_3^{(i+1)}, \mathbf{a}_4^{(i+1)}\right]_{\leq}$ is very precisely known and we can show that there remain $O(1)$ loop iterations.

More precisely, by using Lemma A.28, we show that :

Lemma A.12 *There exist two constants $K, \varepsilon > 0$ such that the following holds. Consider an execution of the four-dimensional greedy algorithm, and a loop iteration $i+1$ for which :*

1. $p_{i+1} \leq p_i$,
2. $|x_{\pi_i}| = 2$ *and* $\left(\left|x_{\sigma(1)}\right|, \left|x_{\sigma(2)}\right|, \left|x_{\sigma(3)}\right|\right) = (1,1,2)$ *for some* $\sigma \in \mathcal{S}_3$,
3. $\left[\mathbf{a}_1^{(i+1)}, \mathbf{a}_2^{(i+1)}, \mathbf{a}_3^{(i+1)}\right]_{\leq}$ *is ε-greedy-reduced,*
4. $\left\|\mathbf{a}_{\pi_i}^{(i+1)}\right\| \geq (1-\varepsilon) \cdot \left\|\mathbf{a}_4^{(i+1)}\right\|$.

Then either $\left\|\mathbf{a}_4^{(i+1)}\right\| \geq (1+K) \cdot \left\|\mathbf{b}_4^{(i+1)}\right\|$ *or :*

$$\left\| \frac{1}{\left\|\mathbf{a}_4^{(i+1)}\right\|^2} \left| G\left(\mathbf{a}_{\sigma(1)}^{(i+1)}, \mathbf{a}_{\sigma(2)}^{(i+1)}, \mathbf{a}_{\sigma(3)}^{(i+1)}, \mathbf{a}_4^{(i+1)}\right)\right| - A \right\|_{\infty} \leq \varepsilon, \text{ with } A = \begin{bmatrix} 1 & 0 & \frac{1}{2} & 0 \\ 0 & 1 & \frac{1}{2} & 0 \\ \frac{1}{2} & \frac{1}{2} & 1 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & 1 \end{bmatrix}.$$

To prove this result, we restrict more and more the possible geometry of the basis $\left[\mathbf{a}_1^{(i)}, \mathbf{a}_2^{(i)}, \mathbf{a}_3^{(i)}, \mathbf{a}_4^{(i)}\right]_{\leq}$. Notice that this critical geometry corresponds to the root lattice D_4. This last case is considered in Lemma A.13.

Proof. The proof essentially relies on Lemma A.28. We choose $\varepsilon, C > 0$ according to Lemma A.28. The constant K will depend on C and ε. We first show that wlog we can suppose that the basis vectors have similar lengths, that is $(1-\varepsilon)^2 \cdot \left\|\mathbf{a}_4^{(i+1)}\right\| \leq \left\|\mathbf{a}_1^{(i+1)}\right\|$. We know that $|x_{\pi_i}| = 2$ and the two other $|x_i|$'s are 1. By hypothesis, we have $\left\|\mathbf{a}_{\pi_i}^{(i+1)}\right\| \geq (1-\varepsilon) \cdot \left\|\mathbf{a}_4^{(i+1)}\right\|$. If $\pi_i = 1$, we are done. If $\pi_i = 2$, then we apply Lemma A.28 2), and if $\pi_i = 3$ we apply Lemma A.28 1), in both cases along with the Pythagorean theorem. So far, we have proved that one of the following holds :

1. $\|\mathbf{a}_4^{(i+1)}\|^2 \geq (1+C)\|\mathbf{b}_4^{(i+1)}\|^2$,
2. $(1-\varepsilon)^2\|\mathbf{a}_4^{(i+1)}\| \leq \|\mathbf{a}_1^{(i+1)}\| \leq \|\mathbf{a}_2^{(i+1)}\| \leq \|\mathbf{a}_3^{(i+1)}\| \leq \|\mathbf{a}_4^{(i+1)}\|$.

The remainder of the proof is the same for any of the possible configurations of (x_1, x_2, x_3), thus, for the sake of simplicity, we suppose now that $(x_1, x_2, x_3) = (2, 1, 1)$. The following step of the proof is to show that we can suppose that the Gram matrix of $\left[\mathbf{a}_1^{(i+1)}, \mathbf{a}_2^{(i+1)}, \mathbf{a}_3^{(i+1)}\right]_{\leq}$ is approximately :

$$\left\|\mathbf{a}_4^{(i+1)}\right\|^2 \cdot \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & 1 & 0 \\ -\frac{1}{2} & 0 & 1 \end{bmatrix}.$$

This directly follows from Lemma A.28 and the Pythagorean theorem, which give that at least one of the following holds :

1. $\left\|\mathbf{a}_4^{(i+1)}\right\|^2 \geq (1+C)\cdot\left\|\mathbf{b}_4^{(i+1)}\right\|^2$,
2. $\left|\langle\mathbf{a}_2^{(i+1)},\mathbf{a}_3^{(i+1)}\rangle\right| \leq \varepsilon\cdot\left\|\mathbf{a}_3^{(i+1)}\right\|^2$ and $\left|\langle\mathbf{a}_1^{(i+1)},\mathbf{a}_j^{(i+1)}\rangle + \frac{1}{2}\cdot\left\|\mathbf{a}_3^{(i+1)}\right\|^2\right| \leq \varepsilon\cdot\left\|\mathbf{a}_3^{(i+1)}\right\|^2$ for $j\in\{2,3\}$.

It remains to consider the scalar products $\langle\mathbf{a}_4^{(i+1)},\mathbf{a}_k^{(i+1)}\rangle$ for $k\in\{1,2,3\}$. Recall that the orthogonal projection of the vector $\mathbf{b}_4^{(i+1)} = \mathbf{a}_4^{(i+1)} - \mathbf{a}_3^{(i+1)} - \mathbf{a}_2^{(i+1)} - 2\mathbf{a}_1^{(i+1)}$ onto the linear span of $\left[\mathbf{a}_1^{(i+1)},\mathbf{a}_2^{(i+1)},\mathbf{a}_3^{(i+1)}\right]_{\leq}$ belongs to the Voronoï cell $\mathrm{Vor}\left(\mathbf{a}_1^{(i+1)},\mathbf{a}_2^{(i+1)},\mathbf{a}_3^{(i+1)}\right)$. Wlog we can suppose that $\left\|\mathbf{a}_4^{(i+1)}\right\| \geq \left\|\mathbf{b}_4^{(i+1)}\right\| \geq (1-\varepsilon)\cdot\left\|\mathbf{a}_4^{(i+1)}\right\|$, since otherwise have the result for $K=\frac{1}{1-\varepsilon}$. By expanding $\|\mathbf{b}_4\|^2$, we get :

$$\begin{aligned}(1+19\varepsilon)\cdot\left\|\mathbf{a}_4^{(i+1)}\right\|^2 &\geq 3\left\|\mathbf{a}_4^{(i+1)}\right\|^2 - 2\langle\mathbf{a}_4^{(i+1)}, 2\mathbf{a}_1^{(i+1)}+\mathbf{a}_2^{(i+1)}+\mathbf{a}_3^{(i+1)}\rangle\\ &\geq (1-19\varepsilon)\cdot\left\|\mathbf{a}_4^{(i+1)}\right\|^2,\end{aligned}$$

where we used our knowledge of the Gram matrix of $\left[\mathbf{a}_1^{(i+1)},\mathbf{a}_2^{(i+1)},\mathbf{a}_3^{(i+1)}\right]_{\leq}$. Thus we have :

$$\left|\langle\mathbf{a}_4^{(i+1)}, 2\mathbf{a}_1^{(i+1)}+\mathbf{a}_2^{(i+1)}+\mathbf{a}_3^{(i+1)}\rangle - \left\|\mathbf{a}_4^{(i+1)}\right\|\right| \leq 19\varepsilon\cdot\left\|\mathbf{a}_4^{(i+1)}\right\|.$$

This last equation gives that in order to end the proof, it is sufficient to prove that the scalar products $\langle\mathbf{a}_4^{(i+1)},\mathbf{a}_2^{(i+1)}\rangle$ and $\langle\mathbf{a}_4^{(i+1)},\mathbf{a}_3^{(i+1)}\rangle$ are small. Let $j\in\{2,3\}$. By hypothesis, for any $x\in\mathbb{Z}$, we have $\left\|\mathbf{a}_4^{(i+1)} - 2\mathbf{a}_1^{(i+1)} - x\mathbf{a}_j^{(i+1)}\right\| \geq \left\|\mathbf{a}_4^{(i+1)} - 2\mathbf{a}_1^{(i+1)} - \mathbf{a}_2^{(i+1)} - \mathbf{a}_3^{(i+1)}\right\|$. In particular, by choosing $x=0$ and $x=2$ and expanding the norms and using the knowledge of the Gram matrix of $\left[\mathbf{a}_1^{(i+1)},\mathbf{a}_2^{(i+1)},\mathbf{a}_3^{(i+1)}\right]_{\leq}$, one can obtain an explicit positive integer k such that $\left|\langle\mathbf{a}_4^{(i+1)},\mathbf{a}_1^{(i+1)}\rangle\right| \leq k\varepsilon\cdot\left\|\mathbf{a}_4^{(i+1)}\right\|$.

Let $K = \min\left(1+C, \frac{1}{(1-\varepsilon)^2}\right)$. Altogether, we have proved that there exists a constant $K' = \max(K,k,16)$ such that :

$$\left\|\frac{1}{\left\|\mathbf{a}_4^{(i+1)}\right\|^2}G\left(\mathbf{a}_1^{(i+1)},\mathbf{a}_2^{(i+1)},\mathbf{a}_3^{(i+1)},\mathbf{a}_4^{(i+1)}\right) - A\right\|_\infty \leq K'\varepsilon.$$

This completes the proof of the lemma. □

At this point of the analysis of the 211-case, we have shown that we can suppose that the shape of the basis $\left[\mathbf{a}_1^{(i+1)},\mathbf{a}_2^{(i+1)},\mathbf{a}_3^{(i+1)},\mathbf{a}_4^{(i+1)}\right]_{\leq}$ is very specific : its Gram matrix is very close to A. We treat this last case by applying the following lemma, which roughly says that if the Gram matrix of a basis is sufficiently close to some invertible matrix, then the number of short vectors generated by the basis remains bounded. Since the greedy algorithm always creates smaller bases for the lexicographic order based on the lengths, if the Gram matrix of the current basis is close to the matrix A, then it remains $O(1)$ loop iterations.

Lemma A.13 *Let A be a $d\times d$ invertible matrix, and $B>0$. Then there exist $\varepsilon, N>0$ such that for any basis $(\mathbf{b}_1,\ldots,\mathbf{b}_d)$ satisfying $\left\|\frac{1}{\|\mathbf{b}_d\|^2}G(\mathbf{b}_1,\ldots,\mathbf{b}_d) - A\right\|_\infty \leq \varepsilon$, then :*

$$|\{(x_1,\ldots,x_d), \|x_1\mathbf{b}_1+\cdots+x_d\mathbf{b}_d\| \leq B\|\mathbf{b}_d\|\}| \leq N.$$

Proof. Let $\varepsilon > 0$ such that for any G, if $\|G - A\|_\infty \leq \varepsilon$, then G is invertible (such an ε does exist since the set of invertible matrices is open). In that case, if $X = (x_1, \ldots, x_d)$, then

$$\left| \frac{1}{\|\mathbf{b}_d\|^2} \|x_1\mathbf{b}_1 + \cdots + x_d\mathbf{b}_d\|^2 - XAX^t \right| = |X(G-A)X^t| \leq d^2\varepsilon \cdot (XX^t),$$

where $G = \frac{1}{\|\mathbf{b}_d\|^2} G(\mathbf{b}_1, \ldots, \mathbf{b}_d)$. Therefore, if $\|x_1\mathbf{b}_1 + \cdots + x_d\mathbf{b}_d\| \leq B\|\mathbf{b}_d\|$, then, by the triangular inequality, we obtain $|XAX^t| \leq B^2 + d^2\varepsilon \cdot (XX^t)$. But $|XAX^t| \geq \frac{1}{\|A^{-1}\|}(XX^t)$, where $\|B\|$ is defined as $\max(YBY^t, Y \in \mathbb{R}^n$ and $\|Y\| = 1)$, which is positive. Therefore :

$$(XX^t) \cdot \left(\frac{1}{\|A^{-1}\|} - d^2\varepsilon \right) \leq B^2.$$

We set $\varepsilon < \frac{1}{d^2\|A^{-1}\|}$. The x_i's are integers such that the quantity (XX^t) remains bounded, so that we are considering integer points in a d-dimensional hypersphere. There can only be finitely many such points. □

A.9 The Geometry of Low-Dimensional Lattices

In this section, we give some results about Voronoï cells in dimensions two and three, which are crucial to the complexity analysis of the greedy algorithm described in Section A.4. More precisely, the analysis is based on the Gap Lemma (given in Subsection A.9.3), which is derived from the study of Voronoï cells in the case of ε-greedy-reduced vectors (see Subsection A.9.2), itself derived from the study of Voronoï cells for Minkowski-reduced bases (in Subsection A.9.1).

A.9.1 Voronoï Cells in the Case of Minkowski-Reduced Bases

We start by giving some simple bounds on the diameter of the Voronoï cell and on the Gram-Schmidt orthogonalisation of a Minkowski-reduced basis :

Lemma A.14 *Let $d \geq 1$. Let $[\mathbf{b}_1, \ldots, \mathbf{b}_d]_\leq$ be a basis of a lattice L. Then $\rho(L) \leq \frac{\sqrt{d}}{2} \cdot \|\mathbf{b}_d\|$. As a consequence, if the basis $[\mathbf{b}_1, \ldots, \mathbf{b}_d]_\leq$ is a Minkowski-reduced basis, then $\|\mathbf{b}_d^*\| \geq \frac{\sqrt{5-d}}{2} \cdot \|\mathbf{b}_d\|$.*

Proof. The first part of the lemma is very classical and several different proofs can be found in the literature. We give here a proof based on transference bounds. Wlog we suppose that $\|\mathbf{b}_d\| = 1$. We have the transference bound (see [199]) : $\rho(L) \cdot \lambda_1(L^*) \leq \sqrt{d}/2$, where L^* is the dual of the lattice L, i.e., $(\forall \mathbf{b} \in L^*)(\forall i \leq d)(\langle \mathbf{b}, \mathbf{b}_i \rangle \in \mathbb{Z})$. By the definition of L^* and the fact that $\mathbf{b}_1, \ldots, \mathbf{b}_d$ are of lengths less than 1, we have $\lambda_1(L^*) \geq 1$, which gives the bound on $\rho(L)$.

Suppose now that the basis $[\mathbf{b}_1, \ldots, \mathbf{b}_d]_\leq$ is Minkowski-reduced. Then the orthogonal projection of the vector $\mathbf{b}_d$ onto the linear span of $[\mathbf{b}_1, \ldots, \mathbf{b}_d]_\leq$ is in $\mathrm{Vor}(\mathbf{b}_1, \ldots, \mathbf{b}_d)$. Therefore, by the Pythagorean theorem, we get : $\|\mathbf{b}_d^*\|^2 \geq \|\mathbf{b}_d\|^2 - \frac{d-1}{4}\|\mathbf{b}_{d-1}\|^2$. The fact that $\|\mathbf{b}_{d-1}\| \leq \|\mathbf{b}_d\|$ completes the proof. □

The following lemma provides the possible Voronoï vectors of a lattice of dimension two given by a Minkowski-reduced basis (recall that we cannot directly use Theorem A.2 because it only considers strict Voronoï coordinates). Such a basis confines the coordinates of the Voronoï vectors :

Lemma A.15 *In dimension two, the possible Voronoï coords are $(1, 0)$ and $(1, 1)$, modulo any change of signs and permutation of coordinates, i.e., any nonzero $(\varepsilon_1, \varepsilon_2)$ where $|\varepsilon_1|, |\varepsilon_2| \leq 1$.*

The proof relies on a detailed study of the quantity $\|(2x_1+\varepsilon_1)\cdot\mathbf{b}_1+(2x_2+\varepsilon_2)\cdot\mathbf{b}_2\|^2 - \|\varepsilon_1\mathbf{b}_1+\varepsilon_2\mathbf{b}_2\|^2$, where the basis $[\mathbf{b}_1, \mathbf{b}_2]_\leq$ is Minkowski-reduced, $\varepsilon_1, \varepsilon_2 \in \{0, 1\}$ and $x_1, x_2 \in \mathbb{Z}$. Indeed, since the

Voronoï coords of a lattice L are given by the minima of the non-zero cosets of $L/2L$, it suffices to show that if $x_1 \neq 0$ or $x_2 \neq 0$, then this expression is strictly positive.

Proof. Recall that the possible Voronoï coords can be obtained by considering the short elements of $L/2L$, given a Minkowski-reduced basis of L. Let $[\mathbf{b}_1, \mathbf{b}_2]_{\leq}$ be a reduced basis. By eventually replacing the vector $\mathbf{b}_i$ by $-\mathbf{b}_i$ for $i \in \{1, 2\}$, it is clearly sufficient to show that for any $x_1, x_2 \geq 0$, and for any $\varepsilon_1, \varepsilon_2 \in \{0, 1\}$, if $x_1 \geq 1$ or $x_2 \geq 1$, then :

$$\|(2x_1 + \varepsilon_1) \cdot \mathbf{b}_1 + (2x_2 + \varepsilon_2) \cdot \mathbf{b}_2\|^2 > \|\varepsilon_1 \cdot \mathbf{b}_1 + \varepsilon_2 \cdot \mathbf{b}_2\|^2.$$

First of all :

$$\begin{aligned} \|(2x_1 + \varepsilon_1) \cdot \mathbf{b}_1 + (2x_2 + \varepsilon_2) \cdot \mathbf{b}_2\|^2 - \|\varepsilon_1 \cdot \mathbf{b}_1 &+ \varepsilon_2 \cdot \mathbf{b}_2\|^2 \\ &= \big((2x_1 + \varepsilon_1)^2 - \varepsilon_1^2\big) \cdot \|\mathbf{b}_1\|^2 + \big((2x_2 + \varepsilon_2)^2 - \varepsilon_2^2\big) \cdot \|\mathbf{b}_2\|^2 \\ &+ 2((2x_1 + \varepsilon_1)(2x_2 + \varepsilon_2) - \varepsilon_1\varepsilon_2) \cdot \langle \mathbf{b}_1, \mathbf{b}_2 \rangle. \end{aligned}$$

Since $x_1, x_2 \geq 0$, we have that $(2x_2 + \varepsilon_2)^2 - \varepsilon_2^2 \geq 0$ and $(2x_1 + \varepsilon_1)(2x_2 + \varepsilon_2) - \varepsilon_1\varepsilon_2 \geq 0$. Moreover, the basis $[\mathbf{b}_1, \mathbf{b}_2]_{\leq}$ is reduced and therefore $\|\mathbf{b}_2\| \geq \|\mathbf{b}_1\|$ and $2\langle \mathbf{b}_1, \mathbf{b}_2 \rangle \geq -\|\mathbf{b}_1\|^2$. From these facts, we obtain :

$$\|(2x_1+\varepsilon_1)\cdot\mathbf{b}_1+(2x_2+\varepsilon_2)\cdot\mathbf{b}_2\|^2-\|\varepsilon_1\mathbf{b}_1+\varepsilon_2\mathbf{b}_2\|^2 \geq 2\left[2x_1^2 + 2x_2^2 - 2x_1x_2 + x_1(2\varepsilon_1 - \varepsilon_2) + x_2(2\varepsilon_2 - \varepsilon_1)\right]\cdot\|\mathbf{b}_1\|^2.$$

This last expression is strictly positive as long as $(x_1, x_2) \neq (0, 0)$. Indeed :

- if $\varepsilon_1 = \varepsilon_2 = 0$, the factor is $4\big((x_1 - x_2)^2 + x_1x_2\big)$,
- if $\varepsilon_1 = 0$ and $\varepsilon_2 = 1$, the factor is $2\big(x_2^2 + 2x_2 + (x_2 - x_1)^2 + (x_1^2 - x_1)\big)$,
- the case $\varepsilon_1 = 1$ and $\varepsilon_2 = 0$ is symmetric,
- if $\varepsilon_1 = \varepsilon_2 = 1$, the factor is $2\big((x_2 - x_1)^2 + (x_2^2 + x_2) + (x_1^2 + x_1)\big)$.

□

We generalise this analysis to the three-dimensional case. The underlying ideas of the proof are the same, but the increase of the number of variables makes the analysis more tedious.

Lemma A.16 *In dimension three, the possible Voronoï coordinates are among* $(1, 0, 0)$, $(1, 1, 0)$, $(1, 1, 1)$ *and* $(2, 1, 1)$, *modulo any change of signs and permutation of coordinates.*

Proof. We generalise the proof of Lemma A.15. We show that for any integers x_1, x_2, x_3 and any $\varepsilon_1, \varepsilon_2, \varepsilon_3 \in \{0, 1\}$, if $(2x_1 + \varepsilon_1, 2x_2 + \varepsilon_2, 2x_3 + \varepsilon_3)$ is not in the desired list of Voronoï coords, then :

$$\|(2x_1 + \varepsilon_1) \cdot \mathbf{b}_1 + (2x_2 + \varepsilon_2) \cdot \mathbf{b}_2 + (2x_3 + \varepsilon_3) \cdot \mathbf{b}_3\|^2 - \|\varepsilon_1\mathbf{b}_1 + \varepsilon_2\mathbf{b}_2 + \varepsilon_3\mathbf{b}_3\|^2 > 0.$$

By replacing the vector $\mathbf{b}_i$ by $-\mathbf{b}_i$, we see that wlog the proof can be restricted to the case $x_1, x_2, x_3 \geq 0$. Moreover, because we already considered the two-dimensional case in Lemma A.15, we can suppose that for any $i \in \{1, 2, 3\}$, we have $(x_i, \varepsilon_i) \neq (0, 0)$.

From Lemma A.14, we know that since the basis $[\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3]_{\leq}$ is Minkowski-reduced, we have $\|\mathbf{b}_3^*\| \geq \|\mathbf{b}_3\|/\sqrt{2}$. As a consequence, if $2x_3 + \varepsilon_3 \geq 5$, then :

$$\|(2x_1 + \varepsilon_1) \cdot \mathbf{b}_1 + (2x_2 + \varepsilon_2) \cdot \mathbf{b}_2 + (2x_3 + \varepsilon_3) \cdot \mathbf{b}_3\|^2 \geq 25 \cdot \|\mathbf{b}_3^*\|^2 \geq \frac{25}{2} \cdot \|\mathbf{b}_3\|^2,$$

and the triangular inequality gives that $\|\varepsilon_1 \cdot \mathbf{b}_1 + \varepsilon_2 \cdot \mathbf{b}_2 + \varepsilon_3 \cdot \mathbf{b}_3\|^2 \leq 9 \cdot \|\mathbf{b}_3\|^2$. This gives the result when $2x_3 + \varepsilon_3 \geq 5$. The same argument holds for $(x_3, \varepsilon_3) = (2, 0)$, and for $(x_3, \varepsilon_3) \in \{(1, 1), (1, 0)\}$ with $\varepsilon_1 \cdot \varepsilon_2 = 0$. Therefore, it remains to consider the three cases $(x_3, \varepsilon_3) = (1, 1)$ with $\varepsilon_1 = \varepsilon_2 = 1$, $(x_3, \varepsilon_3) = (1, 0)$ with $\varepsilon_1 = \varepsilon_2 = 1$, and $(x_3, \varepsilon_3) = (0, 1)$.

Case 1 : Suppose that $(x_3, \varepsilon_3) = (1, 1)$ and $\varepsilon_1 = \varepsilon_2 = 1$. Since the basis $[\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3]_{\leq}$ is Minkowski-reduced, we have $\langle \mathbf{b}_i, \mathbf{b}_j \rangle \geq -\|\mathbf{b}_i\|^2/2$ for any $1 \leq i < j \leq 3$, which gives that

$$\|(2x_1 + 1) \cdot \mathbf{b}_1 + (2x_2 + 1) \cdot \mathbf{b}_2 + 3 \cdot \mathbf{b}_3\|^2 - \|\mathbf{b}_1 + \mathbf{b}_2 + \mathbf{b}_3\|^2$$

is equal to

$$4\left(x_1^2+x_1\right)\cdot\|\mathbf{b}_1\|^2+4\left(x_2^2+x_2\right)\cdot\|\mathbf{b}_2\|^2+8\cdot\|\mathbf{b}_3\|^2+4(3x_1+1)\cdot\langle\mathbf{b}_1,\mathbf{b}_3\rangle \\ +4(3x_2+1)\cdot\langle\mathbf{b}_2,\mathbf{b}_3\rangle+4(2x_1x_2+x_1+x_2)\cdot\langle\mathbf{b}_1,\mathbf{b}_2\rangle,$$

which is

$$\geq\left(4x_1^2-4x_1x_2-4x_1-2x_2-2\right)\cdot\|\mathbf{b}_1\|^2+\left(4x_2^2-2x_2-2\right)\|\mathbf{b}_2\|^2+8\|\mathbf{b}_3\|^2.$$

If $x_2=0$, it suffices to lower-bound $\left(x_1^2-x_1\right)\cdot\|\mathbf{b}_1\|^2+\|\mathbf{b}_3\|^2$, which is always greater than $\|\mathbf{b}_3\|^2$ and therefore strictly positive. Suppose now that $x_2\geq 1$. Then $4x_2^2-2x_2-2\geq 0$ and we obtain that

$$\|(2x_1+1)\mathbf{b}_1+(2x_2+1)\mathbf{b}_2+3\mathbf{b}_3\|^2-\|\mathbf{b}_1+\mathbf{b}_2+\mathbf{b}_3\|^2$$

is

$$\geq 4\left(x_1^2+x_2^2-x_1x_2-x_1-x_2+1\right)\|\mathbf{b}_1\|^2\geq 4\left((x_1-x_2)^2+(x_1x_2-x_1-x_2)+1\right)\|\mathbf{b}_1\|^2.$$

It is clear that this last expression is strictly positive for any $x_1,x_2\geq 0$ except when $x_1=x_2=1$. In this last situation, we use the fact that $\|3\cdot\mathbf{b}_1+3\cdot\mathbf{b}_2+3\cdot\mathbf{b}_3\|^2-\|\mathbf{b}_1+\mathbf{b}_2+\mathbf{b}_3\|^2=8\cdot\|\mathbf{b}_1+\mathbf{b}_2+\mathbf{b}_3\|^2$.

Case 2 : Suppose now that $(x_3,\varepsilon_3)=(1,0)$ and $\varepsilon_1=\varepsilon_2=1$. Similarly,

$$\|(2x_1+1)\mathbf{b}_1+(2x_2+1)\mathbf{b}_2+2\cdot\mathbf{b}_3\|^2-\|\mathbf{b}_1+\mathbf{b}_2\|^2$$

is equal to

$$4\left(x_1^2+x_1\right)\|\mathbf{b}_1\|^2+4\left(x_2^2+x_2\right)\|\mathbf{b}_2\|^2+4\cdot\|\mathbf{b}_3\|^2+4(2x_1+1)\langle\mathbf{b}_1,\mathbf{b}_3\rangle \\ +4(2x_2+1)\langle\mathbf{b}_2,\mathbf{b}_3\rangle+4(2x_1x_2+x_1+x_2)\langle\mathbf{b}_1,\mathbf{b}_2\rangle$$

which is

$$\geq\left(4x_1^2-4x_1x_2-2x_1-2x_2-2\right)\|\mathbf{b}_1\|^2+\left(4x_2^2-2\right)\|\mathbf{b}_2\|^2+4\|\mathbf{b}_3\|^2.$$

If $x_2=0$, it suffices to lower-bound $\left(2x_1^2-x_1-1\right)\cdot\|\mathbf{b}_1\|^2+\|\mathbf{b}_3\|^2$, which is strictly positive if $x_1\geq 1$. If $x_1=0$, then we have one of the possible Voronoï coords. Suppose now that $x_2\geq 1$. In that case $4x_2^2-2\geq 0$, which ensures that :

$$\begin{aligned}\|(2x_1+1)\mathbf{b}_1+(2x_2+1)\mathbf{b}_2+2\mathbf{b}_3\|^2-\|\mathbf{b}_1+\mathbf{b}_2\|^2 &\geq 2\left(2x_1^2+2x_2^2-2x_1x_2-x_1-x_2\right)\|\mathbf{b}_1\|^2\\ &\geq 2\left((x_1-x_2)^2+(x_1^2-x_1)+(x_2^2-x_2)\right)\|\mathbf{b}_1\|^2.\end{aligned}$$

If $x_1\geq 2$ or $x_2\geq 2$ or $x_1\neq x_2$, this is strictly positive. Therefore it remains to consider the case $x_1=x_2=1$. We have :

$$\|3\cdot\mathbf{b}_1+3\cdot\mathbf{b}_2+2\cdot\mathbf{b}_3\|^2-\|\mathbf{b}_1+\mathbf{b}_2\|^2=4\cdot\left(\|\mathbf{b}_3\|^2+3\cdot\langle\mathbf{b}_3,\mathbf{b}_1+\mathbf{b}_2\rangle+2\cdot\|\mathbf{b}_1+\mathbf{b}_2\|^2\right).$$

Since the basis $[\mathbf{b}_1,\mathbf{b}_2,\mathbf{b}_3]_{\leq}$ is reduced, we have $\langle\mathbf{b}_3,\mathbf{b}_1+\mathbf{b}_2\rangle\geq-\|\mathbf{b}_1+\mathbf{b}_2\|^2/2$, which gives the expected result.

Case 3 : Suppose now that $(x_3,\varepsilon_3)=(0,1)$. Similarly, we have the inequalities :

$$\begin{aligned}\|(2x_1+\varepsilon_1)\mathbf{b}_1+(2x_2+\varepsilon_2)\mathbf{b}_2+\mathbf{b}_3\|^2 &- \|\varepsilon_1\mathbf{b}_1+\varepsilon_2\mathbf{b}_2+\mathbf{b}_3\|^2\\ &= 4\left(x_1^2+\varepsilon_1x_1\right)\|\mathbf{b}_1\|^2+4\left(x_2^2+\varepsilon_2x_2\right)\|\mathbf{b}_2\|^2+4x_1\langle\mathbf{b}_1,\mathbf{b}_3\rangle\\ &+ 4x_2\langle\mathbf{b}_2,\mathbf{b}_3\rangle+4(2x_1x_2+x_1\varepsilon_2+x_2\varepsilon_1)\langle\mathbf{b}_1,\mathbf{b}_2\rangle\\ &\geq \left(4x_1^2-4x_1x_2+(4\varepsilon_1-2-2\varepsilon_2)x_1-2x_2\varepsilon_1\right)\|\mathbf{b}_1\|^2\\ &+ \left(4x_2^2+(4\varepsilon_2-2)x_2\right)\|\mathbf{b}_2\|^2.\end{aligned}$$

If $x_2 = 0$, it suffices to lower-bound $4x_1^2 + (4\varepsilon_1 - 2 - 2\varepsilon_2)x_1$. It is strictly positive as soon as $x_1 \geq 1$ except in the case $(x_1, \varepsilon_1, \varepsilon_2) = (1, 0, 1)$, which corresponds to one of the possible Voronoï coords. If $x_1 = 0$, we only have possible Voronoï coords. Suppose now that $x_2 \geq 1$. In that case $4x_2^2 + (4\varepsilon_2 - 2)x_2 \geq 0$ and :

$$\begin{aligned} \|(2x_1 + \varepsilon_1)\mathbf{b}_1 &+ (2x_2 + \varepsilon_2)\mathbf{b}_2 + \mathbf{b}_3\|^2 - \|\varepsilon_1\mathbf{b}_1 + \varepsilon_2\mathbf{b}_2 + \mathbf{b}_3\|^2 \\ &\geq \left(4x_1^2 + 4x_2^2 - 4x_1x_2 + (4\varepsilon_1 - 2 - 2\varepsilon_2)x_1 + (4\varepsilon_2 - 2 - 2\varepsilon_1)x_2\right)\|\mathbf{b}_1\|^2. \end{aligned}$$

For $(\varepsilon_1, \varepsilon_2) = (0, 0)$, we get $2\left((x_1 - x_2)^2 + (x_1^2 - x_1) + (x_2^2 - x_2)\right)$, which is strictly positive as soon as $x_1 \neq x_2$ or $x_1 \geq 2$ or $x_2 \geq 2$. The only remaining case that does not provide a possible Voronoï coord is $x_1 = x_2 = 1$. Notice that $\|2\mathbf{b}_1 + 2\mathbf{b}_2 + \mathbf{b}_3\|^2 > \|\mathbf{b}_3\|^2$ is equivalent to $\|\mathbf{b}_1 + \mathbf{b}_2\|^2 + \langle \mathbf{b}_1 + \mathbf{b}_2, \mathbf{b}_3\rangle > 0$, which is implied by $\langle \mathbf{b}_1 + \mathbf{b}_2, \mathbf{b}_3\rangle \geq -\|\mathbf{b}_1 + \mathbf{b}_2\|^2/2$ (the vector $\mathbf{b}_3$ has its orthogonal projection onto the span of $[\mathbf{b}_1, \mathbf{b}_2]$ inside $\mathrm{Vor}(\mathbf{b}_1, \mathbf{b}_2)$).

For $(\varepsilon_1, \varepsilon_2) = (1, 0)$, we get $2(x_1 - x_2)^2 + 2(x_2^2 - 2x_2) + 2x_1^2 + 2x_1$. If $x_2 \geq 2$, this is clearly strictly positive. If $x_2 = 1$, we obtain $4x_1^2 - 2x_1$, which is strictly positive unless $x_1 = 0$ (one of the possible Voronoï coords). We have already considered the case $x_2 = 0$. The case $(\varepsilon_1, \varepsilon_2) = (0, 1)$ is symmetric.

Finally, if $(\varepsilon_1, \varepsilon_2) = (1, 1)$, we obtain $2\left((x_1 - x_2)^2 + x_1^2 + x_2^2\right)$, which is strictly positive unless $x_1 = x_2 = 0$ (one of the possible Voronoï coords). This completes the proof of the lemma. □

The possible Voronoï coord $(2, 1, 1)$ creates difficulties when analysing the greedy algorithm in dimension four because it contains a two, which cannot be handled with the greedy argument used for the ones. We tackle this problem as follows : we show that when $(2, 1, 1)$ happens to be a Voronoï coord, the lattice has a very specific shape, for which the behaviour of the algorithm is well-understood.

Lemma A.17 *Suppose the basis $[\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3]_\leq$ is Minkowski-reduced.*

1. *If $(s_1, s_2, 2)$ is a Voronoï coord with $s_i = \pm 1$ for $i \in \{1, 2\}$, then $\|\mathbf{b}_1\| = \|\mathbf{b}_2\| = \|\mathbf{b}_3\|$, $\langle \mathbf{b}_1, \mathbf{b}_2\rangle = 0$ and $\langle \mathbf{b}_i, \mathbf{b}_3\rangle = -\frac{s_i}{2} \cdot \|\mathbf{b}_1\|^2$ for $i \in \{1, 2\}$.*
2. *If $(s_1, 2, s_3)$ is a Voronoï coord with $s_i = \pm 1$ for $i \in \{1, 3\}$, then $\|\mathbf{b}_1\| = \|\mathbf{b}_2\|$. Moreover, if $\|\mathbf{b}_1\| = \|\mathbf{b}_2\| = \|\mathbf{b}_3\|$, then $\langle \mathbf{b}_1, \mathbf{b}_3\rangle = 0$ and $\langle \mathbf{b}_i, \mathbf{b}_2\rangle = -\frac{s_i}{2} \cdot \|\mathbf{b}_1\|^2$ for $i \in \{1, 3\}$.*
3. *If $(2, s_2, s_3)$ is a Voronoï coord with $s_i = \pm 1$ for $i \in \{2, 3\}$ and $\|\mathbf{b}_1\| = \|\mathbf{b}_2\| = \|\mathbf{b}_3\|$, then $\langle \mathbf{b}_2, \mathbf{b}_3\rangle = 0$ and $\langle \mathbf{b}_i, \mathbf{b}_1\rangle = -\frac{s_i}{2} \cdot \|\mathbf{b}_1\|^2$ for $i \in \{2, 3\}$.*

Proof. Wlog we suppose that for any i, we have $s_i = 1$. In the first situation we consider the inequality $\|\mathbf{b}_1 + \mathbf{b}_2 + 2 \cdot \mathbf{b}_3\|^2 \leq \|\mathbf{b}_1 + \mathbf{b}_2\|^2$: it is equivalent to $\|\mathbf{b}_3\|^2 + \langle \mathbf{b}_1, \mathbf{b}_3\rangle + \langle \mathbf{b}_2, \mathbf{b}_3\rangle \leq 0$. Since the basis is reduced, we must have $\langle \mathbf{b}_1, \mathbf{b}_3\rangle \geq -\|\mathbf{b}_1\|^2/2$ and $\langle \mathbf{b}_2, \mathbf{b}_3\rangle \geq -\|\mathbf{b}_2\|^2/2$. Thus $2 \cdot \|\mathbf{b}_3\|^2 - \|\mathbf{b}_1\|^2 - \|\mathbf{b}_2\|^2 \leq 0$. Consequently, the length equalities hold and the inequalities on the scalar products above are equalities. It remains to prove that $\langle \mathbf{b}_1, \mathbf{b}_2\rangle = 0$. Since $\mathbf{b}_1$ is a shortest vector, we have $\|\mathbf{b}_3 + \mathbf{b}_2 + \mathbf{b}_1\|^2 \geq \|\mathbf{b}_1\|^2$, which is equivalent to $\langle \mathbf{b}_1, \mathbf{b}_2\rangle \geq 0$. Moreover, we have $\langle \mathbf{b}_1, \mathbf{b}_2\rangle \leq 0$ from the expansion of $\|\mathbf{b}_1 + \mathbf{b}_2 + 2 \cdot \mathbf{b}_3\|^2 \leq \|\mathbf{b}_1 - \mathbf{b}_2\|^2$.

In the second situation, expanding $\|\mathbf{b}_1 + 2 \cdot \mathbf{b}_2 + \mathbf{b}_3\|^2 \leq \|\mathbf{b}_1 + \mathbf{b}_3\|^2$ gives the length equality. In the case of the additional hypothesis $\|\mathbf{b}_1\| = \|\mathbf{b}_2\| = \|\mathbf{b}_3\|$, the basis $[\mathbf{b}_1, \mathbf{b}_3, \mathbf{b}_2]_\leq$ is also reduced and we can apply the result of the first situation. This last argument also holds in the third situation. □

A.9.2 Voronoï Cells in the Case of ε-Greedy-Reduced Vectors

We extend the results of the previous subsection to the case of ε-greedy-reduced vectors. The idea is that if we compactify the set of Minkowski-reduced bases and slightly enlarge it, the possible Voronoï coords remain the same. Unfortunately, by doing so, some of the vectors we consider may be zero and this creates an infinity of possible Voronoï coords : for example, if $\mathbf{b}_1 = \mathbf{0}$, any pair $(x_1, 0)$ is a Voronoï coord of $[\mathbf{b}_1, \mathbf{b}_2]_\leq$. To tackle this problem, we restrict to vectors $\mathbf{b}_i$ with "similar" lengths. More precisely, we use the so-called Topological Lemma : if we can guarantee that the possible Voronoï

coords of the enlargement of the initial compact set of bases are bounded, then for a sufficiently small enlargement, the possible Voronoï coords remain the same. We first give rather simple results on ε-greedy-reduced vectors and their Gram-Schmidt orthogonalisation, then we introduce the Topological Lemma (Lemma A.20), from which we finally derive the relaxed versions of Lemmata A.15, A.16 and A.17.

Lemma A.18 *For any $\varepsilon > 0$, if $[\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3]_{\leq}$ are ε-greedy-reduced, then the following inequalities hold :*

$$\begin{aligned} \forall i < j,\ |\langle \mathbf{b}_i, \mathbf{b}_j \rangle| &\leq \frac{1+\varepsilon}{2} \cdot \|\mathbf{b}_i\|^2, \\ \forall s_1, s_2 \in \{-1, 1\}, |\langle \mathbf{b}_3, s_1 \cdot \mathbf{b}_1 + s_2 \cdot \mathbf{b}_2 \rangle| &\leq \frac{1+\varepsilon}{2} \cdot \|s_1 \cdot \mathbf{b}_1 + s_2 \cdot \mathbf{b}_2\|^2. \end{aligned}$$

Proof. Since $[\mathbf{b}_1, \mathbf{b}_2]_{\leq}$ are ε-greedy-reduced, we have that $\mathbf{b}'_2 \in (1+\varepsilon) \cdot \mathrm{Vor}(\mathbf{b}_1)$, where $\mathbf{b}'_2$ is the orthogonal projection of the vector $\mathbf{b}_2$ onto the span of $\mathbf{b}_1$. As a consequence, we can write $\mathbf{b}'_2 = (1+\varepsilon) \cdot \mathbf{u}$ with $\mathbf{u} \in \mathrm{Vor}(\mathbf{b}_1)$. By expanding the inequalities $\|\mathbf{u} \pm \mathbf{b}_1\|^2 \geq \|\mathbf{u}\|^2$, we obtain that $|\langle \mathbf{u}, \mathbf{b}_1 \rangle| \leq \|\mathbf{b}_1\|^2/2$. Therefore, $|\langle \mathbf{b}_3, \mathbf{b}_1 \rangle| \leq \frac{1+\varepsilon}{2} \cdot \|\mathbf{b}_1\|^2$.

Moreover $[\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3]_{\leq}$ are ε-greedy-reduced, so that $\mathbf{b}'_3 \in (1+\varepsilon) \cdot \mathrm{Vor}(\mathbf{b}_1, \mathbf{b}_2)$, where $\mathbf{b}'_3$ is the orthogonal projection of the vector $\mathbf{b}_3$ onto the span of $[\mathbf{b}_1, \mathbf{b}_2]_{\leq}$. As a consequence, we can write $\mathbf{b}'_3 = (1+\varepsilon) \cdot \mathbf{u}$ with $\mathbf{u} \in \mathrm{Vor}(\mathbf{b}_1, \mathbf{b}_2)$. We proceed exactly as above by expanding the inequalities $\|\mathbf{u} + s_1 \cdot \mathbf{b}_1 + s_2 \cdot \mathbf{b}_2\|^2 \geq \|\mathbf{u}\|^2$ for any $s_1, s_2 \in \{-1, 0, 1\}$, and this provides the result. □

The previous lemma implies that if $[\mathbf{b}_1, \ldots, \mathbf{b}_d]_{\leq}$ are ε-greedy-reduced, the only case for which the $\mathbf{b}_i$'s can be linearly dependent is when some of them are zero, but this case cannot be avoided since we need compactifying the set of Minkowski-reduced bases. The following lemma generalises Lemma A.14. It shows that even with ε-greedy-reduced vectors, if the dimension is below four then the Gram-Schmidt orthogonalisation process cannot arbitrarily decrease the lengths of the initial vectors.

Lemma A.19 *There exists $C > 0$ such that for any $1 \leq d \leq 4$ and any sufficiently small $\varepsilon > 0$, if $[\mathbf{b}_1, \ldots, \mathbf{b}_d]_{\leq}$ are ε-greedy-reduced vectors, then we have $\|\mathbf{b}^*_d\| \geq C \cdot \|\mathbf{b}_d\|$.*

Proof. Since $[\mathbf{b}_1, \ldots, \mathbf{b}_d]_{\leq}$ are ε-greedy-reduced, we know that if $\mathbf{b}'_d$ is the orthogonal projection of the vector $\mathbf{b}_d$ onto the span of $[\mathbf{b}_1, \ldots, \mathbf{b}_{d-1}]_{\leq}$, then $\mathbf{b}'_d \in (1+\varepsilon) \cdot \mathrm{Vor}(\mathbf{b}_1, \ldots, \mathbf{b}_{d-1})$. Therefore, because of Lemma A.14 and because the vectors are ordered,

$$\|\mathbf{b}'_d\| \leq (1+\varepsilon)\frac{\sqrt{d-1}}{2} \cdot \|\mathbf{b}_{d-1}\| \leq (1+\varepsilon)\frac{\sqrt{d-1}}{2} \cdot \|\mathbf{b}_d\|.$$

Besides, from the Pythagorean theorem, we have $\|\mathbf{b}_d\|^2 = \|\mathbf{b}^*_d\|^2 + \|\mathbf{b}'_d\|^2$, which completes the proof. □

The Topological Lemma is the key argument when extending the results on possible Voronoï coords from Minkowski-reduced bases to ε-greedy-reduced vectors. When applying it, X_0 will correspond to the x_i's, K_0 to the $\mathbf{b}_i$'s that are ε-greedy-reduced, X to the possible Voronoï coordinates, K to a compact subset of the Minkowski-reduced bases, and f to the continuous function of real variables $f : (y_i)_{i \leq d}, (\mathbf{b}_i)_{i \leq d} \longrightarrow \|y_1 \mathbf{b}_1 + \ldots + y_d \mathbf{b}_d\|$.

Lemma A.20 (Topological Lemma) *Let $n, m \geq 1$. Let X_0 and K_0 be compact sets of $\mathbb{R}^n$ and $\mathbb{R}^m$. Let f be a continuous function from $K_0 \times X_0$ to $\mathbb{R}$. For any $a \in K_0$ we define $M_a = \{x \in X_0 \cap \mathbb{Z}^n, f(a,x) = \min_{x' \in X_0 \cap \mathbb{Z}^n} f(a,x')\}$. Let $K \subset K_0$ be a compact and $X = \cup_{a \in K} M_a \subset X_0 \cap \mathbb{Z}^n$. With these notations, there exists $\varepsilon > 0$ such that if $b \in K_0$ satisfies $\mathrm{dist}\,(b, K) \leq \varepsilon$, we have $M_b \subset X$.*

Proof. First, all the notations of the result make sense : $X_0 \cap \mathbb{Z}^n$ is finite so the minimum of $f(a,\cdot)$ over it does exist and M_a is finite. Since $X \subset X_0 \cap \mathbb{Z}^n$, X is finite. Finally, since K is compact, the notation $\mathrm{dist}\,(\cdot, K)$ makes sense too.

For each $x \in X_0$ we define $K_x = \{a \in K_0, x \in M_a\}$. The set K_x is compact. Indeed, it is obviously bounded, and if (a_k) is a sequence of elements of K_x that converges towards an $a \in K_0$, we show that $s \in K_x$. For all $x' \in X_0 \cap \mathbb{Z}^n$ and for all k, we have $f(a_k, x) \leq f(a_k, x')$. By continuity, this holds for a too, which proves that $x \in M_a$.

Now we fix an $x \in X_0 \cap \mathbb{Z}^n \setminus X$. Since K_x and K are both compact and $x \notin X$ (which implies $K \cap K_x = \emptyset$), $\mathrm{dist}\,(K_x, K) > 0$. Since $(X_0 \cap \mathbb{Z}^n) \setminus X$ is finite, in order to end the proof it is sufficient to set $\varepsilon = \frac{1}{2} \min\,(dist(K_x, K), x \in (X_0 \cap \mathbb{Z}^n) \setminus X)$. □

In order to apply the Topological Lemma, we need to map the relaxed bases into a compact set. For any $\varepsilon \geq 0$ and any $\alpha \in [0,1]$, we define :

$$\begin{aligned} K_2(\varepsilon,\alpha) &= \{(\mathbf{b}_1,\mathbf{b}_2), \mathbf{b}_1, \mathbf{b}_2 \ \varepsilon\text{-greedy-reduced}, \ \alpha \leq \|\mathbf{b}_1\| \leq \|\mathbf{b}_2\| = 1\} \\ K_3(\varepsilon,\alpha) &= \{(\mathbf{b}_1,\mathbf{b}_2,\mathbf{b}_3), \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3 \ \varepsilon\text{-greedy-reduced}, \ \alpha \leq \|\mathbf{b}_1\| \leq \|\mathbf{b}_2\| \leq \|\mathbf{b}_3\| = 1\}. \end{aligned}$$

Lemma A.21 *If $\varepsilon \geq 0$ and $\alpha \in [0,1]$, $K_2(\varepsilon,\alpha)$ and $K_3(\varepsilon,\alpha)$ are compact sets.*

The following lemma is the relaxed version of Lemma A.15. It can also be viewed as a reciprocal to Lemma A.18.

Lemma A.22 *There exists $\varepsilon > 0$ such that for any $\alpha \in (0,1]$, the possible Voronoï coords of $[\mathbf{b}_1,\mathbf{b}_2]_\leq \in K_2(\varepsilon,\alpha)$ are the same as for Minkowski-reduced bases, i.e., $(1,0)$ and $(1,1)$, modulo any change of signs and permutation of coordinates.*

Proof. Recall that there is a set of possible Voronoï coords for each non-zero coset of $(\mathbb{Z}/2\mathbb{Z})^2$: we look at the minima of the cosets of $L/2L$. Since there is a finite number of such cosets (three in dimension two), we treat them separately. Let $(a_1,a_2) \in \{0,1\}^2$. We are looking for the pairs $(k_1,k_2) \in \mathbb{Z}^2$ that minimise the quantity $\|(a_1+2k_1)\cdot\mathbf{b}_1 + (a_2+2k_2)\cdot\mathbf{b}_2\|$, where $[\mathbf{b}_1,\mathbf{b}_2]_\leq$ are ε-greedy-reduced. We first prove that the minimum over (k_1,k_2) can be taken over a finite domain. Notice that :

$$2 \geq \|\mathbf{b}_1\| + \|\mathbf{b}_2\| \geq \|a_1\cdot\mathbf{b}_1 + a_2\cdot\mathbf{b}_2\| \geq \|(a_1+2k_1)\cdot\mathbf{b}_1 + (a_2+2k_2)\cdot\mathbf{b}_2\|.$$

Moreover, Lemma A.19 gives that :

$$\|(a_1+2k_1)\cdot\mathbf{b}_1 + (a_2+2k_2)\cdot\mathbf{b}_2\| \geq |a_2+2k_2|\cdot\|\mathbf{b}_2^*\| \geq |a_2+2k_2|C,$$

which gives the result for k_2. By applying the triangular inequality, we get the result for k_1 :

$$|a_1+2k_1|\alpha \leq \|(a_1+2k_1)\cdot\mathbf{b}_1\| \leq 2 + \|(a_2+2k_2)\cdot\mathbf{b}_2\| \leq 2 + |a_2+2k_2|.$$

From this we deduce that $(k_1,k_2) \in \mathbb{Z}^2$ can be bounded independently of $(\mathbf{b}_1,\mathbf{b}_2)$. From Lemma A.21, we know that $K_2(\varepsilon,\alpha)$ is compact and therefore we can apply the Topological Lemma. This gives the expected result. □

We now relax Lemma A.16 in the same manner. To do this, we proceed exactly like in the proof above.

Lemma A.23 *There exists $\varepsilon > 0$ such that for any $\alpha \in (0,1]$, the possible Voronoï coords of $[\mathbf{b}_1,\mathbf{b}_2,\mathbf{b}_3]_\leq \in K_3(\varepsilon,\alpha)$ are the same as for Minkowski-reduced bases.*

Proof. We consider each non-zero coset of $L/2L$ separately (there are seven of them in dimension three). Let $(a_1, a_2, a_3) \in \{0,1\}^3$. We are looking for the triples $(k_1, k_2, k_3) \in \mathbb{Z}^3$ minimising the quantity $\|(a_1+2k_1)\cdot\mathbf{b}_1 + (a_2+2k_2)\cdot\mathbf{b}_2 + (a_3+2k_3)\cdot\mathbf{b}_3\|$, where $[\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3]_{\leq}$ are ε-greedy-reduced. In order to apply Lemma A.21, from which the result can be deduced easily, it is sufficient to prove that the minimum over (k_1, k_2, k_3) can be taken over a finite domain. Notice that :

$$3 \geq \|a_1\cdot\mathbf{b}_1 + a_2\cdot\mathbf{b}_2 + a_3\cdot\mathbf{b}_3\| \geq \|(a_1+2k_1)\cdot\mathbf{b}_1 + (a_2+2k_2)\cdot\mathbf{b}_2 + (a_3+2k_3)\cdot\mathbf{b}_3\|.$$

Moreover, Lemma A.19 gives that :

$$\|(a_1+2k_1)\cdot\mathbf{b}_1 + (a_2+2k_2)\cdot\mathbf{b}_2 + (a_3+2k_3)\cdot\mathbf{b}_3\| \geq |a_3+2k_3|\cdot\|\mathbf{b}_3^*\| \geq |a_3+2k_3|C,$$

for any sufficiently small $\varepsilon > 0$. This gives the result for k_3.

From the triangular inequality, Lemma A.19, and the fact that $\|\mathbf{b}_2\| \geq \alpha$, we have :

$$3 + |a_3+2k_3| \geq \|(a_1+2k_1)\cdot\mathbf{b}_1 + (a_2+2k_2)\cdot\mathbf{b}_2\| \geq |a_2+2k_2|\cdot\|\mathbf{b}_2^*\| \geq |a_2+2k_2|C\alpha.$$

The fact that k_3 is bounded ensures that k_2 is bounded.

To obtain the result on k_1, it suffices to apply the triangular inequality once more :

$$3 + |a_3+2k_3| + |a_2+2k_2| \geq \|(a_1+2k_1)\cdot\mathbf{b}_1\| \geq |a_1+2k_1|\alpha.$$

□

The following result generalises Lemma A.17 about the possible Voronoï coord $(1,1,2)$. As opposed to the two previous results, there is no need using the Topological Lemma in this case, because only a finite number of (x_1, x_2, x_3)'s is considered.

Lemma A.24 *There exists $c > 0$ such that for any sufficiently small $\varepsilon > 0$, if $[\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3]_{\leq}$ are ε-greedy-reduced and $\|\mathbf{b}_3\| = 1$, then :*

1. *If $(s_1, s_2, 2)$ is a Voronoï coord with $s_i = \pm 1$ for $i \in \{1,2\}$, then : $\|\mathbf{b}_1\| \geq 1 - c\varepsilon$, $|\langle\mathbf{b}_1, \mathbf{b}_2\rangle| \leq c\varepsilon$ and $\left|\langle\mathbf{b}_i, \mathbf{b}_3\rangle + \frac{s_i}{2}\cdot\|\mathbf{b}_1\|^2\right| \leq c\varepsilon$ for $i \in \{1,2\}$.*
2. *If $(s_1, 2, s_3)$ is a Voronoï coord with $s_i = \pm 1$ for $i \in \{1,3\}$, then $(1-c\varepsilon)\cdot\|\mathbf{b}_2\| \leq \|\mathbf{b}_1\| \leq \|\mathbf{b}_2\|$. Moreover, if $\|\mathbf{b}_1\| \geq 1-\varepsilon$, then : $|\langle\mathbf{b}_1, \mathbf{b}_3\rangle| \leq c\varepsilon$ and $\left|\langle\mathbf{b}_i, \mathbf{b}_2\rangle + \frac{s_i}{2}\cdot\|\mathbf{b}_1\|^2\right| \leq c\varepsilon$ for $i \in \{1,3\}$.*
3. *If $(2, s_2, s_3)$ is a Voronoï coord with $s_i = \pm 1$ for $i \in \{2,3\}$ and if $\|\mathbf{b}_1\| \geq 1-\varepsilon$, then : $|\langle\mathbf{b}_2, \mathbf{b}_3\rangle| \leq c\varepsilon$ and $\left|\langle\mathbf{b}_i, \mathbf{b}_1\rangle + \frac{s_i}{2}\cdot\|\mathbf{b}_1\|^2\right| \leq c\varepsilon$ for $i \in \{2,3\}$.*

The proof of this result is a straightforward modification of the proof of Lemma A.17.

Proof. Wlog we suppose that for any i, we have s_i=1. The proofs of the other cases are very similar. We only prove the statement in the case of the first situation.

We consider the inequality $\|\mathbf{b}_1+\mathbf{b}_2+2\cdot\mathbf{b}_3\|^2 \leq \|\mathbf{b}_1+\mathbf{b}_2\|^2$: it is equivalent to $\|\mathbf{b}_3\|^2 + \langle\mathbf{b}_1, \mathbf{b}_3\rangle + \langle\mathbf{b}_2, \mathbf{b}_3\rangle \leq 0$. Since $[\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3]_{\leq}$ are ε-greedy-reduced, by using Lemma A.18 we obtain that we must have $\langle\mathbf{b}_1, \mathbf{b}_3\rangle \geq -\frac{1+\varepsilon}{2}\cdot\|\mathbf{b}_1\|^2$ and $\langle\mathbf{b}_2, \mathbf{b}_3\rangle \geq -\frac{1+\varepsilon}{2}\cdot\|\mathbf{b}_2\|^2$. Thus $2\cdot\|\mathbf{b}_3\|^2 - (1+\varepsilon)\cdot\|\mathbf{b}_1\|^2 - (1+\varepsilon)\cdot\|\mathbf{b}_2\|^2 \leq 0$. Consequently, the length "quasi-equality" holds, and the inequalities on the scalar products above are "quasi-equalities".

It remains to prove that $|\langle\mathbf{b}_1, \mathbf{b}_2\rangle|$ is very small. Expanding the relation $\|\mathbf{b}_1+\mathbf{b}_2+2\cdot\mathbf{b}_3\|^2 \leq \|\mathbf{b}_1 - \mathbf{b}_2\|^2$ gives that $\langle\mathbf{b}_1, \mathbf{b}_2\rangle \leq c_1\varepsilon$ for some constant $c_1 > 0$. Similarly, expanding the relation $|\langle\mathbf{b}_3, \mathbf{b}_1 + \mathbf{b}_2\rangle| \leq \frac{1+\varepsilon}{2}\cdot\|\mathbf{b}_1+\mathbf{b}_2\|^2$ (which comes from Lemma A.18 gives $\langle\mathbf{b}_1, \mathbf{b}_2\rangle \geq -c_2\varepsilon$ for some constant $c_2 > 0$.

□

A.9.3 The Gap Lemma

The goal of this subsection is to prove that even with relaxed bases, if one adds a lattice vector with not too small coordinates to a vector of the Voronoï cell, this vector becomes significantly longer. This result was used the other way round : if the x_i's found at Step 5 of the recursive greedy algorithm are not too small, then the vector $\mathbf{b}_d$ is significantly shorter than the vector $\mathbf{a}_d$. We first generalise the compact sets K_2 and K_3. For any $\varepsilon \geq 0$ and any $\alpha \in [0,1]$, we define :

$$\begin{aligned} K_2'(\varepsilon,\alpha) &= \{(\mathbf{b}_1,\mathbf{b}_2,\mathbf{u}), (\mathbf{b}_1,\mathbf{b}_2) \in K_2(\varepsilon,\alpha) \text{ and } \mathbf{u} \in \mathrm{Vor}(\mathbf{b}_1,\mathbf{b}_2)\} \\ K_3'(\varepsilon,\alpha) &= \{(\mathbf{b}_1,\mathbf{b}_2,\mathbf{b}_3,\mathbf{u}), (\mathbf{b}_1,\mathbf{b}_2,\mathbf{b}_3) \in K_3(\varepsilon,\alpha) \text{ and } \mathbf{u} \in \mathrm{Vor}(\mathbf{b}_1,\mathbf{b}_2)\}. \end{aligned}$$

Lemma A.25 *If $\varepsilon \geq 0$ and $\alpha \in [0,1]$, the sets $K_2'(\varepsilon,\alpha)$ and $K_3'(\varepsilon,\alpha)$ are compact.*

Proof. Since the proofs in the two and three dimensional cases are the same, we only consider $K_2'(\varepsilon,\alpha)$. It is sufficient to show that $K_2'(\varepsilon,\alpha)$ is closed and bounded. The fact it is bounded is obvious since $\|\mathbf{b}_1\| \leq \|\mathbf{b}_2\| = 1$ and $\|\mathbf{u}\| \leq \|\mathbf{b}_1\| + \|\mathbf{b}_2\| \leq 2$. We suppose now that $K_2'(\varepsilon,\alpha)$ is not closed and we look for a contradiction. Let $\left(\mathbf{b}_1^{(n)}, \mathbf{b}_2^{(n)}, \mathbf{u}^{(n)}\right)$ be a sequence of elements of $K_2'(\varepsilon,\alpha)$ that converges to $(\mathbf{b}_1,\mathbf{b}_2,\mathbf{u}) \notin K_2'(\varepsilon,\alpha)$. By definition of $K_2'(\varepsilon,\alpha)$, there exists $(x_1,x_2) \neq (0,0)$ such that $\|x_1 \cdot \mathbf{b}_1 + x_2 \cdot \mathbf{b}_2 + \mathbf{u}\| > \|\mathbf{u}\|$. Thus there exists an integer $n \geq 0$ such that $\left\|x_1\mathbf{b}_1^{(n)} + x_2\mathbf{b}_2^{(n)} + \mathbf{u}^{(n)}\right\| > \left\|\mathbf{u}^{(n)}\right\|$. In that case, we have $\mathbf{u}^{(n)} \notin \mathrm{Vor}\left(\mathbf{b}_1^{(n)}, \mathbf{b}_2^{(n)}\right)$, which is impossible. □

The next result is the two-dimensional version of the Gap Lemma.

Lemma A.26 *There exist two constants $\varepsilon, C > 0$ such that for any ε-greedy-reduced vectors $[\mathbf{b}_1,\mathbf{b}_2]_\leq$ and any $\mathbf{u} \in \mathrm{Vor}(\mathbf{b}_1,\mathbf{b}_2)$, if at least one of the following conditions holds, then : $\|\mathbf{u}+x_1\mathbf{b}_1+x_2\mathbf{b}_2\|^2 \geq \|\mathbf{u}\|^2 + C\|\mathbf{b}_2\|^2$.*

1. *$|x_2| \geq 2$,*
2. *$|x_1| \geq 2$ and $\|\mathbf{b}_1\| \geq (1-\varepsilon) \cdot \|\mathbf{b}_2\|$.*

Proof. The proof involves three claims. The first one helps compactifying the set of the variables $[\mathbf{b}_1,\mathbf{b}_2]$. The second one shows that we can suppose that the vectors $\mathbf{b}_1$ and $\mathbf{b}_2$ have similar lengths, and the third one is the key step when showing that we can use Lemma A.22 to end the proof.

Claim 1 : Wlog we can suppose $\|\mathbf{b}_2\| = 1$.

Let $(*)$ be the following statement : "There exist two constants $\varepsilon, C > 0$ such that for any ε-greedy-reduced vectors $[\mathbf{b}_1,\mathbf{b}_2]_\leq$ with $\|\mathbf{b}_2\| = 1$, and any $\mathbf{u} \in \mathrm{Vor}(\mathbf{b}_1,\mathbf{b}_2)$, if one of the following conditions is not satisfied then $\|\mathbf{u} + x_1 \cdot \mathbf{b}_1 + x_2 \cdot \mathbf{b}_2\|^2 \geq \|\mathbf{u}\|^2 + C$.
– A- $|x_2| \geq 2$,
– B- $|x_1| \geq 2$ and $\|\mathbf{b}_1\| \geq 1 - \varepsilon$."
We keep the same constants $\varepsilon, C > 0$. Let $[\mathbf{b}_1,\mathbf{b}_2]_\leq$ be ε-greedy-reduced vectors. If $\|\mathbf{b}_2\| = 0$, the result is obvious. Otherwise, let $\mathbf{b}_i' = \frac{1}{\|\mathbf{b}_2\|} \cdot \mathbf{b}_i$ for $i \in \{1,2\}$. We apply $(*)$ to the ε-greedy-reduced vectors $[\mathbf{b}_1',\mathbf{b}_2']_\leq$. Let $\mathbf{u} \in \mathrm{Vor}(\mathbf{b}_1,\mathbf{b}_2)$ and $\mathbf{u}' = \frac{1}{\|\mathbf{b}_2\|} \cdot \mathbf{u}$. Then $\mathbf{u}' \in \mathrm{Vor}(\mathbf{b}_1',\mathbf{b}_2')$. If condition (1) or condition (2) is satisfied, then $\|\mathbf{u}' + x_1 \cdot \mathbf{b}_1' + x_2 \cdot \mathbf{b}_2'\|^2 \geq \|\mathbf{u}'\|^2 + C$. Multiplying both sides by $\|\mathbf{b}_2\|^2$ gives the result. □

We now distinguish two cases : either the vector $\mathbf{b}_1$ has approximately the length of the vector $\mathbf{b}_2$, or it is far shorter (which cannot happen in situation (2)). The idea of the following claim is that

when $\mathbf{b}_1$ converges to $\mathbf{0}$, $(\mathbf{b}_1, \mathbf{b}_2, \mathbf{u})$ converges to $(0, \mathbf{b}_2, \mathbf{u}')$ where $\mathbf{u}'$ is close to $\mathrm{Vor}(\mathbf{b}_2)$.

Claim 2 : There exist $C, \alpha, \varepsilon > 0$ such that for any $(\mathbf{b}_1, \mathbf{b}_2, \mathbf{u}) \in K_2'(\varepsilon, 0)$ with $\|\mathbf{b}_1\| \leq \alpha$, as soon as $|x_2| \geq 2$:

$$\|\mathbf{u} + x_1 \cdot \mathbf{b}_1 + x_2 \cdot \mathbf{b}_2\|^2 - \|\mathbf{u}\|^2 \geq C.$$

Since $\mathbf{u} \in \mathrm{Vor}(\mathbf{b}_1, \mathbf{b}_2)$, we have $|\langle \mathbf{u}, \mathbf{b}_i \rangle| \leq \|\mathbf{b}_i\|^2/2$ for $i \in \{1, 2\}$. Therefore :

$$\begin{aligned} \|\mathbf{u} + x_1 \cdot \mathbf{b}_1 + x_2 \cdot \mathbf{b}_2\|^2 - \|\mathbf{u}\|^2 &= \|x_1 \cdot \mathbf{b}_1 + x_2 \cdot \mathbf{b}_2\|^2 + 2\langle \mathbf{u}, x_1 \cdot \mathbf{b}_1 + x_2 \cdot \mathbf{b}_2 \rangle \\ &\geq \|x_1 \cdot \mathbf{b}_1 + x_2 \cdot \mathbf{b}_2\|^2 - |x_1| \cdot \|\mathbf{b}_1\|^2 - |x_2| \\ &\geq (x_1^2 - |x_1|) \cdot \|\mathbf{b}_1\|^2 + (x_2^2 - |x_2|) - 2|x_1 x_2| \cdot |\langle \mathbf{b}_1, \mathbf{b}_2 \rangle|. \end{aligned}$$

Since $[\mathbf{b}_1, \mathbf{b}_2]_{\leq}$ are ε-greedy-reduced, by Lemma A.18 we have $|\langle \mathbf{b}_1, \mathbf{b}_2 \rangle| \leq \frac{1+\varepsilon}{2} \cdot \|\mathbf{b}_1\|^2$, from which we get :

$$\|\mathbf{u} + x_1 \cdot \mathbf{b}_1 + x_2 \cdot \mathbf{b}_2\|^2 - \|\mathbf{u}\|^2 \geq |x_1|(|x_1| - |x_2|(1+\varepsilon) - 1) \cdot \|\mathbf{b}_1\|^2 + (x_2^2 - |x_2|).$$

We now minimise this last expression as regard to the variable $|x_1|$ (this is a degree-2 polynomial), and with the extremal choice "$|x_1| = \frac{(1+\varepsilon)|x_2|+1}{2}$" we obtain :

$$\begin{aligned} \|\mathbf{u} + x_1 \cdot \mathbf{b}_1 + x_2 \cdot \mathbf{b}_2\|^2 - \|\mathbf{u}\|^2 &\geq -\frac{(|x_2|(1+\varepsilon)+1^2}{4} \cdot \|\mathbf{b}_1\|^2 + (x_2^2 - |x_2|) \\ &\geq \left(1 - \frac{\alpha^2(1+\varepsilon)^2}{4}\right)|x_2|^2 - \left(1 + \frac{\alpha^2(1+\varepsilon)}{2}\right)|x_2| - \frac{\alpha^2}{4} \end{aligned}$$

For a small enough α, the minimum of this degree-2 polynomial in $|x_2|$ is reached for "$|x_2| \leq 2$", and is increasing as regard to $|x_2| \geq 2$. By hypothesis $|x_2| \geq 2$, therefore we have :

$$\|\mathbf{u} + x_1 \cdot \mathbf{b}_1 + x_2 \cdot \mathbf{b}_2\|^2 - \|\mathbf{u}\|^2 \geq 2 - \alpha^2(9/4 + 3\varepsilon + \varepsilon^2),$$

which gives the result. □

The constant $\alpha > 0$ is fixed to satisfy the constraint of Claim 2. We fix $\varepsilon > 0$ to satisfy the constraints of Claim 2 and Lemmata A.19 and A.22. Consider a sequence $\left(\mathbf{b}_1^{(k)}, \mathbf{b}_2^{(k)}, \mathbf{u}^{(k)}, x_1^{(k)}, x_2^{(k)}\right)_k$ such that :

1. $\left(\mathbf{b}_1^{(k)}, \mathbf{b}_2^{(k)}, \mathbf{u}^{(k)}\right) \in K_2'(\varepsilon, \alpha)$,
2. $x_1^{(k)}, x_2^{(k)}$ are integers with $|x_2^{(k)}| \geq 2$ (respectively $|x_1^{(k)}| \geq 2$),
3. $\left\|\mathbf{u}^{(k)} + x_1^{(k)} \cdot \mathbf{b}_1^{(k)} + x_2^{(k)} \cdot \mathbf{b}_2^{(k)}\right\|^2 - \left\|\mathbf{u}^{(k)}\right\|^2 \to 0$ when $k \to \infty$.

If no such sequence exists, then (1) (respectively (2)) is proved. To end the proof of Lemma A.26, suppose to the contrary that such a sequence does exist : there will be a contradiction with Voronoï coords.

Claim 3 : For any sufficiently small $\varepsilon > 0$ and for any $\alpha > 0$, the sequences $x_2^{(k)}$ and $x_1^{(k)}$ remain bounded.

Suppose that this is not the case. We show that this implies that $\left\|\mathbf{u}^{(k)} + x_1^{(k)} \cdot \mathbf{b}_1^{(k)} + x_2^{(k)} \cdot \mathbf{b}_2^{(k)}\right\|^2 - \left\|\mathbf{u}^{(k)}\right\|^2$ is not bounded, which is impossible. By the Cauchy-Schwarz inequality, we have :

$$\left\|\mathbf{u}^{(k)} + x_1^{(k)}\mathbf{b}_1^{(k)} + x_2^{(k)}\mathbf{b}_2^{(k)}\right\|^2 - \left\|\mathbf{u}^{(k)}\right\|^2 \geq \left\|x_1^{(k)}\mathbf{b}_1^{(k)} + x_2^{(k)}\mathbf{b}_2^{(k)}\right\| \left(\left\|x_1^{(k)}\mathbf{b}_1^{(k)} + x_2^{(k)}\mathbf{b}_2^{(k)}\right\| - 2\left\|\mathbf{u}^{(k)}\right\|\right).$$

Since $\left\|\mathbf{u}^{(k)}\right\|$ is bounded, it suffices to show that $\left\|x_1^{(k)} \cdot \mathbf{b}_1^{(k)} + x_2^{(k)} \cdot \mathbf{b}_2^{(k)}\right\|$ is not bounded. From Lemma A.19, we have :

$$\left\|x_1^{(k)} \cdot \mathbf{b}_1^{(k)} + x_2^{(k)} \cdot \mathbf{b}_2^{(k)}\right\| \geq \left|x_2^{(k)}\right| \cdot \left\|\mathbf{b}_2^{(k)*}\right\| \geq C \left|x_2^{(k)}\right|.$$

Therefore, the sequence $x_2^{(k)}$ is bounded. The triangular inequality and the fact that $\left\|\mathbf{b}_1^{(k)}\right\| \geq \alpha$ ensure that the sequence $x_1^{(k)}$ remains bounded too. □

The previous claim and Lemma A.25 imply that the sequence $\left(\mathbf{b}_1^{(k)}, \mathbf{b}_2^{(k)}, \mathbf{u}^{(k)}, x_1^{(k)}, x_2^{(k)}\right)$ remains in a compact subset of $\mathbb{R}^{3n} \times \mathbb{Z}^2$. Therefore we can extract a subsequence that converges to a limit $(\mathbf{b}_1, \mathbf{b}_2, \mathbf{u}, x_1, x_2)$ that lies in the same compact and satisfies : $\|\mathbf{u} + x_1 \cdot \mathbf{b}_1 + x_2 \cdot \mathbf{b}_2\| = \|\mathbf{u}\|$. This means that $x_1 \cdot \mathbf{b}_1 + x_2 \cdot \mathbf{b}_2$ is a Voronoï vector of the lattice spanned by $[\mathbf{b}_1, \mathbf{b}_2]_{\leq} \in K_2'(\varepsilon, \alpha)$, which contradicts Lemma A.22. □

We now give the three-dimensional Gap Lemma, on which relies the analysis of the four-dimensional greedy algorithm.

Lemma A.27 *There exist two constants $\varepsilon, C > 0$ such that for any ε-greedy-reduced vectors $[\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3]_{\leq}$ and any $\mathbf{u} \in \mathrm{Vor}(\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$, if at least one of the following conditions holds, then :*

$$\|\mathbf{u} + x_1 \cdot \mathbf{b}_1 + x_2 \cdot \mathbf{b}_2 + x_3 \cdot \mathbf{b}_3\|^2 \geq \|\mathbf{u}\|^2 + C \cdot \|\mathbf{b}_3\|^2.$$

1. $|x_3| \geq 3$, *or* $|x_3| = 2$ *with* $(|x_1|, |x_2|) \neq (1, 1)$.
2. $\|\mathbf{b}_2\| \geq (1 - \varepsilon) \cdot \|\mathbf{b}_3\|$ *and :* $|x_2| \geq 3$, *or* $|x_2| = 2$ *with* $(|x_1|, |x_3|) \neq (1, 1)$.
3. $\|\mathbf{b}_1\| \geq (1 - \varepsilon) \cdot \|\mathbf{b}_3\|$ *and :* $|x_1| \geq 3$, *or* $|x_1| = 2$ *with* $(|x_2|, |x_3|) \neq (1, 1)$.

Proof. It is easy to see that similarly to Lemma A.26 we can suppose that $\|\mathbf{b}_3\| = 1$. We consider three cases : both vectors $\mathbf{b}_1$ and $\mathbf{b}_2$ are significantly shorter than the vector $\mathbf{b}_3$, the vector $\mathbf{b}_1$ is very short but the vectors $\mathbf{b}_2$ and $\mathbf{b}_3$ have similar lengths, and finally all the vectors have similar lengths.

Claim 1 : There exist $C, \alpha, \varepsilon > 0$ such that for any $(\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \mathbf{u}) \in K_3'(\varepsilon, 0)$ with $\|\mathbf{b}_2\| \leq \alpha$, as soon as $|x_3| \geq 2$, we have $\|\mathbf{u} + x_1 \cdot \mathbf{b}_1 + x_2 \cdot \mathbf{b}_2 + x_3 \cdot \mathbf{b}_3\|^2 - \|\mathbf{u}\|^2 \geq C$.

Because the vector $\mathbf{u}$ lies in the Voronoï cell of the lattice spanned by the vectors $\mathbf{b}_1, \mathbf{b}_2$ and $\mathbf{b}_3$, and because of Lemma A.18, we have the following inequalities :

$$\begin{aligned}
\|\mathbf{u} + x_1\mathbf{b}_1 + x_2\mathbf{b}_2 + x_3\mathbf{b}_3\|^2 - \|\mathbf{u}\|^2 &= \|x_1\mathbf{b}_1 + x_2\mathbf{b}_2 + x_3\mathbf{b}_3\|^2 + 2\langle \mathbf{u}, x_1\mathbf{b}_1 + x_2\mathbf{b}_2 + x_3\mathbf{b}_3\rangle \\
&\geq \left(x_1^2 - (1+\varepsilon)|x_1|(|x_2| + |x_3|) - |x_1|\right) \|\mathbf{b}_1\|^2 \\
&+ \left(x_2^2 - (1+\varepsilon)|x_2\|x_3| - |x_2|\right) \|\mathbf{b}_2\|^2 + x_3^2 - |x_3|.
\end{aligned}$$

We minimise this degree-2 polynomial of the variable $|x_1|$, and with the choice "$x_1 = \frac{(1+\varepsilon)(|x_2|+|x_3|)+1}{2}$", we obtain :

$$\begin{aligned}
\|\mathbf{u} &+ x_1\mathbf{b}_1 + x_2\mathbf{b}_2 + x_3\mathbf{b}_3\|^2 - \|\mathbf{u}\|^2 \\
&\geq -\frac{((1+\varepsilon)(|x_2| + |x_3|) + 1)^2}{4} \|\mathbf{b}_1\|^2 + \left(x_2^2 - (1+\varepsilon)|x_2||x_3| - |x_2|\right) \|\mathbf{b}_2\|^2 + x_3^2 - |x_3| \\
&\geq \left(\frac{3 - 2\varepsilon - \varepsilon^2}{4} x_2^2 - \left(\frac{3 + 4\varepsilon + \varepsilon^2}{2}|x_3| + \frac{3+\varepsilon}{2}\right)|x_2| - \frac{((1+\varepsilon)|x_3| + 1)^2}{4}\right) \|\mathbf{b}_2\|^2 + x_3^2 - |x_3|,
\end{aligned}$$

because $\|\mathbf{b}_1\| \leq \|\mathbf{b}_2\|$. The left-hand term is a degree-2 polynomial in $|x_2|$, whose first coefficient is positive (for a small enough $\varepsilon > 0$). It is lower-bounded by its minimum over $\mathbb{Z}$ and the minimum is reached for $|x_2| = |x_3| + 1$ when $\varepsilon = 0$. This gives :

$$\|\mathbf{u} + x_1 \cdot \mathbf{b}_1 + x_2 \cdot \mathbf{b}_2 + x_3 \cdot \mathbf{b}_3\|^2 - \|\mathbf{u}\|^2 \geq \big(a_2(\varepsilon)x_3^2 + a_1(\varepsilon)|x_3| + a_0(\varepsilon)\big) \cdot \|\mathbf{b}_2\|^2 + x_3^2 - |x_3|,$$

where $a_2(\varepsilon) \to -1, a_2(\varepsilon) \to -2$ and $a_0(\varepsilon) \to -1$ when $\varepsilon \to 0$. When $\varepsilon > 0$ is small enough, the factor in front of $\|\mathbf{b}_2\|^2$ is negative, and since $\|\mathbf{b}_2\| \leq \alpha$, we get :

$$\|\mathbf{u} + x_1 \cdot \mathbf{b}_1 + x_2 \cdot \mathbf{b}_2 + x_3 \cdot \mathbf{b}_3\|^2 - \|\mathbf{u}\|^2 \geq \big(1 + \alpha^2 a_2(\varepsilon)\big) x_3^2 + \big(-1 + \alpha^2 a_1(\varepsilon)\big) |x_3| + \alpha^2 a_0(\varepsilon).$$

For small enough constants $\alpha, \varepsilon > 0$, this degree-2 polynomial of the variable $|x_3|$ is strictly increasing over $[2, \infty)$ so that we can replace $|x_3|$ by 2 in the right hand side :

$$\|\mathbf{u} + x_1 \cdot \mathbf{b}_1 + x_2 \cdot \mathbf{b}_2 + x_3 \cdot \mathbf{b}_3\|^2 - \|\mathbf{u}\|^2 \geq 4\big(1 + \alpha^2 a_2(\varepsilon)\big) - 2\big(1 - \alpha^2 a_1(\varepsilon)\big) + \alpha^2 a_0(\varepsilon).$$

When $\alpha > 0$ is small enough, this quantity becomes larger than a constant $C > 0$.

Claim 2 : There exist $C, c'', \varepsilon, \alpha > 0$ such that for any $(\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \mathbf{u}) \in K_3'(\varepsilon, 0)$ with $\|\mathbf{b}_1\| \leq \alpha$ and $\|\mathbf{b}_2\| \geq c''\alpha$, as soon as $|x_3| \geq 2$, $\|\mathbf{u} + x_1 \cdot \mathbf{b}_1 + x_2 \cdot \mathbf{b}_2 + x_3 \cdot \mathbf{b}_3\|^2 - \|\mathbf{u}\|^2 \geq C$.

The proof looks like the one of Claim 2 of Lemma A.26, but is slightly more technical. We have the following inequalities :

$$\begin{aligned}
\|\mathbf{u} + x_1\mathbf{b}_1 + x_2\mathbf{b}_2 \;+\; & x_3\mathbf{b}_3\|^2 - \|\mathbf{u}\|^2 = \|x_1\mathbf{b}_1 + x_2\mathbf{b}_2 + x_3\mathbf{b}_3\|^2 + 2\langle \mathbf{u}, x_1\mathbf{b}_1 + x_2\mathbf{b}_2 + x_3\mathbf{b}_3\rangle \\
&\geq \big(x_1^2 - (1+\varepsilon)|x_1|(|x_2| + |x_3|) - |x_1|\big) \|\mathbf{b}_1\|^2 \\
&+ \|x_2\mathbf{b}_2 + x_3\mathbf{b}_3\|^2 + 2\langle \mathbf{u}, x_2\mathbf{b}_2 + x_3\mathbf{b}_3\rangle \\
&\geq -\frac{\alpha^2}{4}\left((1+\varepsilon)(|x_2| + |x_3|) + 1\right)^2 + \|x_2\mathbf{b}_2 + x_3\mathbf{b}_3\|^2 + 2\langle \mathbf{u}, x_2\mathbf{b}_2 + x_3\mathbf{b}_3\rangle.
\end{aligned}$$

We write $\mathbf{u} = \mathbf{u}' + \mathbf{u}''$, where $\mathbf{u}'$ is in the span of $[\mathbf{b}_2, \mathbf{b}_3]$, and $\mathbf{u}''$ is orthogonal to it. It is clear that the vector $\mathbf{u}'$ is in the Voronoï cell of $L[\mathbf{b}_2, \mathbf{b}_3]$. By Lemma A.14, we know that $\|\mathbf{u}'\| \leq 1/\sqrt{2}$. Besides, as soon as $|x_3| \geq 2$:

$$\begin{aligned}
\|x_2 \cdot \mathbf{b}_2 + x_3 \cdot \mathbf{b}_3\|^2 &\geq \big(x_2^2 - (1+\varepsilon)|x_2x_3|\big) \cdot \|\mathbf{b}_2\|^2 + x_3^2 \\
&\geq -\frac{((1+\varepsilon)|x_3|)^2}{4} \cdot \|\mathbf{b}_2\|^2 + x_3^2 \\
&\geq x_3^2\left(1 - \frac{(1+\varepsilon)^2}{4}\right) \\
&\geq 3 - 2\varepsilon - \varepsilon^2.
\end{aligned}$$

Consequently $\sqrt{2} \leq \sqrt{\frac{2}{3-2c\varepsilon-c^2\varepsilon^2}} \cdot \|x_2 \cdot \mathbf{b}_2 + x_3 \cdot \mathbf{b}_3\|$, which gives, by using the Cauchy-Schwarz inequality :

$$\langle \mathbf{u}, x_2 \cdot \mathbf{b}_2 + x_3 \cdot \mathbf{b}_3\rangle \geq -\frac{\sqrt{2}}{2} \cdot \|x_2 \cdot \mathbf{b}_2 + x_3 \cdot \mathbf{b}_3\| \geq -\frac{1}{2}\sqrt{\frac{2}{3-2\varepsilon-\varepsilon^2}}\|x_2 \cdot \mathbf{b}_2 + x_3 \cdot \mathbf{b}_3\|^2.$$

From this, we get :

$$\begin{aligned}
\|\mathbf{u} \;+\; & x_1 \cdot \mathbf{b}_1 + x_2 \cdot \mathbf{b}_2 + x_3 \cdot \mathbf{b}_3\|^2 - \|\mathbf{u}\|^2 \\
&\geq -\frac{\alpha^2}{4}((1+\varepsilon)(|x_2| + |x_3|) + 1)^2 + \left(1 - \sqrt{\frac{2}{3-2\varepsilon-\varepsilon^2}}\right) \cdot \|x_2 \cdot \mathbf{b}_2 + x_3 \cdot \mathbf{b}_3\|^2 \\
&\geq -\frac{\alpha^2}{4}((1+\varepsilon)(|x_2| + |x_3|) + 1)^2 \\
&+ \left(1 - \sqrt{\frac{2}{3-2\varepsilon-\varepsilon^2}}\right)\left((x_2^2 - (1+2\varepsilon)|x_2\|x_3|) \cdot \|\mathbf{b}_2\|^2 + x_3^2\right).
\end{aligned}$$

Because $|x_2x_3| \leq (x_2^2+x_3^2)/2$, we obtain, for any sufficiently small $\varepsilon > 0$, a lower bound of the form :

$$\begin{aligned}\|\mathbf{u} + x_1 \cdot \mathbf{b}_1 &+ x_2 \cdot \mathbf{b}_2 + x_3 \cdot \mathbf{b}_3\|^2 - \|\mathbf{u}\|^2 \\ &\geq \left(\alpha^2 f_1(\varepsilon) + g_1(\varepsilon) \cdot \|\mathbf{b}_2\|^2\right) x_2^2 + \left(\alpha^2 f_2(\varepsilon) + g_2(\varepsilon) \cdot \|\mathbf{b}_2\|^2 + g_3(\varepsilon)\right) x_3^2 + \alpha^2 f_3(\varepsilon),\end{aligned}$$

where, when ε converges to zero : $f_i(\varepsilon) = O(1)$, $\lim_\varepsilon g_1(\varepsilon) > 0$, $0 \leq |\lim_\varepsilon g_2(\varepsilon)| < \lim_\varepsilon g_3(\varepsilon)$. It follows that for a suitable $c'' > 0$, there exists $C' > 0$ such that for any sufficiently small $\alpha > 0$ and any sufficiently small $\varepsilon > 0$, and any $(\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \mathbf{u}) \in K_3'(\varepsilon, 0)$ with $\|\mathbf{b}_1\| \leq \alpha$ and $\|\mathbf{b}_2\| \geq c''\alpha$, in the lower bound above, the coefficient of x_2^2 is non-negative and the coefficient of x_3^2 is greater than C'. This completes the proof of Claim 2. □

The rest of the proof is identical to the end of the proof of Lemma A.26 : we choose $\alpha > 0$ satisfying the conditions of the previous two claims and consider a sequence $\left(\mathbf{b}_1^{(k)}, \mathbf{b}_2^{(k)}, \mathbf{b}_3^{(k)}, \mathbf{u}^{(k)}, x_1^{(k)}, x_2^{(k)}, x_3^{(k)}\right)_k$ such that :

1. $\left(\mathbf{b}_1^{(k)}, \mathbf{b}_2^{(k)}, \mathbf{b}_3^{(k)}, \mathbf{u}^{(k)}\right) \in K_3'(\varepsilon, \alpha)$,
2. $x_1^{(k)}, x_2^{(k)}, x_3^{(k)}$ are integers with $\left|x_3^{(k)}\right| \geq 3$ or $\left|x_3^{(k)}\right| = 2$ and $\left(|x_1^{(k)}|, |x_2^{(k)}|\right) \neq (1,1)$ (respectively $\left|x_i^{(k)}\right| \geq 3$ or etc. for $i \in \{1,2\}$),
3. $\left\|\mathbf{u}^{(k)} + x_1^{(k)} \cdot \mathbf{b}_1^{(k)} + x_2^{(k)} \cdot \mathbf{b}_2^{(k)} + x_3^{(k)} \cdot \mathbf{b}_3^{(k)}\right\|^2 - \left\|\mathbf{u}^{(k)}\right\|^2 \to 0$ when $k \to \infty$.

If no such sequence exists, then the lemma is proved. We assume that there is one and we look for a contradiction with Voronoï coords.

Claim 3 : For any sufficiently small $\varepsilon, \alpha > 0$, the sequences $x_3^{(k)}, x_2^{(k)}$ and $x_1^{(k)}$ are bounded.

Suppose this is not the case. We show that this implies that $\left\|\mathbf{u}^{(k)} + x_1^{(k)}\mathbf{b}_1^{(k)} + x_2^{(k)}\mathbf{b}_2^{(k)} + x_3^{(k)}\mathbf{b}_3^{(k)}\right\|^2 - \left\|\mathbf{u}^{(k)}\right\|^2$ is not bounded, which is impossible. By the Cauchy-Schwarz inequality, we have :

$$\left\|\mathbf{u}^{(k)} + x_1^{(k)}\mathbf{b}_1^{(k)} + x_2^{(k)}\mathbf{b}_2^{(k)} + x_3^{(k)}\mathbf{b}_3^{(k)}\right\|^2 - \left\|\mathbf{u}^{(k)}\right\|^2$$

is

$$\geq \left\|x_1^{(k)}\mathbf{b}_1^{(k)} + x_2^{(k)}\mathbf{b}_2^{(k)} + x_3^{(k)}\mathbf{b}_3^{(k)}\right\| \cdot \left(\left\|x_1^{(k)}\mathbf{b}_1^{(k)} + x_2^{(k)}\mathbf{b}_2^{(k)} + x_3^{(k)}\mathbf{b}_3^{(k)}\right\| - 2\left\|\mathbf{u}^{(k)}\right\|\right).$$

Since the sequence $\left\|\mathbf{u}^{(k)}\right\|$ is bounded, it suffices to show that $\left\|x_1^{(k)} \cdot \mathbf{b}_1^{(k)} + x_2^{(k)} \cdot \mathbf{b}_2^{(k)} + x_3^{(k)} \cdot \mathbf{b}_3^{(k)}\right\|$ is not bounded. We take a small enough $\varepsilon > 0$ to apply Lemma A.19. Let $C > 0$ denote the corresponding constant. We have :

$$\left\|x_1^{(k)} \cdot \mathbf{b}_1^{(k)} + x_2^{(k)} \cdot \mathbf{b}_2^{(k)} + x_3^{(k)} \cdot \mathbf{b}_3^{(k)}\right\| \geq \left|x_3^{(k)}\right| \cdot \left\|\mathbf{b}_3^{(k)*}\right\| \geq C \cdot \left|x_3^{(k)}\right|.$$

Therefore, the sequence $x_3^{(k)}$ is bounded. If the sequence $x_2^{(k)}$ is bounded, by the triangular inequality, so is the sequence $x_1^{(k)}$. Now we show that the sequence $x_2^{(k)}$ remains bounded. We have the following inequality :

$$\left\|x_1^{(k)} \cdot \mathbf{b}_1^{(k)} + x_2^{(k)} \cdot \mathbf{b}_2^{(k)} + x_3^{(k)} \cdot \mathbf{b}_3^{(k)}\right\| \geq \left|x_2^{(k)}\right| \cdot \left\|\mathbf{b}_2^{(k)*}\right\| - \left|x_3^{(k)}\right| \cdot \left\|\mathbf{b}_3^{(k)}\right\|.$$

Since $\left[\mathbf{b}_1^{(k)}, \mathbf{b}_2^{(k)}\right]_\leq$ are ε-greedy-reduced and $1 \geq \left\|\mathbf{b}_2^{(k)}\right\| \geq \left\|\mathbf{b}_1^{(k)}\right\| \geq \alpha$, we have a situation similar to the third claim of Lemma A.26 and this gives us a strictly positive lower bound on $\|\mathbf{b}_2^{(k)*}\|$ that

depends on ε and α. This completes the proof of the claim. □

This claim and Lemma A.25 imply that $\left(\mathbf{b}_1^{(k)}, \mathbf{b}_2^{(k)}, \mathbf{b}_3^{(k)}, \mathbf{u}^{(k)}, x_1^{(k)}, x_2^{(k)}, x_3^{(k)}\right)$ remains in a compact subset of $\mathbb{R}^{4n+3}$. Therefore we can extract a subsequence converging to a limit $(\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \mathbf{u}, x_1, x_2, x_3)$ that is in the same compact and satisfies : $\|\mathbf{u} + x_1 \cdot \mathbf{b}_1 + x_2 \cdot \mathbf{b}_2 + x_3 \cdot \mathbf{b}_3\| = \|\mathbf{u}\|$. This means that $x_1 \cdot \mathbf{b}_1 + x_2 \cdot \mathbf{b}_2 + x_3 \cdot \mathbf{b}_3$ is a Voronoï vector of the lattice spanned by $[\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3]_{\leq} \in K_3'(\varepsilon, \alpha)$. From Lemma A.23, this is impossible. □

Like in previous subsections, we now consider the case of the possible Voronoï coords $(\pm 1, \pm 1, \pm 2)$ modulo any permutation of coordinates.

Lemma A.28 *There exist two constants $\varepsilon, C > 0$ such that for any ε-greedy-reduced vectors $[\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3]_{\leq}$ and any vector $\mathbf{u} \in \mathrm{Vor}(\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$, if at least one of the following conditions holds, then :*

$$\|\mathbf{u} + x_1 \cdot \mathbf{b}_1 + x_2 \cdot \mathbf{b}_2 + x_3 \cdot \mathbf{b}_3\|^2 \geq \|\mathbf{u}\|^2 + C \cdot \|\mathbf{b}_3\|^2.$$

1. $(x_1, x_2, x_3) = (s_1, s_2, 2)$, $|s_i| = 1$ *for* $i \in \{1, 2\}$ *and at least one of the following conditions holds :*
   $\|\mathbf{b}_1\| \leq (1-\varepsilon) \cdot \|\mathbf{b}_3\|$, *or* $\|\mathbf{b}_2\| \leq (1-\varepsilon) \cdot \|\mathbf{b}_3\|$, *or* $|\langle \mathbf{b}_1, \mathbf{b}_2 \rangle| \geq \varepsilon \cdot \|\mathbf{b}_3\|^2$, *or* $\left|\langle \mathbf{b}_1, \mathbf{b}_3 \rangle + \frac{s_1}{2} \cdot \|\mathbf{b}_1\|^2\right| \geq \varepsilon \cdot \|\mathbf{b}_3\|^2$, *or* $\left|\langle \mathbf{b}_2, \mathbf{b}_3 \rangle + \frac{s_2}{2} \cdot \|\mathbf{b}_1\|^2\right| \geq \varepsilon \cdot \|\mathbf{b}_3\|^2$.
2. $(x_1, x_2, x_3) = (s_1, 2, s_3)$, $|s_i| = 1$ *for* $i \in \{1, 3\}$ *and* $\|\mathbf{b}_1\| \leq (1 - \varepsilon) \cdot \|\mathbf{b}_2\|$.
3. $(x_1, x_2, x_3) = (s_1, 2, s_3)$, $|s_i| = 1$ *for* $i \in \{1, 3\}$, $\|\mathbf{b}_1\| \geq (1 - \varepsilon) \cdot \|\mathbf{b}_3\|$ *and at least one of the following conditions holds :* $|\langle \mathbf{b}_1, \mathbf{b}_3 \rangle| \geq \varepsilon \cdot \|\mathbf{b}_3\|^2$, *or* $\left|\langle \mathbf{b}_1, \mathbf{b}_2 \rangle + \frac{s_1}{2} \cdot \|\mathbf{b}_1\|^2\right| \geq \varepsilon \cdot \|\mathbf{b}_3\|^2$, *or* $\left|\langle \mathbf{b}_3, \mathbf{b}_2 \rangle + \frac{s_3}{2} \cdot \|\mathbf{b}_1\|^2\right| \geq \varepsilon \cdot \|\mathbf{b}_3\|^2$.
4. $(x_1, x_2, x_3) = (2, s_2, s_3)$, $|s_i| = 1$ *for* $i \in \{2, 3\}$, $\|\mathbf{b}_1\| \geq (1 - \varepsilon) \cdot \|\mathbf{b}_3\|$ *and at least one of the following conditions holds :* $|\langle \mathbf{b}_2, \mathbf{b}_3 \rangle| \geq \varepsilon \cdot \|\mathbf{b}_3\|^2$, *or* $\left|\langle \mathbf{b}_2, \mathbf{b}_1 \rangle + \frac{s_2}{2} \cdot \|\mathbf{b}_1\|^2\right| \geq \varepsilon \cdot \|\mathbf{b}_3\|^2$, *or* $\left|\langle \mathbf{b}_3, \mathbf{b}_1 \rangle + \frac{s_3}{2} \cdot \|\mathbf{b}_1\|^2\right| \geq \varepsilon \cdot \|\mathbf{b}_3\|^2$.

Proof. Wlog we suppose that $\|\mathbf{b}_3\| = 1$. We consider each subcase and each triple (x_1, x_2, x_3) separately (there is a finite number of subcases and of triples to consider). The constant $\varepsilon > 0$ is fixed such that if any of the conditions of the considered subcase is not fulfilled, then the result of Lemma A.24 is wrong. In that case, the triple (x_1, x_2, x_3) cannot be a Voronoï coord for the ε-greedy-reduced vectors $[\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3]_{\leq}$. This implies that $\mathrm{dist}\,(V + x_1 \cdot \mathbf{b}_1 + x_2 \cdot \mathbf{b}_2 + x_3 \cdot \mathbf{b}_3, V) > 0$, where $V = \mathrm{Vor}(\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$. The facts that the function $\mathrm{dist}\,(V + x_1 \cdot \mathbf{b}_1 + x_2 \cdot \mathbf{b}_2 + x_3 \cdot \mathbf{b}_3, V)$ is continuous as regard to the variables $(\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$ and that the variables $(\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$ belong to a compact set provide the expected result. □

A.10 Difficulties Arising in Dimension 5

We have seen that the greedy algorithm can output arbitrarily bad bases in dimension 5. By using Minkowski's conditions, it is easy to see that the following generalisation of the greedy algorithm computes a Minkowski-reduced basis in dimensions five and six : at Steps 2 and 3 of the iterative version of the greedy algorithm, instead of shortening the vector $\mathbf{b}_k$ by using an integer linear combination of $\mathbf{b}_1, \ldots, \mathbf{b}_{k-1}$, the algorithm may also use $\mathbf{b}_{k+1}, \ldots, \mathbf{b}_d$, see Figure A.7. We know very little about this algorithm when the dimension is higher than six : does it compute a Minkowski-reduced basis ? is there a good bound on the number of loop iterations ? does this algorithm admit a polynomial-time complexity bound ?

Despite the fact that the greedy algorithm does not return a Minkowski-reduced basis, one may wonder if the quadratic complexity bound remains valid. To make the technique developed in Section A.7 ready for use, it suffices to show that the iterative greedy algorithm in dimension 5 admits

Input : An basis $[\mathbf{b}_1, \ldots, \mathbf{b}_d]_{\leq}$ with its Gram matrix.
Output : An ordered basis of $L[\mathbf{b}_1, \ldots, \mathbf{b}_d]$ with its Gram matrix.

1. $k \longleftarrow 2$. While $k \leq d$, do :
2. Compute a vector $\mathbf{c}$ closest to $\mathbf{b}_k$, in $L[\mathbf{b}_1, \ldots, \mathbf{b}_{k-1}, \mathbf{b}_{k+1}, \ldots, \mathbf{b}_d]$,
3. $\mathbf{b}_k \longleftarrow \mathbf{b}_k - \mathbf{c}$ and update the Gram matrix,
4. If $\|\mathbf{b}_k\| \geq \|\mathbf{b}_{k-1}\|$, $k \longleftarrow k+1$
5. Else insert $\mathbf{b}_k$ at his length rank k', update the Gram matrix, $k \longleftarrow k'+1$.

FIG. A.7 – A generalisation of the greedy algorithm.

a linear bound (as regard to the input size) on its number of loop iterations. In dimensions below four, it was possible to use both local and global approaches.

With the global approach, it seems possible to show that the number of loop iterations of the recursive version of the greedy algorithm in dimension five is linear, which does not suffice. Besides, the result seems hard to get : Lemma A.5 is not valid anymore. Nevertheless, it seems possible to determine precisely the bad cases : roughly speaking, these are the bases that resemble the one given in Lemma A.4. It could then be shown that if we are not in this situation then Lemma A.5 is correct, and that we can use Lemma A.6 (it remains valid in dimension five). If we are in the bad situation, the geometry of the current basis could be made precise, and it should be possible to show that two loop iterations after the end of the η-phase, there is some significant length decrease.

The local analysis in dimensions two, three and four essentially relies on the fact that if one of the x_j's found at Step 5, of the recursive version of the greedy algorithm, has absolute value higher than 2, then $\left\|\mathbf{b}_d^{(i)}\right\|$ is significantly shorter than $\left\|\mathbf{a}_d^{(i)}\right\|$. This fact is derived from the so-called Gap Lemma. In dimension four, this was only partly true, but the exception (the 211-case) occurred in very few cases and could be dealt by considering the very specific shape of the lattices for which it could go wrong. Things worsen in dimension five. Indeed, for Minkowski-reduced bases, $(1,1,1,2)$ and $(1,1,2,2)$ — modulo any change of sign and permutation of coordinates — are possible Voronoï coords. Here is an example of a lattice for which $(1,1,2,2)$ is a Voronoï coord :

$$\begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The lattice basis given by the columns is Minkowski-reduced (since it is greedy-reduced), but :

$$\|\mathbf{b}_1 + \mathbf{b}_2 + 2\cdot\mathbf{b}_3 + 2\cdot\mathbf{b}_4\| = 2 = \|\mathbf{b}_1 + \mathbf{b}_2\| \leq \|(2k_1+1)\cdot\mathbf{b}_1 + (2k_2+1)\cdot\mathbf{b}_2 + 2k_3\cdot\mathbf{b}_3 + 2k_4\cdot\mathbf{b}_4\|,$$

for any $k_1, k_2, k_3, k_4 \in \mathbb{Z}$. Notice that $(1,1,2,2)$ cannot be a strict Voronoï coord : if $\mathbf{b}_1+\mathbf{b}_2+2\cdot\mathbf{b}_3+2\cdot\mathbf{b}_4$ reaches the length minimum of its coset of $L/2L$, then so does $\mathbf{b}_1+\mathbf{b}_2$. Thus it might be possible to work around the difficulty coming from $(1,1,2,2)$ like in the 211-case. However, the case $(1,1,1,2)$ would still remain, and this possible Voronoï coordinate can be strict.

Acknowledgements.

We thank Ali Akhavi, Florian Hess, Igor Semaev, Jacques Stern and Gilles Villard for helpful discussions and comments. Part of the writing of the present paper was performed while the second author was visiting the University of Bristol and the University of Sydney, whose hospitalities are gratefully acknowledged.

Annexe B

An LLL Algorithm with Quadratic Complexity

EUROCRYPT 2005
Version complète de [264], avec Damien Stehlé (LORIA)

Abstract: *The Lenstra-Lenstra-Lovász lattice basis reduction algorithm (called LLL or L^3) is a fundamental tool in computational number theory and theoretical computer science. Given an integer d-dimensional lattice basis with vectors of norm less than B in an n-dimensional space, the L^3 algorithm outputs a reduced basis in polynomial time $O(d^5 n \log^3 B)$, using arithmetic operations on integers of bit-length $O(d \log B)$. This worst-case complexity is problematic for applications where d or/and $\log B$ are often large. As a result, the original L^3 algorithm is almost never used in practice, except in tiny dimension. Instead, one applies floating-point variants where the long-integer arithmetic required by Gram-Schmidt orthogonalization is replaced by floating-point arithmetic. Unfortunately, this is known to be unstable in the worst case : the usual floating-point L^3 algorithm is not even guaranteed to terminate, and the output basis may not be L^3-reduced at all. In this article, we introduce the L^2 algorithm, a new and natural floating-point variant of the L^3 algorithm which provably outputs L^3-reduced bases in polynomial time $O(d^4 n(d + \log B) \log B)$. This is the first L^3 algorithm whose running time (without fast integer arithmetic) provably grows only quadratically with respect to $\log B$, like Euclid's gcd algorithm and Lagrange's two-dimensional algorithm.*

B.1 Introduction

Let $\mathbf{b}_1, \ldots, \mathbf{b}_d$ be linearly independent vectors in $\mathbb{R}^n$ with $n \geq d$: often, $n = d$ or $n = O(d)$. We denote by $L(\mathbf{b}_1, \ldots, \mathbf{b}_d) = \left\{ \sum_{i=1}^{d} x_i \mathbf{b}_i \mid x_i \in \mathbb{Z} \right\}$ the set of all integer linear combinations of the $\mathbf{b}_i$'s. This set is called a *lattice* of $\mathbb{R}^n$ and $[\mathbf{b}_1, \ldots, \mathbf{b}_d]$ a *basis* of that lattice. A lattice basis is usually not unique, but all the bases have the same number d of elements, called the *dimension* of the lattice. If $d \geq 2$, there are infinitely many bases, but some are more interesting than others : they are called *reduced*. Roughly speaking, a reduced basis is a basis made of reasonably short vectors which are almost orthogonal. Finding good reduced bases has proved invaluable in many fields of computer science and mathematics (see the books [129, 65]), particularly in cryptology (see [267, 240]). This problem is known as *lattice reduction* and can intuitively be viewed as a vectorial generalization of gcd computations.

Hermite [142] invented in 1845 the first lattice reduction algorithm in arbitrary dimension, by trying to generalize Lagrange's two-dimensional algorithm [201] (often wrongly attributed to Gauss).

In his famous letters [142] to Jacobi, Hermite actually described two reduction algorithms : the first letter presented an algorithm to show the existence of Hermite's constant (which guarantees the existence of short lattice vectors), while the second letter presented a slightly different algorithm to further prove the existence of lattice bases with bounded orthogonality defect. The subject had a revival around 1981 with Lenstra's celebrated work on integer programming [207, 208], which was, among others, based on a novel lattice reduction technique. This reduction technique, which can be found in the preliminary version [207] of [208], turns out to be a relaxed variant of Hermite's second algorithm, perhaps because the reduction goal of [207] was to bound the orthogonality defect. Lenstra's algorithm is only polynomial-time for fixed dimension, which was however sufficient in [207]. This inspired Lovász to develop a polynomial-time variant of the algorithm, which reached a final form in the seminal paper [205] where Lenstra, Lenstra and Lovász applied it to factor rational polynomials in polynomial time, from which the name LLL or L^3 comes : the L^3 algorithm was the first polynomial-time reduction algorithm, and it provides bases almost as reduced as Hermite's second algorithm. Further refinements of the L^3 algorithm were later proposed, notably by Schnorr [302, 303] and Gama *et al.* [103] : [303] improves the running-time of L^3, and [302, 103] improve the output quality of L^3 while keeping polynomial-time complexity. Reduction algorithms (in particular L^3) have arguably become the most popular tool in public-key cryptanalysis (see the survey [267]). In the past twenty-five years, they have been used to break many public-key cryptosystems, including knapsack cryptosystems [274], RSA in particular settings [70, 43, 40], DSA and similar signature schemes in particular settings [159, 261], etc.

Given as input an integer d-dimensional lattice basis whose n-dimensional vectors have norm less than B, the L^3 algorithm outputs a so-called L^3-reduced basis in time $O(d^5 n \log^3 B)$ without fast integer arithmetic, using arithmetic operations on integers of bit-length $O(d \log B)$. This worst-case complexity turns out to be problematic in practice, especially for lattices arising in cryptanalysis where d or/and $\log B$ are often large. For instance, in a typical RSA application of Coppersmith's lattice-based theorem [70], we may need to reduce a 64-dimensional lattice with vectors having RSA-type coefficients (1024-bit), in which case the complexity becomes "$d^5 n \log^3 B = 2^{66}$ ". As a result, the original L^3 algorithm is seldom used in practice. Instead, one applies floating-point variants, where the long-integer arithmetic required by Gram-Schmidt orthogonalization (which plays a central role in L^3) is replaced by floating-point arithmetic on much smaller numbers. The use of floating-point arithmetic in the L^3 algorithm dates back to the early eighties when the L^3 algorithm was used to solve low-density knapsacks [200]. Unfortunately, floating-point arithmetic may lead to stability problems, both in theory and practice, especially when the dimension increases : the running time of floating-point variants of the L^3 algorithm such as Schnorr-Euchner's [307] is not guaranteed to be polynomial nor even finite, and the output basis may not be L^3-reduced at all. This phenomenon is well-known to L^3 practitioners, and is usually solved by sufficiently increasing the precision. For instance, experimental problems arose during the cryptanalyses [252, 250] of lattice-based cryptosystems, which led to improvements in Shoup's NTL library [319].

There is however one provable floating-point variant of L^3, due to Schnorr [303], which significantly improves the worst-case complexity. Schnorr's variant outputs an approximate L^3-reduced basis in time $O(d^3 n \log B (d + \log B)^2)$, using $O(d + \log B)$ precision floating-point numbers. However, this algorithm is mostly of theoretical interest and is not implemented in any of the main computational libraries [319, 216, 22, 213]. This is perhaps explained by the following reasons : it is not clear which floating-point arithmetic model is used, the algorithm is not easy to describe, and the hidden complexity constants are rather large. More precisely, the required precision of floating-point numbers in [303] seems to be higher than $12d + 7 \log_2 B$.

Our Results. We present the L^2 algorithm, a new and simple floating-point variant of the L^3 algorithm in a standard floating-point arithmetic model, which provably outputs approximate L^3-reduced bases in polynomial time. More precisely, its complexity is without fast integer arithmetic

$O(d^4 n(d + \log B) \log B)$ using only a $(d \log_2 3)$-bit precision, which is independent of $\log B$. This is the first L^3 algorithm whose running time grows only quadratically with respect to $\log B$ (hence the name L^2), whereas the growth is cubic – without fast integer arithmetic – for all other provable L^3 algorithms known. This improvement is significant for lattices where $\log B$ is larger than d, for example those arising from minimal polynomials [65] and Coppersmith's technique [70]. Interestingly, the L^3 algorithm can be viewed as a generalization of the famous Euclid gcd algorithm and Lagrange's two-dimensional algorithm [201] whose complexities are quadratic without fast integer arithmetic, not cubic like the original L^3. This arguably makes L^2 closer to Euclid's algorithm.

	L^3 [205]	Schnorr [303]	L^2
Required precision	$O(d \log B)$	$> 12d + 7 \log_2 B$	$d \log_2 3 \approx 1.58d$
Complexity	$O(d^5 n \log^3 B)$	$O(d^3 n(d + \log B)^2 \log B)$	$O(d^4 n(d + \log B) \log B)$

FIG. B.1 – Comparison of different L^3 algorithms.

The L^2 algorithm is based on several improvements, both in the L^3 algorithm itself and more importantly in its analysis. From an algorithmic point of view, we improve the accuracy of the usual Gram-Schmidt computations by a systematic use of the Gram matrix, and we adapt Babai's nearest plane algorithm [17] to floating-point arithmetic in order to stabilise the so-called size-reduction process extensively used in L^3. We give tight bounds on the accuracy of Gram-Schmidt computations to prove the correctness of L^2. The analysis led to the discovery of surprisingly bad lattices : for instance, we found a 55-dimensional lattice with 100-bit vectors which makes NTL's LLL_FP [319] (an improved version of [307]) loop forever, which contradicts [194] where it is claimed that double precision is sufficient in [307] to L^3-reduce lattices up to dimension 250 with classical Gram-Schmidt. However, for random looking lattice bases, stability problems seem to arise only in dimension much higher than 55, due perhaps to the well-known experimental fact that for such input bases, the L^3 algorithm outputs better bases than for the worst-case. Finally, to establish a quadratic running time, we generalise a well-known cascade phenomenon in the complexity analysis of the Gaussian and Euclidean algorithms. This was inspired by the so-called greedy lattice reduction algorithm of [263], which is quadratic in low dimension thanks to another cascade. The cascade analysis is made possible by the efficiency of our floating-point variant of Babai's algorithm, and cannot be adapted to the standard L^3 algorithm : it is unlikely that the complexity of the original L^3 algorithm could be proved quadratic in $\log B$.

RELATED WORK. Much work [310, 303, 336, 193, 194, 306] has been devoted to improve L^3, specifically the exponent of d in the complexity, but none has improved the $\log^3 B$ factor (except [358, 311] for dimension two). We hope that some of these improvements might be adaptable to L^2.

Floating-point stability has long been a mysterious issue in L^3. When it was realized during experiments that classical Gram-Schmidt orthogonalization could be very unstable, it was suggested in the late nineties to use well-known alternative techniques (see [203, 123]) like Givens rotations (implemented in NTL) or Householder reflections, which are more expensive but seem to be more stable in practice. However, from a theoretical point of view, the best results known on the worst-case accuracy of such techniques are not significantly better than the so-called Modified Gram-Schmidt algorithm. Besides, most numerical analysis results refer to backward stability and not accuracy : such a mistake is made in [194], where a theorem from [203] is incorrectly applied. At the moment, it is therefore not clear how to provably exploit known results on Givens rotations and Householder reflections to improve the L^3 algorithm theoretically, though Schnorr [306] provides heuristic arguments suggesting it might be possible. This is why L^2 only uses a process close to classical Gram-Schmidt.

ROAD MAP. In Section B.2 we provide necessary background on lattices and L^3. We describe the

L^2 algorithm in Section B.3. Section B.4 proves the correctness of L^2, while Section B.5 analyses its complexity. In Section B.6, we generalize the L^2 algorithm to generating sets.

B.2 Preliminaries

NOTATION. All logarithms are in base 2. Let $\|\cdot\|$ and $\langle\cdot,\cdot\rangle$ be the Euclidean norm and inner product of $\mathbb{R}^n$. The notation $\lceil x \rfloor$ denotes a closest integer to x. Bold variables are vectors. All the lattices we consider are integer lattices, as usual. The complexity model we use is the RAM model and the computational cost is measured in elementary operations on bits, without fast integer arithmetic [312]. Our floating-point arithmetic model is a smooth extension of the IEEE-754 standard [162], as provided by NTL [319] (RR class) and MPFR [289]. With an ℓ-bit working precision, a *fp*-number is of the form $x = \pm m_x \cdot 2^{e_x}$ where the mantissa $m_x \in [1/2, 1)$ is ℓ-bit long and the exponent e_x is an integer. We expect all four basic floating-point operations to be correctly rounded : the returned value $\diamond(a \text{ op } b)$ for $\text{op} \in \{+, -, /, *\}$ is a closest *fp*-number to $(a \text{ op } b)$. In our complexity analysis, we do not consider the cost of the arithmetic on the exponents : it can be checked easily that the exponents are integers of length $O(\log(d + \log B))$, so that the cost is indeed negligible.

We now recall a few basic notions from algorithmic geometry of numbers (see [240]), before describing the classical LLL algorithm.

B.2.1 Lattice Background

Gram matrix. Let $\mathbf{b}_1, \ldots, \mathbf{b}_d$ be vectors. Their *Gram matrix* $G(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ is the $d \times d$ symmetric positive definite matrix $(\langle \mathbf{b}_i, \mathbf{b}_j \rangle)_{1 \le i,j \le d}$ formed by all the inner products.

Lattice volume. A lattice L has infinitely many lattice bases as soon as $\dim L \ge 2$. Any two bases are related to each other by an integral matrix of determinant ± 1, and therefore the determinant of the Gram matrix of a basis only depends on the lattice. The square root of this determinant is called the *volume* vol L (or *determinant*) of the lattice. This volume is sometimes called *co-volume* because it is the volume of the torus $\text{span}(L)/L$.

Gram-Schmidt orthogonalization. Let $\mathbf{b}_1, \ldots, \mathbf{b}_d$ be linearly independent vectors. Their *Gram-Schmidt orthogonalization* (GSO) $\mathbf{b}_1^*, \ldots, \mathbf{b}_d^*$ is the orthogonal family defined recursively as follows : the vector $\mathbf{b}_i^*$ is the component of the vector $\mathbf{b}_i$ which is orthogonal to the linear span of the vectors $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$. We have $\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^*$ where $\mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\|\mathbf{b}_j^*\|^2}$. For $i \le d$ we let $\mu_{i,i} = 1$. The reason why the Gram-Schmidt orthogonalization is widely used in lattice reduction is because the matrix representing the vectors $\mathbf{b}_1, \ldots, \mathbf{b}_d$ with respect to the orthonormal basis $(\mathbf{b}_1^*/\|\mathbf{b}_1^*\|, \ldots, \mathbf{b}_d^*/\|\mathbf{b}_d^*\|)$ is lower triangular, with diagonal coefficients $\|\mathbf{b}_1^*\|, \ldots, \|\mathbf{b}_d^*\|$. It follows that the lattice L spanned by the $\mathbf{b}_i$'s satisfies vol $L = \prod_{i=1}^{d} \|\mathbf{b}_i^*\|$.

Notice that the GSO family depends on the order of the vectors. If the $\mathbf{b}_i$'s are integer vectors, the $\mathbf{b}_i^*$'s and the $\mu_{i,j}$'s are rational. We also define the variables $r_{i,j}$ for $i \ge j$ as follows : for any $i \in [\![1, d]\!]$, we let $r_{i,i} = \|\mathbf{b}_i^*\|^2$, and for any $i \ge j$ we let $r_{i,j} = \mu_{i,j} \left\|\mathbf{b}_j^*\right\|^2$. In what follows, the *GSO family* denotes the $r_{i,j}$'s and $\mu_{i,j}$'s. Some information is redundant in rational arithmetic, but in the context of our floating-point calculations, it is useful to have all these variables.

Size-reduction. A basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ is *size-reduced* with factor $\eta \ge 1/2$ if its GSO family satisfies $|\mu_{i,j}| \le \eta$ for all $1 \le j < i \le d$. The i-th vector $\mathbf{b}_i$ is *size-reduced* if $|\mu_{i,j}| \le \eta$ for all $j \in [\![1, i-1]\!]$. Size-reduction usually refers to $\eta = 1/2$, but it is essential for the L^2 algorithm to allow at least slightly larger factors η.

B.2.2 From Hermite's Reductions to the Lenstra-Lenstra-Lovász Reduction

Lattice reduction in dimension two. A pair of linearly independent vectors $(\mathbf{b}_1, \mathbf{b}_2)$ is *Lagrange-reduced* [201] if $\|\mathbf{b}_1\| \leq \|\mathbf{b}_2\|$ and $|\langle \mathbf{b}_1, \mathbf{b}_2 \rangle| \leq \|\mathbf{b}_1\|^2/2$. This definition is often wrongly attributed to Gauss [109]. Lagrange's algorithm [201] shows that any two-dimensional lattice has a Lagrange-reduced basis : the algorithm is very similar to Euclid's gcd algorithm. Furthermore, such a basis $(\mathbf{b}_1, \mathbf{b}_2)$ satisfies the following property :

$$\|\mathbf{b}_1\| \cdot \|\mathbf{b}_2\| \leq \sqrt{4/3} \cdot \mathrm{vol}\left(L(\mathbf{b}_1, \mathbf{b}_2)\right),$$

which means that the vectors $\mathbf{b}_1$ and $\mathbf{b}_2$ are almost orthogonal. It also follows that :

$$\|\mathbf{b}_1\| \leq (4/3)^{1/4} \cdot \mathrm{vol}\left(L(\mathbf{b}_1, \mathbf{b}_2)\right)^{1/2},$$

which gives a tight upper bound on Hermite's constant of dimension 2.

Hermite's reductions. In his famous letters to Jacobi [142], Hermite described two reduction notions (along with algorithms) in the language of quadratic forms. We will briefly describe both, since Hermite's algorithms can be viewed as the ancestors of the L^3 algorithm.

The first reduction notion (H1) is described at the end of the first letter [142], dated from August 6, 1845. To the best of our knowledge, it is the first reduction notion in arbitrary dimension, which was introduced to prove the existence of Hermite's constant, by establishing an upper bound called Hermite's inequality. The H1 reduction is defined by induction :

- A single vector $\mathbf{b}_1 \neq \mathbf{0}$ is always H1-reduced.
- A d-dimensional basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ is H1-reduced if and only if :
  - The basis is size-reduced with factor $1/2$, that is, the GSO satisfies all the inequalities $|\mu_{i,j}| \leq 1/2$ for all $i > j$.
  - The first basis vector $\mathbf{b}_1$ satisfies Hermite's inequality :

$$\|\mathbf{b}_1\| \leq (4/3)^{(d-1)/4} \cdot \mathrm{vol}(L(\mathbf{b}_1, \ldots, \mathbf{b}_d))^{1/d}.$$

  - By induction, the projected $(d-1)$-tuple $(\mathbf{b}'_2, \ldots, \mathbf{b}'_d)$ is itself H1-reduced, where for any $i \in [\![2, d]\!]$ the vector $\mathbf{b}'_i$ is the component of the vector $\mathbf{b}_i$ which is orthogonal to the vector $\mathbf{b}_1$.

This H1 reduction notion is only useful to prove Hermite's inequality : the first vector of an H1-reduced basis may be arbitrarily far from the first minimum of the lattice, and the orthogonality defect of the basis may be arbitrarily large. To prove the existence of H1-reduced bases, Hermite presented the following recursive algorithm :

- Let $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ be a basis of a lattice L.
- Apply recursively the algorithm to the projected basis $(\mathbf{b}'_2, \ldots, \mathbf{b}'_d)$, in such a way that all the basis vectors $\mathbf{b}_2, \ldots, \mathbf{b}_d$ are size-reduced with respect to $\mathbf{b}_1$: $|\mu_{i,1}| \leq 1/2$ for all $i \geq 2$.
- If $\mathbf{b}_1$ satisfies Hermite's inequality, the algorithm terminates. Otherwise, it can be shown that $\|\mathbf{b}_2\| < \|\mathbf{b}_1\|$, so exchange $\mathbf{b}_1$ and $\mathbf{b}_2$, and restart from the beginning.

The main differences with this algorithm and L^3 are the following : L^3 starts working with the first two basis vectors, but Hermite will start working with the last two basis vectors ; and Hermite's algorithm uses Hermite's inequality instead of the so-called Lovász condition. Hermite proved that his algorithm must terminate. However, because his algorithm did not match Lagrange's algorithm in dimension two, and perhaps also because the orthogonality defect of a H1-reduced basis can be arbitrarily large, Hermite presented a slightly different algorithm in his second letter [142] to Jacobi :

- Let $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ be a basis of a lattice L.
- Ensure that $\mathbf{b}_1$ has minimal norm among all $\mathbf{b}_i$'s : otherwise, swap $\mathbf{b}_1$ with the shortest basis vector $\mathbf{b}_i$.
- Apply recursively the algorithm to the projected basis $(\mathbf{b}'_2, \ldots, \mathbf{b}'_d)$, in such a way that all the basis vectors $\mathbf{b}_2, \ldots, \mathbf{b}_d$ are size-reduced with respect to $\mathbf{b}_1$: $|\mu_{i,1}| \leq 1/2$ for all $i \geq 2$.

- If $\mathbf{b}_1$ has minimal norm among all $\mathbf{b}_i$'s, the algorithm terminates. Otherwise, swap $\mathbf{b}_1$ with the shortest basis vector $\mathbf{b}_i$, and restart from the beginning.

Hermite also proved that this algorithm must terminate. One can note that this algorithm matches Lagrange's algorithm when $d = 2$. But one can also note that this second algorithm achieves a reduction notion (H2) which is stronger than H1 :

- A single vector $\mathbf{b}_1 \neq \mathbf{0}$ is always H2-reduced.
- A d-dimensional basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ is H2-reduced if and only if :
  - The basis is size-reduced with factor $1/2$, that is, the GSO satisfies all the inequalities $|\mu_{i,j}| \leq 1/2$ for all $i > j$.
  - The first basis vector $\mathbf{b}_1$ has minimal norm among all basis vectors : $\|\mathbf{b}_1\| \leq \|\mathbf{b}_i\|$ for all i.
  - By induction, the $(d-1)$-tuple $(\mathbf{b}'_2, \ldots, \mathbf{b}'_d)$ is itself H2-reduced, where for any $i \in [\![2, d]\!]$ the vector $\mathbf{b}'_i$ is the component of the vector $\mathbf{b}_i$ which is orthogonal to the vector $\mathbf{b}_1$.

As opposed to H1, this reduction notion implies a bounded orthogonality defect : more precisely, an H2-reduced basis satisfies $\prod_{i=1}^{d} \|\mathbf{b}_i\| \leq (4/3)^{\frac{d(d-1)}{4}} \cdot \mathrm{vol}(L)$. Surprisingly, Lenstra's reduction notion [207] (resp. his algorithm) turns out to be a relaxed variant of H2 (resp. Hermite's second algorithm) : more precisely, one replaces the conditions $\|\mathbf{b}_1\| \leq \|\mathbf{b}_i\|$ by $c\|\mathbf{b}_1\| \leq \|\mathbf{b}_i\|$ for some constant $1/4 < c < 1$. This allowed Lenstra [207] to prove that his algorithm was polynomial time in fixed dimension d. The H2 reduction was also rediscovered by Schnorr and Euchner [307] in 1994 with their L^3 algorithm with deep insertion : Schnorr-Euchner's algorithm is different from Hermite's second algorithm, but both try to achieve the same reduction notion. It is unknown if Hermite's algorithms are polynomial time in varying dimension.

The Lenstra-Lenstra-Lovász reduction. Roughly speaking, the Lenstra-Lenstra-Lovász reduction [205] modifies Hermite's second reduction notion by replacing the conditions $\|\mathbf{b}_1\| \leq \|\mathbf{b}_i\|$ by the single condition $c\|\mathbf{b}_1\| \leq \|\mathbf{b}_2\|$ for some constant $1/4 < c < 1$. More precisely, a basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ is *L^3-reduced* with factor (δ, η) where $\delta \in (1/4, 1]$ and $\eta \in \left]1/2, \sqrt{\delta}\right)$ if the basis is size-reduced with factor η and if its GSO satisfies the $(d-1)$ Lovász conditions : for all $2 \leq \kappa \leq d$,

$$\left(\delta - \mu_{\kappa,\kappa-1}^2\right) \cdot r_{\kappa-1,\kappa-1} \leq r_{\kappa,\kappa},$$

or equivalently $\delta \cdot \left\|\mathbf{b}^*_{\kappa-1}\right\|^2 \leq \left\|\mathbf{b}^*_{\kappa} + \mu_{\kappa,\kappa-1}\mathbf{b}^*_{\kappa-1}\right\|^2$. This implies that the norms $\|\mathbf{b}^*_1\|, \ldots, \|\mathbf{b}^*_d\|$ of the GSO vectors never drop too much : intuitively, the vectors are not far from being orthogonal. Such bases have very useful properties, like providing approximations to the shortest vector problem and the closest vector problem. In particular, their first vector is relatively short. More precisely :

Theorem B.10 ([205]) *Let $\delta \in (1/4, 1]$ and $\eta \in \left]1/2, \sqrt{\delta}\right)$. Let $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ be a (δ, η)-L^3-reduced basis of a lattice L. Then :*

$$\begin{aligned} \|\mathbf{b}_1\| &\leq \left(1/(\delta - \eta^2)\right)^{\frac{d-1}{4}} \cdot (\mathrm{vol}\ L)^{\frac{1}{d}}, \\ \prod_{i=1}^{d} \|\mathbf{b}_i\| &\leq (1/(\delta - \eta^2))^{\frac{d(d-1)}{4}} \cdot (\mathrm{vol}\ L). \end{aligned}$$

Proof. We have $r_{i-1,i-1} \leq (\delta - \eta^2)^{-1} \cdot r_{i,i}$ for any $i \in [\![2, d]\!]$. This implies that for any $j \leq i$:

$$r_{j,j} \leq (\delta - \eta^2)^{-(i-j)} \cdot r_{i,i}.$$

Taking $j = 1$ and multiplying these inequalities for all $i \in [\![1, d]\!]$ gives the first statement of the

theorem. Since the basis is size-reduced, we have, for any i :

$$\begin{aligned}
\|\mathbf{b}_i\|^2 = r_{i,i} + \sum_{j=1}^{i-1} \mu_{i,j}^2 r_{j,j} &\leq \left(1 + \sum_{j=1}^{i-1} \eta^2 \frac{r_{j,j}}{r_{i,i}}\right) \cdot r_{i,i} \\
&\leq \left(1 + \eta^2 \sum_{j=1}^{i-1} \left(\delta - \eta^2\right)^{-(i-j)}\right) \cdot r_{i,i} \\
&\leq \left(1 + \eta^2 \frac{\left(\delta - \eta^2\right)^{-i} - \left(\delta - \eta^2\right)^{-1}}{\left(\delta - \eta^2\right)^{-1} - 1}\right) \cdot r_{i,i} \\
&\leq \left(\delta - \eta^2\right)^{-i+1} \cdot r_{i,i}.
\end{aligned}$$

By multiplying these d inequalities, we obtain the second statement of the theorem. □

The L^3-reduction usually refers to the factor $(3/4, 1/2)$ initially chosen in [205], in which case the approximation constant $1/(\delta - \eta^2)$ is equal to 2. But the closer δ and η are respectively to 1 and $1/2$, the shorter the vector $\mathbf{b}_1$ should be. In practice, one usually selects $\delta \approx 1$ and $\eta \approx 1/2$, so that we almost have $\|\mathbf{b}_1\| \leq (4/3)^{\frac{d-1}{4}} \cdot (\mathrm{vol}\, L)^{\frac{1}{d}}$. The L^3 algorithm obtains in polynomial time a basis reduced with factor $(\delta, 1/2)$ where $\delta < 1$ can be chosen arbitrarily close to 1. The new L^2 algorithm achieves a factor (δ, η), where $\delta < 1$ can be arbitrarily close to 1 and $\eta > 1/2$ arbitrarily close to $1/2$. It is unknown whether or not $\delta = 1$ can be achieved in polynomial time (attempts to prove it can be found in [9] and [211]). However, it can be noted that $\eta = 1/2$ can be achieved in quadratic time like the L^2 algorithm : first, run the L^2 algorithm on the given input basis with the same factor δ and a factor $\eta \in (1/2, \sqrt{\delta})$;, then run the L^3 algorithm on the output basis. Because the first reduction outputs an almost-reduced basis, the second reduction will only perform size-reduction operations, in which case L^3 has the same complexity bound as the L^2 algorithm given in Theorem B.11.

The L^3 algorithm. The usual L^3 algorithm [205] is described in Figure B.2. It computes an L^3-reduced basis in an iterative fashion : the index κ is such that at any stage of the algorithm, the truncated basis $(\mathbf{b}_1, \ldots, \mathbf{b}_{\kappa-1})$ is L^3-reduced. At each loop iteration, the index κ is either incremented or decremented : the loop stops when the index κ reaches the value $d + 1$, in which case the entire basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ is L^3-reduced.

Input : A basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ and $\delta \in (1/4, 1)$.
Output : An L^3-reduced basis with factor $(\delta, 1/2)$.

1. Compute the rational GSO, i.e., all the $\mu_{i,j}$'s and $r_{i,i}$'s.
2. $\kappa \longleftarrow 2$. While $\kappa \leq d$ do
3. Size-reduce the vector $\mathbf{b}_\kappa$ using the size-reduction algorithm of Figure B.3, which updates the GSO.
4. $\kappa' \longleftarrow \kappa$. While $\kappa \geq 2$ and $\delta \cdot r_{\kappa-1,\kappa-1} \geq r_{\kappa',\kappa'} + \sum_{i=\kappa-1}^{\kappa'-1} \mu_{\kappa',i}^2 r_{i,i}$, do $\kappa \longleftarrow \kappa - 1$.
5. Insert the vector $\mathbf{b}_{\kappa'}$ right before the vector $\mathbf{b}_\kappa$ and update the GSO accordingly.
6. $\kappa \longleftarrow \kappa + 1$.
7. Output $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$.

FIG. B.2 – The L^3 algorithm.

The L^3 algorithm performs two kinds of operations : swaps of consecutive vectors and size-reductions (see Figure B.3), which consist of at most d translations of the form $\mathbf{b}_\kappa \longleftarrow \mathbf{b}_\kappa - m \cdot \mathbf{b}_i$, where m is some integer and $i < \kappa$. Swaps are used to achieve Lovász's conditions, while the size-reduction algorithm is used to size-reduce vectors. Intuitively, size-reductions intend to shorten (or

upper bound) the projection of $\mathbf{b}_\kappa$ over the linear span of $(\mathbf{b}_1, \ldots, \mathbf{b}_{\kappa-1})$, while swaps shorten the projection of $\mathbf{b}_\kappa$ over the orthogonal complement of $(\mathbf{b}_1, \ldots, \mathbf{b}_{\kappa-1})$, that is $\mathbf{b}_\kappa^*$. We explain Steps 4–6 : if Lovász's condition is satisfied, nothing happens in Step 5 and the index κ is incremented like in more classical descriptions of the L^3 algorithm. Otherwise, Step 4 finds the right index to insert the vector $\mathbf{b}_\kappa$, by collecting consecutive failures of Lovász's test.

Input : A basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$, its GSO and an index κ.
Output : The basis where the vector $\mathbf{b}_\kappa$ is size-reduced, and the updated GSO.

1. For $i = \kappa - 1$ down to 1 do
2. $\quad \mathbf{b}_\kappa \longleftarrow \mathbf{b}_\kappa - \lceil \mu_{\kappa,i} \rfloor \cdot \mathbf{b}_i$.
3. $\quad$ Update the GSO accordingly.

FIG. B.3 – The size-reduction algorithm.

If the L^3 algorithm terminates, it is clear that the output basis is L^3-reduced. What is less clear *a priori* is why this algorithm has a polynomial-time complexity. A standard argument shows that each swap decreases the quantity $\Delta = \prod_{i=1}^{d} \|\mathbf{b}_i^*\|^{2(d-i+1)}$ by at least a factor $\delta < 1$. On the other hand, we have that $\Delta \geq 1$ because the $\mathbf{b}_i$'s are integer vectors and Δ can be viewed as a product of squared volumes of lattices spanned by some subsets of the $\mathbf{b}_i$'s. This proves that there can be no more than $O(d^2 \log B)$ swaps, and therefore loop iterations, where B is an upper bound on the norms of the input basis vectors. It remains to estimate the cost of each loop iteration. This cost turns out to be dominated by $O(dn)$ arithmetic operations on the basis matrix and GSO coefficients $\mu_{i,j}$ and $r_{i,i}$ which are rational numbers of bit-length $O(d \log B)$. Thus, the overall complexity of the L^3 algorithm described in Figure B.2 without fast integer arithmetic is :

$$O\left(\left(d^2 \log B\right) \cdot dn \cdot (d \log B)^2\right) = O\left(d^5 n \log^3 B\right).$$

B.2.3 The L^3 Algorithm with Floating-Point Arithmetic

The cost of the L^3 algorithm is dominated by the arithmetic operations on the GSO coefficients which are rationals with huge numerators and denominators. It is therefore tempting to replace the exact GSO coefficients by floating-point approximations, that will be represented with much fewer bits. But doing so in a straightforward manner leads to instability. The algorithm is no longer guaranteed to be polynomial-time : it may not even terminate, because the quantity Δ used to bound the complexity of L^3 algorithm no longer necessarily decreases at each swap : it could be that the new algorithm performs a swap when the initial L^3 algorithm would not have performed such a swap. And if ever the algorithm terminates, the output basis may not be L^3-reduced, due to potential inaccuracy in the GSO coefficients. Prior to this work, the only provable floating-point L^3 algorithm was the one of Schnorr [303], which simulates the behaviour of the L^3 algorithm using floating-point approximations of the coefficients of the inverse matrix of the $\mu_{i,j}$'s. The number of loop iterations and the number of arithmetic operations (in each iteration) remain the same as in the L^3 algorithm (up to a constant factor) : only the cost of each arithmetic operation related to the GSO decreases. Instead of handling integers of length $O(d \log B)$, Schnorr's algorithm uses floating-point numbers with $O(d + \log B)$-bit long mantissæ (with large hidden constants, as mentioned in the introduction). This decreases the worst-case complexity of the L^3 algorithm to $O(d^3 n \log B(d + \log B)^2)$. This is still cubic in $\log B$. Because this algorithm is mostly of theoretical interest, the main number theory computer packages [22, 216, 319] used to implement heuristic floating-point variants of the L^3 algorithm à la Schnorr-Euchner [307] which suffer from stability problems in high dimension. This is no longer the case in [216], which contains an implementation of the L^2 algorithm.

B.3 The L^2 Algorithm

In this section, we present the L^2 algorithm. Its complexity analysis is postponed to the next section.

B.3.1 Overview

The L^2 algorithm is a natural floating-point variant of the L^3 algorithm, which follows the same structure as the L^3 algorithm [205] described in Figure B.2, with two important differences :

- Instead of keeping the GSO in exact rational arithmetic, we will only keep a sufficiently good floating-point approximation, and we will try to simulate the execution of the rational L^3 algorithm. If the floating-point precision is too large, it will be too expensive to compute with the GSO. But if the floating-point precision is too small, the approximation might become too inaccurate, to the point of being meaningless : accuracy is crucial for the size-reductions and for checking the Lovász conditions. We will select a floating-point precision linear in d only, whereas Schnorr's floating-point L^3 algorithm described in [303] uses a larger precision (linear in both d and $\log B$). The fact that the floating-point precision is independent of $\log B$ is crucial to the quadratic complexity of L^2.
- We replace the size-reduction algorithm described in Figure B.3 by an algorithm better suited to floating-point arithmetic. The new algorithm will perform more operations, but it will be more stable : it will tolerate an approximation of the GSO. We will use the fact that when L^2 calls the size-reduction algorithm, we already know that the first $\kappa - 1$ vectors are almost L^3-reduced.

When computing a floating-point approximation of the GSO, it is very important to use exact scalar products $\langle \mathbf{b}_i, \mathbf{b}_j \rangle$: we will only keep a floating-point approximation of the GSO, but we will also keep the exact Gram matrix formed by $\langle \mathbf{b}_i, \mathbf{b}_j \rangle$, and update it during the reduction.

B.3.2 Gram-Schmidt Computations

It is important for the L^2 algorithm to have accurate formulæ for the computation of the GSO coefficients. In [307], the following recursive formulæ were used :

$$\mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j \rangle - \sum_{k=1}^{j-1} \mu_{j,k} \cdot \mu_{i,k} \cdot \|\mathbf{b}_k^*\|^2}{\left\|\mathbf{b}_j^*\right\|^2} \quad \text{and} \quad \|\mathbf{b}_i^*\|^2 = \|\mathbf{b}_i\|^2 - \sum_{j=1}^{i-1} \mu_{i,j}^2 \cdot \left\|\mathbf{b}_j^*\right\|^2 .$$

In these formulæ, the inner products $\langle \mathbf{b}_i, \mathbf{b}_j \rangle$ are computed in floating-point arithmetic, which possibly leads to an absolute error of $2^{-\ell} \cdot \|\mathbf{b}_i\| \cdot \|\mathbf{b}_j\|$, where ℓ is the chosen precision. This happens for example when the vectors $\mathbf{b}_i$ and $\mathbf{b}_j$ are almost orthogonal, i.e., when their scalar product is very small compared to the product of their norms. This has the following drawback : to ensure that the basis returned by the L^2 algorithm is size-reduced, absolute error bounds on the $\mu_{i,j}$'s are required ; if the absolute error on $\langle \mathbf{b}_i, \mathbf{b}_j \rangle$ can be larger than $2^{-\ell}\|\mathbf{b}_i\|\|\mathbf{b}_j\|$, the precision ℓ must be $\Omega(\log B)$ in the worst case (for example when the vector $\mathbf{b}_i$ is very long while $\|\mathbf{b}_j\| = O(1)$). The analyses of [303, 306] do not tackle this issue. We solve this problem by computing the exact Gram matrix at the beginning of the execution of the L^2 algorithm, and by updating it every time the basis matrix is modified : in fact, the transformations are computed from the Gram matrix and applied to the basis matrix. The additional cost is only proportional to the cost of the update of the basis matrix. The advantage is that it allows us to require a floating-point precision of only $O(d)$ bits within the underlying orthogonalization process.

Besides, we use slightly different formulæ by introducing the quantities $r_{i,j} = \mu_{i,j} \cdot \left\|\mathbf{b}_j^*\right\|^2 = \langle \mathbf{b}_i, \mathbf{b}_j^* \rangle$ for all $i \geq j$:

$$r_{i,j} = \langle \mathbf{b}_i, \mathbf{b}_j \rangle - \sum_{k=1}^{j-1} \mu_{j,k} \cdot r_{i,k} \quad \text{and} \quad \mu_{i,j} = \frac{r_{i,j}}{r_{j,j}}.$$

The accuracy is improved because the inner products are extracted from the exact Gram matrix and because each term of the sum now only requires a single multiplication instead of two. For $i = j$, we have $r_{i,i} = \|\mathbf{b}_i\|^2 - \sum_{k=1}^{i-1} \mu_{i,k} \cdot r_{i,k}$, which suggests to define the quantities $s_j^{(i)} = \|\mathbf{b}_i\|^2 - \sum_{k=1}^{j-1} \mu_{i,k} \cdot r_{i,k}$ for all $j \in [\![1, i]\!]$. In particular, we have $s_1^{(i)} = \|\mathbf{b}_i\|^2$ and $s_i^{(i)} = \|\mathbf{b}_i^*\|^2 = r_{i,i}$. The quantities $s_j^{(i)}$ will be useful to check consecutive Lovász's conditions. Indeed, Lovász's condition $\left(\delta - \mu_{\kappa,\kappa-1}^2\right) \cdot \left\|\mathbf{b}_{\kappa-1}^*\right\|^2 \leq \|\mathbf{b}_\kappa^*\|^2$ can be rewritten as $\delta \cdot \left\|\mathbf{b}_{\kappa-1}^*\right\|^2 \leq \|\mathbf{b}_\kappa^*\|^2 + \mu_{\kappa,\kappa-1}^2 \left\|\mathbf{b}_{\kappa-1}^*\right\|^2$, i.e.,

$$\delta \cdot r_{\kappa-1,\kappa-1} \leq s_{\kappa-1}^{(\kappa)}.$$

Whenever the condition is not satisfied, the L^3 algorithm would swap the vectors $\mathbf{b}_{\kappa-1}$ and $\mathbf{b}_\kappa$ and check the following Lovász's condition :

$$\delta \cdot r_{\kappa-2,\kappa-2} \leq s_{\kappa-2}^{(\kappa)}.$$

Thus, storing the $s_j^{(\kappa)}$'s allows us to check consecutive Lovász's conditions (when consecutive swaps occur) without any additional cost since they appear in the calculation of $r_{\kappa,\kappa}$. Notice that at any moment in the execution of the algorithm, no more than κ such quantities are stored. The computation of the $r_{i,j}$'s, $\mu_{i,j}$'s and $s_j^{(d)}$'s is summarized in the so-called Cholesky Factorization Algorithm (CFA) of Figure B.4.

Input : The Gram matrix of $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$.
Output : All the $r_{i,j}$'s, $\mu_{i,j}$'s and $s_j^{(d)}$'s defined by the GSO.

1. For $i = 1$ to d do
2. For $j = 1$ to i do
3. $r_{i,j} \longleftarrow \langle \mathbf{b}_i, \mathbf{b}_j \rangle$,
4. For $k = 1$ to $j - 1$ do $r_{i,j} \longleftarrow r_{i,j} - \mu_{j,k} \cdot r_{i,k}$,
5. $\mu_{i,j} \longleftarrow r_{i,j} / r_{j,j}$.
6. $s_1^{(d)} \longleftarrow \|\mathbf{b}_d\|^2$. For $j = 2$ to d do $s_j^{(d)} \longleftarrow s_{j-1}^{(d)} - \mu_{d,j-1} \cdot r_{d,j-1}$.
7. $r_{d,d} \longleftarrow s_d^{(d)}$.

Fig. B.4 – The Cholesky factorization algorithm (CFA).

Of course, because one uses floating-point arithmetic, the exact values are unknown. Instead, one computes floating-point approximations $\bar{r}_{i,j}$, $\bar{\mu}_{i,j}$ and $\bar{s}_j^{(d)}$. Steps 4–6 are performed in the following way :

$$\begin{aligned}
\bar{r}_{i,j} &\longleftarrow \diamond\left(\bar{r}_{i,j} - \diamond(\bar{\mu}_{j,k} \cdot \bar{r}_{i,k})\right), \\
\bar{\mu}_{i,j} &\longleftarrow \diamond\left(\bar{r}_{i,j} / \bar{r}_{j,j}\right) \\
\bar{s}_j^{(d)} &\longleftarrow \diamond\left(\bar{s}_{j-1}^{(d)} - \diamond(\bar{\mu}_{d,j-1} \cdot \bar{r}_{d,j-1})\right).
\end{aligned}$$

We will not use the CFA directly in the L^2 algorithm. Instead, we will use parts of it during the execution of the algorithm : because the orthogonalization is performed vector by vector, there is no need recomputing everything from scratch if the $r_{i,j}$'s and $\mu_{i,j}$'s are already known for any i and j below some threshold. The CFA will also prove useful in Section B.4, as a first step in the proof of correctness of the L^2 algorithm.

B.3.3 A Lazy Floating-Point Size-Reduction Algorithm

The core of the L^2 algorithm is a lazy floating-point version of the size-reduction algorithm, described in Figure B.5. Instead of size-reducing the vector $\mathbf{b}_\kappa$ at once like in Figure B.3, our floating-point version tries to do the same progressively in several small steps that use the CFA of Figure B.4. Size-reducing in a single step requires a very high accuracy for the floating-point calculations : the required mantissa size could possibly be linear in $\log B$ (see the discussion after Theorem B.13). Rather than doing this, we perform the size-reduction progressively : we make the $\mu_{i,j}$'s decrease a bit, we recompute them with more accuracy (with the CFA), and we go on until they are small enough and known with good accuracy. If we consider the integer translation coefficients x_i that would have been computed if we had performed the size-reduction at once, it intuitively means that we discover them progressively, from the most significant bits to the least significant ones. Although the lazy size-reduction requires more steps than the initial size-reduction, it will overall be less expensive because replacing the exact rational arithmetic of the GSO by floating-point arithmetic outweighs the cost.

Input : A factor $\eta > 1/2$, a floating-point precision ℓ, an index κ, a basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ with its Gram matrix, and floating-point numbers $\bar{r}_{i,j}$'s and $\bar{\mu}_{i,j}$'s for $j \leq i < \kappa$.
Output : Floating-point numbers $\bar{r}_{\kappa,j}$, $\bar{\mu}_{\kappa,j}$ and $\bar{s}_j^{(\kappa)}$ for $j \leq \kappa$, a basis $(\mathbf{b}_1, \ldots, \mathbf{b}_{\kappa-1}, \mathbf{b}'_\kappa, \mathbf{b}_{\kappa+1}, \ldots, \mathbf{b}_d)$ with its Gram matrix, where $\mathbf{b}'_\kappa = \mathbf{b}_\kappa - \sum_{i<\kappa} x_i \cdot \mathbf{b}_i$ for some integers x_i and : $|\langle \mathbf{b}'_\kappa, \mathbf{b}_i^* \rangle| \leq \eta \cdot \|\mathbf{b}_i^*\|^2$ for any $i < \kappa$.

1. $\bar{\eta} \longleftarrow \frac{\eta+1/2}{2}$. Repeat
2. Compute the $\bar{r}_{\kappa,j}$'s, $\bar{\mu}_{\kappa,j}$'s, $\bar{s}_j^{(\kappa)}$'s with Steps 2–7 of the CFA with "$i = \kappa$".
3. For $i = \kappa - 1$ down to 1 do
4. If $|\bar{\mu}_{\kappa,i}| \geq \bar{\eta}$, then $X_i \longleftarrow \lfloor \bar{\mu}_{\kappa,i} \rceil$, else $X_i \longleftarrow 0$,
5. For $j = 1$ to $i - 1$, $\bar{\mu}_{\kappa,j} \longleftarrow \diamond(\bar{\mu}_{\kappa,j} - \diamond(X_i \cdot \bar{\mu}_{i,j}))$.
6. $\mathbf{b}_\kappa \longleftarrow \mathbf{b}_\kappa - \sum_{i=1}^{\kappa-1} X_i \cdot \mathbf{b}_i$, update $G(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ accordingly.
7. Until all the X_i's are zero.

FIG. B.5 – The lazy size-reduction algorithm.

At Step 4, we use $\bar{\eta} = \frac{\eta+1/2}{2} \in (1/2, \eta)$ instead of η to take into account the fact that the $\mu_{\kappa,i}$'s are known only approximately. At Step 6, it suffices to update the scalar products $\langle \mathbf{b}_i, \mathbf{b}_\kappa \rangle$ for $i \leq d$ with the following relations :

$$\begin{aligned} \left\|\mathbf{b}'_\kappa\right\|^2 &= \|\mathbf{b}_\kappa\|^2 + \sum_{j \neq \kappa} X_j^2 \cdot \|\mathbf{b}_j\|^2 - 2 \sum_{j \neq \kappa} X_j \cdot \langle \mathbf{b}_j, \mathbf{b}_\kappa \rangle + 2 \sum_{j \neq \kappa, i \neq \kappa} X_i \cdot X_j \cdot \langle \mathbf{b}_i, \mathbf{b}_j \rangle \\ \langle \mathbf{b}_i, \mathbf{b}'_\kappa \rangle &= \langle \mathbf{b}_i, \mathbf{b}_\kappa \rangle - \sum_{j \neq \kappa} X_j \cdot \langle \mathbf{b}_i, \mathbf{b}_j \rangle \quad \text{for} \quad i \neq \kappa. \end{aligned}$$

B.3.4 Main Results

A description of the L^2 algorithm is given in Figure B.6.

There is no need keeping approximations of all the GSO coefficients : because the algorithm is iterative, it suffices to have approximations up to the threshold κ. Notice that the cost of the first step is bounded by $O(d^2 n \log^2 B)$ and is thus negligible compared to the rest of the execution. At Step 4, we use $\bar{\delta} = \frac{\delta+1}{2} \in (\delta, 1)$ instead of δ to take into account the fact that the quantities $\bar{r}_{\kappa-1,\kappa-1}$ and $\bar{s}_{\kappa-1}^{(\kappa')}$ are known only approximately. The main result of this paper is the following :

Theorem B.11 *Let (δ, η) such that $1/4 < \delta < 1$ and $1/2 < \eta < \sqrt{\delta}$. Let $c > \log \frac{(1+\eta)^2}{\delta - \eta^2}$ be a constant. Given as input a d-dimensional lattice basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ in $\mathbb{Z}^n$ with $\max_i \|\mathbf{b}_i\| \leq B$,*

Input : A valid pair (δ, η) like in Theorem B.11, a basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ and a fp-precision ℓ.
Output : An L^3-reduced basis with factor pair (δ, η).
Variables : An integral matrix G, floating-point numbers $\bar{r}_{i,j}$, $\bar{\mu}_{i,j}$, and $\bar{s}_i$.

1. Compute exactly $G = G(\mathbf{b}_1, \ldots, \mathbf{b}_d)$.
2. $\bar{\delta} \longleftarrow \frac{\delta+1}{2}, \bar{r}_{1,1} \longleftarrow \diamond(\langle \mathbf{b}_1, \mathbf{b}_1 \rangle), \kappa \longleftarrow 2$. While $\kappa \leq d$, do
3. Size-reduce the vector $\mathbf{b}_\kappa$ using the algorithm of Figure B.5, updating the floating-point GSO.
4. $\kappa' \longleftarrow \kappa$. While $\kappa \geq 2$ and $\bar{\delta} \cdot \bar{r}_{\kappa-1,\kappa-1} \geq \bar{s}_{\kappa-1}^{(\kappa')}$, do $\kappa \longleftarrow \kappa - 1$.
5. For $i = 1$ to $\kappa - 1$ do $\bar{\mu}_{\kappa,i} \longleftarrow \bar{\mu}_{\kappa',i}, \bar{r}_{\kappa,i} \longleftarrow \bar{r}_{\kappa',i}, \bar{r}_{\kappa,\kappa} \longleftarrow \bar{s}_\kappa^{(\kappa')}$.
6. Insert the vector $\mathbf{b}_{\kappa'}$ right before the vector $\mathbf{b}_\kappa$ and update G accordingly.
7. $\kappa \longleftarrow \kappa + 1$.
8. Output $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$.

FIG. B.6 – The L^2 algorithm.

the L^2 algorithm of Figure B.6 with precision $\ell = cd + o(d)$ outputs a (δ, η)-L^3-reduced basis in time $O\left(d^4 n \log B(d + \log B)\right)$. More precisely, if τ denotes the number of loop iterations, then the running time is $O\left(d^2 n(\tau + d \log dB)(d + \log B)\right)$.

Let us make a few remarks.

1. The L^2 algorithm decreases the complexity bound $O(d^3 n \log B(d + \log B)^2)$ of [303] by a factor $\frac{d+\log B}{d}$.
2. We can choose δ arbitrarily close to 1 and η arbitrarily close to $1/2$, so that the coefficient $c > \log \frac{(1+\eta)^2}{\delta - \eta^2}$ can be chosen arbitrarily close to $\log 3 < 1.585$. More precisely, we can show the following complexity bound, where the $O(\cdot)$-constant is universal :

$$O\left(\frac{c^2 d^3 n \log B(d + \log B)\left(d - \log\left(\eta - \frac{1}{2}\right)\right)}{(1-\delta) \cdot \left(c - \log \frac{(1+\eta)^2}{\delta - \eta^2}\right)}.\right).$$

   We can therefore choose $\eta = 1/2 + 2^{-d}$ and $\delta = 1 - \frac{1}{\log d}$, while keeping almost the same complexity bound as in Theorem B.11.
3. The additional statement of the theorem that is related to the number of loop iterations is useful for certain lattices which arise in practice, like lattices arising in knapsacks and minimal polynomials, where we possibly have $\tau = O(d \log B)$ instead of the worst-case bound $O(d^2 \log B)$.
4. Finally, the $o(d)$ term in the condition $\ell = c \cdot d + o(d)$ can be made explicit (see Theorem B.15) and may be used backwards : if we perform calculations with a fixed precision (e.g., in the normalized 53-bit mantissæ double precision), the correctness will be guaranteed up to some computable dimension.

The two following sections are devoted to prove Theorem B.11. We will obtain the correctness property in Section B.4, by studying successively the CFA when given as input bases appearing during a L^3-reduction (Theorem B.12), the lazy size-reduction algorithm (Theorem B.14), to finally obtain the desired result in Theorem B.15. The complexity statement of Theorem B.11 will be proved in Section B.5.

B.4 Correctness of the L^2 Algorithm

To guarantee the correctness of the L^2 algorithm, we need to estimate the accuracy of the floating-point approximations at various stages.

B.4.1 Accuracy of the Gram-Schmidt Computations

In general, the classical Gram-Schmidt algorithm is known to have very poor numerical stability properties [123, 203, 357]. However, it must be stressed that in the context of the L^3 algorithm, the bases are reduced in an iterative fashion, which implies that we can study the accuracy of Gram-Schmidt computations under the hypothesis that the first $d-1$ vectors of the input basis are L^3-reduced. In this particular setting, because an L^3-reduced basis is almost orthogonal, the following result shows that a working precision of $\approx d \log 3$ bits is sufficient for the CFA if the reduction factor (δ, η) is sufficiently close to the optimal pair $(1, 1/2)$.

Theorem B.12 *Let (δ, η) be a valid factor pair like in Theorem B.11, $\rho = \frac{(1+\eta)^2+\varepsilon}{\delta-\eta^2}$ with $\varepsilon \in (0, 1/2]$ and $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ in $\mathbb{Z}^n$ be a d-dimensional lattice basis whose Gram matrix is given as input to the CFA described in Figure B.4. Assume that $(\mathbf{b}_1, \ldots, \mathbf{b}_{d-1})$ is (δ, η)-L^3-reduced. In the case of floating-point arithmetic with a precision ℓ satisfying $\ell \geq 5 + 2\log d - \log \varepsilon + d \log \rho$, the output floating-point numbers satisfy the following equations : for all $j \leq i < d$,*

$$\frac{|\bar{r}_{i,j} - r_{i,j}|}{r_{j,j}} \leq d\rho^j 2^{-\ell+2} \quad \text{and} \quad |\bar{\mu}_{i,j} - \mu_{i,j}| \leq d\rho^j 2^{-\ell+4}.$$

Furthermore, if $M = \max_{j<d} |\mu_{d,j}|$, then we have for any $j < d$:

$$\frac{|\bar{r}_{d,j} - r_{d,j}|}{r_{j,j}} \leq d\rho^j 2^{-\ell+2} \cdot M \quad \text{and} \quad |\bar{\mu}_{d,j} - \mu_{d,j}| \leq d\rho^j 2^{-\ell+4} \cdot M.$$

Finally, if the vector $\mathbf{b}_d$ is η-size-reduced with respect to $(\mathbf{b}_1, \ldots, \mathbf{b}_{d-1})$, then for any $j \leq d$:

$$\left| \bar{s}_j^{(d)} - s_j^{(d)} \right| \leq d\rho^j 2^{-\ell+9} \cdot r_{j,j} + d2^{-\ell} \cdot s_j^{(d)}.$$

The second set of inequalities is useful for the analysis of the size-reduction algorithm, while the last set provides guarantees when checking Lovász's conditions.

It is crucial in the theorem to assume that the first vectors are L^3-reduced. More precisely, if the Lovász condition or the size-reduction condition is not satisfied, the result is no longer valid. Heuristically, the Lovász condition ensures that when some $r_{i,j}$ is computed by using the $r_{i,k}$'s with $k < j$, the error on $r_{i,k}$ relatively to $r_{k,k}$ cannot be arbitrarily large relatively to $r_{j,j}$ (since $r_{j,j}$ cannot be arbitrarily small compared to $r_{k,k}$). The size-reduction condition allows to bound the error on $r_{i,j}$ relatively to $r_{j,j}$, independently of the vector $\mathbf{b}_i$ (this vector could be arbitrarily longer than the vector $\mathbf{b}_j$). At the end, it provides an absolute error on the $\mu_{i,j}$'s.

Before giving the proof of Theorem B.12, we first give a sketch of it for the case $\eta \approx 1/2$ and $\delta \approx 1$. Most of the accuracy loss comes from Step 4, which amplifies the error. We define $err_j = \max_{i<d} \frac{|\bar{r}_{i,j} - r_{i,j}|}{r_{j,j}}$, i.e., the error on $r_{i,j}$ relatively to $r_{j,j}$, and we bound its growth as j increases. Obviously :

$$err_1 \leq \frac{|\diamond \langle \mathbf{b}_i, \mathbf{b}_1 \rangle - \langle \mathbf{b}_i, \mathbf{b}_1 \rangle|}{\|\mathbf{b}_1\|^2} \leq 2^{-\ell} \cdot \max_{i<d} \frac{|\langle \mathbf{b}_i, \mathbf{b}_1 \rangle|}{\|\mathbf{b}_1\|^2} \leq 2^{-\ell},$$

because of the size-reduction condition. We now choose $j \in [\![2, d-1]\!]$. The result for $i = d$ can be derived from the proof for $i \leq d-1$, intuitively by replacing "$\mathbf{b}_d$" by "$\frac{1}{M}\mathbf{b}_d$" in it. Because of Step 5, we have, for any $i < d$ and any $k < j$:

$$|\bar{\mu}_{i,k} - \mu_{i,k}| \leq \left| \frac{r_{k,k}}{\bar{r}_{k,k}} \right| err_k + |r_{i,k}| \left| \frac{1}{\bar{r}_{k,k}} - \frac{1}{r_{k,k}} \right| \lesssim \left(\frac{3}{2} \right) err_k,$$

where we neglected low-order terms and used the fact that $|r_{i,k}| \lesssim \frac{1}{2}\|\mathbf{b}_k\|^2$, which comes from the size-reduction condition. This implies that :

$$\begin{aligned} |\diamond(\bar{\mu}_{j,k}\cdot\bar{r}_{i,k}) - \mu_{j,k}r_{i,k}| &\leq |\bar{\mu}_{j,k}-\mu_{j,k}|\cdot|\bar{r}_{i,k}| + |\mu_{j,k}|\cdot|\bar{r}_{i,k}-r_{i,k}| \\ &\lesssim \left(\frac{5}{4}\right) err_k \cdot \|\mathbf{b}_k^*\|^2, \end{aligned}$$

where we also neglected low-order terms and used the size-reduction condition twice. Thus,

$$err_j \lesssim \left(\frac{5}{4}\right)\sum_{k<j}\frac{\|\mathbf{b}_k^*\|^2}{\left\|\mathbf{b}_j^*\right\|^2} err_k \lesssim \left(\frac{5}{4}\right)\sum_{k<j}\left(\frac{4}{3}\right)^{j-k} err_k,$$

by using Lovász's conditions. This last inequality finally gives :

$$err_j \lesssim 3^j \cdot err_1 \lesssim 3^j 2^{-\ell}.$$

Proof. We start with the first inequalities. The goal is to bound the error that we make while computing the $r_{i,j}$'s. At Step 4 of the CFA, we compute the sum $\langle\mathbf{b}_i,\mathbf{b}_j\rangle - \sum_{k=1}^{j-1}\mu_{j,k}\cdot r_{i,k}$ in a naive way. It is classical (see [144] for example) that the error performed during the summation is bounded by :

$$\frac{d2^{-\ell-1}}{1-d2^{-\ell-1}}\cdot\left(|\diamond\langle\mathbf{b}_i,\mathbf{b}_j\rangle| + \sum_{k=1}^{j-1}|\diamond(\bar{\mu}_{j,k}\cdot\bar{r}_{i,k})|\right).$$

As a consequence, we have, by using the triangular inequality, that the quantity $|\bar{r}_{i,j}-r_{i,j}|$ is :

$$\begin{aligned} &\leq \frac{d2^{-\ell-1}}{1-d2^{-\ell-1}}\cdot\left(|\diamond\langle\mathbf{b}_i,\mathbf{b}_j\rangle| + \sum_{k=1}^{j-1}|\diamond(\bar{\mu}_{j,k}\cdot\bar{r}_{i,k})|\right) \\ &\quad + \sum_{k=1}^{j-1}|\diamond(\bar{\mu}_{j,k}\cdot\bar{r}_{i,k}) - \mu_{j,k}r_{i,k}| + |\diamond\langle\mathbf{b}_i,\mathbf{b}_j\rangle - \langle\mathbf{b}_i,\mathbf{b}_j\rangle| \\ &\leq \frac{1}{1-d2^{-\ell-1}}\left(|\diamond\langle\mathbf{b}_i,\mathbf{b}_j\rangle - \langle\mathbf{b}_i,\mathbf{b}_j\rangle| + \sum_{k=1}^{j-1}|\diamond(\bar{\mu}_{j,k}\bar{r}_{i,k}) - \mu_{j,k}r_{i,k}| + \frac{d}{2^{\ell+1}}(|\langle\mathbf{b}_i,\mathbf{b}_j\rangle| + \sum_{k=1}^{j-1}|\mu_{j,k}r_{i,k}|)\right) \\ &\leq \frac{1}{1-d2^{-\ell-1}}\left(|\diamond\langle\mathbf{b}_i,\mathbf{b}_j\rangle - \langle\mathbf{b}_i,\mathbf{b}_j\rangle| + \sum_{k=1}^{j-1}|\diamond(\bar{\mu}_{j,k}\bar{r}_{i,k}) - \mu_{j,k}r_{i,k}| + \frac{d}{2^{\ell+1}}\cdot(|\langle\mathbf{b}_i,\mathbf{b}_j\rangle| + \sum_{k=1}^{j-1}r_{k,k})\right), \quad (*) \end{aligned}$$

where we used the fact that $\eta < 1$, which implies that $|\mu_{j,k}r_{i,k}| \leq \eta\cdot r_{k,k} \leq r_{k,k}$. We show by induction on $j < d$ the following error bound :

$$\forall i \in [\![j, d-1]\!], \ \frac{|\bar{r}_{i,j}-r_{i,j}|}{r_{j,j}} \leq d\rho^j 2^{-\ell+2}.$$

To do this, we define $err_j = \max_{i\in[\![j,d-1]\!]}\frac{|\bar{r}_{i,j}-r_{i,j}|}{r_{j,j}}$ and $E = d\rho^d 2^{-\ell+2} \leq \varepsilon/16$. We first consider the case $j = 1$. We have :

$$|\diamond\langle\mathbf{b}_i,\mathbf{b}_1\rangle - \langle\mathbf{b}_i,\mathbf{b}_1\rangle| \leq 2^{-\ell-1}|\mu_{i,1}|r_{1,1} \leq 2^{-\ell-1}\eta r_{1,1} \leq 2^{-\ell-1}r_{1,1}.$$

Assume now that $j \in [\![2, d-1]\!]$ and that we know that the result holds for all $k < j$. If $k < j \leq i$, we have :

$$\begin{aligned}
|\bar{\mu}_{i,k} - \mu_{i,k}| &\leq \left|\bar{\mu}_{i,k} - \frac{\bar{r}_{i,k}}{\bar{r}_{k,k}}\right| + \left|\frac{\bar{r}_{i,k}}{\bar{r}_{k,k}} - \frac{r_{i,k}}{r_{k,k}}\right| \\
&\leq \left(1 + 2^{-\ell-1}\right) \cdot \left|\frac{\bar{r}_{i,k}}{\bar{r}_{k,k}} - \frac{r_{i,k}}{r_{k,k}}\right| + 2^{-\ell-1}\eta \\
&\leq \left(1 + 2^{-\ell-1}\right) \cdot \left(\frac{|\bar{r}_{i,k} - r_{i,k}|}{r_{k,k}}\frac{r_{k,k}}{\bar{r}_{k,k}} + |r_{i,k}|\frac{|\bar{r}_{k,k} - r_{k,k}|}{r_{k,k}}\frac{1}{\bar{r}_{k,k}}\right) + 2^{-\ell-1} \\
&\leq \left(1 + 2^{-\ell-1}\right) err_k(1+\eta)\frac{r_{k,k}}{\bar{r}_{k,k}} + 2^{-\ell-1} \\
&\leq err_k(1+\eta)\frac{1 + 2^{-\ell-1}}{1 - E} + 2^{-\ell-1},
\end{aligned}$$

because of the induction hypothesis. Therefore, if $k < j \leq i$, we have :

$$\begin{aligned}
\frac{|\diamond(\bar{\mu}_{j,k} \cdot \bar{r}_{i,k}) - \mu_{j,k} r_{i,k}|}{r_{k,k}} &\leq (1 + 2^{-\ell-1}) \cdot \frac{|\bar{\mu}_{j,k}\bar{r}_{i,k} - \mu_{j,k} r_{i,k}|}{r_{k,k}} + 2^{-\ell-1} \\
&\leq (1 + 2^{-\ell-1}) \cdot \left(\frac{|\bar{r}_{i,k} - r_{i,k}|}{r_{k,k}}\bar{\mu}_{j,k} + |\bar{\mu}_{j,k} - \mu_{j,k}|\frac{r_{i,k}}{r_{k,k}}\right) + 2^{-\ell-1} \\
&\leq (1 + 2^{-\ell-1}) \cdot (\eta \cdot err_k + (\eta + E)|\bar{\mu}_{j,k} - \mu_{j,k}|) + 2^{-\ell-1} \\
&\leq (1 + 2^{-\ell-1}) \cdot err_k \left(\eta(2+\eta) + \frac{8E}{1-E}\right) + 129 \cdot 2^{-\ell-7} \\
&\leq err_k \left(\eta(2+\eta) + 2^{-\ell+2} + 16E\right) + 129 \cdot 2^{-\ell-7} \\
&\leq err_k \left(\eta(2+\eta) + \varepsilon\right) + 129 \cdot 2^{-\ell-7},
\end{aligned}$$

where we used the induction hypothesis and the inequalities $E \leq \varepsilon/16 \leq 1/32, \ell \geq 6, \varepsilon \leq 1/2$ and $\eta < 1$. As a consequence, if $j \leq i$, we have :

$$\begin{aligned}
\frac{\sum_{k=1}^{j-1} |\diamond(\bar{\mu}_{j,k} \cdot \bar{r}_{i,k}) - \mu_{j,k} r_{i,k}|}{r_{j,j}} &\leq \sum_{k=1}^{j-1} \left((\eta(2+\eta) + \varepsilon)\, err_k \frac{r_{k,k}}{r_{j,j}} + 129 \cdot 2^{-\ell-7}(\delta - \eta^2)^{k-j}\right) \\
&\leq (\eta(2+\eta) + \varepsilon) \cdot \sum_{k=1}^{j-1} \left(err_k(\delta - \eta^2)^{k-j}\right) + 3 \cdot 129 \cdot 2^{-\ell-7}(\delta - \eta^2)^{-j},
\end{aligned}$$

where we used the fact that $\delta - \eta^2 \leq 3/4$. Similarly, we have :

$$\frac{|\langle \mathbf{b}_i, \mathbf{b}_j \rangle|}{r_{j,j}} \leq \frac{|r_{i,j}|}{r_{j,j}} + \sum_{k=1}^{j-1} |\mu_{j,k}||\mu_{i,k}|\frac{r_{k,k}}{r_{j,j}} \leq \sum_{k=1}^{j} (\delta - \eta^2)^{k-j} \leq 3(\delta - \eta^2)^{-j}.$$

Finally, by using the inequalities $d2^{-\ell-1} \leq 2^{-7}$ and $d \geq 2$, we get, with Equation $(*)$:

$$\begin{aligned}
err_j &\leq \frac{1}{1-d2^{-\ell-1}} \cdot \left(2^{-\ell-1} + (\eta(2+\eta)+\varepsilon) \sum_{k=1}^{j-1} \left(err_k (\delta-\eta^2)^{k-j} \right) + 6d2^{-\ell-1}(\delta-\eta^2)^{-j} \right) \\
&\leq \frac{\eta(2+\eta)+\varepsilon}{1-d2^{-\ell-1}} \cdot \sum_{k=1}^{j-1} \left(err_k (\delta-\eta^2)^{k-j} \right) + \frac{d2^{-\ell+2}}{1-d2^{-\ell-1}} (\delta-\eta^2)^{-j} \\
&\leq \frac{d2^{-\ell+2}}{1-d2^{-\ell-1}} (\delta-\eta^2)^{-j} \cdot \left(1 + \frac{\eta(2+\eta)+\varepsilon}{1-d2^{-\ell-1}} \right)^{j-1} \\
&\leq \frac{d2^{-\ell+2}}{1-d2^{-\ell-1}} \cdot \left(\frac{1+\eta(2+\eta)+\varepsilon}{\delta-\eta^2} \right)^{j} \cdot \frac{1}{1+\eta(2+\eta)+\varepsilon} \cdot \left(1 + d2^{-\ell+3} \right)^{j} \\
&\leq d2^{-\ell+2}\rho^j,
\end{aligned}$$

where we used the fact that $(1+d2^{-\ell+3})^d \leq 2$, itself implied by the inequality $d^2 2^{-\ell+5} \leq 1$.

The result on the μ's is straightforward if we use the facts that $E \leq 1/32$ and $\ell \geq 6$. Furthermore, it is easy to slightly modify the proof to get the desired results on the $r_{d,j}$'s and $\mu_{d,j}$'s for $j < d$.

We now consider the third set of inequalities. We assume that the vector $\mathbf{b}_d$ is size-reduced for the factor η. The analysis is similar to the one just above. If $j \leq d$, we can bound the quantity $\left| \bar{s}_j^{(d)} - s_j^{(d)} \right|$ by :

$$\begin{aligned}
&\leq \frac{d2^{-\ell-1}}{1-d2^{-\ell-1}} \left(\diamond \|\mathbf{b}_d\|^2 + \sum_{k=1}^{j-1} |\diamond(\bar{\mu}_{d,k} \cdot \bar{r}_{d,k})| \right) + \sum_{k=1}^{j-1} |\diamond(\bar{\mu}_{d,k} \cdot \bar{r}_{d,k}) - \mu_{d,k} r_{d,k}| + \left| \diamond \|\mathbf{b}_d\|^2 - \|\mathbf{b}_d\|^2 \right| \\
&\leq \frac{d2^{-\ell-1}}{1-d2^{-\ell-1}} \left(s_j^{(d)} + 2 \sum_{k=1}^{j-1} r_{k,k} \right) + \frac{1}{1-d2^{-\ell-1}} \left(\sum_{k=1}^{j-1} |\diamond(\bar{\mu}_{d,k} \cdot \bar{r}_{d,k}) - \mu_{d,k} r_{d,k}| + 2^{-\ell-1} \|\mathbf{b}_d\|^2 \right) \\
&\leq d2^{-\ell} \cdot s_j^{(d)} + d2^{-\ell+1} \cdot \sum_{k=1}^{j-1} r_{k,k} + d2^{-\ell+6} \cdot \sum_{k=1}^{j-1} r_{k,k} \rho^k,
\end{aligned}$$

where we used the second set of inequalities of the theorem. As a consequence, we have :

$$\begin{aligned}
\left| \bar{s}_j^{(d)} - s_j^{(d)} \right| &\leq d2^{-\ell} \cdot s_j^{(d)} + r_{j,j} d2^{-\ell+4} (\delta-\eta^2)^{-j} + r_{j,j} d2^{-\ell+6} \cdot \sum_{k=1}^{j-1} (\delta-\eta^2)^{k-j} \rho^k \\
&\leq d2^{-\ell} s_j^{(d)} + r_{j,j} d2^{-\ell+9} \rho^j.
\end{aligned}$$

This ends the proof of the theorem. □

The bound in Theorem B.12 seems to be tight in the worst case : the classical Gram-Schmidt algorithm or the CFA become experimentally inaccurate with a precision $\leq d \log 3$ bits for some lattice bases. Consider indeed the L^3-reduced lattice basis given by the rows of the following $d \times d$ matrix L :

$$L_{i,j} = \begin{cases} \left(\sqrt{4/3} \right)^{d-i} & \text{if} \quad i = j \\ (-1)^{i-j+1} L_{j,j} \cdot \text{random}[0.49, 0.5] & \text{if} \quad j < i \\ 0 & \text{if} \quad j > i. \end{cases}$$

To obtain an integral lattice, one can multiply the matrix L by a large scaling factor and round its entries. This matrix is already L^3-reduced. With double precision calculations, i.e., with 53-bit mantissæ, the error on the $\mu_{i,j}$'s becomes significant (higher than 0.5) in dimension 35.

These bases show the tightness of the $d \log 3$ bound. By adding a suitable random vector to such a basis, it is possible to make the LLL_FP routine of NTL loop forever in dimension 55 (available at the URL `http ://www.loria.fr/ stehle/FPLLL.html`). This invalidates the claims of [307] and [194] which state that double precision suffices for lattices of dimension up to ≈ 250 using classical Gram-Schmidt. Nevertheless, these lattice bases seem to be pathological and the floating-point calculations seem to behave much more nicely in practice than in the worst case (see [265] for more details).

B.4.2 Accuracy of Babai's Nearest Plane Algorithm

To estimate the accuracy of the lazy floating-point size-reduction algorithm given in Figure B.5 and used in the L^2 algorithm, we first study a simpler floating-point version that is described in Figure B.7.

Input : A floating-point precision ℓ, a basis $[\mathbf{b}_1, \ldots, \mathbf{b}_d]$, its Gram matrix $G(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ and floating-point numbers $\bar{r}_{i,j}$ and $\bar{\mu}_{i,j}$ for $j \leq i < d$.
Output : Integers $x_1, \ldots, x_{d-1}$, a vector $\mathbf{b}'_d = \mathbf{b}_d - \sum_{i<d} x_i \mathbf{b}_i$ and $G(\mathbf{b}_1, \ldots, \mathbf{b}_{d-1}, \mathbf{b}'_d)$.

1. Compute the $\bar{\mu}_{d,j}$'s for $j < d$ with Steps 2–7 of the CFA with "$i = d$".
2. $\bar{\eta} \longleftarrow \frac{\eta+1/2}{2}$. For $i = d-1$ down to 1 do
3. If $|\bar{\mu}_{d,i}| \geq \bar{\eta}$, then $x_i \longleftarrow \lfloor \bar{\mu}_{d,i} \rceil$, else $x_i \longleftarrow 0$,
4. For $j = 1$ to $i-1$ do $\bar{\mu}_{d,j} \longleftarrow \diamond(\bar{\mu}_{d,j} - \diamond(x_i \cdot \bar{\mu}_{i,j}))$.
5. Compute $\mathbf{b}'_d$ and $G(\mathbf{b}_1, \ldots, \mathbf{b}_{d-1}, \mathbf{b}'_d)$.

FIG. B.7 – A floating-point size-reduction algorithm.

We use Theorem B.12 to show stability properties of the algorithm of Figure B.7.

Theorem B.13 *Let (δ, η) be a valid reduction factor (like in Theorem B.11) and $\rho = \frac{(1+\eta)^2+\varepsilon}{\delta-\eta^2}$ with $\varepsilon \in (0, 1/2]$. Let $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ in $\mathbb{Z}^n$ be a d-dimensional lattice basis given as input to the algorithm of Figure B.7, and $B = \max_i \|\mathbf{b}_i\|$. Assume that the truncated basis $(\mathbf{b}_1, \ldots, \mathbf{b}_{d-1})$ is (δ, η)-L^3-reduced and that the given Gram-Schmidt coefficients $\bar{r}_{i,j}$'s and $\bar{\mu}_{i,j}$'s are those that would have been returned by the CFA with working precision ℓ. Let $M = \max_j |\mu_{d,j}|$. If ℓ satisfies $\ell \geq 5 + 2\log d - \log \varepsilon + d \log \rho$, the algorithm of Figure B.7 finds integers $x_1, \ldots, x_{d-1}$ such that for any $i < d$:*

$$|x_i| \leq 3(1+\eta)^{d-1-i}(M+1) \quad \textit{and} \quad \frac{|\langle \mathbf{b}'_d, \mathbf{b}^*_i \rangle|}{\|\mathbf{b}^*_i\|^2} \leq \bar{\eta} + d\rho^d (M+1) 2^{-\ell+11}.$$

Furthermore, the execution finishes in $O(dn\ell(\ell + d + \log B))$ bit operations.

Proof. We define $\mu^{(i)}_{d,j} = \mu_{d,j} - \sum_{k=i}^{d-1} x_k \mu_{k,j}$ for $i > j$. By using the first set of inequalities of Theorem B.12, we have :

$$\begin{aligned} |\diamond(x_i \cdot \bar{\mu}_{i,j}) - x_i \mu_{i,j}| &\leq \left(1 + 2^{-\ell-1}\right) |x_i| |\bar{\mu}_{i,j} - \mu_{i,j}| + 2^{-\ell-1} |x_i| |\mu_{i,j}| \\ &\leq |x_i| 2^{-\ell-1} \left(1 + 2^5 (1 + 2^{-\ell-1}) d\rho^j\right) \\ &\leq |x_i| d\rho^j 2^{-\ell+5}, \end{aligned}$$

where we used the inequalities $\eta < 1$ and $\ell \geq 6$. At Step 4, we update the quantity $\mu_{d,j}$. It is easy to see that in fact we compute sequentially the sum $\bar{\mu}^{(i)}_{d,j} = \bar{\mu}_{d,j} - \sum_{k=i}^{d-1} x_k \cdot \bar{\mu}_{k,j}$. By using the error

bound of [144] for this naive summation, we obtain that we can bound the quantity $\left|\bar{\mu}_{d,j}^{(i)} - \mu_{d,j}^{(i)}\right|$ by :

$$\begin{aligned}
&\leq \frac{d2^{-\ell-1}}{1-d2^{-\ell-1}}\left(|\bar{\mu}_{d,j}| + \sum_{k=i}^{d-1} |\diamond(x_k \cdot \bar{\mu}_{k,j})|\right) + \sum_{k=i}^{d-1} |\diamond(x_k \cdot \bar{\mu}_{k,j}) - x_k\mu_{k,j}| + |\bar{\mu}_{d,j} - \mu_{d,j}| \\
&\leq \frac{d2^{-\ell-1}}{1-d2^{-\ell-1}}\left(M + \sum_{k=i}^{d-1} |x_k\mu_{k,j}|\right) + \frac{1}{1-d2^{-\ell-1}} \sum_{k=i}^{d-1} |\diamond(x_k \cdot \bar{\mu}_{k,j}) - x_k\mu_{k,j}| + \frac{|\bar{\mu}_{d,j} - \mu_{d,j}|}{1-d2^{-\ell-1}} \\
&\leq \frac{d2^{-\ell-1}}{1-d2^{-\ell-1}}\left(M + \sum_{k=i}^{d-1} |x_k|\right) + \frac{d2^{-\ell+5}\rho^j}{1-d2^{-\ell-1}} \sum_{k=i}^{d-1} |x_k| + \frac{d2^{-\ell+4}\rho^j M}{1-d2^{-\ell-1}} \\
&\leq d2^{-\ell+6}\rho^j \left(M + \sum_{k=i}^{d-1} |x_k|\right),
\end{aligned}$$

where we used the second set of inequalities of Theorem B.12. Assume that $x_j \neq 0$. By taking $i = j+1$ in the previous inequality, we get :

$$\begin{aligned}
|x_j| &\leq \frac{1}{2} + \left|\bar{\mu}_{d,j}^{(j+1)}\right| \\
&\leq \frac{1}{2} + \left|\bar{\mu}_{d,j}^{(j+1)} - \mu_{d,j}^{(j+1)}\right| + \left|\mu_{d,j}^{(j+1)}\right| \\
&\leq \frac{1}{2} + \left(1 + d2^{-\ell+6}\rho^j\right) M + \eta \sum_{k=i}^{d-1} |x_k| \\
&\leq (1+\eta)^{d-1-j}\left(\frac{1}{2} + \left(1 + d2^{-\ell+6}\rho^j\right) M\right) \\
&\leq \frac{3}{2}(1+\eta)^{d-1-j}(1+M),
\end{aligned}$$

where we used the inequalities $\bar{\eta} > 1$ and $d2^{-\ell+6}\rho^j \leq 1/2$.

So far, we have proved the first statement of the theorem. Besides, for any $i > j$, we have :

$$\begin{aligned}
\left|\bar{\mu}_{d,j}^{(i)} - \mu_{d,j}^{(i)}\right| &\leq \left(d2^{-\ell+6}\rho^j\right)\left(M + \frac{3}{2}(1+M)\sum_{k=i}^{d-1}(1+\eta)^{d-1-k}\right) \\
&\leq d2^{-\ell+8}\rho^{d-1}(1+M)(1+\eta)^{d-i},
\end{aligned}$$

where we used the fact that $\eta > 1/2$. This allows us to prove the second statement of the theorem, and to bound the numbers occurring during the computation.

$$\begin{aligned}
\frac{|\langle \mathbf{b}'_d, \mathbf{b}_i^* \rangle|}{\|\mathbf{b}_i^*\|^2} &\leq \bar{\eta} + d2^{-\ell+7}\rho^d(1+M)(1+\eta)^{d-i}, \\
\left|\bar{\mu}_{d,j}^{(i)}\right| &\leq d2^{-\ell+8}\rho^{d-1}(1+M)(1+\eta)^{d-i} + M + \sum_{k=i}^{d-1} |x_k| = 2^{O(d)}(1+M).
\end{aligned}$$

We now consider the costs of the diverse arithmetic operations. The most expensive operations involving floating-point numbers are the additions. The computations on the exponents that correspond to these operations cost $O\left(\log\log(2^{O(d)}(1+M))\right)$ bit operations. From now on, we only consider the bit costs coming from the operations on the mantissæ. Step 1 costs $O(d^2\ell^2)$ elementary operations on bits. The cost of Step 3 is lower than the cost of Step 4. There are at most $O(d^2)$ arithmetic operations during the successive Steps 4. Among them, the multiplications $\diamond(x_i \cdot \bar{\mu}_{i,j})$ are

the most expensive. Since $x_i = \left\lfloor \bar{\mu}_{d,i}^{(i+1)} \right\rceil$, the integer x_i can be represented on $O(\ell)$ bits : if $\ell < d$, then x_i can be written $2^{t_i} x_i'$ for some integers x_i' and $t_i = O(\log(d + \log(1 + M)))$. As a consequence, The total cost of the successive Steps 4 is $O(d^2\ell^2)$. At Step 5, we have $O(dn)$ arithmetic operations involving integers only. By using the relation $|\langle \mathbf{b}_i, \mathbf{b}_d \rangle| \leq \|\mathbf{b}_i\| \cdot \|\mathbf{b}_d\|$ and our bound on the x_i's, we see that all the involved integers are of length $2^{O(d)} \cdot B^2$. The most costly operations are of the type $x_i \cdot \langle \mathbf{b}_d, \mathbf{b}_i \rangle$, and since the x_i's are of length $O(\ell)$, the cost of Step 5 is $O(dn\ell(d + \log B))$. □

Notice that by using the relation $\log M = O(d + \log B)$ (coming from the fact that the $d - 1$ first vectors are L^3-reduced), the last theorem implies that taking $\ell = O(d + \log B)$ is sufficient to make the $|\mu_{d,i}|$'s smaller than η. The drawback of this approach is that one should have previously computed the $r_{i,j}$'s and $\mu_{i,j}$'s with precision $O(d + \log B)$. This seems an overkill since $O(d + \log M)$ bits suffice and M is usually far smaller than B. Indeed, in the case of Euclid's gcd algorithm, the analogy is that the quotients would be computed by using all the bits of the remainders, instead of only the most significant ones (see below for more details).

The lazy size-reduction algorithm from Figure B.5 is a way to work around the difficulty that M cannot be tightly bounded in advance. Using only a $O(d)$-bit precision, it finds the x_j's progressively by performing successive size-reduction steps, each one making $\log M$ decrease by $\Omega(d)$, until we reach $M \leq \bar{\eta}$. This strategy is somewhat similar to the size-reduction routine of the floating-point L^3 algorithm of NTL [319], which repeatedly applies the size-reduction algorithm until nothing happens.

The lazy size-reduction algorithm uses a precision $\ell = \left(\log \frac{(1+\eta)^2}{\delta - \eta^2} + C\right) d + o(d)$ with an arbitrary $C > 0$. The CFA with working precision ℓ gives the input $\bar{r}_{i,j}$'s and $\bar{\mu}_{i,j}$'s, which by Theorem B.12 have their $\approx Cd$ leading bits correct. Therefore, the $r_{i,j}$'s and $\mu_{i,j}$'s may not be known sufficiently well to perform the size-reduction in one single step, but Theorem B.13 gives that their approximations suffice to make $M = \max_{i<\kappa} \frac{|\langle \mathbf{b}_\kappa, \mathbf{b}_i^* \rangle|}{\|\mathbf{b}_i^*\|^2}$ decrease by $\approx Cd$ bits. By making $O\left(1 + \frac{\log M}{d}\right)$ such incomplete size-reductions, size-reduction can be achieved.

Theorem B.14 *Let (δ, η) be a valid pair (like in Theorem B.11) and $\rho = \frac{(1+\eta)^2 + \varepsilon}{\delta - \eta^2}$ with $\varepsilon \in (0, 1/2]$. Let $C > 0$ be a constant. Let $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ be a d-dimensional lattice basis in $\mathbb{Z}^n$ given as input to the algorithm of Figure B.5, and $B = \max_i \|\mathbf{b}_i\|$. Assume that the truncated basis $(\mathbf{b}_1, \ldots, \mathbf{b}_{\kappa-1})$ is (δ, η)-L^3-reduced and that the given $\bar{r}_{i,j}$'s and $\bar{\mu}_{i,j}$'s are those that would have been returned by the CFA with precision ℓ. Let $M = \max_{j<\kappa} |\mu_{\kappa,j}|$. If ℓ satisfies the inequality $\ell \geq 9 + 2\log d - \log \min\left(\varepsilon, \eta - \frac{1}{2}\right) + d\,(C + \log \rho)$, the algorithm of Figure B.5 provides a correct output and the returned $\bar{r}_{\kappa,j}$'s, $\bar{\mu}_{\kappa,j}$'s and $\bar{s}_j^{(\kappa)}$'s are those that would have been returned by the CFA with precision ℓ. Furthermore, if $\ell = O(d)$, the execution finishes in $O(dn(d + \log B)(d + \log M))$ bit operations.*

Proof. We start by the correctness properties of the algorithm. At the last iteration of the main loop, the computed X_j's are all zero, which implies that nothing happens during the corresponding Steps 3–6. This gives that for any $j < \kappa$, we have $|\bar{\mu}_{\kappa,j}| \leq \bar{\eta}$. By using the second set of inequalities of Theorem B.12, we obtain :

$$\max_{j<\kappa} |\mu_{\kappa,j}| \leq \bar{\eta} + d\rho^{d-1} 2^{-\ell+4}.$$

With the hypothesis on ℓ, we obtain that $\frac{|\langle \mathbf{b}'_\kappa, \mathbf{b}_i^* \rangle|}{\|\mathbf{b}_i^*\|^2} \leq \eta$, for all $j < \kappa$. This also gives the correctness of the returned $\bar{r}_{\kappa,j}$'s, $\bar{\mu}_{\kappa,j}$'s and $\bar{s}_j^{(\kappa)}$'s.

We now consider the impact on M of one iteration of the main loop. Let M_1 be the "new M" after the loop iteration. Theorem B.13 and the hypothesis on ℓ give the inequalities :

$$M_1 \leq \bar{\eta} + d\rho^d (1 + M) 2^{-\ell+7} \leq \frac{3/2 + 5\eta}{8} + 2^{-Cd} \frac{\eta - 1/2}{8} M.$$

As a consequence, the k-th iterate of M is smaller than $\frac{3+10\eta}{15-\eta}+2^{-Cdk}M$. Since $\eta-\frac{3+10\eta}{15-\eta} \leq \frac{2}{5}(\eta-1/2)$, there cannot be more than $1+\frac{1}{Cd}O\left(-\log(\eta-1/2)+\log M\right)$ loop iterations.

Theorem B.13 gives that the cost of one loop iteration is bounded by $O(d^2 n(d+\log B))$, since during the execution of the algorithm, the entries of the Gram matrix remain integers of length bounded by $O(d+\log B)$. The fact that we have additional vectors in the basis (namely $\mathbf{b}_{\kappa+1},\ldots,\mathbf{b}_d$) is taken into account in the complexity bound. Finally, the overall cost of the algorithm is bounded by :

$$O\left(d^2 n(d+\log B)\left(1+\frac{\log M}{d}\right)\right) = O(dn(d+\log B)(d+\log M)).$$

□

B.4.3 Application to the L^2 algorithm

We now prove the correctness of the L^2 algorithm. In fact, it follows easily from the following theorem.

Theorem B.15 *Let (δ,η) be a valid pair (like in Theorem B.11) and $\rho=\frac{(1+\eta)^2+\varepsilon}{\delta-\eta^2}$ with $\varepsilon\in(0,1/2]$. Let $C>0$ be a constant. Let $\left(\mathbf{b}_1^{(1)},\ldots,\mathbf{b}_d^{(1)}\right)$ in $\mathbb{Z}^n$ be a lattice basis given as input to the L^2 algorithm. For any loop iteration t, let $\left(\mathbf{b}_1^{(t)},\ldots,\mathbf{b}_d^{(t)}\right)$ denote the current basis at the beginning of the t-th iteration. Assume that the floating-point precision ℓ satisfies $d^2\rho^d 2^{-\ell+9+Cd} \leq \min\left(\varepsilon,\eta-\frac{1}{2},1-\delta\right)$. Then we have :*

1. *For any t, the truncated basis $\left(\mathbf{b}_1^{(t)},\ldots,\mathbf{b}_{\kappa(t)-1}^{(t)}\right)$ is L^3-reduced with factor pair (δ,η).*
2. *For any $i\leq d$, $\max_{j\leq i}\left\|\mathbf{b}_j^{(t)*}\right\| \leq \max_{j\leq i}\left\|\mathbf{b}_j^{(1)*}\right\|$ and $\left\|\mathbf{b}_i^{(t)}\right\| \leq \sqrt{d}\cdot\max_{j\leq i}\left\|\mathbf{b}_i^{(1)}\right\|$.*

Proof. We show the result by induction on t. Clearly, all these properties are valid for $t=1$, since $\kappa(1)=2$. Assume now that we are performing the t-th loop iteration. We show that Lovász's tests work as desired. Recall that the vector $\mathbf{b}_\kappa$ is swapped with the vector $\mathbf{b}_i$ for $i<\kappa$ if and only if for any $j\in[\![i,\kappa-1]\!]$ we have $\bar{s}_j^{(\kappa)} \leq \bar{\delta}\bar{r}_{j,j}$. Assume first that the vector $\mathbf{b}_\kappa$ is not swapped with the vector $\mathbf{b}_i$. Theorem B.12 gives that :

$$s_i^{(\kappa)}(1-d2^{-\ell}) \;\leq\; r_{i,i}\left(\bar{\delta}\left(1+d\rho^{d-1}2^{-\ell+2}\right)+d\rho^{d-1}2^{-\ell+9}\right).$$

Therefore, since $d^2\rho^d 2^{-\ell+9}\leq 1-\delta$, if the vector $\mathbf{b}_\kappa$ is swapped with the vector $\mathbf{b}_i$, then they would have also been swapped with the classical L^3 algorithm with some factor $\frac{3-\delta}{2}\in(\delta,1)$. On the opposite, assume that the vectors $\mathbf{b}_\kappa$ and $\mathbf{b}_i$ are not swapped. Theorem B.12 gives :

$$s_i^{(\kappa)}(1+d2^{-\ell}) \;\geq\; r_{i,i}\left(\bar{\delta}\left(1+d\rho^{d-1}2^{-\ell+2}\right)+d\rho^{d-1}2^{-\ell+9}\right).$$

As a consequence, if the vectors $\mathbf{b}_\kappa$ and $\mathbf{b}_i$ are not swapped, they would not have been swapped with the classical L^3 algorithm with the factor δ. This gives the first statement of the theorem.

For the second statement, observe that during a swap between the vectors $\mathbf{b}_\kappa$ and $\mathbf{b}_{\kappa-1}$, we have :

1. $\left\|\mathbf{b}_{\kappa-1}^{*new}\right\| \leq \left\|\mathbf{b}_{\kappa-1}^{*old}\right\|$ because of Lovász's condition,
2. $\left\|\mathbf{b}_\kappa^{*new}\right\| \leq \left\|\mathbf{b}_{\kappa-1}^{*old}\right\|$ because the vector $\mathbf{b}_\kappa^{*new}$ is an orthogonal projection of the vector $\mathbf{b}_{\kappa-1}^{*old}$,

which gives the first part of the second statement. Finally, if the vector $\mathbf{b}_i^{(t)}$ appears during the execution of the algorithm and is size-reduced, we have :

$$\left\|\mathbf{b}_i^{(t)}\right\|^2 \leq d\cdot\max_{j\leq i}\left\|\mathbf{b}_j^{(t)*}\right\|^2 \leq d\cdot\max_{j\leq i}\left\|\mathbf{b}_j^{(1)*}\right\|^2 \leq d\cdot\max_{j\leq i}\left\|\mathbf{b}_j^{(1)}\right\|^2.$$

This proves the second part of the statement for all $i < \kappa(t)$. If $i \geq \kappa(t)$, we consider the largest $t' < t$ such that $\kappa(t'+1) - 1 = i$. The loop iteration t' was the one during which was created the vector $\mathbf{b}_i^{(t)}$. If t' does not exist, this vector is an initial vector and the result is obvious. Otherwise the vector $\mathbf{b}_i^{(t)} = \mathbf{b}_{\kappa(t'+1)-1}^{(t'+1)}$ is size-reduced at the $(t'+1)$-th loop iteration. Thus we have $\left\|\mathbf{b}_i^{(t)}\right\|^2 \leq d \cdot \max_{j \leq \kappa(t'+1)-1} \left\|\mathbf{b}_j^{(1)}\right\|^2$. Since κ cannot increase by more than 1 in a single iteration, we must have $i \geq \kappa(t'+1) - 1$, which ends the proof of the theorem. □

B.5 Complexity Analysis of the L^2 Algorithm

In Section B.4.3, we showed that an accuracy of $O(d)$ bits suffices to check Lovász's conditions. For any Lovász test, either the index κ increases or decreases by one and when it decreases, the quantity $\prod_{i=1}^{d} \|\mathbf{b}_i^*\|^{2(d-i)}$ decreases by a factor of at least $\frac{3-\delta}{2} > 1$. It is a standard L^3 argument that this quantity is actually an integer (it is a product of squared volumes of integer lattices) and is initially bounded by $B^{O(d^2)}$. But during the execution of the algorithm, the difference between the numbers of decreases and increases of κ is $O(d)$, so there are $O(d^2 \log B)$ loop iterations.

In this section we show how to achieve the claimed complexity bound $O(d^4 n \log B(d+\log B))$. This is done by generalising a cascade phenomenon appearing in the analysis of Euclid's gcd algorithm.

B.5.1 Analysing Euclid's Gcd Algorithm

As mentioned in the introduction, the L^3 algorithm can be viewed as a high-dimensional generalization of Euclid's algorithm to compute gcds. But this analogy is not completely satisfiying : indeed, Euclid's algorithm has a quadratic complexity bound without fast integer arithmetic, whereas the L^3 algorithm is cubic for any fixed dimension without fast integer arithmetic.

Recall Euclid's algorithm : given as input two integers $r_0 > r_1 > 0$, it successively computes the quotients q_i and remainders r_i defined by :

$$q_i = \lfloor r_{i-1}/r_i \rfloor \quad \text{and} \quad r_{i+1} = r_{i-1} - q_i r_i,$$

until $r_{\tau+1} = 0$ for some loop iteration τ. Then r_τ is the gcd of the integers r_0 and r_1. It is well-known that the remainders decrease at least geometrically, so that the number of Euclidean divisions is $\tau = O(\log r_0)$. A very naive analysis of Euclid's algorithm states that the algorithm performs $O(\log r_0)$ arithmetic operations on integers of lengths bounded by $O(\log r_0)$, so that the overall cost is bounded by $O(\log^3 r_0)$. A well-known more subtle analysis notices that the cost of computing q_i and r_{i+1} without fast integer arithmetic is bounded by $O(\log r_{i-1} \cdot (1 + \log q_i)) = O(\log r_0 \cdot (1 + \log r_{i-1} - \log r_i))$. Summed over all the steps, all but two terms "$\log r_i$" vanish, leading to the classical quadratic complexity bound.

Unfortunately, this amortized analysis of Euclid's algorithm cannot be extended to the standard L^3 algorithm, because the GSO coefficients stay big. In Euclid's algorithm, the numbers get smaller and smaller : in L^3, the basis vectors get shorter and shorter, but there still remain very large GSO coefficients. Surprisingly, we will show that the amortized analysis of Euclid's algorithm can be extended to the L^2 algorithm, and this would not be possible without a floating-point precision independent of $\log B$. The main difficulty is to generalize the cancellation of all but a very few terms in the sum of the costs of consecutive loop iterations. In Euclid's algorithm, this cancellation is trivial because two consecutive costs compensate each other directly. The phenomenon is much more complicated in higher dimension : the cost of the t-th iteration will not necessarily be balanced by the cost of the previous $(t-1)$-th iteration, but by the cost of the t'-th iteration for some $t' < t$. Special care must be taken so that the t''s do not collide.

B.5.2 An Amortized Analysis in Arbitrary Dimension

We now complete the proof of Theorem B.11. We already know that the number of loop iterations is $\tau = O(d^2 \log B)$. Thanks to Theorem B.14, we also know that the t-th loop iteration costs $O(dn(d+\log B)(d+\log M(t)))$ bit operations, where $M(t) = \max_{j<\kappa(t)} \left|\mu^{(t)}_{\kappa(t),j}\right|$. By analogy with Euclid's gcd algorithm, we make terms cancel out in the sum over the loop iterations of the "$\log M(t)$'s". For this purpose, we define the index $\alpha(t)$ as the smallest swapping index since the last time the index κ was at least $\kappa(t)$.

Lemma B.29 *Let t be a loop iteration. Let $\varphi(t) = \max(t' < t, \kappa(t') \geq \kappa(t))$ if it exists and 1 otherwise, and let $\alpha(t) = \min(\kappa(t'), t' \in [\![\varphi(t), t-1]\!]) - 1$. Then we have :*

$$\log M(t) \leq O(d) + \log \left\|\mathbf{b}^{(t)}_{\kappa(t)}\right\| - \log \left\|\mathbf{b}^{(t)}_{\alpha(t)}\right\|.$$

Proof. Between the loop iterations $\varphi(t)$ and t, the vectors $\mathbf{b}_1, \ldots, \mathbf{b}_{\alpha(t)-1}$ remain unchanged and because of the size-reductions, each vector created during these loop iterations is size-reduced with respect to the vectors $\mathbf{b}_1, \ldots, \mathbf{b}_{\alpha(t)-1}$. This includes the vector $\mathbf{b}^{(t)}_{\kappa(t)}$. As a consequence, by using the fact that $\left(\mathbf{b}^{(t)}_1, \ldots, \mathbf{b}^{(t)}_{\kappa(t)-1}\right)$ is L^3-reduced (thanks to Theorem B.15), we have :

$$\begin{aligned}
M(t) = \max_{i<\kappa(t)} |\mu_{\kappa(t),i}| &= \max\left(\max_{i<\alpha(t)} |\mu_{\kappa(t),i}|, \max_{i\in[\![\alpha(t),\kappa(t)-1]\!]} |\mu_{\kappa(t),i}|\right) \\
&\leq \max\left(\eta, \max_{i\in[\![\alpha(t),\kappa(t)-1]\!]} \frac{\left\|\mathbf{b}^{(t)}_{\kappa(t)}\right\|}{\left\|\mathbf{b}^{(t)*}_{i}\right\|}\right) \\
&\leq \eta + (\delta-\eta^2)^{-(\kappa(t)-\alpha(t))/2} \frac{\left\|\mathbf{b}^{(t)}_{\kappa(t)}\right\|}{\left\|\mathbf{b}^{(t)*}_{\alpha(t)}\right\|} \\
&\leq \eta + \sqrt{d}(\delta-\eta^2)^{-d/2} \frac{\left\|\mathbf{b}^{(t)}_{\kappa(t)}\right\|}{\left\|\mathbf{b}^{(t)}_{\alpha(t)}\right\|},
\end{aligned}$$

since $\left\|\mathbf{b}^{(t)}_{\alpha(t)}\right\| \leq \sqrt{d} \cdot \max_{i\leq\alpha(t)} \left\|\mathbf{b}^{(t)*}_{i}\right\| \leq \sqrt{d}(\delta-\eta^2)^{-\alpha(t)/2} \left\|\mathbf{b}^{(t)*}_{\alpha(t)}\right\|$. □

We are to subdivide the sum of the $\log M(t)$'s over the successive loop iterations into $O(d)$ subsums according to the value of the index $\kappa(t)$:

$$\sum_{t\leq\tau} \left(d + \log \left\|\mathbf{b}^{(t)}_{\kappa(t)}\right\| - \log \left\|\mathbf{b}^{(t)}_{\alpha(t)}\right\|\right) \leq \tau d + \sum_{k=2}^{d} \sum_{t,\kappa(t)=k} \left(\log \left\|\mathbf{b}^{(t)}_{k}\right\| - \log \left\|\mathbf{b}^{(t)}_{\alpha(t)}\right\|\right).$$

For each of these subsums, we keep $k-1$ positive terms and $k-1$ negative terms, and make the others vanish in a progressive cancellation. Terms proportional to d can appear from such cancellations, but they will be dominated by the above "τd". The crucial point to do this is the following :

Lemma B.30 *Let $k \in [\![2, d]\!]$ and $t_1 < \ldots < t_k$ be loop iterations of the L^2 algorithm such that for any $j \leq k$, we have $\kappa(t_j) = k$. Then there exists $j < k$ with :*

$$d(\delta-\eta^2)^{-d} \left\|\mathbf{b}^{(t_j)}_{\alpha(t_j)}\right\| \geq \left\|\mathbf{b}^{(t_k)}_{k}\right\|.$$

To prove this result, we need the following technical fact :

Lemma B.31 *Let T and j be integers such that $\kappa(T) \geq j \geq \kappa(T+1)$. We have :*

$$\max_{i \leq j} \left\| \mathbf{b}_i^{(T+1)*} \right\| \leq \max_{i<j} \left\| \mathbf{b}_i^{(T)*} \right\| \quad \textit{and} \quad \max_{i \leq j} \left\| \mathbf{b}_i^{(T+1)} \right\| \leq \sqrt{d} \cdot \max_{i<j} \left\| \mathbf{b}_i^{(T)} \right\|.$$

Proof. If $i \leq j$ is different from $\kappa(T+1)-1$, then the vector $\mathbf{b}_i^{(T+1)}$ is a vector $\mathbf{b}_{i'}^{(T)}$ for some $i' \leq j-1$. Thanks to the size-reduction and to Theorem B.15, it suffices to show that $\left\| \mathbf{b}_{\kappa(T+1)-1}^{(T+1)*} \right\| \leq \max_{i<j} \left\| \mathbf{b}_i^{(T)*} \right\|$. Since the Lovász test has failed for the loop iteration T and the index $\kappa(T+1)-1$, we have :

$$\left\| \mathbf{b}_{\kappa(T+1)-1}^{(T+1)*} \right\| \leq \left\| \mathbf{b}_{\kappa(T+1)-1}^{(T)*} \right\|.$$

□

Proof of Lemma B.30. We choose $j = \max(i \leq k, \alpha(t_i) \geq i)$. This definition is valid because the maximized set is not empty (it contains $i=1$). Since $\alpha(t_k) < k$ and $\kappa(t_k) = \kappa(t_{k-1}) = k$, there exists a first loop iteration $T_k \in [\![t_{k-1}, t_k - 1]\!]$ such that $\kappa(T_k) \geq k \geq \kappa(T_k+1)$. Because of Theorem B.15 (for the first inequality) and Lemma B.31 (for the second inequality), we have :

$$\left\| \mathbf{b}_k^{(t_k)} \right\| \leq \sqrt{d} \cdot \max_{i \leq k} \left\| \mathbf{b}_i^{(T_k+1)} \right\| \leq d \cdot \max_{i \leq k-1} \left\| \mathbf{b}_i^{(T_k)} \right\|.$$

By definition of T_k, the vectors $\mathbf{b}_1, \ldots, \mathbf{b}_{k-1}$ do not change between the loop iterations t_{k-1} and T_k. Since the vectors $\mathbf{b}_1^{(t_{k-1})}, \ldots, \mathbf{b}_{k-1}^{(t_{k-1})}$ are L^3-reduced, we have :

$$\left\| \mathbf{b}_k^{(t_k)} \right\| \leq d \cdot \max_{i \leq k-1} \left\| \mathbf{b}_i^{(t_{k-1})} \right\| \leq d(\delta - \eta^2)^{-d/2} \cdot \max_{i \leq k-1} \left\| \mathbf{b}_i^{(t_{k-1})*} \right\|.$$

If $j = k-1$, we have the result because in this case :

$$\max_{i \leq k-1} \left\| \mathbf{b}_i^{(t_{k-1})*} \right\| \leq (\delta - \eta^2)^{-d/2} \left\| \mathbf{b}_{\alpha(t_{k-1})}^{*} \right\| \leq (\delta - \eta^2)^{-d/2} \left\| \mathbf{b}_{\alpha(t_{k-1})} \right\|.$$

Otherwise there exists a first loop iteration $T_{k-1} \in [\![t_{k-2}, t_{k-1} - 1]\!]$ such that $\kappa(T_{k-1}) \geq k-1 \geq \kappa(T_{k-1}+1)$. From Lemma B.31, we have :

$$\begin{aligned} \left\| \mathbf{b}_k^{(t_k)} \right\| &\leq d(\delta - \eta^2)^{-d/2} \cdot \max_{i \leq k-1} \left\| \mathbf{b}_i^{(T_{k-1}+1)*} \right\| \\ &\leq d(\delta - \eta^2)^{-d/2} \cdot \max_{i \leq k-2} \left\| \mathbf{b}_i^{(T_{k-1})*} \right\| \\ &\leq d(\delta - \eta^2)^{-d/2} \cdot \max_{i \leq k-2} \left\| \mathbf{b}_i^{(t_{k-2})*} \right\| \end{aligned}$$

If $j = k-2$, we have the result, otherwise we go on constructing loop iterations T_i's in a similar fashion to obtain the result. □

It is now possible to complete the complexity analysis of the L^2 algorithm. Let $k \in [\![2, d]\!]$ and $t_1 < \ldots < t_{\tau_k} = \{t \leq \tau, \kappa(t) = k\}$. We extract from the global sum the terms corresponding to these loop iterations. Theorem B.15 and the fact we are dealing with an integer lattice ensure that :

$$\sum_{i=1}^{\tau_k} \log \frac{\left\| \mathbf{b}_k^{(t_i)} \right\|}{\left\| \mathbf{b}_{\alpha(t_i)}^{(t_i)} \right\|} \leq (k-1)\log(\sqrt{d}B) + \sum_{i=k}^{\tau_k} \log \left\| \mathbf{b}_k^{(t_i)} \right\| - \sum_{i=1}^{\tau_k - k + 1} \log \left\| \mathbf{b}_{\alpha(t_i)}^{(t_i)} \right\|.$$

Lemma B.30 helps to tightly bound the right-hand term above. First, we apply it with $t_1, \ldots, t_k$. This shows that there exists $j < k$ such that $\left\| \mathbf{b}_k^{(t_k)} \right\| \leq d(\delta - \eta^2)^{-d} \left\| \mathbf{b}_{\alpha(t_j)}^{(t_j)} \right\|$. The indices "$i = k$" in the

positive sum and "$i = j$" in the negative sum cancel out and a term "$-d\log(\delta-\eta^2)+\log d$" appears. Then we use Lemma B.30 with t_{k+1} and the $k-1$ first t_i's that remain in the negative sum. It is easy to see that t_{k+1} is larger than any of them, so that we can have another "positive-negative" pair which cancels out in another "$-d\log(\delta-\eta^2)+\log d$". We perform this operation $\tau_k - k + 1$ times, to obtain :

$$\sum_{i=1}^{\tau_k} \log \frac{\left\|\mathbf{b}_k^{(t_i)}\right\|}{\left\|\mathbf{b}_{\alpha(t_i)}^{(t_i)}\right\|} \leq (k-1)\log(\sqrt{d}B) + \tau_k\left(-d\log(\delta-\eta^2)+\log d\right).$$

The fact that $\sum_k \tau_k = \tau$ finally gives :

$$\sum_{t\leq\tau} (O(d) + \log M(t)) = O(\tau d + d^2 \log dB).$$

B.6 The L^2 Algorithm for Linearly Dependent Vectors

We show in this section that the L^2 algorithm can be extended to linearly dependent vectors. The modification resembles the one described in [286]. It can be used to find integer linear relations between linearly dependent lattice vectors. We add a new index ζ that has the following meaning : any vector of index $< \zeta$ is a zero vector, while any vector of index $\geq \zeta$ is non-zero. At any loop iteration of the algorithm, the vectors $\mathbf{b}_{\zeta+1}, \ldots, \mathbf{b}_{\kappa-1}$ are L^3-reduced. The rest of the modified algorithm is the same as in the L^2 algorithm of Figure B.6. The modified algorithm is given in Figure B.8.

Input : A valid pair (δ, η) like in Th. B.11, non-zero vectors $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ and a fp-precision ℓ.
Output : An L^3-reduced basis with factor pair (δ, η).
Variables : An integral matrix G, floating-point numbers $\bar{r}_{i,j}$, $\bar{\mu}_{i,j}$ and $\bar{s}_i$.

1. Compute exactly $G = G(\mathbf{b}_1, \ldots, \mathbf{b}_d)$.
2. $\bar{\delta} \longleftarrow \frac{\delta+1}{2}, \bar{r}_{1,1} \longleftarrow \diamond(\langle \mathbf{b}_1, \mathbf{b}_1\rangle), \kappa \longleftarrow 2, \zeta \longleftarrow 0$. While $\kappa \leq d$, do
3. Size-reduce the vector $\mathbf{b}_\kappa$ using the algorithm of Figure B.5 for the vectors $\mathbf{b}_{\zeta+1}, \ldots, \mathbf{b}_\kappa$. It updates the floating-point GSO.
4. $\kappa' \longleftarrow \kappa$. While $\kappa \geq \zeta + 2$ and $\bar{\delta}\cdot\bar{r}_{\kappa-1,\kappa-1} \geq \bar{s}_{\kappa-1}^{(\kappa')}$, do $\kappa \longleftarrow \kappa - 1$.
5. For $i = \zeta + 1$ to $\kappa - 1$ do $\bar{\mu}_{\kappa,i} \longleftarrow \bar{\mu}_{\kappa',i}, \bar{r}_{\kappa,i} \longleftarrow \bar{r}_{\kappa',i}, \bar{r}_{\kappa,\kappa} \longleftarrow \bar{s}_\kappa^{(\kappa')}$.
6. Insert $\mathbf{b}_{\kappa'}$ right before $\mathbf{b}_\kappa$ and update G accordingly.
7. If $\mathbf{b}_\kappa = \mathbf{0}$, $\zeta \longleftarrow \zeta + 1$.
8. $\kappa \longleftarrow \max(\zeta + 2, \kappa + 1)$.
9. Output $(\mathbf{b}_{\zeta+1}, \ldots, \mathbf{b}_d)$.

FIG. B.8 – The modified L^2 algorithm.

The following theorem gives the bit complexity of the modified L^2 algorithm.

Theorem B.16 *Let (δ, η) such that $1/4 < \delta < 1$ and $1/2 < \eta < \sqrt{\delta}$. Let $c > \log\frac{(1+\eta)^2}{\delta-\eta^2}$ be a constant. Given as input d vectors $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ in $\mathbb{Z}^n$ with $\max_i \|\mathbf{b}_i\| \leq B$, the modified L^2 algorithm of Figure B.8 with precision $\ell = cd + o(d)$ outputs a (δ, η)-L^3-reduced basis of the lattice L generated by the $\mathbf{b}_i$'s in $O(k^2 n(k + \log B)(k^2 \log B + d(d-k)))$ bit operations, where $k = \dim L \leq d$. More precisely, if τ denotes the number of loop iterations, then the running time is $O(k^2 n(\tau + k\log kB)(k + \log B))$.*

Since most of the proof of this theorem is identical to the one of Theorem B.11, we only point out the differences. Notice first that at any moment of the algorithm, there can be no more than k non-zero vectors of index smaller than κ : Theorem B.15 remains valid, so that the vectors $\mathbf{b}_{\zeta+1}, \ldots, \mathbf{b}_{\kappa-1}$ are

L^3-reduced, but L^3-reduced vectors must be linearly independent. This implies that in the modified version of Theorem B.14, the complexity bound of the lazy size-reduction becomes :

$$O(kn(k + \log B)(k + \log M)) \quad \text{bit operations.}$$

It remains to bound the number of loop iterations of the modified L^2 algorithm. For all $i \leq k$, let d_i be the product of the i first non-zero $\|\mathbf{b}_i^*\|^2$'s. The d_i's are positive integers since they are the determinants of the Gram matrices of subsets of the $\mathbf{b}_i$'s. We also define :

$$D = \left(\prod_{i \leq \dim L} d_i \right) \cdot \left(\prod_{i, \mathbf{b}_i^* = \mathbf{0}} 2^i \right).$$

The quantity D is an integer that is initially bounded by $B^{O(k^2)} \cdot 4^{d(d-k)}$. We are to show that this quantity decreases by some constant factor each time there is a swap in the modified L^2 algorithm of Figure B.8. This will imply that there can be no more than $O(k^2 \log B + d(d-k))$ loop iterations.

Assume that the vectors $\mathbf{b}_\kappa$ and $\mathbf{b}_{\kappa-1}$ are not swapped. We consider three cases :

1. Both vectors $\mathbf{b}_{\kappa-1}^*$ and $\mathbf{b}_\kappa^*$ are non-zero. Then the right side of the quantity D is constant, and the left side decreases by a factor $\geq 1/\delta$: the first d_i's do not change because none of the terms of the corresponding products is changing ; the last d_i's do not change either because the only terms of the corresponding products that change are $\left\|\mathbf{b}_{\kappa-1}^*\right\|$ and $\|\mathbf{b}_\kappa^*\|$ but their product remains the same ; the remaining d_i has a single term that is changing, the term $\left\|\mathbf{b}_{\kappa-1}^*\right\|^2$, that is decreasing by a factor $\geq 1/\delta$.
2. If $\mathbf{b}_{\kappa-1}^* \neq \mathbf{0}$ and $\mathbf{b}_\kappa^* = \mathbf{0}$ and if the new vector $\mathbf{b}_{\kappa-1}^*$ is zero, after the swap we will have $\mathbf{b}_{\kappa-1}^* = \mathbf{0}$ and $\mathbf{b}_\kappa^* \neq \mathbf{0}$. More precisely, the new vector $\mathbf{b}_\kappa^*$ will exactly be the last vector $\mathbf{b}_{\kappa-1}^*$. As a consequence, the left side of the quantity D is constant, while the right side decreases by a factor 2.
3. If $\mathbf{b}_{\kappa-1}^* \neq \mathbf{0}$ and $\mathbf{b}_\kappa^* = \mathbf{0}$ and if the new vector $\mathbf{b}_{\kappa-1}^*$ is non-zero, then the new vector $\mathbf{b}_\kappa^*$ must be zero since the dimension of the lattice $L(\mathbf{b}_1, \ldots, \mathbf{b}_\kappa)$ shall remain constant. So the right side of the quantity D is constant. On the opposite, all the d_i's above some threshold are decreasing by a factor $\geq 1/\delta$: none of the $\|\mathbf{b}_i^*\|^2$'s is changing except $\left\|\mathbf{b}_{\kappa-1}^*\right\|^2$ that decreases by a factor $\geq 1/\delta$.

The case $\mathbf{b}_{\kappa-1}^* = \mathbf{0}$ cannot occur, since at the loop iteration for which the vector $\mathbf{b}_{\kappa-1}$ would have been created and inserted at a rank $\kappa - 1 \geq \zeta + 1$, Lovász's condition between this vector and the previous one would not have been satisfied.

Remark. Like the L^3 algorithm, the L^2 algorithm and its modified version work on the underlying quadratic form. In particular, the modified L^2 algorithm of Figure B.8 can easily be adapted to the case of possibly non-definite and possibly non-positive integer quadratic forms, by adding absolute values in the Lovász conditions (see [329] for more details).

Acknowledgements

We thank Richard Brent, Guillaume Hanrot, Claus Peter Schnorr and Paul Zimmermann for numerous discussions. The writing of the present article was completed while the second author was visiting the University of Sydney, whose hospitality is gratefully acknowledged.

Annexe C

LLL on the Average

ANTS-VII (2006)
[265] avec Damien Stehlé (LORIA)

Abstract: *Despite their popularity, lattice reduction algorithms remain mysterious in many ways. It has been widely reported that they behave much more nicely than what was expected from the worst-case proved bounds, both in terms of the running time and the output quality. In this article, we investigate this puzzling statement by trying to model the average case of lattice reduction algorithms, starting with the celebrated Lenstra-Lenstra-Lovász algorithm (L^3). We discuss what is meant by lattice reduction on the average, and we present extensive experiments on the average case behavior of L^3, in order to give a clearer picture of the differences/similarities between the average and worst cases. Our work is intended to clarify the practical behavior of L^3 and to raise theoretical questions on its average behavior.*

C.1 Introduction

Lattices are discrete subgroups of $\mathbb{R}^n$. A basis of a lattice L is a set of $d \leq n$ linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_d$ in $\mathbb{R}^n$ such that L is the set $L[\mathbf{b}_1, \ldots, \mathbf{b}_d] = \left\{ \sum_{i=1}^{d} x_i \mathbf{b}_i, x_i \in \mathbb{Z} \right\}$ of all integer linear combinations of the $\mathbf{b}_i$'s. The integer d matches the dimension of the linear span of L : it is called the dimension of the lattice L. A lattice has infinitely many bases (except in trivial dimension ≤ 1), but some are more useful than others. The goal of *lattice reduction* is to find interesting lattice bases, such as bases consisting of reasonably short and almost orthogonal vectors. Finding good reduced bases has proved invaluable in many fields of computer science and mathematics (see [65, 129]), particularly in cryptology (see [240, 267]).

The first lattice reduction algorithm in arbitrary dimension is due to Hermite [142]. It was introduced to show the existence of Hermite's constant and of lattice bases with bounded orthogonality defect. Very little is known on the complexity of Hermite's algorithm : the algorithm terminates, but its polynomial-time complexity remains an open question. The subject had a revival with Lenstra's celebrated work on integer programming [207, 208], which used an approximate variant of Hermite's algorithm. Lenstra's variant was only polynomial-time for fixed dimension, which was however sufficient in [207]. This inspired Lovász to develop a polynomial-time variant of the algorithm, which reached a final form in [205] where Lenstra, Lenstra and Lovász applied it to factor rational polynomials in polynomial time, from whom the name L^3 comes. Further refinements of L^3 were later proposed, notably by Schnorr [302, 303]. Currently, the most efficient provable variant of L^3 known

in case of large entries, called L^2, is due to Nguyen and Stehlé [264], and is based on floating-point arithmetic. Like L^3, it can be viewed as a relaxed version of Hermite's algorithm.

Our Contribution. One of the main reasons why lattice reduction has proved invaluable in many fields is the widely reported experimental fact that lattice reduction algorithms, L^3 in particular, behave much more nicely than what could be expected from the worst-case proved bounds, both in terms of the running time and the output quality. However, to our knowledge, this mysterious phenomenon has never been described in much detail. In this article, we try to give a clearer picture and to give heuristic arguments that explain the situation. We start by discussing what is meant by the average case of lattice reduction, which is related to notions of random lattices and random bases. We then focus on L^3. Regarding the output quality, it seems as if the only difference between the average and worst cases of L^3 in high dimension is a change of constants : while the worst-case behavior of L^3 is closely related to Hermite's constant in dimension two $\gamma_2 = \sqrt{4/3}$, the average case involves a smaller constant whose value is only known experimentally : ≈ 1.04. So while L^3 behaves better than expected, it does not behave that much better : the approximation factors seem to remain exponential in d. Regarding the running time, there is no surprise for the so-called integer version of L^3, except when the input lattice has a special shape such as knapsack-type lattices. However, there can be significant changes with the floating-point variants of L^3. We give a family of bases for which the average running time should be asymptotically close to the worst-case bound, and explain why for reasonable input sizes the executions are faster.

Applications. Guessing the quality of the bases output by L^3 is very important for several reasons. First, all lattice reduction algorithms known rely on L^3 at some stage and their behavior is therefore strongly related to that of L^3. A better understanding of their behavior should provide a better understanding of stronger reduction algorithms such as Schnorr's BKZ [302] and is thus useful to estimate the hardness of lattice problems (which is used in several public-key cryptosystems, such as NTRU [149] and GGH [118]). Besides, if after running L^3, one obtains a basis which is worse than expected, then one should randomize the basis and run L^3 again. Another application comes from the so-called floating-point (*fp* for short) versions of L^3. These are very popular in practice because they are usually much faster. They can however prove tricky to use because they require tuning : if the precision used in *fp*-arithmetic is not chosen carefully, the algorithm may no longer terminate, and if it terminates, it may not give an L^3-reduced basis. On the other hand, the higher the precision, the slower the execution. Choosing the right precision for *fp*-arithmetic is thus important in practice and it turns out to be closely related to the average-case quality of the bases output by the L^3 algorithm.

The table below sums up our results, for d-dimensional lattices whose initial basis vectors are of lengths smaller than B, with $n = \Theta(d)$ and $d = O(\log B)$.

	$\frac{\|\mathbf{b}_1\|}{(\det L)^{1/d}}$	Running time of L^2	Required prec. for L^2
Worst-case bound	$(4/3)^{d/4}$	$O(d^5 \log^2 B)$	$\approx 1.58d + o(d)$
Average-case estim.	$(1.02)^d$	$O(d^4 \log^2 B) \rightarrow O(d^5 \log^2 B)$	$0.18d + o(d)$

Road map. In Section C.2 we provide necessary background on L^3. We discuss random lattices and random bases in Section C.3. Then we describe our experimental observations on the quality of the computed bases (Section C.4), the running time (Section C.5) and the numerical behavior (Section C.6).

Additional Material. All experiments were performed with the `fplll-1.2` software, which is available at `http ://www.loria.fr/~stehle/practLLL.html`. The data used to draw the figures of the paper and some others are also available at this URL.

C.2 Background

NOTATION. All logarithms are in base 2. Let $\|\cdot\|$ and $\langle\cdot,\cdot\rangle$ be the Euclidean norm and inner product of $\mathbb{R}^n$. The notation $\lceil x \rfloor$ denotes a closest integer to x. Bold variables are vectors. All the lattices we consider are integer lattices, as usual. All our complexity results are given for the bit complexity model, without fast integer arithmetic. Our *fpa*-model is a smooth extension of the IEEE-754 standard, as provided by NTL [319] (RR class) and MPFR [289].

We recall basic notions from algorithmic geometry of numbers (see [240]).

First minimum. If L is a lattice, we denote by $\lambda(L)$ its *first minimum*.

Gram matrix. Let $\mathbf{b}_1, \ldots, \mathbf{b}_d$ be vectors. Their *Gram matrix* $G(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ is the $d \times d$ symmetric matrix $(\langle \mathbf{b}_i, \mathbf{b}_j \rangle)_{1 \leq i,j \leq d}$ formed by all the inner products.

Gram-Schmidt orthogonalization. Let $\mathbf{b}_1, \ldots, \mathbf{b}_d$ be linearly independent vectors. The *Gram-Schmidt orthogonalization* (GSO) $[\mathbf{b}_1^*, \ldots, \mathbf{b}_d^*]$ is the orthogonal family defined as follows : $\mathbf{b}_i^*$ is the component of $\mathbf{b}_i$ orthogonal to the linear span of $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$. We have $\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^*$ where $\mu_{i,j} = \langle \mathbf{b}_i, \mathbf{b}_j^* \rangle / \|\mathbf{b}_j^*\|^2$. For $i \leq d$, we let $\mu_{i,i} = 1$. The lattice L spanned by the $\mathbf{b}_i$'s satisfies $\det L = \prod_{i=1}^{d} \|\mathbf{b}_i^*\|$. The GSO family depends on the order of the vectors. If the $\mathbf{b}_i$'s are integer vectors, the $\mathbf{b}_i^*$'s and the $\mu_{i,j}$'s are rational. In what follows, the *GSO family* denotes the $\mu_{i,j}$'s, together with the quantities $r_{i,j}$'s defined as : $r_{i,i} = \|\mathbf{b}_i^*\|^2$ and $r_{i,j} = \mu_{i,j} r_{j,j}$ for $j < i$.

Size-reduction. A basis $[\mathbf{b}_1, \ldots, \mathbf{b}_d]$ is *size-reduced* with factor $\eta \geq 1/2$ if its GSO family satisfies $|\mu_{i,j}| \leq \eta$ for all $j < i$. The i-th vector $\mathbf{b}_i$ is *size-reduced* if $|\mu_{i,j}| \leq \eta$ for all $j < i$. Size-reduction usually refers to $\eta = 1/2$, but it is essential for *fp* variants of L^3 to allow larger η.

L^3-reduction. A basis $[\mathbf{b}_1, \ldots, \mathbf{b}_d]$ is L^3-reduced with factor (δ, η) with $1/4 < \delta \leq 1$ and $1/2 \leq \eta < \sqrt{\delta}$ if the basis is η-size-reduced and if its GSO satisfies the $(d-1)$ Lovász conditions $(\delta - \mu_{\kappa,\kappa-1}^2) r_{\kappa-1,\kappa-1} \leq r_{\kappa,\kappa}$ (or equivalently $\delta \|\mathbf{b}_{\kappa-1}^*\|^2 \leq \|\mathbf{b}_\kappa^* + \mu_{\kappa,\kappa-1} \mathbf{b}_{\kappa-1}^*\|^2$), which implies that the $\|\mathbf{b}_\kappa^*\|$'s never drop too much. Such bases have useful properties (see [205]), like providing approximations to the shortest and closest vector problems. In particular, the first vector is relatively short : $\|\mathbf{b}_1\| \leq \beta^{(d-1)/4} (\det L)^{1/d}$, where $\beta = 1/(\delta - \eta^2)$. And the first basis vector is at most exponentially far away from the first minimum : $\|\mathbf{b}_1\| \leq \beta^{(d-1)/2} \lambda(L)$. L^3-reduction usually refers to the factor $(3/4, 1/2)$ initially chosen in [205], in which case $\beta = 2$. But the closer (δ, η) is to $(1, 1/2)$, the shorter $\mathbf{b}_1$ should be. In practice, one usually selects $\delta \approx 1$ and $\eta \approx 1/2$, so that $\beta \approx 4/3$ and therefore $\|\mathbf{b}_1\| \lesssim (4/3)^{(d-1)/4} (\det L)^{1/d}$. The L^3 algorithm obtains in polynomial time a $(\delta, 1/2)$-L^3-reduced basis where $\delta < 1$ can be chosen arbitrarily close to 1. The L^2 algorithm achieves a factor (δ, η), where $\delta < 1$ can be arbitrarily close to 1 and $\eta > 1/2$ arbitrarily close to $1/2$.

Input : A basis $[\mathbf{b}_1, \ldots, \mathbf{b}_d]$ and $\delta \in (1/4, 1)$.
Output : An L^3-reduced basis with factor $(\delta, 1/2)$.
1. Compute the rational GSO, i.e., all the $\mu_{i,j}$'s and $r_{i,i}$'s.
2. $\kappa \longleftarrow 2$. While $\kappa \leq d$ do
3. Size-reduce $\mathbf{b}_\kappa$ using the algorithm of Figure C.2, that updates the GSO.
4. $\kappa' \longleftarrow \kappa$. While $\kappa \geq 2$ and $\delta r_{\kappa-1,\kappa-1} \geq r_{\kappa',\kappa'} + \sum_{i=\kappa-1}^{\kappa'-1} \mu_{\kappa',i}^2 r_{i,i}$, do $\kappa \longleftarrow \kappa - 1$.
5. For $i = 1$ to $\kappa - 1$, $\mu_{\kappa,i} \longleftarrow \mu_{\kappa',i}$. Insert $\mathbf{b}_{\kappa'}$ right before $\mathbf{b}_\kappa$.
6. $\kappa \longleftarrow \kappa + 1$.
7. Output $[\mathbf{b}_1, \ldots, \mathbf{b}_d]$.

FIG. C.1 – The L^3 Algorithm.

The L^3 algorithm. The L^3 algorithm [205] is described in Figure C.1. It computes an L^3-reduced basis in an iterative fashion : the index κ is such that at any stage of the algorithm, the truncated basis $[\mathbf{b}_1, \ldots, \mathbf{b}_{\kappa-1}]$ is L^3-reduced. At each loop iteration, κ is either incremented or decremented :

Input : A basis $[\mathbf{b}_1, \ldots, \mathbf{b}_d]$, its GSO and an index κ.
Output : The basis with $\mathbf{b}_\kappa$ size-reduced and the updated GSO.
1. For $i = \kappa - 1$ down to 1 do
2. $\quad \mathbf{b}_\kappa \longleftarrow \mathbf{b}_\kappa - \lceil \mu_{\kappa,i} \rfloor \mathbf{b}_i$.
3. $\quad$ For $j = 1$ to i do $\mu_{\kappa,j} \longleftarrow \mu_{\kappa,j} - \lceil \mu_{\kappa,i} \rfloor \mu_{i,j}$.
4. Update the GSO accordingly.

FIG. C.2 – The size-reduction algorithm.

the loop stops when κ reaches the value $d+1$, in which case the entire basis $[\mathbf{b}_1, \ldots, \mathbf{b}_d]$ is L^3-reduced.

If L^3 terminates, it is clear that the output basis is L^3-reduced. What is less clear *a priori* is why L^3 has a polynomial-time complexity. A standard argument shows that each swap decreases the quantity $\Delta = \prod_{i=1}^{d} \|\mathbf{b}_i^*\|^{2(d-i+1)}$ by at least a factor $\delta < 1$, while $\Delta \geq 1$ because the $\mathbf{b}_i$'s are integer vectors and Δ can be viewed as a product of squared volumes of lattices spanned by some of the $\mathbf{b}_i$'s. This proves that there are $O(d^2 \log B)$ swaps, and therefore loop iterations, where B is an upper bound on the norms of the input basis vectors. It remains to estimate the cost of each loop iteration. This cost turns out to be dominated by $O(dn)$ arithmetic operations on the basis matrix and GSO coefficients $\mu_{i,j}$ and $r_{i,i}$ which are rational numbers of bit-length $O(d \log B)$. Thus, the overall complexity of L^3 is $O((d^2 \log B) \cdot dn \cdot (d \log B)^2)) = O(d^5 n \log^3 B)$.

L^3 with *fpa*. The cost of L^3 is dominated by the operations on the GSO coefficients which are rationals with huge numerators and denominators. It is therefore tempting to replace the exact GSO coefficients by *fp* approximations. But doing so in a straightforward manner leads to numerical anomalies. The algorithm is no longer guaranteed to be polynomial-time : it may not even terminate. And if ever it terminates, the output basis may not be L^3-reduced. The main number theory computer packages [22, 216, 319] contain heuristic *fp*-variants of L^3 *à la* Schnorr-Euchner [307] suffering from stability problems. On the theoretic side, the fastest provable *fp* variant of L^3 is Nguyen-Stehlé's L^2 [264], whose running time is $O(d^4 n(d + \log B) \log B)$. The main algorithmic differences with Schnorr-Euchner's *fp* L^3 are that the integer Gram matrix is updated during the execution (thus avoiding cancellations while computing scalar products with *fpa*), and that the size-reduction algorithm is replaced by a lazy variant (this idea was already in Victor Shoup's NTL code). In L^2, the worst-case required precision for *fpa* is $\leq 1.59d + o(d)$. The proved variant of `fplll-1.2` implements L^2.

C.3 Random Lattices

In this section, we give the main methods known to generate random lattices and random bases, and describe the random bases we use in our experiments.

C.3.1 Random Lattices

When experimenting with L^3, it seems natural to work with random lattices, but what is a random lattice ? From a practical point of view, one could just select randomly generated lattices of interest, such as lattices used in cryptography or in algorithmic number theory. This would already be useful but one might argue that it would be insufficient to draw conclusions, because such lattices may not be considered random in a mathematical sense. For instance, in many cryptanalyses, one applies reduction algorithms to lattices whose first minimum is much shorter than all the other minima.

From a mathematical point of view, there is a natural notion of random lattice, which follows from a measure on n-dimensional lattices with determinant 1 introduced by Siegel [325] back in 1945, to provide an alternative proof of the Minkowski-Hlwaka theorem. Let $X_n = SL_n(\mathbb{R})/SL_n(\mathbb{Z})$ be the space of (full-rank) lattices in $\mathbb{R}^n$ modulo scale. The group $G = SL_n(\mathbb{R})$ possesses a unique (up to

scale) bi-invariant Haar measure, which can be thought of as the measure it inherits as a hypersurface in $\mathbb{R}^{n^2}$. When mapping G to the quotient $X_n = G/SL_n(\mathbb{Z})$, the Haar measure projects to a finite measure μ on the space X_n which we can normalize to have total volume 1. This measure μ is G-invariant : if $A \subseteq X_n$ is measurable and $g \in G$, then $\mu(A) = \mu(gA)$. In fact, μ can be characterized as the unique G invariant Borel probability measure on X_n. This gives rise to a natural notion of random lattices. The recent articles [6, 3, 121] propose efficient ways to generate lattices which are random in this sense. For instance, Goldstein and Mayer [121] show that for large N, the (finite) set $\mathcal{L}_{n,N}$ of n-dimensional integer lattices of determinant N is uniformly distributed in X_n in the following sense : given any measurable subset $A \subseteq X_n$ whose boundary has zero measure with respect to μ, the fraction of lattices in $\mathcal{L}_{n,N}/N^{1/n}$ that lie in A tends to $\mu(A)$ as N tends to infinity.

Thus, to generate lattices that are random in a natural sense, it suffices to generate uniformly at random a lattice in $\mathcal{L}_{n,N}$ for large N. This is particularly easy when N is prime. Indeed, when p is a large prime, the vast majority of lattices in $\mathcal{L}_{n,p}$ are lattices spanned by row matrices of the following form :

$$R_p^n = \begin{pmatrix} p & 0 & 0 & \dots & 0 \\ x_1 & 1 & 0 & \dots & 0 \\ x_2 & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ x_{n-1} & 0 & \dots & 0 & 1 \end{pmatrix},$$

where the x_i's are chosen independently and uniformly in $\{0, \dots, p-1\}$.

C.3.2 Random Bases

Once a lattice has been selected, it would be useful to select a random basis, among the infinitely many bases. This time however, there is no clear definition of what is a random basis, since there is no finite measure on $SL_n(\mathbb{Z})$. Since we mostly deal with integer lattices, one could consider the Hermite normal form (HNF) of the lattice, and argue that this is the basis which gives the least information on the lattice, because it can be computed in polynomial time from any basis. However, it could also be argued that the HNF may have special properties, depending on the lattice. For instance, the HNF of NTRU lattices [149] is already reduced in some sense, and does not look like a random basis at all. A random basis should consist of long vectors : the orthogonality defect should not be bounded, since the number of bases with bounded orthogonality defect is bounded. In other words, a random basis should not be reduced at all.

A heuristic approach was used for the GGH cryptosystem [118]. Namely, a secret basis was transformed into a large public basis of the same lattice by multiplying generators of $SL_n(\mathbb{Z})$ in a random manner. However, it is difficult to control the size of the entries, and it looks hard to obtain theoretical results.

One can devise a less heuristic method as follows. Consider a full-rank integer lattice $L \subseteq \mathbb{Z}^n$. If B is much bigger than $(\det L)^{1/n}$, it is possible to sample efficiently and uniformly points in $L \cap [-B, B]^n$ (see [4]). For instance, if $B = (\det L)/2$, one can simply take an integer linear combination $x_1\mathbf{b}_1 + \cdots + x_n\mathbf{b}_n$ of a basis, with large coefficients x_i, and reduce the coordinates modulo $\det L = [\mathbb{Z}^n : L]$. That is easy for lattices in the previous set $\mathcal{L}_{n,p}$ where p is prime. Once we have such a sampling procedure, we note that n vectors randomly chosen in such a way will with overwhelming probability be linearly independent. Though they are unlikely to form a lattice basis (rather, they will span a sublattice), one can easily lift such a full-rank set of linearly independent vectors of norm $\leq B$ to a basis made of vectors of norm $\leq B\sqrt{n}/2$ using Babai's nearest plane algorithm [17] (see [4] or [240, Lemma 7.1]). In particular, if one considers the lattices of the class $\mathcal{L}_{n,p}$, it is easy to generate plenty of bases in a random manner in such a way that all the coefficients of the basis vectors are $\leq p\sqrt{n}/2$.

C.3.3 Random L^3-Reduced Bases

There are two natural notions of random L^3 bases. One is derived from the mathematical definition. An L^3-reduced basis is necessarily Siegel-reduced (following the definition of [58]), that is, its $\mu_{i,j}$'s and $\|\mathbf{b}_i^*\|/\|\mathbf{b}_{i+1}^*\|$'s are bounded. This implies [58] that the number of L^3-reduced bases of a given lattice is finite (for any reduction parameters), and can be bounded independently of the lattice. Thus, one could define a random L^3 basis as follows : select a random lattice, and among all the finitely many L^3-reduced bases of that lattice, select one uniformly at random. Unfortunately, the latter process is impractical, but it might be interesting to prove probabilistic statements on such bases. Instead, one could try the following in practice : select a random lattice, then select a random basis, and eventually apply the L^3 algorithm. The output basis will not necessarily be random in the first sense, since the L^3 algorithm may bias the distribution. However, intuitively, it could also be viewed as some kind of random L^3 basis. In the previous process, it is crucial to select a random-looking basis (unlike the HNF of NTRU lattices). For instance, if we run the L^3 algorithm on already reduced (or almost reduced) bases, the output basis will differ from a typical L^3-reduced basis.

C.3.4 Random Bases in Our Experiments

In our experiments, besides the Goldstein-Mayer [121] bases of random lattices, we considered two other types of random bases. The Ajtai-type bases of dimension d and factor α are given by the rows of a lower triangular random matrix B with :

$$B_{i,i} = \lfloor 2^{(2d-i+1)^\alpha} \rfloor \quad \text{and} \quad B_{j,i} = \text{rand}(-B_{i,i}/2, B_{i,i}/2) \quad \text{for all } j > i.$$

Similar bases have been used by Ajtai in [7] to show the tightness of worst-case bounds of [302]. The bases used in Coppersmith's root-finding method [70] bear some similarities with what we call Ajtai-type bases.

We define the knapsack-type bases as the rows of the $d \times (d+1)$ matrices :

$$\begin{pmatrix} A_1 & 1 & 0 & \dots & 0 \\ A_2 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_d & 0 & 0 & \dots & 1 \end{pmatrix},$$

where the A_i's are sampled independently and uniformly in $[-B, B]$, for some given bound B. Such bases often occur in practice, e.g., in cryptanalyses of knapsack-based cryptosystems, reconstructions of minimal polynomials and detections of integer relations between real numbers. The behavior of L^3 on this type of bases and on the above R_p^{d+1}'s look alike.

Interestingly, we did not notice any significant change in the output quality or in the geometry of reduced bases between all three types of random bases.

C.4 The Output Quality of L^3

For fixed parameters δ and η, the L^3 and L^2 algorithms output bases $\mathbf{b}_1, \dots, \mathbf{b}_d$ such that

$$\|\mathbf{b}_{i+1}^*\|^2/\|\mathbf{b}_i^*\|^2 \geq \beta = 1/(\delta - \eta^2)$$

for all $i < d$, which implies that :

$$\|\mathbf{b}_1\| \leq \beta^{(d-1)/4}(\det L)^{1/d} \quad \text{and} \quad \|\mathbf{b}_1\| \leq \beta^{(d-1)/2}\lambda(L).$$

It is easy to prove that these bounds are tight in the worst case : both are reached for some reduced bases of some particular lattices. However, there is a common belief that they are not tight in practice.

For instance, Odlyzko wrote in [274] :

This algorithm [...] *usually finds a reduced basis in which the first vector is much shorter than guaranteed* [theoretically]. *(In low dimensions, it has been observed empirically that it usually finds the shortest non-zero vector in a lattice.)*

We argue that the quantity $\|\mathbf{b}_1\|/(\det L)^{1/d}$ remains exponential on the average, but is indeed far smaller than the worst-case bound : for δ close to 1 and η close to 1/2, one should replace $\beta^{1/4} \approx (4/3)^{1/4}$ by ≈ 1.02, so that the approximation factor $\beta^{(d-1)/4}$ becomes $\approx 1.02^d$. As opposed to the worst-case bounds, the ratio $\|\mathbf{b}_1\|/\lambda(L)$ should also be $\approx 1.02^d$ on the average, rather than being the square of $\|\mathbf{b}_1\|/(\det L)^{1/d}$. Indeed, if the Gaussian heuristic holds for a lattice L, then $\lambda(L) \approx \sqrt{\frac{d}{2\pi e}}(\det L)^{1/d}$. The Gaussian heuristic is only a heuristic in general, but it can be proved for random lattices (see [6, 3]), and it is unlikely to be wrong by an exponential factor, unless the lattice is very special.

Heuristic 1 *Let δ be close to 1 and η be close to 1/2. Given as input a random basis of almost any lattice L of sufficiently high dimension (e.g., larger than 40), L^3 and L^2 with parameters δ and η output a basis whose first vector $\mathbf{b}_1$ satisfies $\|\mathbf{b}_1\|/(\det L)^{1/d} \approx (1.02)^d$.*

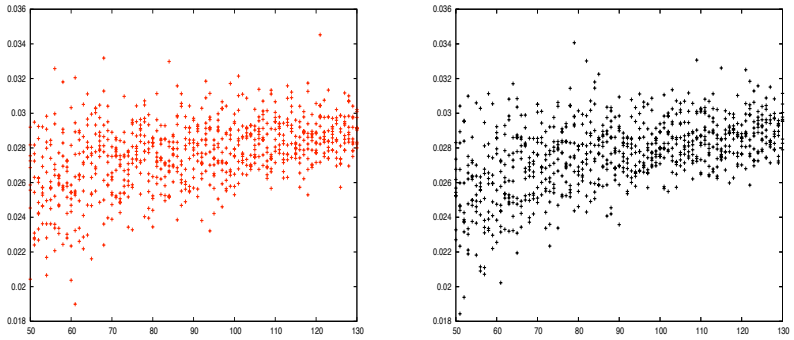


FIG. C.3 – Variation of $\frac{1}{d}\log\frac{\|\mathbf{b}_1\|}{(\det L)^{1/d}}$ as a function of d.

C.4.1 A Few Experiments

In Figure C.3, we consider the variations of the quantity $\frac{1}{d}\log\frac{\|\mathbf{b}_1\|}{(\det L)^{1/d}}$ as the dimension d increases. On the left side of the figure, each point is a sample of the following experiment : generate a random knapsack-type basis with $B = 2^{100\cdot d}$ and reduce it with L^2 (the fast variant of `fplll-1.2` with $(\delta, \eta) = (0.999, 0.501)$). The points on the right side correspond to the same experiments, but starting with Ajtai-type bases, with $\alpha = 1.2$. The two sides of Figure C.3 are similar and the quantity $\frac{1}{d}\log\frac{\|\mathbf{b}_1\|}{(\det L)^{1/d}}$ seems to converge slightly below 0.03 (the corresponding worst-case constant is ≈ 0.10). This means that the first output vector $\mathbf{b}_1$ usually satisfies $\|\mathbf{b}_1\| \approx (1.02)^d(\det L)^{1/d}$. The exponential quantity $(1.02)^d$ remains tiny even in moderate dimensions : e.g., $(1.02)^{50} \approx 2.7$ and $(1.02)^{100} \approx 7.2$. These data may explain why in the 80's, cryptanalysts used to believe that L^3 returns vectors surprisingly small compared to the worst-case bound.

C.4.2 The Configuration of Local Bases

To understand the shape of the bases that are computed by L^3, it is tempting to consider the local bases of the output bases, i.e., the pairs $(\mathbf{b}_i^*, \mu_{i+1,i}\mathbf{b}_i^* + \mathbf{b}_{i+1}^*)$ for $i < d$. These pairs are the components of $\mathbf{b}_i$ and $\mathbf{b}_{i+1}$ which are orthogonal to $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$. We experimentally observe that after the reduction, local bases seem to share a common configuration, independently of the index i. In Figure C.4, a point corresponds to a local basis (its coordinates are $\mu_{i+1,i}$ and $\|\mathbf{b}_{i+1}^*\|/\|\mathbf{b}_i^*\|$) of a basis returned by the fast variant of `fplll-1.2` with parameters $\delta = 0.999$ and $\eta = 0.501$, starting from a knapsack-type basis with $B = 2^{100 \cdot d}$. The 2100 points correspond to 30 reduced bases of 71-dimensional lattices. This distribution seems to stabilize between the dimensions 40 and 50.

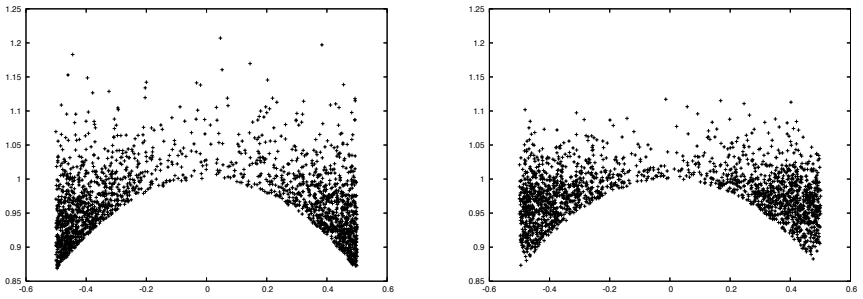


FIG. C.4 – Distribution of the local bases after L^3 (left) and deep-L^3 (right).

Figure C.4 is puzzling. First of all, the $\mu_{i+1,i}$'s are not uniformly distributed in $[-\eta, \eta]$, as one may have thought *a priori*. As an example, the uniform distribution was used as an hypothesis Theorem 2 in [194]. Our observation therefore invalidates this result. This non-uniformity is surprising because the other $\mu_{i,j}$'s seem to be uniformly distributed in $[-\eta, \eta]$, in particular when $i - j$ becomes larger, as it is illustrated by Figure C.5. The mean value of the $|\mu_{i+1,i}|$'s is close to 0.38. Besides, the mean value of $\|\mathbf{b}_i^*\|/\|\mathbf{b}_{i+1}^*\|$ is close to 1.04, which matches the 1.02 constant of the previous subsection. Indeed, if the local bases behave independently, we have :

$$\frac{\|\mathbf{b}_1\|^d}{\det L} = \prod_{i=1}^{d} \frac{\|\mathbf{b}_1\|}{\|\mathbf{b}_i^*\|} = \prod_{i=1}^{d-1} \left(\frac{\|\mathbf{b}_i^*\|}{\|\mathbf{b}_{i+1}^*\|}\right)^{d-i+1} \approx (1.04)^{d^2/2} \approx (1.02)^{d^2}.$$

A possible explanation of the shape of the pairs $(\mathbf{b}_i^*, \mu_{i+1,i}\mathbf{b}_i^* + \mathbf{b}_{i+1}^*)$ is as follows. During the execution of L^3, the ratios $\|\mathbf{b}_i^*\|/\|\mathbf{b}_{i+1}^*\|$ are decreasing steadily. At some moment, the ratio $\|\mathbf{b}_i^*\|/\|\mathbf{b}_{i+1}^*\|$ becomes smaller than $\sqrt{4/3}$. When it does happen, relatively to $\mathbf{b}_i^*$, either $\mu_{i+1,i}\mathbf{b}_i^* + \mathbf{b}_{i+1}^*$ lies in one of the corners of Figure C.4 or is close to the vertical axis. In the first case, it does not change since $(\mathbf{b}_i^*, \mu_{i+1,i}\mathbf{b}_i^* + \mathbf{b}_{i+1}^*)$ is reduced. Otherwise $\mathbf{b}_i$ and $\mathbf{b}_{i+1}$ are to be swapped since $\mu_{i+1,i}\mathbf{b}_i^* + \mathbf{b}_{i+1}^*$ is not in the fundamental domain.

C.4.3 Schnorr-Euchner's Deep Insertion

The study of local bases helps to understand the behavior of the Schnorr-Euchner deep insertion algorithm [307]. In deep-L^3, instead of having the Lovász conditions satisfied for the pairs $(i, i+1)$, one requires that they are satisfied for all pairs (i, j) with $i < j$, i.e. :

$$\|\mathbf{b}_j^* + \mu_{j,j-1}\mathbf{b}_{j-1}^* + \ldots + \mu_{j,i}\mathbf{b}_i^*\|^2 \geq \delta \|\mathbf{b}_i^*\|^2 \text{ for } j > i.$$

This is stronger than the L^3-reduction, but no polynomial-time algorithm to compute it is known. Yet in practice, if we deep-L^3-reduce an already L^3-reduced basis and if the dimension is not too high, it terminates reasonably fast. On the right side of Figure C.4, we did the same experiment as on the left side, except that instead of only L^3-reducing the bases, we L^3-reduced them and then

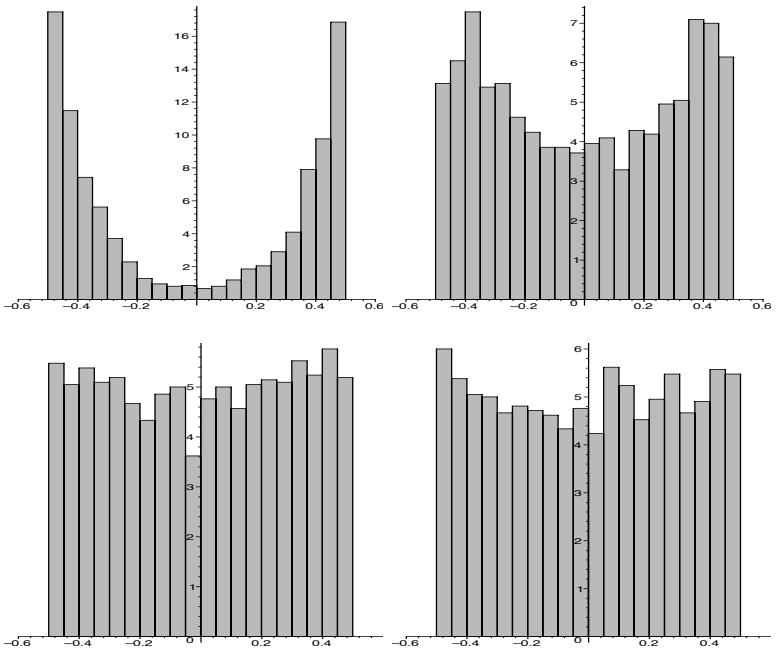


FIG. C.5 – Distribution of $\mu_{i,i-1}$ (top left), $\mu_{i,i-2}$ (top right), $\mu_{i,i-5}$ (bottom left) and $\mu_{i,i-10}$ (bottom right) for 71-dimensional lattices after L^3.

deep-L^3-reduced the obtained bases. The average value of $\|\mathbf{b}_i^*\|/\|\mathbf{b}_{i+1}^*\|$'s is closer to 1 than in the case of L^3 : the 1.04 and 1.02 constants become respectively ≈ 1.025 and 1.012. These data match the observations of [18].

We explain this phenomenon as follows. Assume that, relatively to $\mathbf{b}_i^*$, the vector $\mu_{i+1,i}\mathbf{b}_i^* + \mathbf{b}_{i+1}^*$ lies in a corner in the left side of Figure C.4. Then the Lovász condition between $\mathbf{b}_i$ and $\mathbf{b}_{i+2}$ is less likely to be fulfilled, and the vector $\mathbf{b}_{i+2}$ is more likely to be changed. Indeed, the component of $\mathbf{b}_{i+2}$ onto $\mathbf{b}_{i+1}^*$ will be smaller than usual (because $\|\mathbf{b}_{i+1}^*\|/\|\mathbf{b}_i^*\|$ is small), and thus $\mu_{i+2,i+1}\mathbf{b}_{i+1}^*$ will be smaller. As a consequence, the vector $\mu_{i+2,i}\mathbf{b}_i^* + \mu_{i+2,i+1}\mathbf{b}_{i+1}^* + \mathbf{b}_{i+2}^*$ is more likely to be shorter than $\mathbf{b}_i^*$, and thus $\mathbf{b}_{i+2}$ is more likely to change. Since the corner local bases arise with high frequency, deep-L^3 often performs insertions of depths higher than 2 that would not be performed by L^3.

C.5 The Practical Running Time of L^3

In this section, we argue that the worst case complexity bound $O(d^4(d+n)(d+\log B)\log B)$ is asymptotically reached for some classes of random bases, and explain how and why the running time is better in practice. Here we consider bases for which $n = \Theta(d) = O(\log B)$, so that the bound above becomes $O(d^5 \log^2 B)$. Notice that the heuristic codes do not have any asymptotic meaning since they do not terminate when the dimension increases too much (in particular, the working precision must increase with the dimension). Therefore, all the experiments described in this section were performed using the proved variant of `fplll-1.2`.

We draw below a heuristic worst-case complexity analysis of L^2 that will help us to explain the difference between the worst case and the practical behavior :

- There are $O(d^2 \log B)$ loop iterations.
- In a given loop iteration, there are usually two iterations within the lazy size-reduction : the first one makes the $|\mu_{\kappa,i}|$'s smaller than η and the second one recomputes the $\mu_{\kappa,i}$'s and $r_{\kappa,i}$'s with better

accuracy. This is incorrect in full generality (in particular for knapsack-type bases as we will see below), but is the case most often.

- In each iteration of the size-reduction, there are $O(d^2)$ arithmetic operations.
- Among these, the most expensive ones are those related to the coefficients of the basis and Gram matrices : these are essentially multiplications between integers of lengths $O(\log B)$ and the x_i's, of lengths $O(d)$.

We argue that the analysis above is tight for Ajtai-type random bases.

Heuristic 2 *Let $\alpha > 1$. When d grows to infinity, the average cost of the L^2 algorithm given as input a randomly and uniformly chosen d-dimensional Ajtai-type basis with parameter α is $\Theta(d^{5+2\alpha})$.*

In this section, we also claim that the bounds of the heuristic worst-case analysis are tight in practice for Ajtai-type random bases, except the $O(d)$ bound on size of the x_i's. Finally, we detail the case of knapsack-type random bases.

C.5.1 L^2 on Ajtai-Type Random Bases

Firstly, the $O(d^2 \log B)$ bound on the loop iterations seems to be tight in practice, as suggested by Figure C.6. The left side corresponds to Ajtai-type random bases with $\alpha = 1.2$: the points are the experimental data and the continuous line is the `gnuplot` interpolation of the type $f(d) = a \cdot d^{3.2}$ (we have $\log B = O(d^{1.2})$). The right side has been obtained similarly, for $\alpha = 1.5$, and $g(d) = b \cdot d^{3.5}$. With Ajtai-type bases, size-reductions contain extremely rarely more than two iterations. For example, for $d \leq 75$ and $\alpha = 1.5$, fewer than 0.01% of the size-reductions involve more than two iterations. The third bound of the heuristic worst case analysis is also reached.

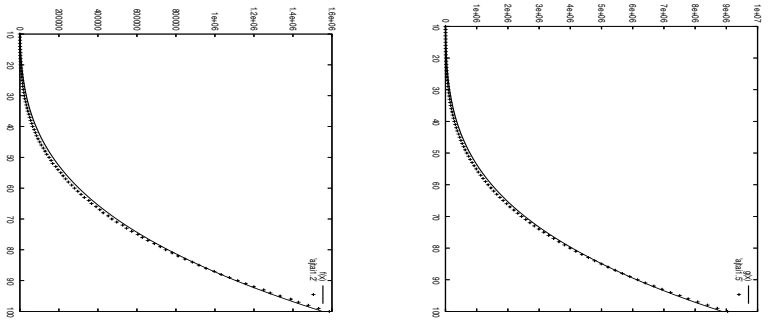


FIG. C.6 – Number of loop iterations of L^2 as a function of d, for Ajtai-type random bases.

These similarities between the worst and average cases do not go on for the size of the integers involved in the arithmetic operations. The x_i's computed during the size-reductions are most often shorter than a machine word, which makes it difficult to observe the $O(d)$ factor in the complexity bound coming from them. For an Ajtai-type basis with $d \leq 75$ and $\alpha = 1.5$, fewer than 0.2% of the non-zero x_i's are longer than 64 bits. In the worst case [264], we have $|x_i| \lesssim (3/2)^{\kappa-i} M$, where M is the maximum of the $\mu_{\kappa,j}$'s before the lazy size-reduction starts, and κ is the current L^3 index. In practice, M happens to be small most of the time. We argue that the average situation is $|x_i| \approx (1.04)^{\kappa-i} M$. This bound remains exponential, but for a small M, x_i becomes larger than a machine word only in dimensions higher than several hundreds. We define :

$$\mu_{\kappa,i}^{(\text{final})} = \mu_{\kappa,i}^{(\text{initial})} - \sum_{j=i+1}^{\kappa-1} x_j \mu_{j,i} = \mu_{\kappa,i}^{(\text{initial})} - \sum_{j=i+1}^{\kappa-1} \left\lfloor \mu_{\kappa,j}^{(\text{final})} \right\rceil \mu_{j,i}.$$

We model the $\left(\mu_{\kappa,\kappa-i}^{(\text{final})}\right)_i$'s by the random variables U_i defined as follows :

$$U_0 = R_0 \text{ and } U_i = R_i + \sum_{j=1}^{i-1} U_j R'_{i,j} \text{ if } i \geq 1,$$

where the R_i's and $R'_{i,j}$'s are uniformly distributed respectively in $[-a, a]$ for some constant a and in $[-\eta, \eta]$. We assume that the R_i's and $R'_{i,j}$'s are pairwise independent. These hypotheses on the $\mu_{i,j}$'s are strong. In particular we saw in Section C.4 that the $\mu_{i,i-1}$'s are not uniformly distributed in $[-\eta, \eta]$. Nevertheless, this simplification does not significantly change the asymptotic behavior of the sequence (U_i) and simplifies the technicalities. Besides, to make the model closer to the reality, we could have rounded the U_j's, but since these quantities are growing to infinity, this should not change much the asymptotic behavior. The independence of the R_i's and $R'_{i,j}$'s and their symmetry give :

$$\mathbb{E}\left[U_i\right] = 0,\ \mathbb{E}\left[U_i^2\right] = \mathbb{E}\left[R_i^2\right] + \sum_{j=1}^{i-1} \mathbb{E}\left[U_j^2\right] \cdot \mathbb{E}\left[R_{i,j}'^2\right] = \frac{2a^3}{3} + \frac{2\eta^3}{3} \sum_{j=1}^{i-1} \mathbb{E}\left[U_j^2\right].$$

As a consequence, for i growing to infinity, we have $\mathbb{E}[U_i^2] \approx \left(\frac{2\eta^3}{3} + 1\right)^i$. If we choose $\eta \approx 1/2$, we get $\frac{2\eta^3}{3} + 1 \approx \frac{13}{12} \approx 1.08$. We thus expect the $|x_i|$'s to be of length $\lesssim (\log_2 1.04) \cdot d \approx 0.057 \cdot d$. To sum up, the x_i's should have length $O(d)$ in practice, but the $O(\cdot)$-constant is tiny. For example, the quantity $(1.04)^d$ becomes larger than 2^{64} for $d \geq 1100$. Since we cannot reduce lattice bases which simultaneously have this dimension and reach the other bounds of the heuristic worst-case complexity analysis, it is at the moment impossible to observe the asymptotic behavior. The practical running time is rather to $O(d^4 \log^2 B)$.

C.5.2 L^2 on Knapsack-Type Bases

In the case of knapsack-type bases there are fewer loop iterations than in the worst case : the quantity $\Delta = \prod_{i=1}^{d} \|\mathbf{b}_i^*\|^{d-i+1}$ of L^3's analysis satisfies $\Delta = B^{O(d)}$ instead of $\Delta = B^{O(d^2)}$. This ensures that there are $O(d \log B)$ loop iterations, so that the overall cost of L^2 for these lattice bases is $O(d^4 \log^2 B)$. Here we argue that asymptotically one should expect a better complexity bound.

Heuristic 3 *When d and $\log B$ grow to infinity with $\log B = \Omega(d^2)$, the average cost of the L^2 algorithm given as input a randomly and uniformly chosen d-dimensional knapsack-type basis with entries of length $\leq \log B$ is $\Theta(d^3 \log^2 B)$.*

In practice for moderate dimensions, the phenomenon described in the previous subsection makes the cost even lower : close to $O(d^2 \log^2 B)$ when $\log B$ is significantly larger than d.

First, there are $\Theta(d \log B)$ main loop iterations. These iterations are equally distributed among the different values of $\kappa_{\max}$: we define κ_{max} as the maximum of the indices κ since the beginning of the execution of the algorithm, i.e., the number of basis vectors that have been considered so far. We have $\kappa_{\max} = 2$ at the beginning, then $\kappa_{\max}$ is gradually incremented up to $d+1$, when the execution of L^2 is over. The number of iterations for each $\kappa_{\max}$ is roughly the same, approximately $\Theta(\log B)$. We divide the execution into $d-1$ phases, according to the value of $\kappa_{\max}$. We observe experimentally that at the end of the phase of a given $\kappa_{\max}$, the current basis has the following shape :

$$\begin{pmatrix}
a_{1,1} & a_{1,2} & \dots & a_{1,\kappa_{\max}+1} & 0 & 0 & \dots & 0 \\
a_{2,1} & a_{2,2} & \dots & a_{2,\kappa_{\max}+1} & 0 & 0 & \dots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
a_{\kappa_{\max},1} & a_{\kappa_{\max},2} & \dots & a_{\kappa_{\max},\kappa_{\max}+1} & 0 & 0 & \dots & 0 \\
A_{\kappa_{\max}+1} & 0 & \dots & 0 & 1 & 0 & \dots & 0 \\
A_{\kappa_{\max}+2} & 0 & \dots & 0 & 0 & 1 & \dots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
A_d & 0 & \dots & 0 & 0 & 0 & \dots & 1
\end{pmatrix},$$

where the top left $a_{i,j}$'s satisfy : $|a_{i,j}| = O\left(B^{\frac{1}{\kappa_{\max}}}\right)$.

We subdivide each $\kappa_{\max}$-phase in two subphases : the first subphase is the first loop iteration of L^2 for which $\kappa = \kappa_{\max}$, and the second one is made of the other iterations with the same $\kappa_{\max}$. The first subphase shortens the vector $\mathbf{b}_{\kappa_{\max}}$: its length decreases from $\approx B$ to $\leq \sqrt{\kappa_{\max}(\max_{i<\kappa_{\max}} \|\mathbf{b}_i\|^2) + 1} \lesssim B^{\frac{1}{\kappa_{\max}-1}}$. This subphase costs $O(d \log^2 B)$ bit operations (see [264]) : there are $O(\log B/d)$ loop iterations in the lazy size-reduction ; each one involves $O(d^2)$ arithmetic operations ; among them, the most costly are the integer multiplications between the x_i's (that are $O(d)$-bit long) and the coefficients of the basis and Gram matrices (their lengths are $O(\log B/d)$, except the $\langle \mathbf{b}_\kappa, \mathbf{b}_i \rangle$'s which occur with frequency $1/O(\kappa)$). The global cost of the first subphases is $O(d^2 \log^2 B)$. This is negligible in comparison to the overall cost of the second subphases.

Let $\mathbf{b}'_i$ be the vector obtained from $\mathbf{b}_i$ after the first subphase of the phase for which $\kappa_{\max} = i$, that is, right after its first size-reduction. Let $C(d, B)$ be the overall cost of the second subphases in dimension d and for input A_i's satisfying $|A_i| \leq B$. We divide the execution of the algorithm as follows : it starts by reducing a knapsack-type basis of dimension $\lfloor d/2 \rfloor$; let $\left(\mathbf{b}''_1, \ldots, \mathbf{b}''_{\lfloor d/2 \rfloor}\right)$ be the corresponding L^3-reduced vectors ; if we exclude the $\lceil d/2 \rceil$ remaining first subphases, then L^2 reduces the basis $\left(\mathbf{b}''_1, \ldots, \mathbf{b}''_{\lfloor d/2 \rfloor}, \mathbf{b}'_{\lfloor d/2+1 \rfloor}, \ldots, \mathbf{b}'_d\right)$, where all the lengths of the vectors are bounded by $\approx B^{2/d}$. As a consequence, we have :

$$C(d, B) = C(d/2, B) + O(d^5 (\log B/d)^2) = C(d/2, B) + O(d^3 \log^2 B),$$

from which one easily gets $C(d, B) = O(d^3 \log^2 B)$, as long as $d^2 = O(\log B)$.

C.5.3 Many Parameters Can Influence the Running Time

We list below a few tunings that should be performed if one wants to optimize L^3 and L^2 for particular instances :

- Firstly, use as less multiprecision arithmetic as possible. If you are in a medium dimension (e.g., less than 170), you may avoid multiprecision *fpa* (see Section C.6). If your input basis is made of short vectors, like for NTRU lattices, try using chip integers instead of multiprecision integers.
- Detect if there are scalar products cancellations : if these cancellations happen very rarely, use a heuristic variant that does not require the Gram matrix. Otherwise, if such cancellations happen frequently, a proved variant using the Gram matrix may turn out to be cheaper than a heuristic one recomputing exactly many scalar products.
- It is sometimes recommended to weaken δ and η. Indeed, if you increase η and/or decrease δ, it will possibly decrease the number of iterations within the lazy size-reduction and the number of global iterations. However, relaxed L^3-factors require a higher precision : for a given precision, the dimension above which L^2 might loop forever decreases (see Section C.6).

C.6 "Numerical Stability" of L^3

In this section, we discuss problems that may arise when one uses *fpa* within L^3. The motivation is to get a good understanding of the "standard" numerical behavior, in order to keep the double precision as long as possible with low chances of failure. Essentially, two different phenomena may be encountered : a lazy size-reduction or consecutive Lovász tests may be looping forever. The output may also be incorrect, but most often if something goes wrong, the execution loops within a size-reduction. We suppose here that either the Gram matrix is maintained exactly during the execution or that the problems arising from scalar product cancellations do not show up.

It is shown in [264] that for some given parameters δ and η, a precision of $\left(\log \frac{(1+\eta)^2}{\delta - \eta^2} + \varepsilon\right) \cdot d + o(d)$ is sufficient for L^2 to work correctly, for any constant $\varepsilon > 0$. For δ close to 1 and η close to 1/2, it gives that a precision of $1.6 \cdot d + o(d)$ suffices. A family of lattice bases for which this bound seems to

be tight is also given. Nevertheless, in practice the algorithm seems to work correctly with a much lower precision : for example, the double precision (53 bit-long mantissæ) seems sufficient most of the time up to dimension 180. We argue here that the average required precision grows linearly with the dimension, but with a significantly lower constant.

Heuristic 4 *Let δ be close to 1 and η be close to 1/2. For almost every lattice, with a precision of $0.18 \cdot d + o(d)$ bits for the* fp*-calculations, the L^2 algorithm performs correctly when given almost any input basis.*

This heuristic has direct consequences for a practical implementation of L^2 : it helps guessing what precision should be sufficient in a given dimension, and thus a significant constant factor can be saved for the running time.

We now give a justification for the heuristic above. For a fixed size of mantissa, we evaluate the dimension for which things should start going wrong. First, we evaluate the error made on the Gram-Schmidt coefficients and then we will use these results for the behavior of L^3 : to do this, we will say that L^3 performs plenty of Gram-Schmidt calculations (during the successive loop iterations), and that things go wrong if at least one of these calculations is erroneous.

We consider the following random model, which is a simplified version of the one described in Section C.4 (the simplification should not change the asymptotic results but helps for the analysis).

- The $\mu_{i,j}$'s for $i > j$ are chosen randomly and independently in $[-\eta, \eta]$. They share a distribution that is symmetric towards 0. This implies that $\mathbb{E}[\mu] = 0$. We define $\mu_2 = \mathbb{E}[\mu^2]$ and $\mu_{i,i} = 1$.
- The $\frac{r_{i,i}}{r_{i+1,i+1}}$'s are chosen randomly and independently in $(0, \beta]$. These choices are independent of those of the $\mu_{i,j}$'s. We define $\alpha = \mathbb{E}\left[\frac{r_{i,i}}{r_{i+1,i+1}}\right]$.
- The random variables $\mu_{i,j}$ and $\frac{r_{i,i}}{r_{i+1,i+1}}$ determine the Gram matrix of the initial basis. Let $r_{1,1}$ be an arbitrary constant. We define the following random variables, for $i \geq j$:

$$\langle \mathbf{b}_i, \mathbf{b}_j \rangle = r_{1,1} \sum_{k=1}^{j} \mu_{j,k}\mu_{i,k} \prod_{l=1}^{k-1} (r_{l,l}/r_{l+1,l+1})^{-1}.$$

- We define the random variables $r_{i,j} = r_{1,1}\mu_{i,j} \prod_{l=1}^{j-1} (r_{l,l}/r_{l+1,l+1})^{-1}$ (for $i \geq j$).
- We assume that we do a relative error $\varepsilon = 2^{-\ell}$ (with ℓ the working precision) while translating the exact value $\|\mathbf{b}_1\|^2$ into a *fp* number : $\Delta\|\mathbf{b}_1\|^2 = \varepsilon\|\mathbf{b}_1\|^2$.

We have selected a way to randomly choose the Gram matrix and to perform a rounding error on $\|\mathbf{b}_1\|^2$. To simplify the analysis, we suppose that there is no rounding error performed on the other $\langle \mathbf{b}_i, \mathbf{b}_j \rangle$'s. Our goal is to estimate the amplification of the rounding error $\Delta\|\mathbf{b}_1\|^2$ during the calculations of approximations of the $r_{i,j}$'s and $\mu_{i,j}$'s. We neglect high-order error terms. More precisely, we study the following random variables, defined recursively :

$$\begin{aligned} \Delta r_{1,1} &= \Delta\|\mathbf{b}_1\|^2 = \varepsilon\|\mathbf{b}_1\|^2, \\ \Delta r_{i,j} &= -\sum_{k=1}^{j-1} (\Delta r_{i,k}\mu_{j,k} + \Delta r_{j,k}\mu_{i,k} - \Delta r_{k,k}\mu_{i,k}\mu_{j,k}) \text{ when } i \geq j, \end{aligned}$$

The $\mu_{a,b}$'s and $\frac{r_{b,b}}{r_{b+1,b+1}}$'s that may not be independent with $\Delta r_{i,k}$ are those for which $b < k$. As a consequence, $\Delta r_{i,k}$, $\mu_{j,k}$, $\frac{r_{j-1,j-1}}{r_{j,j}}$, $\frac{r_{j-2,j-2}}{r_{j-1,j-1}}$, ..., $\frac{r_{k,k}}{r_{k+1,k+1}}$ are pairwise independent, $\Delta r_{j,k}, \mu_{i,k}$, $\frac{r_{j-1,j-1}}{r_{j,j}}$, $\frac{r_{j-2,j-2}}{r_{j-1,j-1}}$, ..., $\frac{r_{k,k}}{r_{k+1,k+1}}$ are pairwise independent, and $\Delta r_{k,k}$, $\mu_{i,k}$, $\mu_{j,k}$, $\frac{r_{j-1,j-1}}{r_{j,j}}$, $\frac{r_{j-2,j-2}}{r_{j-1,j-1}}$, ..., $\frac{r_{k,k}}{r_{k+1,k+1}}$ are pairwise independent, for all (i, j, k) satisfying $i > j > k$. Therefore, for any $i > j$:

$\mathbb{E}\left[\frac{\Delta r_{i,j}}{r_{j,j}}\right] = -\sum_{k=1}^{j-1}\left(\prod_{l=k}^{j-1}\mathbb{E}\left[\frac{r_{l,l}}{r_{l+1,l+1}}\right]\right)\left(\mathbb{E}\left[\frac{\Delta r_{i,k}}{r_{k,k}}\right]\mathbb{E}[\mu_{j,k}] + \mathbb{E}\left[\frac{\Delta r_{j,k}}{r_{k,k}}\right]\mathbb{E}[\mu_{i,k}]\right.$

Because $\mathbb{E}[\mu_{j,k}] = \mathbb{E}[\mu_{i,k}] = 0$, we get $\mathbb{E}\left[\frac{\Delta r_{i,j}}{r_{j,j}}\right] = 0$, for all $i > j$. Similarly, we have, for $i > 1$:

$\mathbb{E}\left[\frac{\Delta r_{i,i}}{r_{i,i}}\right] = \mu_2\sum_{k=1}^{j-1}\left(\prod_{l=k}^{i-1}\mathbb{E}\left[\frac{r_{l,l}}{r_{l+1,l+1}}\right]\right)\mathbb{E}\left[\frac{\Delta r_{k,k}}{r_{k,k}}\right] = \mu_2\sum_{k=1}^{j-1}\alpha^{i-k}\mathbb{E}\left[\frac{\Delta r_{k,k}}{r_{k,k}}\right].$

We obtain that $\mathbb{E}\left[\frac{\Delta r_{i,i}}{r_{i,i}}\right]$ is close to $(\alpha(1+\mu_2))^i\varepsilon$. For example, if the $\mu_{i,j}$'s are uniformly chosen in $[-1/2, 1/2]$, if $\alpha = 1.04$ (as observed in Section C.4), and if $\varepsilon \approx 2^{-53}$, we get $\mathbb{E}\left[\frac{\Delta r_{i,i}}{r_{i,i}}\right] \approx 1.13^i \cdot 2^{-53}$. For $i = 200$, this is close to 2^{-17}.

We have analyzed very roughly the influence of the rounding error made on $\|\mathbf{b}_1\|^2$, within the Gram-Schmidt orthogonalization for L^3-reduced bases. If we want to adapt this analysis to L^2, we must take into account the number of $r_{i,j}$'s and $\mu_{i,j}$'s that are computed during the execution. To simplify we consider only the $r_{d,d}$'s, which are *a priori* the less accurate. We suppose that all the computations of $r_{d,d}$ through the execution are independent. Let K be the number of iterations for which $\kappa = d$. We consider that an error on $r_{d,d}$ is significant if $\frac{\Delta r_{d,d}}{r_{d,d}}$ is at least 2^{-3}. If such an error occurs, the corresponding Lovász test is likely to be erroneous. Under such hypotheses, the probability of failure is of the order of $1 - (1 - 2^{-17+3})^K \approx K2^{-14}$. In case of several millions of Lovász tests, it is likely that there is one making L^3 behave unexpectedly.

The above analysis is completely heuristic and relies on very strong hypotheses, but it provides orders of magnitude that one seems to encounter in practice. For random bases, we observe infinite loops in double precision arising around dimensions 175 to 185, when there are a few millions Lovász tests.

Acknowledgments. We thank Guillaume Hanrot for helpful discussions. The writing of this paper was completed while the second author was visiting the University of Sydney, whose hospitality is gratefully acknowledged.

Annexe D

Symplectic Lattice Reduction and NTRU

EUROCRYPT 2006
[104] avec Nicolas Gama (ENS) et Nick Howgrave-Graham (NTRU Cryptosystems)

Abstract: *NTRU is a very efficient public-key cryptosystem based on polynomial arithmetic. Its security is related to the hardness of lattice problems in a very special class of lattices. This article is motivated by an interesting peculiar property of NTRU lattices. Namely, we show that NTRU lattices are proportional to the so-called symplectic lattices. This suggests to try to adapt the classical reduction theory to symplectic lattices, from both a mathematical and an algorithmic point of view. As a first step, we show that orthogonalization techniques (Cholesky, Gram-Schmidt, QR factorization, etc.) which are at the heart of all reduction algorithms known, are all compatible with symplecticity, and that they can be significantly sped up for symplectic matrices. Surprisingly, by doing so, we also discover a new integer Gram-Schmidt algorithm, which is faster than the usual algorithm for all matrices. Finally, we study symplectic variants of the celebrated LLL reduction algorithm, and obtain interesting speed ups.*

D.1 Introduction

The NTRU cryptosystem [149] is one of the fastest public-key cryptosystems known, offering both encryption (under the name NTRUEncrypt) and digital signatures (under the name NTRUSign [148]). Besides efficiency, another interesting feature of NTRU compared to traditional public-key cryptosystems based on factoring or discrete logarithm is its potential resistance to quantum computers : no efficient quantum algorithm is known for NP-hard lattice problems. The security and insecurity of NTRU primitives has been a popular research topic in the past 10 years, and NTRU is now being considered by the *IEEE P1363.1* standards [164].

The security of NTRU is based on the hardness of two famous lattice problems, namely the shortest and closest vector problems (see for instance the survey [267]), in a very particular class of lattices called convolution modular lattices by [225]. More precisely, it was noticed by the authors of NTRU and by Coppersmith and Shamir [72] that ideal lattice reduction algorithms could heuristically recover NTRU's secret key from the public key. This does not necessarily imply that NTRU is insecure, since there is a theoretical and experimental gap between existing reduction algorithms (such as LLL [205] or its block improvements by Schnorr [302]) and ideal lattice reduction (which is assumed to be solving NP-hard lattice problems), while NTRU is so far the only lattice-based cryptosystem known that can cope with high dimensions without sacrificing performances. Nor does

it mean that the security of NTRU primitives is strictly equivalent to the hardness of lattice problems. In fact, the main attacks on NTRU primitives have bypassed the hard lattice problems : this was notably the case for the decryption failure attacks [158] on NTRUEncrypt, the attacks [111, 112] on the ancestor NSS [150] of NTRUSign [148], as well as the recent attack [259] on NTRUSign [148] without perturbation. Almost ten years after the introduction of NTRU [149], no significant weakness on NTRU lattices has been found, despite the very particular shape of NTRU lattice bases : both the public and secret NTRU bases are $2N \times 2N$ matrices formed by four blocks of $N \times N$ circulant matrices. It is this compact representation that makes NTRU much more efficient than other lattice-based or knapsack-based schemes (see the survey [267]). A fundamental open question is whether this particular shape makes NTRU lattices easier to reduce or not.

Our Results. We propose to exploit the structure of NTRU lattices in lattice reduction algorithms. As a starting point, we observe a peculiar property of NTRU lattices : we show that NTRU lattices are proportional to the so-called *symplectic* lattices (see the survey [28]). As their name suggests, symplectic lattices are related to the classical symplectic group [354] : a lattice is said to be symplectic if it has at least one basis whose Gram matrix is symplectic, which can only occur in even dimension. Such lattices are *isodual* : there exists an isometry of the lattice onto its dual. Interestingly, most of the well-known lattices in low even dimension are proportional to symplectic lattices, *e. g.* the roots lattices $\mathbb{A}_2$, $\mathbb{D}_4$ and $\mathbb{E}_8$, the Barnes lattice P_6, the Coxeter-Todd lattice K_{12}, the Barnes-Wall lattice BW_{16} and the Leech lattice Λ_{24} (see the bible of lattices [68]). Besides, there is a one-to-one correspondence between symplectic lattices and principally polarized complex Abelian varieties, among which Jacobians form an interesting case (see [54]). This has motivated the study of symplectic lattices in geometry of numbers.

However, to our knowledge, symplectic lattices have never been studied in reduction theory. The long-term goal of this paper is to explore the novel concept of symplectic lattice reduction in which the classical reduction theory is adapted to symplectic lattices, from both a mathematical and an algorithmic point of view in order to speed up reduction algorithms. As a first step, we show that the Gram-Schmidt orthogonalization process – which is at the heart of all lattice reduction algorithms known – preserves symplecticity, and that is made possible by a slight yet essential change on the classical definition of a symplectic matrix, which is fortunately compatible with the standard theory of the symplectic group. We then exploit this property to speed up its computation for symplectic lattices. In doing so, we actually develop a new and faster method to compute integral Gram-Schmidt, which is applicable to all matrices, and not just symplectic matrices. The method is based on duality : it is faster than the classical method, because it significantly reduces the number of long-integer divisions. When applied to symplectic matrices, a further speed up is possible thanks to the links between symplecticity and duality : in practice, the method then becomes roughly 30 times faster than the classical GS method, which is roughly the time it would take on a matrix of halved dimension. Finally, we study symplectic versions of the celebrated LLL lattice basis reduction algorithm [205] and obtain a speedup of 6 for NTRU lattices of standard size. We restrict to the so-called integral version of LLL to facilitate comparisons : it might be difficult to compare two floating-point variants with different stability properties. We leave the cases of floating-point variants [264] and improved reduction algorithms [302] to future work, but the present work seems to suggest that reduction algorithms might be optimized to NTRU lattices in such a way that a $2n$-dimensional NTRU lattice would not take more time to reduce than an αn-dimensional lattice for some $\alpha < 2$. This is the case for Gram-Schmidt orthogonalization and LLL.

Related Work. Incidentally, the compatibility of the symplectic group with respect to standard matrix factorizations has recently been studied in [215] : however, because they rely on the classical definition of a symplectic matrix, they fail to obtain compatibility with Gram-Schmidt orthogonalization or the QR decomposition.

Road map. The paper is organized as follows. In Section D.2, we provide necessary background

on lattice reduction and the symplectic group. In Section D.3, we explain the relationship between NTRU lattices and symplecticity. In Section D.4, we show that the Gram-Schmidt orthogonalization process central to all lattice reduction algorithms known is fully compatible with symplecticity. In Section D.5, we present a new integral Gram-Schmidt algorithm, which leads to significant speed-ups for symplectic matrices. The final section D.6 deals with symplectic variants of integral LLL.

Acknowledgements. Part of this work, as well as a visit of the second author to the ENS, were supported by the Commission of the European Communities through the IST program under contract IST-2002-507932 ECRYPT. We would like to thank Joe Silverman, Mike Szydlo and William Whyte for useful conversations.

D.2 Background

Let $\|.\|$ and $\langle .,. \rangle$ be the Euclidean norm and inner product of $\mathbb{R}^n$. Vectors will be written in bold, and we will use row-representation for matrices. The notations $\mathcal{M}_n(\mathbb{R})$ represents the $n \times n$-dimensional matrices over $\mathbb{R}$, and $GL_n(\mathbb{R})$ the n-dimensional invertible matrices of $\mathcal{M}_n(\mathbb{R})$. For a matrix M whose name is a capital letter, we will usually denote its coefficients by $m_{i,j}$: if the name is a Greek letter like μ, we will keep the same symbol for both the matrix and its coefficients. The matrix norm $|M|$ represents the maximum of the euclidean norms of the rows of M. The notation $\lceil x \rfloor$ denotes a closest integer to x.

D.2.1 Lattices

We refer to the survey [267] for a bibliography on lattices. In this paper, by the term lattice, we mean a discrete subgroup of $\mathbb{R}^n$. The simplest lattice is $\mathbb{Z}^n$. It turns out that in any lattice L, not just $\mathbb{Z}^n$, there must exist linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_d \in L$ such that :

$$L = \left\{ \sum_{i=1}^{d} n_i \mathbf{b}_i \mid n_i \in \mathbb{Z} \right\}.$$

Any such d-tuple of vectors $\mathbf{b}_1, \ldots, \mathbf{b}_d$ is called a *basis* of L : a lattice can be represented by a basis, that is, a row matrix. Two lattice bases are related to one another by some matrix in $GL_d(\mathbb{Z})$. The *dimension* of a lattice L is the dimension d of the linear span of L. Let $[\mathbf{b}_1, \ldots, \mathbf{b}_d]$ be vectors : the lattice is full-rank if $d = n$, which is the usual case. We denote by $G(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ their *Gram matrix*, that is the $d \times d$ symmetric matrix $(\langle \mathbf{b}_i, \mathbf{b}_j \rangle)_{1 \leq i,j \leq d}$ formed by all the inner products. The *volume* $\mathrm{vol}(L)$ (or *determinant*) of the lattice L is the square root of the determinant of the Gram matrix of any basis of L : here, the Gram matrix is symmetric definite positive. The dual lattice $L^\times$ of a lattice L is :

$$L^\times = \{ \mathbf{v} \in \mathrm{span} L,\ \forall \mathbf{u} \in L,\ \langle \mathbf{u}, \mathbf{v} \rangle \in \mathbb{Z} \}.$$

They have the same dimension and their volumes satisfy $\mathrm{vol}(L) \cdot \mathrm{vol}(L^\times) = 1$. If $B = [\mathbf{b}_1, ..., \mathbf{b}_d]$ is a basis of L, and $\delta_{i,j}$ the Kronecker symbol, then the dual family $B^\times = [\mathbf{b}_1^\times, ..., \mathbf{b}_d^\times]$ with $\mathbf{b}_i^\times \in \mathrm{span}(L)$ satisfying $\langle \mathbf{b}_i^\times, \mathbf{b}_j \rangle = \delta_{i,j}$ is a basis of $L^\times$ called the dual basis of B. The Gram matrices of B and $B^\times$ are inversed, and when L is a full rank lattice, $B^\times = B^{-t}$.

D.2.2 The Symplectic Group

The symplectic group is one of the classical groups [354], whose name is due to Weyl. Given four matrices $A, B, C, D \in \mathcal{M}_n(\mathbb{R})$ we denote by $Q[A, B, C, D]$ the $(2n) \times (2n)$ matrix with A, B, C, D as its quadrants :

$$Q[A, B, C, D] = \begin{pmatrix} A & B \\ C & D \end{pmatrix}.$$

Symplectic matrices are matrices preserving a nondegenerate antisymmetric bilinear form. Let σ be an isometry of $\mathbb{R}^{2n}$. Then $\sigma^2 = -1$ if and only if there exists an orthonormal basis of $\mathbb{R}^{2n}$ over which the matrix of σ is $J_{2n} = Q[0, I_n, -I_n, 0]$, where I_n is the $n \times n$ identity matrix. Usually, a matrix $M \in \mathcal{M}_{2n}(\mathbb{R})$ is said to be *symplectic* if and only if

$$M^t J_{2n} M = J_{2n}. \tag{D.1}$$

where M^t is the transpose of M. This is equivalent to M being invertible and having inverse equal to :

$$M^{-1} = -J_{2n} M^t J_{2n}. \tag{D.2}$$

The set of such matrices is denoted by $Sp(2n, \mathbb{R})$, which is a subgroup of the special linear group $SL_{2n}(\mathbb{R})$: a symplectic matrix has determinant +1. A matrix is symplectic if and only if its transpose is symplectic. The matrix $Q[A, B, C, D]$ is symplectic if and only if $AD^t - BC^t = I_n$ and both the matrices AB^t and CD^t are symmetric. It follows that a triangular matrix $Q[A, 0, C, D]$ may only be symplectic if A and D are diagonal, which is too restrictive to make the symplectic group fully compatible with standard matrix factorizations involving triangular matrices.

To fix this, we consider a variant of the usual symplectic group obtained by equation (D.1) with another matrix J_{2n}. Fortunately, this is allowed by the theory, as while as J_{2n} is a nonsingular, skew-symmetric matrix. From now on, we thus let $J_{2n} = Q[0, R_n, -R_n, 0]$ where R_n is the reversed identity matrix : the identity where the rows (or the columns) are in reverse order, that is, the (i, j)-th coefficient is the Kronecker symbol $\delta_{i,n+1-j}$. This new matrix J_{2n} still satisfies $J_{2n}^2 = -I_{2n}$, and is therefore compatible with symplecticity. From now on, by a symplectic matrix, we will mean a matrix satisfying equation (D.1) or (D.2) with this choice of J_{2n}, and this will be our symplectic group $Sp(2n, \mathbb{R})$. Now, a matrix $Q[A, B, C, D]$ is symplectic if and only if the following conditions hold :

$$BA^s = AB^s,\ DC^s = CD^s, AD^s - BC^s = I_n \tag{D.3}$$

where $M^s = R_n M^t R_n$ for any $M \in \mathcal{M}_n(\mathbb{R})$, which corresponds to reflecting the entries in the off-diagonal $m_{i,j} \leftrightarrow m_{n+1-j,n+1-i}$. The matrix $R_n M$ reverses the rows of M, while MR_n reverses the columns. In other words, compared to the usual definition of symplectic matrices, we have replaced the transpose operation M^t and I_n by respectively the reflection M^s and R_n. This will be the general rule to switch from the usual symplectic group to our variant. In fact, it can be checked that the reflection $M \mapsto M^s$ is an involution of $Sp(2n, \mathbb{R})$: M^s is symplectic (though R_n is not symplectic), and $(M^s)^s = M$. Now a triangular matrix $Q[A, 0, C, D]$ may be symplectic without requiring A and D to be diagonal. Naturally, M^{-s} will mean the inverse of M^s.

To conclude this subsection, let us give a few examples of symplectic matrices with our own definition of $Sp(2n, \mathbb{R})$, which will be very useful in the rest of the paper :

- Any element of $SL_2(\mathbb{R})$, that is, any 2x2 matrix with determinant 1.
- A diagonal matrix of $\mathcal{M}_{2n}(\mathbb{R})$ with coefficients $d_1, ..., d_{2n}$ is symplectic if and only if $d_i = 1/d_{2n+1-i}$ for all i.
- Any $\begin{bmatrix} A & 0 & B \\ 0 & M & 0 \\ C & 0 & D \end{bmatrix}$ including $\begin{bmatrix} I & 0 & 0 \\ 0 & M & 0 \\ 0 & 0 & I \end{bmatrix}$ where $\begin{cases} M \in Sp(2n, \mathbb{R}) \\ Q[A, B, C, D] \in Sp(2m, \mathbb{R}) \end{cases}$.
- $Q[U, 0, 0, U^{-s}]$ for any invertible matrix $U \in GL_n(\mathbb{R})$.
- $Q[I_n, 0, A, I_n]$ for any $A \in \mathcal{M}_n(\mathbb{R})$ such that $A = A^s$, that is, A is *reversed-symmetric*.

The symplecticity can be checked by equations (D.1), (D.2) or (D.3). In particular, these equations prove the following elementary lemma, which gives the structure of symplectic triangular matrices.

Lemma D.32 *A lower-triangular $2n$-dimensional matrix L can always be decomposed as follows :*

$$L = \begin{bmatrix} \alpha & 0 & 0 \\ \mathbf{u}^t & M & 0 \\ \beta & \mathbf{v} & \gamma \end{bmatrix} \text{ where } \begin{cases} \alpha, \beta, \gamma \in \mathbb{R} \\ \mathbf{u}, \mathbf{v} \in \mathbb{R}^{2n-2} \\ M \in \mathcal{M}_{2n-2}(\mathbb{R}) \text{ is triangular} \end{cases}.$$

Then the matrix L is symplectic if and only if M is symplectic (and triangular), $\gamma = \frac{1}{\alpha}$ and $\mathbf{u} = -\alpha \mathbf{v} J_{2n-2} M^t$.

D.2.3 Symplectic Lattices

A lattice L is said to be *isodual* if there exists an isometry σ of L onto its dual (see the survey [28]). One particular case of isodualities is when $\sigma^2 = -1$, in which case the lattice is called "*symplectic*". There exists an orthogonal basis of span(L) over which the matrix of σ is J_{2n} there is at least a basis of L whose Gram matrix is symplectic.

A symplectic lattice has volume equal to 1. In this paper, we will say that an integer full-rank lattice $L \in \mathbb{Z}^{2n}$ is *q-symplectic* if the lattice $L/\sqrt{q}$ is symplectic where $q \in \mathbb{N}^*$. Its volume is equal to q^n. Our q-symplectic lattices seem to be a particular case of the modular lattices introduced by Quebbemann [291], which are connected to modular forms.

D.2.4 Orthogonalization

CHOLESKY. Let $G \in \mathcal{M}_n(\mathbb{R})$ be symmetric definite positive. There exists a unique lower triangular matrix $L \in \mathcal{M}_n(\mathbb{R})$ with strictly positive diagonal such that $G = LL^t$. The matrix L is the *Cholesky factorization* of G, and its Gram matrix is G.

THE $\mu D \mu^t$ FACTORIZATION. This factorization is the analogue of the so-called "LDL decomposition" in [123, Chapter 4.1]. Let $G \in \mathcal{M}_n(\mathbb{R})$ be symmetric definite. There exists a unique lower triangular matrix $\mu \in \mathcal{M}_n(\mathbb{R})$ with unit diagonal and a unique diagonal matrix $D \in \mathcal{M}_n(\mathbb{R})$ such that $G = \mu D \mu^t$. The couple (μ, D) is the *$\mu D \mu^t$ factorization* of G. When G is positive definite, then D is positive diagonal, and the relation between the $\mu D \mu^t$ and Cholesky factorizations of G is $L = \mu\sqrt{D}$.

QR OR LQ. Let $M \in GL_n(\mathbb{R})$. There exists a unique pair $(Q, R) \in \mathcal{M}_n(\mathbb{R})^2$ such that $M = QR$, where Q is unitary and R is upper triangular with strictly positive diagonal. This is the standard QR factorization. Since we deal with row matrices, we prefer to have a lower triangular matrix, which can easily be achieved by transposition. It follows that there also exists a unique pair $(L, Q) \in \mathcal{M}_n(\mathbb{R})^2$ such that $M = LQ$, where Q is unitary and L is lower triangular with strictly positive diagonal. Note that L is the Cholesky factorization of the Gram matrix MM^t of M.

GRAM-SCHMIDT. Let $[\mathbf{b}_1, \ldots, \mathbf{b}_d]$ be linearly independent vectors represented by the $d \times n$ matrix B. Their *Gram-Schmidt orthogonalization* (GSO) is the orthogonal family $[\mathbf{b}_1^*, \ldots, \mathbf{b}_d^*]$ defined recursively as follows : $\mathbf{b}_1^* = \mathbf{b}_1$ and $\mathbf{b}_i^*$ is the component of $\mathbf{b}_i$ orthogonal to the subspace spanned by $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$. We have $\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^*$ where $\mu_{i,j} = \langle \mathbf{b}_i, \mathbf{b}_j^* \rangle / \|\mathbf{b}_j^*\|^2$ for all $i < j$. We let $\mu \in \mathcal{M}_d(\mathbb{R})$ be the lower triangular matrix whose coefficients are $\mu_{i,j}$ above the diagonal, and 1 on the diagonal. If B^* is the $d \times n$ row matrix representing $[\mathbf{b}_1^*, \ldots, \mathbf{b}_d^*]$, then $B = \mu B^*$. If we let G be the Gram matrix BB^t of B, then μ is exactly the matrix from the $\mu D \mu^t$ decomposition of G, and its Cholesky factorization $L = (\ell_{i,j})$ is related to the GSO by : $\ell_{i,j} = \mu_{i,j} \|\mathbf{b}_j^*\|$ for $i < j$. The matrices L and B have the same Gram matrix, so the GSO can be viewed as a trigonalization of the lattice Λ spanned by B. Note that $\mathrm{vol}(\Lambda) = \prod_{i=1}^{d} \|\mathbf{b}_i^*\|$.

INTEGRAL GRAM-SCHMIDT. In practice, we are interested in the case where the $\mathbf{b}_i$'s are in $\mathbb{Z}^n$. Then the $\mathbf{b}_i^*$'s and the $\mu_{i,j}$'s are in general rational. To avoid rational arithmetic, it is customary to use the following integral quantities (as in [353] and in the complexity analysis of [205]) : for all $1 \leq i \leq d$, let : $\lambda_{i,i} = \prod_{j=1}^{i} \|\mathbf{b}_j^*\|^2 = \mathrm{vol}(\mathbf{b}_1, \ldots, \mathbf{b}_i)^2 \in \mathbb{Z}$. Then let $\lambda_{i,j} = \mu_{i,j} \lambda_{j,j}$ for all $j < i$, so that $\mu_{i,j} = \frac{\lambda_{i,j}}{\lambda_{j,j}}$. It is known that $\lambda_{i,j} \in \mathbb{Z}$. When using the GSO for lattice reduction, one does not need to compute the $\mathbf{b}_i^*$'s themselves : one only needs to compute the $\mu_{i,j}$'s and the $\|\mathbf{b}_i^*\|^2$. Since $\|\mathbf{b}_i^*\|^2 = \lambda_{i,i} / \lambda_{i-1,i-1}$ (if we let $\lambda_{0,0} = 1$), it follows that it suffices to compute the integral matrix $\lambda = (\lambda_{i,j})_{1 \leq i,j \leq d}$ for lattice

reduction purposes. This is done by Algorithm 10, whose running time is $O(nd^4 \log^2 |B|)$ where $|B|$ is an upper bound of the $\|\mathbf{b}_i\|$'s.

Algorithm 10 Standard GS

Input : A set of d linearly independent vectors $[\mathbf{b_1}, ..., \mathbf{b_d}]$ of $\mathbb{Z}^n$
Output : The λ matrix of the GSO of $[\mathbf{b_1}, ..., \mathbf{b_d}]$.

```
for i = 1 to d do
    λ_{i,1} ← ⟨b_i, b_1⟩
    for j = 2 to i do
        S = λ_{i,1} λ_{j,1}
        for k = 2 to j − 1 do
            S ← (λ_{k,k} S + λ_{j,k} λ_{i,k}) / λ_{k−1,k−1}
        end for
        λ_{i,j} ← ⟨b_i, b_j⟩ λ_{j−1,j−1} − S
    end for
end for
```

D.2.5 LLL reduction

Size reduction. A basis $[\mathbf{b}_1, \ldots, \mathbf{b}_d]$ is *size-reduced* with factor $\eta \geq 1/2$ if its GSO family satisfies $|\mu_{i,j}| \leq \eta$ for all $j < i$. An individual vector $\mathbf{b}_i$ is size-reduced if $|\mu_{i,j}| \leq \eta$ for all $j < i$. Size reduction usually refers to $\eta = 1/2$, and is typically achieved by successively size-reducing individual vectors.

LLL reduction. A basis $[\mathbf{b}_1, \ldots, \mathbf{b}_d]$ is LLL-reduced [205] with factor (δ, η) for $1/4 < \delta \leq 1$ and $1/2 \leq \eta < \sqrt{\delta}$ if the basis is size-reduced with factor η and if its GSO satisfies the $(d-1)$ Lovász conditions $(\delta - \mu_{i,i-1}^2) \left\|\mathbf{b}_{i-1}^*\right\|^2 \leq \|\mathbf{b}_i^*\|^2$, which means that the GSO vectors never drop too much. Such bases have several useful properties (see [65, 205]), notably the following one : the first basis vector is relatively short, namely

$$\|\mathbf{b}_1\| \leq \beta^{(d-1)/4} \mathrm{vol}(L)^{1/d} \text{ , where } \beta = 1/(\delta - \eta^2).$$

LLL-reduction usually refers to the factor $(3/4, 1/2)$ because this was the choice considered in the original paper by Lenstra, Lenstra and Lovász [205]. But the closer δ and η are respectively to 1 and 1/2, the more reduced the basis is. The classical LLL algorithm obtains in polynomial time a basis reduced with factor $(\delta, 1/2)$ where δ can be arbitrarily close to 1. Reduction with a factor (1,1/2) is closely related to a reduction notion introduced by Hermite [142].

The LLL algorithm. The basic LLL algorithm [205] computes an LLL-reduced basis in an iterative fashion : there is an index κ such that at any stage of the algorithm, the truncated basis $[\mathbf{b}_1, \ldots, \mathbf{b}_{\kappa-1}]$ is LLL-reduced. At each loop iteration, κ is either incremented or decremented : the loop stops when κ eventually reaches the value $d+1$, in which case the entire basis $[\mathbf{b}_1, \ldots, \mathbf{b}_d]$ is already LLL-reduced.

LLL uses two kinds of operations : swaps of consecutive vectors and Babai's nearest plane algorithm [17], which performs at most d translations of the form $\mathbf{b}_\kappa \leftarrow \mathbf{b}_\kappa - m\mathbf{b}_i$, where m is some integer and $i < \kappa$. Swaps are used to achieve Lovász conditions, while Babai's algorithm is used to size-reduce vectors.

If L is a full-rank lattice of dimension n and $|B|$ is an upper bound on the $\|\mathbf{b}_i\|$'s, then the complexity of the LLL algorithm (using integral Gram-Schmidt) without fast integer arithmetic is $O(n^6 \log^3 |B|)$. The recent L^2 algorithm [264] (based on floating-point Gram-Schmidt) by Nguyen and Stehlé achieves a factor of (δ, ν) arbitrarily close to (1,1/2) in faster polynomial time : the complexity is $O(n^5(n + \log |B|) \log |B|)$ which is essentially $O(n^5 \log^2 |B|)$ for large entries. This is the fastest LLL-type reduction algorithm known.

D.3 NTRU Lattices

The NTRU [149] cryptosystem has many variants. To simplify our exposition, we focus on the usual version, but our results apply to all known variants of NTRU.

Let n be a prime number about several hundreds (*e. g.* 251), and q be a small power of two (*e. g.* 128 or 256). Let $\mathcal{R}$ be the ring $\mathbb{Z}[X]/(X^n - 1)$ whose multiplication is denoted by $*$. The NTRU secret key is a pair of polynomials $(f, g) \in \mathcal{R}^2$ with tiny coefficients compared to q, say 0 and 1. The polynomial f is chosen to be invertible modulo q, so that the polynomial $h = g/f \bmod q$ is well-defined in $\mathcal{R}$. The NTRU public key is the polynomial $h \in \mathcal{R}$ with coefficients modulo q. Its fundamental property is : $f * h \equiv g \bmod q$ in $\mathcal{R}$.

In order to fit multiplicative properties of polynomials of $\mathcal{R}$, we use circulant matrices. The application φ that maps a polynomial in $\mathcal{R}$ to its circulant matrix in $\mathcal{M}_n(\mathbb{Z})$ is defined by :

$$\varphi(\sum_{i=0}^{n-1} h_i X^i) = \begin{bmatrix} h_0 & h_1 & \cdots & h_{n-1} \\ h_{n-1} & h_0 & \cdots & h_{n-2} \\ \vdots & \ddots & \ddots & \vdots \\ h_1 & \cdots & h_{n-1} & h_0 \end{bmatrix}$$

1. This application is a ring morphism.
2. Circulant matrices are reversed-symmetric : $\varphi(a)^s = \varphi(a)$ for any $a \in \mathcal{R}$.

There is a natural lattice Λ in $\mathbb{Z}^{2n}$ corresponding to the set of pairs of polynomials $(u, v) \in \mathcal{R}^2$ such that $v * h \equiv u \bmod q$ (see [72, 267]). This lattice can be defined by the following basis, which is public since it can be derived from the public key :

$$B = Q\left[\varphi(q), \varphi(0), \varphi(h), \varphi(1)\right].$$

This basis is in fact the Hermite normal form of Λ. It follows that the dimension of Λ is $2n$ and its volume is q^n. Notice that $B/\sqrt{q}$ is symplectic by equation (D.1), and therefore the public basis B and the NTRU lattice Λ are q-symplectic.

Because of the fundamental property of the public key h, there is a special lattice vector in Λ corresponding to (g, f), which is heuristically the shortest lattice vector. All the vectors corresponding to the rotations $(g*X^k, f*X^k)$ also belong to Λ. In fact, in NTRUSign [148], the pair (f, g) is selected in such a way that there exists another pair of polynomials $(F, G) \in \mathcal{R}^2$ such that $f * G - g * F = q$ in $\mathcal{R}$. It follows that the following matrix is a secret basis of Λ :

$$C = Q\left[\varphi(g), \varphi(f), \varphi(G), \varphi(F)\right].$$

This is the basis used to sign messages in NTRUSign.

Hence, if a, b, c, d are polynomials in $\mathcal{R}$, the matrix $M = Q[\varphi(a), \varphi(b), \varphi(c), \varphi(d)]$ satisfies :

$$-MJ_{2n}M^tJ_{2n} = Q[\varphi(a * d - b * c), 0, 0, \varphi(a * d - b * c)].$$

In particular, the secret basis satisfies : $-CJ_{2n}C^tJ_{2n} = qI_{2n}$, which proves that C is a q-symplectic matrix like B, only with smaller coefficients. Hence, the unimodular transformation that transforms the public basis B into the secret basis C is symplectic. One may wonder if it is possible to design special (possibly faster) lattice reduction algorithms for NTRU lattices, which would restrict their elementary row transformations to the symplectic subgroup of $GL_{2n}(\mathbb{Z})$. This motivates the study of symplectic lattice reduction.

D.4 Symplectic Orthogonalization

All lattice reduction algorithms known are based on Gram-Schmidt orthogonalization, which we recalled in Section D.2. In this section, we show that Cholesky factorization, LQ decomposition and Gram-Schmidt orthogonalization are compatible with symplecticity. The key result of this section is the following symplectic analogue of the so-called LDL^t factorization of a symmetric matrix :

Theorem D.17 (Symplectic $\mu D\mu^t$) *Let G be a symmetric matrix in $Sp(2n,\mathbb{R})$. There exists a lower-triangular matrix $\mu \in Sp(2n,\mathbb{R})$ whose diagonal is 1, and a diagonal matrix $D \in Sp(2n,\mathbb{R})$ such that, $G = \mu D\mu^t$. And the pair (μ, D) is unique.*

Before proving this theorem, let us give three important consequences on Cholesky factorization, LQ factorization and Gram-Schmidt orthogonalization :

Theorem D.18 (Symplectic Cholesky) *If $G \in S_p(2n,\mathbb{R})$ is a symmetric positive definite matrix, then its Cholesky factorization is symplectic.*

Proof. Apply Theorem D.17 to G, then μ is lower-triangular symplectic with only 1 on the diagonal. Since G is positive definite, the diagonal matrix $D = \mu^{-1}G\mu^{-t}$ is positive definite. But D is also symplectic, so its coefficients satisfy $d_{i,i} = 1/d_{2n+1-i,2n+1-i}$ (see the end of Section D.2.2). For these reasons, the square root C of D (with $c_{i,i} = \sqrt{d_{i,i}}$) is also symplectic. It is clear that $L = \mu C$ is symplectic and satisfies $G = LL^t$. Since the Cholesky factorisation of G is unique, it must be L and it is therefore symplectic. □

Theorem D.19 (Symplectic LQ) *If B is symplectic, then its LQ decomposition is such that both L and Q are symplectic.*

Proof. L is the Cholesky factorization of the matrix BB^t, which is symplectic, so the previous theorem shows that L is symplectic. Then $Q = L^{-1}B$ is also symplectic, because $Sp(2n,\mathbb{R})$ is a group. □

Theorem D.20 (Symplectic Gram-Schmidt) *If B is symplectic, then the μ matrix of its Gram-Schmidt orthogonalisation is also symplectic.*

Proof. Apply Theorem D.17 to $G = BB^t$, then μB represents an orthogonal basis, because its Gram matrix is diagonal. □
Thus, the isometry σ represented by J_{2n} that sends the symplectic basis onto its dual basis is also an isometry between each part of the GSO of the symplectic basis and its dual basis :

Corollary D.1 *Let $[\mathbf{b}_1, ..., \mathbf{b}_{2n}]$ be a symplectic basis of a $2n$-dimensional lattice, then the GSO satisfy for all $i \leq n$, $\mathbf{b}^*_{2n+1-i} = \frac{1}{||\mathbf{b}^*_i||^2}\mathbf{b}^*_i J$ and $\mathbf{b}^*_i = \frac{1}{||\mathbf{b}^*_{2n+1-i}||^2}\mathbf{b}^*_{2n+1-i}J$*

Proof. Consider the LQ factorization of $[\mathbf{b}_1, ..., \mathbf{b}_{2n}]$. The unitary matrix Q is symplectic, therefore equation (D.2) implies that $Q = -JQJ$. Hence, a unitary symplectic matrix always has the form :

$$\begin{pmatrix} C & D \\ -R_nDR_n & R_nCR_n \end{pmatrix} = \begin{pmatrix} A \\ R_nAJ_{2n} \end{pmatrix}.$$

This proves that the directions of the $\mathbf{b}^*_i$ in this corollary are correct. Their norm are the diagonal coefficients of L, and this matrix is lower-triangular symplectic, so Lemma D.32 implies that $||\mathbf{b}^*_{2n+1-i}|| = 1/||\mathbf{b}^*_i||$. □
We now prove Theorem D.17 by induction over n. There is nothing to prove for $n = 0$. Thus, assume that the result holds for $n-1$ and let $G = (g_{i,j})$ be a symmetric matrix in $Sp(2n,\mathbb{R})$. The

main idea is to reduce the first column G with a symplectic transformation, then verify that it automatically reduces the last row, and finally use the induction hypothesis to reduce the remaining $2n-2$ dimensional block in the center. The symplectic transformation has the form :

$$P = \begin{bmatrix} 1 & 0 & 0 \\ (\alpha_2, ..., \alpha_{2n-1})^t & I_{2n-2} & 0 \\ \alpha_{2n} & (\alpha_2, ..., \alpha_{2n-1})J_{2n-2} & 1 \end{bmatrix} \text{ where } \alpha_2, ..., \alpha_{2n-1} \in \mathbb{R}.$$

This is a symplectic matrix because of Lemma D.32. Apply the transformation with $\alpha_i = -\frac{g_{i,1}}{g_{1,1}}$. Then PGP^t has the following shape :

$$PGP^t = \begin{bmatrix} g_{1,1} & 0 & \gamma \\ 0 & S & \mathbf{u}^T \\ \gamma & \mathbf{u} & \beta \end{bmatrix},$$

where S is a $(2n-2) \times (2n-2)$ symmetric matrix, $\mathbf{u}$ is a $(2n-2)$ dimensional row vector, and $\beta \in \mathbb{R}$ and $\gamma = 0$. The coefficient γ in the bottom left corner of PGP^t is equal to zero, because α_{2n} satisfies $\alpha_{2n}g_{1,1} + \alpha_{2n-1}g_{2,1} + ... + \alpha_{n+1}g_{n,1} - \alpha_n g_{n+1,1} - ... - \alpha_2 g_{2n-1,1} + g_{2n,1} = 0$.

Since PGP^t is symplectic, the image by J_{2n} of the first row $\mathbf{r}_1$ of PGP^t has the form $\mathbf{e}_{2n} = (0, ..., 0, g_{1,1})$ and its j^{th} row $\mathbf{r}_j$ satisfies $\langle \mathbf{e}_{2n}, \mathbf{r}_j \rangle = \delta_{2n,j}$ for all $j \geq 2$ (where δ is the Kronecker symbol) : in other words, $\mathbf{u} = 0$ and $\beta = 1/g_{11}$.

$$PGP^t = \begin{bmatrix} g_{1,1} & 0 & 0 \\ 0 & S & 0 \\ 0 & 0 & \frac{1}{g_{1,1}} \end{bmatrix}.$$

As a result, S is symmetric positive definite and symplectic. The induction hypothesis implies the existence of a pair (μ_S, D_S) such that $S = \mu_S D_s \mu_S^t$, and we can extend μ_S to a lower-triangular matrix $U \in Sp(2n, \mathbb{R})$ using the third property at the end of Section D.2.2 :

$$U = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mu_S & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Hence, the product $\mu = UP^{-1}$ is a lower-triangular symplectic matrix whose diagonal is 1, and $D = \mu^{-1}G\mu^{-t} \in Sp(2n, \mathbb{R})$ is diagonal. This concludes the proof of Theorem D.17 by induction.

D.5 Speeding-up Gram-Schmidt by Duality

The standard integral Gram Schmidt algorithm we recalled in Section D.2 is based on a formula which computes $\mu_{i,j}$ from the $\mu_{i,k}$'s $(k < j)$ on the same row. This leads to many integer divisions for each coefficient as in the innerloop rows 5-7 of Algorithm 10.

D.5.1 The general case

We now show that most of these divisions can be avoided. Consider a basis $B = [\mathbf{b}_1, ..., \mathbf{b}_d]$ and its Gram matrix $G = BB^t$. We know that if μ is the Gram-Schmidt orthogonalization of B, then $G = \mu D \mu^t$ where D is a positive diagonal matrix. If we rewrite the previous equation as $\mu = G\mu^{-t}D^{-t}$, it appears that for any integer $k < d$, if we know the $k \times k$ topleft triangle of μ and D, we can compute the $k \times k$ topleft triangle of $\mu^{-t}D^{-t}$ and the first full k columns of μ. The matrix μ^{-t} is just a rotation of μ^{-s}, which is the Gram-Schmidt orthogonalization of the dual basis of $(\mathbf{b}_d, ..., \mathbf{b}_1)$. At the end, we get not only the GSO of B, but also the one of its reverse dual basis : this method which we call "Dual Gram-Schmidt", is surprisingly faster than the classical one despite computing more information.

Theorem D.21 *Let $G \in \mathcal{M}_n(\mathbb{Z})$ be a symmetric (positive) definite matrix, and μ the $\mu D \mu^t$ factorization of G. As in Section D.2, we define $\lambda_{0,0} = 1$ and $\lambda_{k,k} = \det G_k$ where G_k is the $k \times k$ topleft block of G. Let $\lambda = \mu \cdot \mathrm{diag}(\lambda_{1,1}, ..., \lambda_{n,n})$ and $U = \mu^{-t} \cdot \mathrm{diag}(\lambda_{0,0}, ..., \lambda_{n-1,n-1})$. Then the following three relations hold :*

$$\lambda \in \mathcal{M}_n(\mathbb{Z}), \tag{D.4}$$
$$U \in \mathcal{M}_n(\mathbb{Z}), \tag{D.5}$$
$$\lambda = GU. \tag{D.6}$$

Proof. From $G = \mu D \mu^t$, we know that $\mu^{-1} G$ is uppertriangular : for all i and t with $i > t$, then $\sum_{j=1}^{i} \mu_{i,j}^{-1} g_{j,t} = 0$. If we call $G'$the $(i-1) \times (i-1)$ topleft block of G, $\mathbf{v} = (\mu_{i,1}^{-1}, ..., \mu_{i,i-1}^{-1})$ and $\mathbf{g'} = (g_{i,1}, ..., g_{i,i-1})$, then the previous equation is equivalent to $\mathbf{g'} = -\mathbf{v}G'$. By Cramer's rule, we deduce the following relation for all $j < i$, which proves relation (D.5) :

$$u_{j,i} = \det G' \cdot \mathbf{v}_j = (-1)^{i-j} \det \begin{pmatrix} g_{1,1} & \cdots & g_{1,j-1} & g_{1,j+1} & \cdots & g_{1,i} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ g_{i-1,1} & \cdots & g_{i-1,j-1} & g_{i-1,j+1} & \cdots & g_{i-1,i} \end{pmatrix}.$$

The relation (D.6) is obtained by multiplying $\mu = G\mu^{-t}D^{-t}$ by $\mathrm{diag}(\lambda_{1,1}, ..., \lambda_{n,n})$. It also implies that $\lambda_{p,i} = \sum_{k=1}^{i} g_{p,k} u_{k,i}$ is the last row development of an integer determinant :

$$\lambda_{p,i} = \det \begin{pmatrix} g_{1,1} & \cdots & g_{1,i} \\ \vdots & \ddots & \vdots \\ g_{i-1,1} & \cdots & g_{i-1,i} \\ g_{p,1} & \cdots & g_{p,i} \end{pmatrix},$$

which concludes the proof. □

Note that these determinants prove that $\lambda_{p,i} \leq |G|^i$ and $U_{j,i} \leq |G|^{i-1}$.

We derive from $\mu^{-t}\mu^t = \mathrm{Id}$ a column formula to compute the matrix U defined in the previous theorem :

$$(\mu^{-t})_{i,j} = -\mu_{i,j}^t - \sum_{k=j+1}^{i-1} \mu_{i,k}^t (\mu^{-t})_{k,j}. \tag{D.7}$$

From the definition of U, we know that $(\mu^{-t})_{i,j} = \frac{U_{i,j}}{\lambda_{j-1,j-1}}$. Replacing $\mu_{i,j}$ by $\frac{\lambda_{i,j}}{\lambda_{j,j}}$, we may rewrite the formula as : $-U_{i,j} = \left(\sum_{k=j+1}^{i-1} \lambda_{k,i} U_{k,j}\right) / \lambda_{i,i}$. Hence if we know the $i \times (i-1)$ top-left triangle of λ, we can compute the i^{th} column of U using this formula (from the diagonal to the top), and then the i^{th} column of λ using relation (D.6) of Theorem D.21. It is not necessary to keep the i^{th} column of U after that.

We deduce Algorithm 11 to compute the GSO of a basis B. The correctness of this algorithm is a consequence of the results described in this section. If we look at the number of operations requested, there are $i^2/2$ multiplications and one division in the innerloop lines 4-6, and $i(n+1-i)$ small multiplications between the input Gram matrix and U in the innerloop lines 7-9. This gives a total of $n^3/6$ large multiplications and $n^2/2$ divisions. In the Standard GS algorithm, there was as many multiplications, but $\Theta(n^3)$ divisions.

Algorithm 11 Dual Gram Schmidt

Input : A basis $B = (\mathbf{b}_1, ..., \mathbf{b}_d)$ or its Gram matrix G
Output : The GSO decomposition λ of B

for $i = 1$ to d **do**
  $U_i \leftarrow \lambda_{i-1,i-1}$ (for all k, U_k represents $U_{k,i}$)
  $U_{i-1} \leftarrow -\lambda_{i,i-1}$
  for $j = i - 2$ downto 1 **do**
    compute $U_j = -(\sum_{k=j+1}^{i} \lambda_{k,j} U_k)/\lambda_{j,j}$
  end for
  for $j = i$ to n **do**
    compute $\lambda_{j,i} = \sum_{k=1}^{i} \langle \mathbf{b}_j, \mathbf{b_k} \rangle U_k$
  end for
end for

D.5.2 The symplectic case

We derive an algorithm specialized to q-symplectic bases to compute the λ matrix of the GSO, and we show why it is faster than the Dual Gram-Schmidt procedure applied to symplectic bases (see Algorithm 10 of Section D.2). Let B be a q-symplectic basis. We know that L in the LQ decomposition of B is q-symplectic and the μ matrix corresponding to the GSO of B is symplectic. We will also use the integer dual matrix U we introduced in Theorem D.21. Let us denote the quadrants of μ, and λ by :

$$\mu = \begin{pmatrix} \mu_a & 0 \\ \mu_\gamma & \mu_\delta \end{pmatrix}, \lambda = \begin{pmatrix} \lambda_a & 0 \\ \lambda_\gamma & \lambda_\delta \end{pmatrix} \qquad \text{and} \qquad U = \begin{pmatrix} U_\alpha & 0 \\ U_\gamma & U_\delta \end{pmatrix}.$$

Because of Theorem D.20, we know that $\mu_\delta = \mu_\alpha^{-s}$. Together with Corollary D.1, we have $U_\alpha = R_n \lambda_\delta R_n$ for symplectic matrices and the following relation for q-symplectic matrices : $U_\alpha \cdot \text{diag}(q^2, q^4, ..., q^{2n}) = R_n \lambda_\delta R_n$. For this reason, it is only necessary to run DualGS up to the middle of the matrix, and fill the columns of λ_δ using those of U_α. Given as input a symplectic basis $B = (\mathbf{b}_1, ..., \mathbf{b}_{2n})$, Algorithm 12 computes the λ matrix of the GSO of B in time $O(n^5 \log^2 B)$ using standard arithmetic.

Algorithm 12 Symplectic Gram-Schmidt

Input : A q-symplectic basis $B = [\mathbf{b_1}, ..., \mathbf{b_{2n}}]$
Output : The λ matrix of the GSO of B.

precompute : q^{2i} for $i = 1$ to n
for $i = 1$ to n **do**
  $U_i \leftarrow \lambda_{i-1,i-1}$ (for all k, U_k represents $U_{k,i}$)
  $U_{i-1} \leftarrow -\lambda_{i,i-1}$
  for $j = i - 2$ downto 1 **do**
    $U_j \leftarrow -(\sum_{k=j+1}^{i} \lambda_{k,j} U_k)/\lambda_{j,j}$
  end for
  for $j = 1$ to i **do**
    $\lambda_{2n+1-j,2n+1-i} \leftarrow q^{2(n+1-i)} \cdot U_j$
  end for
  for $j = i$ to $2n$ **do**
    compute $\lambda_{j,i} = \sum_{k=1}^{i} G_{j,k} U_k$
  end for
end for

Tab. D.1 – Timing of Gram-Schmidt algorithms on random matrices

n	StandardGS in seconds	DualGS in seconds	**DualGS speedup**
100	37.3	25.1	1.49
200	881	579	1.52
300	5613	3644	1.54

Tab. D.2 – Timing of Gram-Schmidt algorithms on NTRUSign bases

$2n$	Standard GS	DualGS	SympGS	**speedup DualGS**	**speedup SympGS**
502	179	122	8.7	1.46	20.5
802	1822	1254	67	1.45	27.1
1214	12103	8515	390	1.42	31.0

D.5.3 Experiments

We performed tests on randomly generated bases, whose coefficients are uniformly distributed 128-bit integers (see Table D.1 page 186). On these random matrices, the speedup is rather moderate, but it will be much more significant when considering symplectic matrices.

We also performed tests on secret bases of NTRUSign as described in Section D.3 (see Table D.2 page 186). Roughly speaking, Algorithm 11 is at least 3 times as fast as Standard GS, and the specialized algorithm is from 10 to 30 times as fast as Standard GS. We give in function of the dimension of the input matrix, the running time in seconds to compute the GSO for each algorithm on 2Ghz Opteron processors with a 32-bit version of NTL 5.4. Note that the speedup of 31 in Symplectic GS seems to indicate that the cost of computing the GSO of a $2n$-dimensional symplectic basis is roughly the one of computing the GSO of a standard n-dimensional standard matrix. The GMP library contains a division function *divexact* which is faster than the standard *divrem* used in NTL when the integer division is known to be exact. The speedup decreases when using GMP and *divexact* but remains greater than 1. For example it is about 1.2 for "random matrices" like in Table D.1 page 186.

D.6 Symplectic LLL

When applied to a symplectic basis, the standard LLL algorithm will likely not preserve symplecticity, because its elementary operations are not necessarily symplectic. In this section, we show how one can slightly modify the notion of size-reduction used in LLL to make it compatible with symplecticity, and we then deduce a symplectic version of LLL. We do not know if every symplectic lattice contains an LLL-reduced basis which is also symplectic. But we show how to obtain efficiently a symplectic basis which is *effectively LLL-reduced* (as introduced in [155]). Such bases satisfy the most important properties of LLL-reduced bases, those which are sufficient to guarantee the shortness of the first basis vector.

D.6.1 Symplectic size-reduction

The first condition in LLL-reduction is size-reduction, which we recalled in Section D.2. Unfortunately, size-reduction transformations are not necessarily symplectic. However, we show that it is still possible to force half of the coefficients of μ to be very small using symplectic transformations, and at the same time, bound the size of the remaining coefficients.

We say that a matrix $B \in \mathcal{M}_{2n}(\mathbb{R})$ is *semi-size reduced* if its GSO satisfies : for all $j \leq n$, for all $i \in [j+1, 2n+1-j]$, $|\mu_{i,j}| \leq \frac{1}{2}$.

Theorem D.22 *If $B \in Sp(2n, \mathbb{R})$ is semi-size reduced, then its GSO μ is bounded by $||\mu||_\infty \leq n \cdot (\frac{3}{2})^n$.*

Proof. For the block μ_δ , Equation (D.7) gives for $i \geq n+1$ and $j \geq n+1$, $|\mu_{i,j}| \leq \frac{1}{2} + \frac{1}{2}\sum_{k=j+1}^{i-1} |\mu_{i,k}|$, which is bounded by a geometric sequence of ratio $\frac{3}{2}$. Hence, the bottom diagonal block is bounded by $|\mu_{i,j}| \leq \frac{1}{2}(\frac{3}{2})^{i-j-1}$, and this bound can be reached for $\mu_{i\in[1,n],j\in[1,i-1]} = -1/2$.

For the bound on block μ_γ , apply Equation (D.1) to μ^s in order to get a column formula. This gives for $i \geq n+1$ and $j \geq 2n-i$:

$$\begin{aligned}\mu^s_{n+1-j,n+1-i} &= \mu^s_{i,j} + \sum_{k=j+1}^{n} \mu^s_{i,k}\mu^s_{2n+1-j,2n+1-k} \\ &\quad - \sum_{k=n+1}^{i-1} \mu^s_{i,k}\mu^s_{2n+1-j,2n+1-k}.\end{aligned}$$

After reindexing the matrix and applying the triangular inequality to this sum, we obtain

$$|\mu_{i,j}| \leq \frac{1}{2} + \frac{2n+1-2j}{4} + \frac{1}{2}\sum_{k=2n+2-j}^{i-1} |\mu_{k,j}|.$$

It is still bounded by a geometric sequence of ratio $\frac{3}{2}$, but the initial term $\mu_{i+1,i}$ is less than $\frac{2n+3-2j}{4} \leq n$. Thus $|\mu_{i,j}| \leq n \cdot (\frac{3}{2})^{i-2n+j-2}$. □

D.6.2 Lovász conditions and effective reduction

A basis satisfying all Lovász conditions and $|\mu_{i,i-1}| \leq 1/2$ is referred to as *effectively LLL-reduced* in [155]. Such bases have the same provable bounds on the size of $\mathbf{b}_1$ (which is typically the most interesting vector in a basis) as LLL-reduced bases. Besides, it is easy to derive an LLL-reduced basis from an effectively LLL-reduced basis, using just size reductions (no swaps). In general the reason for weakly reducing the other $\mu_{i,j}$ for $1 \leq j < i-1$ is to prevent coefficient explosion in the explicit $\mathbf{b}_i$, but there are many other strategies for this that don't require as strict a condition as $|\mu_{i,j}| \leq 1/2$, $1 \leq j < i-1$ (see [202, 315]). It is not difficult to see that this notion of *"effective LLL-reduction"* can be reached by symplectic transformations.

Lemma D.33 *A symplectic $2n$-dimensional basis B is* effectively-reduced *if and only if its first $n+1$ vectors are* effectively LLL-reduced.

Proof. Let μ be the GSO matrix of B, for $i \leq n$, since μ is symplectic, we know that $\mu_{2n+2+i,2n+1-i} = \mu_{i,i-1}$. Using Corollary D.1, the Lovasz condition for the i^{th} index is equivalent to $\delta\frac{1}{||\mathbf{b}^*_{\mathbf{2n+2-i}}||^2} \leq \frac{1}{||\mathbf{b}^*_{\mathbf{2n+1-i}}||^2} - \mu_{2n+2+i,2n+1-i}\frac{1}{||\mathbf{b}^*_{\mathbf{2n+2-i}}||^2}$, which is precisely the Lovasz condition for the $2n+2-i^{\text{th}}$ index. □

This means that for all $i \leq n$, every operation made on the rows i and $i-1$ that reduces B can be blindly applied on the rows $2n-i+2$ and $2n-i+1$ without knowing the GSO of the second block. A symplectic basis is said to be *symplectic-LLL reduced* if it is both *effectively LLL-reduced* and *semi-size-reduced*.

D.6.3 A Symplectic-LLL algorithm

It is easy to find polynomial algorithms for symplectic-LLL reduction, but the difficulty is to make them run faster than LLL in practice. Here, we present an algorithm which reaches symplectic-LLL reduction with an experimental running-time 6 times smaller than LLL on NTRU public bases of standard size (but the speed up may be larger for higher-dimensional lattices).

Symplectic LLL is an iterative algorithm that reduces a symplectic lattice L from its center. It takes as input the integer GSO λ of a symplectic lattice and outputs the GSO of a symplectic-LLL reduced basis and the unimodular transformation that achieves the reduction. More precisely, it only keeps one half of the GSO of symplectic matrices, since the other half can be easily deduced with (D.1) or Lemma D.32. Here, we chose to keep the left triangle $\lambda' = \lambda_{i,j}, 1 \leq j \leq n, j \leq i \leq 2n+1-j$. During the algorithm, every elementary operation (swap or a linear combination of rows) is deduced from λ', and λ' is updated incrementally like in the standard integer LLL (see [205, 65]). As a result, symplecticLLL can generate the complete sequence of elementary operations of the reduction without knowing the basis. Unfortunately, having only the GSO of the LLL reduced basis is not sufficient to compute its coefficients, so every operation that occur in symplectic LLL algorithm is in fact performed on a third part matrix U. If U is initially equal to the input basis (resp. the identity matrix), then it is transformed into the LLL-reduced basis (resp. the unitary transformation).

In this paragraph, we explain the principles of SymplecticLLL on the projected lattice vectors, but in practice, all operations are done on the GSO λ' (see Algorithm 13 for details). For $1 \leq k \leq n$, let

$$C_k = [\pi_{n+1-k}(\mathbf{b}_{n+1-k}), ..., \pi_{n+1-k}(\mathbf{b}_{n+k-1})].$$

The $2k$-dimensional lattice $L(C_k)$ is symplectic, and its GSO matrix μ is the $2k \times 2k$ block located in the center of the GSO of the basis. When the algorithm begins, the counter k is set to 1. At the start of each loop iteration, C_{k-1} is already symplectic-LLL-reduced (there is no condition if $k = 1$). If $k = 1$ then the projected lattice C_1 is Lagrange-reduced and the counter k is set to 2. If the determinant of the transformation computed by Lagrange reduction is -1, we negate $\mathbf{b}_{n+1}$ to preserve the symplecticity of the basis. In the general case, C_k is semi-size-reduced, which means that $\lambda_{i,n+1-k}$ is made lower than $\frac{1}{2}\lambda_{n+1-k,n+1-k}$ with symplectic combinations of rows for $i = n+2-k$ to $n+k$. If the pair $(n+1-k, n+2-k)$ does not satisfy Lovász condition (by symplecticity neither does the pair $(n+k-1, n+k)$), then the pairs of consecutive vectors $(\mathbf{b}_{n-k+1}, \mathbf{b}_{n-k+2})$ and $(\mathbf{b}_{k-1}, \mathbf{b}_k)$ are swapped and the counter k is decremented, otherwise k is incremented. The loop goes on until k eventually reaches the value $n+1$.

Experiments show that this basic symplecticLLL algorithm is already as fast as LLL in dimension 80, and becomes faster in higher dimension. The quality of the output basis is similar to the one of StandardLLL. (see Figure D.1 page 190). Note also that the curve of $\log \|\mathbf{b}_i^*\|$ obtained after symplecticLLL is symmetric because of Corollary D.1. In other words, both the basis and its dual are reduced in the same time. We now describe optimizations.

D.6.4 Optimizations

The following two optimizations do not modify the output of the algorithm, but considerably speed up the algorithm in practice :

Early reduction. Let C_i be the $2i$-dimensional central projection of the input basis for $2 \leq i \leq n$. Suppose that Algorithm 13 found the unimodular matrix $U_p \in Sp(2p, \mathbb{Z})$ such that $U_p C_p$

Algorithm 13 symplectic LLL

Input : A GSO matrix λ of a q-symplectic basis (at least the left triangle λ')
Output : The GSO λ' of the reduced basis, and the unitary transformation U

$k \leftarrow 1$,$U = I_{2n}$ (or $U = U_{\text{init}}$ initially given by the user)
while $k \leq n$ **do**
  if $k = 1$ **then**
    compute $\lambda_{n+1,n+1} = q^2 \lambda'_{n-1,n-1}$
    find the 2×2 unimodular transformation P that Lagrange-reduces the GSO of C_1
    ensure that the determinant of P is not -1, negate one row of P if necessary
    Apply P on the two middle rows of λ' and U, and update $\lambda'_{n,n}, \lambda'_{n+1,n}$
    $k \leftarrow 2$
  end if
  for $i = n + 2 - k$ to $n + k$ **do**
    $r \leftarrow \lfloor \lambda_{i,n+1-k} / \lambda_{n+1-k,n+1-k} \rceil$
    $\mathbf{u}_i \leftarrow \mathbf{u}_i - r\,\mathbf{u}_{n+1-k}$ and $\lambda'_i \leftarrow \lambda'_i - r\,\lambda'_{n+1-k}$
    $\mathbf{u}_{n+k} \leftarrow \mathbf{u}_{n+k} + r\,\mathbf{u}_{2n+1-i}$ and $\lambda'_{n+k} \leftarrow \lambda'_{n+k} + r\,\lambda'_{2n+1-i}$ **if** $i \leq n$
    $\mathbf{u}_{n+k} \leftarrow \mathbf{u}_{n+k} - r\,\mathbf{u}_{2n+1-i}$ and $\lambda'_{n+k} \leftarrow \lambda'_{n+k} - r\,\lambda'_{2n+1-i}$ **if** $n + 1 \leq i \leq n + k - 1$
  end for
  if Lovász does not hold for the pair $(n - k + 1, n - k + 2)$ **then**
    compute $\lambda_{n+k,n-k+2}$ using Lemma D.32
    swap $\mathbf{u}_{n+k} \leftrightarrow \mathbf{u}_{n+k-1}$ and $\lambda'_{n+k,j} \leftrightarrow \lambda'_{n+k-1,j}$ for $1 \leq j \leq n - k$
    swap $\mathbf{u}_{n-k+2} \leftrightarrow \mathbf{u}_{n-k+1}$ and $\lambda'_{n-k+2,j} \leftrightarrow \lambda'_{n-k+2,j}$ for $1 \leq j \leq n - k$
    update $\lambda'_{n-k+2,n-k+1}$ and $\lambda'_{i,n-k+1}, \lambda'_{i,n-k+2}$ for $n - k + 3 \leq i \leq n + k$ using the same swap formula as standard LLL
    $k \leftarrow k - 1$
  else
    $k \leftarrow k + 1$
  end if
end while

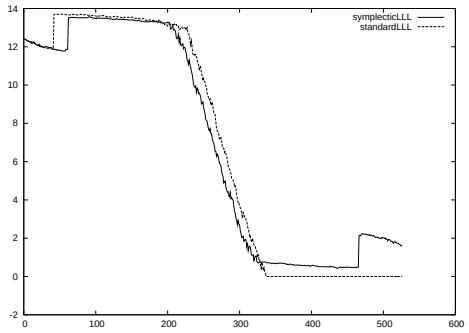
is symplecticLLL reduced and the reduced GSO λ'_p. If we want to reduce the initial C_{p+1} using Algorithm 13, we know that when the counter k reaches $p + 1$ for the first time, the current state U_{p+1} and λ'_{p+1} is :

$$U_{p+1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & U_p & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \text{and} \qquad \lambda'_{p+1} = \begin{pmatrix} \lambda_{n-p,n-p} & 0 \\ U_p \lambda_{.,n-p} & \lambda'_p \\ \lambda_{n+p+1,n-p} & 0 \end{pmatrix}.$$

So we can launch Algorithm 13 on λ'_{p+1} with $U_{\text{init}} = U_{p+1}$ to finish the reduction. Using this simple trick for $p = 2$ to n, the first transformations of Algorithm 13 apply to lower dimensional matrices. On NTRU matrices, the execution is almost two times faster than the basic symplecticLLL. In the Standard LLL algorithm, the analogue is to update only the first p rows of the GSO, where p is the maximum index of vectors already modified by the main loop of LLL since the beginning of the execution.

Integer triangular matrices. This last optimization only works on matrices for which every $\|\mathbf{b}_k^*\|^2$ is an integer (at the beginning). It is the case of all NTRU public key matrices, and all integer triangular matrices. The key result is that in the previous algorithm, each λ'_p is initially divisible by $D_{n-p} = \prod_{i=1}^{n-p} \|\mathbf{b}_i^*\|^2$. The only improvement is to use reduced GSO λ'_p / D_{n-p} instead of λ'_p in the previous algorithm. Then the first transformations of Algorithm 13 apply to matrices of lower

FIG. D.1 – Quality of the input basis ($\log_2 \|\mathbf{b}_i^*\|$ as a function of i)



dimension, but also with smaller coefficients. On NTRU matrices, the execution becomes almost 4 times faster than the basic symplecticLLL.

TAB. D.3 – Experimental results (on 2GHz Opteron with 32-bit NTL 5.4)

n *half of dim.*	*q* *max. coefs*	Standard LLL *in seconds*	SympLLL Early red. *in seconds*	SympLLL Integer triang. optim. *in seconds*	**speedup Early reduction**	**speedup integer triang.**
40	64	3.09	2.27	1.98	1.36	1.56
83	64	26.89	6.62	4.46	4.06	6.02
107	64	44.7	6.13	4.51	7.29	9.91
167	128	410.8	98.86	65.40	4.15	6.28
253	128	2028	553	294	3.66	6.89
317	128	3688	1131	519	3.26	7.10

Annexe E

Rankin's Constant and Blockwise Lattice Reduction

CRYPTO 2006

[103] avec Nicolas Gama (ENS), Nick Howgrave-Graham (NTRU Cryptosystems) et Henrik Koy (Deutsche Bank AG)

Abstract: *Lattice reduction is a hard problem of interest to both public-key cryptography and cryptanalysis. Despite its importance, extremely few algorithms are known. The best algorithm known in high dimension is due to Schnorr, proposed in 1987 as a block generalization of the famous LLL algorithm. This paper deals with Schnorr's algorithm and potential improvements. We prove that Schnorr's algorithm outputs better bases than what was previously known : namely, we decrease all former bounds on Schnorr's approximation factors to their* $(\ln 2)$*-th power. On the other hand, we also show that the output quality may have intrinsic limitations, even if an improved reduction strategy was used for each block, thereby strengthening recent results by Ajtai. This is done by making a connection between Schnorr's algorithm and a mathematical constant introduced by Rankin more than 50 years ago as a generalization of Hermite's constant. Rankin's constant leads us to introduce the so-called smallest volume problem, a new lattice problem which generalizes the shortest vector problem, and which has applications to blockwise lattice reduction generalizing LLL and Schnorr's algorithm, possibly improving their output quality. Schnorr's algorithm is actually based on an approximation algorithm for the smallest volume problem in low dimension. We obtain a slight improvement over Schnorr's algorithm by presenting a cheaper approximation algorithm for the smallest volume problem, which we call transference reduction.*

E.1 Introduction

Lattices are discrete subgroups of $\mathbb{R}^m$. A lattice L can be represented by a basis, that is, a set of $n \leq m$ linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$ in $\mathbb{R}^m$ such that L is equal to the set $L(\mathbf{b}_1, \ldots, \mathbf{b}_n) = \{\sum_{i=1}^{n} x_i \mathbf{b}_i, x_i \in \mathbb{Z}\}$ of all integer linear combinations of the $\mathbf{b}_i$'s. The integer n is the dimension of the lattice L. A lattice has infinitely many bases (except in trivial dimension ≤ 1), but some are more useful than others. The goal of *lattice reduction* is to find interesting lattice bases, such as bases consisting of reasonably short and almost orthogonal vectors : it can intuitively be viewed as a vectorial generalisation of gcd computations. Finding good reduced bases has proved invaluable in many fields of computer science and mathematics (see [129, 65]), particularly in cryptology (see [240, 267]).

Lattice reduction is one of the few potentially hard problems currently in use in public-key cryptography (see [267, 240] for surveys on lattice-based cryptosystems). But the problem is perhaps more well-known in cryptology for its major applications in public-key cryptanalysis (see [267]) : knapsack cryptosystems [274], RSA in special settings [70, 43, 40], DSA signatures in special settings [159, 261], *etc.* Nevertheless, there are very few lattice reduction algorithms, and most of the (recent) theoretical results focus on complexity aspects (see [240]).

The first lattice reduction algorithm in arbitrary dimension is due to Hermite [142], and is based on Lagrange's two-dimensional algorithm [201] (often wrongly attributed to Gauss). It was introduced to show the existence of Hermite's constant (which guarantees the existence of short lattice vectors), as well as proving the existence of lattice bases with bounded orthogonality defect. The celebrated Lenstra-Lenstra-Lovász algorithm [205] (LLL) can be viewed as a relaxed variant of Hermite's algorithm, in order to guarantee a polynomial-time complexity. There are faster variants of LLL based on floating-point arithmetic (see [264, 303]), but none improves the output quality of LLL, which is tightly connected to Hermite's historical (exponential) upper bound on his constant. The only (high-dimensional) polynomial-time reduction algorithm known with better output quality than LLL is due to Schnorr [302]. From a theoretical point of view, only one improvement to Schnorr's block-reduction algorithm has been found since [302] : by plugging the probabilistic AKS sieving algorithm [8], one may increase the blocksize $k = \log n / \log\log n$ to $k = \log n$ and keep polynomial-time complexity, which leads to (slightly) better output quality. Curiously, in practice, one does not use Schnorr's algorithm when LLL turns out to be insufficient : rather, one applies the so-called BKZ variants [307, 308] of Schnorr's algorithm, whose complexity is unknown.

Our Results. We focus on the best high-dimensional lattice reduction algorithm known (Schnorr's semi block-$2k$ algorithm [302]) and potential improvements. Despite its importance, Schnorr's algorithm is not described in any survey or textbook, perhaps due to the technicality of the subject. We first revisit Schnorr's algorithm by rewriting it as a natural generalization of LLL. This enables to analyze both the running time and the output quality of Schnorr's algorithm in much the same way as with LLL. It also leads us to reconsider a certain constant β_k introduced by Schnorr [302], which is tightly related to the output quality of his semi block-$2k$ algorithm. Roughly speaking, β_k plays a role similar to Hermite's constant $\gamma_2 = \sqrt{4/3}$ in LLL.

We improve the best upper bound known for β_k : we show that essentially, $\beta_k \lessapprox 0.38 \times k^{2\ln 2} \approx 0.38 \times k^{1.39}$, while the former upper bound [302] was $4k^2$. This leads to better bounds on the output quality of Schnorr's algorithm : for instance, the approximation factor $(6k)^{n/k}$ given in [302] can be decreased to its $(\ln 2)$-th power (note that $\ln 2 \approx 0.69$). On the other hand, Ajtai [7] recently proved that there exists $\varepsilon > 0$ such that $\beta_k \geq k^{\varepsilon}$, but no explicit value of ε was known. We establish the lower bound $\beta_k \geq k/12$, and our method is completely different from Ajtai's. Indeed, we use a connection between β_k and a mathematical constant introduced by Rankin [292] more than 50 years ago as a generalization of Hermite's constant.

Besides, Rankin's constant is naturally related to a potential improvement of Schnorr's algorithm, which we call block-Rankin reduction, and which may lead to better approximation factors. Roughly speaking, the new algorithm would still follow the LLL framework like Schnorr's algorithm, but instead of using Hermite-Korkine-Zolotarev (HKZ) reduction of $2k$-blocks, it would try to solve the so-called *smallest volume problem* in $2k$-blocks, which is a novel generalization of the shortest vector problem. Here, Rankin's constant plays a role similar to β_k in Schnorr's algorithm. But our lower bound on β_k actually follows from a lower bound on Rankin's constant, which suggests that there are intrinsic limitations to the quality of block-Rankin reduction. However, while Ajtai presented in [7] "worst cases" of Schnorr's algorithm which essentially matched the bounds on the output quality, this is an open question for block-Rankin reduction : perhaps the algorithm may perform significantly better than what is proved, even in the worst case. Finally, we make a preliminary study of the smallest volume problem. In particular, we show that HKZ-reduction does not necessarily solve the

problem, which suggests that block-Rankin reduction might be stronger than Schnorr's semi block reduction. We also present an exact solution of the smallest volume problem in dimension 4, as well as an approximation algorithm for the smallest volume problem in dimension $2k$, which we call *transference reduction*. Because transference reduction is cheaper than the $2k$-dimensional HKZ-reduction used by Schnorr's algorithm, we obtain a slight improvement over Schnorr's algorithm : for a similar cost, we can increase the blocksize and therefore obtain better quality.

Road map. The paper is organized as follows. In Section E.2, we provide necessary background on lattice reduction. In Section E.3, we revisit Schnorr's algorithm and explain its main ideas. Section E.4 deals with Rankin's constant and its connection with Schnorr's algorithm. In Section E.5, we study the smallest volume problem, discuss its application to the so-called block-Rankin reduction, and present transference reduction.

Acknowledgements. Part of this work, as well as a visit of the second author to the ENS, were supported by the Commission of the European Communities through the IST program under contract IST-2002-507932 ECRYPT. We would like to thank Renaud Coulangeon for useful conversations.

E.2 Background

Let $\|.\|$ and $\langle .,. \rangle$ be the Euclidean norm and inner product of $\mathbb{R}^m$. Vectors will be written in bold, and we will use row-representation for matrices. For a matrix M whose name is a capital letter, we will usually denote its coefficients by $m_{i,j}$: if the name is a Greek letter like μ, we will keep the same symbol for both the matrix and its coefficients. The notation $\lceil x \rfloor$ denotes a closest integer to x.

E.2.1 Lattices

We refer to the survey [267] for a bibliography on lattices. In this paper, by the term lattice, we mean a discrete subgroup of some $\mathbb{R}^m$. The simplest lattice is $\mathbb{Z}^n$, and for any linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$, the set $L(\mathbf{b}_1, \ldots, \mathbf{b}_n) = \{\sum_{i=1}^n m_i \mathbf{b}_i \mid m_i \in \mathbb{Z}\}$ is a lattice. It turns out that in any lattice L, not just $\mathbb{Z}^n$, there must exist linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n \in L$ such that $L = L(\mathbf{b}_1, \ldots, \mathbf{b}_n)$. Any such n-tuple of vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$ is called a *basis* of L : a lattice can be represented by a basis, that is, a row matrix. Two lattice bases are related to one another by some matrix in $GL_n(\mathbb{Z})$. The *dimension* of a lattice L is the dimension n of the linear span of L. The lattice is full-rank if n is the dimension of the space. Let $[\mathbf{v}_1, \ldots, \mathbf{v}_k]$ be vectors : we denote by $G(\mathbf{v}_1, \ldots, \mathbf{v}_k)$ their *Gram matrix*, that is, the $k \times k$ symmetric positive definite matrix $(\langle \mathbf{v}_i, \mathbf{v}_j \rangle)_{1 \leq i,j \leq k}$ formed by all the inner products. The *volume* of $[\mathbf{v}_1, \ldots, \mathbf{v}_k]$ is $(\det G(\mathbf{v}_1, \ldots, \mathbf{v}_k))^{1/2}$, which is zero if the vectors are linearly dependent. The *volume* $\mathrm{vol}(L)$ (or *determinant*) of a lattice L is the volume of any basis of L.

Direct sum. Let L_1 and L_2 be two lattices such that $\mathrm{span}(L_1) \cap \mathrm{span}(L_2) = \{\mathbf{0}\}$. Then the set $L_1 \oplus L_2$ defined as $\{\mathbf{u} + \mathbf{v}, \mathbf{u} \in L_1, \mathbf{v} \in L_2\}$ is a lattice, whose dimension is $\dim L_1 + \dim L_2$. It is the smallest lattice containing L_1 and L_2.

Pure Sublattice. A sublattice U of a lattice L is *pure* if there exists a sublattice V of L such that $L = U \oplus V$. A set $[\mathbf{u}_1, \ldots, \mathbf{u}_k]$ of independent lattice vectors of L is *primitive* if and only if $[\mathbf{u}_1, \ldots, \mathbf{u}_k]$ can be extended to a basis of L, which is equivalent to $L(\mathbf{u}_1, \ldots, \mathbf{u}_k)$ being a pure sublattice of L. For any sublattice U of a lattice L, there exists a pure sublattice S of L such that $\mathrm{span}(S) = \mathrm{span}(U)$, in which case $\mathrm{vol}(U)/\mathrm{vol}(S) = [S : U]$ is an integer.

Successive minima. The *successive minima* of an n-dimensional lattice L are the positive quantities $\lambda_1(L), \ldots, \lambda_n(L)$ where $\lambda_r(L)$ is the smallest radius of a zero-centered ball containing r linearly independent vectors of L. The first minimum is the norm of a shortest non-zero vector of L. Note that : $\lambda_1(L) \leq \cdots \leq \lambda_n(L)$.

HERMITE'S CONSTANT. The Hermite invariant of the lattice is defined by $\gamma(L) = \left(\lambda_1(L)/\mathrm{vol}(L)^{\frac{1}{n}}\right)^2$. Hermite's constant γ_n is the maximal value of $\gamma(L)$ over all n-dimensional lattices. Its exact value is known for $1 \leq n \leq 8$ and $n = 24$, and we have [220] : $\gamma_n \leq 1 + \frac{n}{4}$. Asymptotically, the best bounds known are : $\frac{n}{2\pi e} + \frac{\log(\pi n)}{2\pi e} \leq \gamma_n \leq \frac{1.744n}{2\pi e}(1 + \circ(1))$ (see [68, 242]). The lower bound follows from the so-called Minkowski-Hlawka theorem.

PROJECTED LATTICE. Given a basis $[\mathbf{b}_1, ..., \mathbf{b}_n]$ of L, let π_i denote the orthogonal projection over $\mathrm{span}(\mathbf{b}_1, ..., \mathbf{b}_{i-1})^{\perp}$. Then $\pi_i(L)$ is an $(n+1-i)$-dimensional lattice. These projections are stable by composition : if $i > j$, then $\pi_i \circ \pi_j = \pi_j \circ \pi_i = \pi_i$. Note that :

$$\pi_i(L) = \pi_i\left(L(\mathbf{b}_i, \ldots, \mathbf{b}_n)\right) = L\left(\pi_i(\mathbf{b}_i), \ldots, \pi_i(\mathbf{b}_n)\right)$$

E.2.2 Lattice reduction

We will consider two quantities to measure the quality of a basis $[\mathbf{b}_1, \ldots, \mathbf{b}_n]$: the first one is the usual *approximation factor* $\|\mathbf{b}_1\| / \lambda_1(L)$, and the second one is $\|\mathbf{b}_1\| / \mathrm{vol}(L)^{1/n}$, which we call the *Hermite factor*. The smaller these quantities, the shorter the first basis vector. Lovász showed in [214] that any algorithm achieving a Hermite factor $\leq q$ can be used to efficiently find a basis with approximation factor $\leq q^2$ using n calls to the algorithm.

ORTHOGONALIZATION. Given a basis $B = [\mathbf{b}_1, ..., \mathbf{b}_n]$, there exists a unique lower-triangular matrix μ with unit diagonal and an orthogonal family $B^* = [\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*]$ such that $B = \mu B^*$. They can be computed using Gram-Schmidt orthogonalization, and will be denoted the *GSO* of B. Note that $\mathrm{vol}(B) = \prod_{i=1}^n \|\mathbf{b}_i^*\|$, which will often be used. It is well-known [205, 240] that :

$$\lambda_1(L(\mathbf{b}_1, ..., \mathbf{b}_n)) \geq \min_{1 \leq i \leq n} \|\mathbf{b}_i^*\| \tag{E.1}$$

SIZE-REDUCTION. A basis $[\mathbf{b}_1, \ldots, \mathbf{b}_n]$ is *size-reduced* with factor $\eta \geq 1/2$ if its GSO μ satisfies $|\mu_{i,j}| \leq \eta$ for all $1 \leq j < i$. An individual vector $\mathbf{b}_i$ is size-reduced if $|\mu_{i,j}| \leq \eta$ for all $1 \leq j < i$. Size reduction usually refers to $\eta = 1/2$, and is typically achieved by successively size-reducing individual vectors. Size reduction was introduced by Hermite.

LLL-REDUCTION. A basis $[\mathbf{b}_1, \ldots, \mathbf{b}_n]$ is LLL-reduced [205] with factor (δ, η) for $1/4 < \delta \leq 1$ and $1/2 \leq \eta < \sqrt{\delta}$ if the basis is size-reduced with factor η and if its GSO family satisfies the $(n-1)$ Lovász conditions $(\delta - \mu_{i+1,i}^2)\|\mathbf{b}_i^*\|^2 \leq \left\|\mathbf{b}_{i+1}^*\right\|^2$. LLL-reduction usually refers to the factor $(3/4, 1/2)$ because this was the choice considered in the original LLL paper [205]. But the closer δ and η are respectively to 1 and 1/2, the more reduced the basis. Reduction with a factor (1,1/2) is closely related to a reduction notion introduced by Hermite [142].

When the reduction factor is close to (1,1/2), Lovász conditions and size-reduction imply the Siegel conditions [58] : $\|\mathbf{b}_i^*\|^2 \lessapprox \frac{4}{3}\left\|\mathbf{b}_{i+1}^*\right\|^2$for all $1 \leq i \leq n-1$, which limit the drop of the $\|\mathbf{b}_i^*\|$. Here, the $\lessapprox$ symbol means that $\frac{4}{3}$ is actually $\frac{4}{3} + \varepsilon$ for some small $\varepsilon > 0$. In particular, the first vector satisfies $\|\mathbf{b}_1\|^2 \lessapprox \left(\frac{4}{3}\right)^{i-1}\|\mathbf{b}_i^*\|^2$. Hence, the Hermite factor of an LLL-reduced basis is bounded by :

$$\|\mathbf{b}_1\| / \mathrm{vol}(L)^{1/n} \lessapprox \left(\frac{4}{3}\right)^{(n-1)/4} = (\sqrt{\gamma_2})^{n-1},$$

and (E.1) implies that the approximation factor is bounded by :

$$\|\mathbf{b}_1\| / \lambda_1(L) \lessapprox \left(\frac{4}{3}\right)^{(n-1)/2} = (\gamma_2)^{n-1}$$

The LLL algorithm is an iterative algorithm. At the start of each loop iteration, the first i vectors are already LLL-reduced, then the $(i+1)$-th vector is size-reduced ; if it does not satisfy Lovász condition, the consecutive vectors $\mathbf{b}_{i+1}$ and $\mathbf{b}_i$ are swapped and the counter i is decremented, otherwise i is incremented. The loop goes on until i eventually reaches the value n. If L is a full-rank integer lattice of dimension n and B is an upper bound on the $\|\mathbf{b}_i\|$'s, then the complexity of the LLL algorithm described (using integral Gram-Schmidt) without fast integer arithmetic is $O(n^6 \log^3 B)$. The main reason is that the integer $\prod_{k=1}^{n} \|\mathbf{b}_k^*\|^{2(n-k)}$ decreases by at least the geometric factor δ at every swap : thus, the number of swaps is $O(n^2 \log B)$. The recent L^2 algorithm [264] by Nguyen and Stehlé achieves a factor of (δ, ν) arbitrarily close to (1,1/2) in faster polynomial time : the complexity is $O(n^5(n + \log B)\log B)$ which is essentially $O(n^5 \log^2 B)$ for large entries. This is the fastest LLL-type reduction algorithm known for large entries.

HKZ REDUCTION. A basis $[\mathbf{b}_1, \ldots, \mathbf{b}_n]$ of a lattice L is Hermite-Korkine-Zolotarev (HKZ) reduced if it is size-reduced and if $\mathbf{b}_i^*$ is a shortest vector of the projected lattice $\pi_i(L)$ for all $1 \leq i \leq n$. In particular, the first basis vector is a shortest vector of the lattice. Schnorr introduced in [302] a constant to globally measure the drop of the $\|\mathbf{b}_i^*\|$ of $2k$-dimensional HKZ bases :

$$\beta_k = \max_{\substack{L\ 2k\text{-dim. lattice} \\ H\ \text{HKZ-basis of L}}} \left(\frac{\|\mathbf{h}_1^*\| \times \cdots \times \|\mathbf{h}_k^*\|}{\|\mathbf{h}_{k+1}^*\| \times \cdots \times \|\mathbf{h}_{2k}^*\|} \right)^{\frac{2}{k}}$$

which we rewrite more geometrically as,

$$\beta_k = \max_{\substack{L\ 2k\text{-dim. lattice} \\ H\ \text{HKZ-basis of L}}} \left(\frac{\mathrm{vol}(\mathbf{h}_1, \ldots, \mathbf{h}_k)}{\mathrm{vol}(\pi_{k+1}(\mathbf{h}_{k+1}), \ldots, \pi_{k+1}(\mathbf{h}_{2k}))} \right)^{\frac{2}{k}}$$

Schnorr proved that $\beta_k \leq 4k^2$, and Ajtai recently proved in [7] that there exists $\varepsilon > 0$ such that $\beta_k \geq k^{\varepsilon}$, but this is an existential lower bound : no explicit value of ε is known. The value of β_k is very important to bound the output quality of Schnorr's algorithm. One can achieve an n-dimensional HKZ reduction in essentially the same time as finding the shortest vector of an n-dimensional lattice : the deterministic algorithm [302] needs $n^{O(n)}$ polynomial operations, and the probabilistic algorithm [8] needs $2^{O(n)}$ polynomial operations.

E.3 Revisiting Schnorr's Algorithm

In this section, we give an intuitive description of Schnorr's semi block-$2k$-reduction algorithm and show that it is very similar to LLL. The analogy between LLL and Schnorr's algorithm is summarized in Tables E.2 and E.1. We explain the relationship between the constant β_k and the quality of Schnorr reduced basis, and we give the main ideas for its complexity analysis. Here, we assume that the lattice dimension n is a multiple of k.

E.3.1 From LLL to Schnorr

In the LLL algorithm, vectors are considered two by two. At each loop iteration, the 2-dimensional lattice $L_i = [\pi_i(\mathbf{b}_i), \pi_i(\mathbf{b}_{i+1})]$, is partially reduced (through a swap) in order to decrease $\|\mathbf{b}_i^*\|$ by at least some geometric factor. When all L_i are almost reduced, every ratio $\|\mathbf{b}_i^*\| / \left\|\mathbf{b}_{i+1}^*\right\|$ is roughly less than $\gamma_2 = \sqrt{\frac{4}{3}}$, which is Siegel's condition [58].

Schnorr's semi block-$2k$-reduction is a polynomial-time block generalization of the LLL algorithm, where the vectors $\mathbf{b}_i^*$ are "replaced" by k-dimensional blocks $S_i = [\pi_{ik-k+1}(\mathbf{b}_{ik-k+1}), \ldots, \pi_{ik-k+1}(\mathbf{b}_{ik})]$

TAB. E.1 – Analogy between LLL and Schnorr's algorithm

LLL	**Schnorr's semi block-$2k$ reduction**
1: **while** $i \leq n$ **do**	1: **while** $i \leq n/k$ **do**
2: Size-reduce $\mathbf{b}_i$	2a: HKZ-reduce S_i, do the transformations on the basis vectors, not just on the projections 2b: Size-reduce $\mathbf{b}_{ik-k+1}, \ldots, \mathbf{b}_{ik}$.
3: $B' \leftarrow$copy of B 4: Try to decrease $\|\mathbf{b}_i^*\|$ in B': 4a: • by swap of $(\mathbf{b}_i, \mathbf{b}_{i+1})$	3: $B' \leftarrow$copy of B 4: Try to decrease $\mathrm{vol}(S_i)$ in B': 4a: • by swap of $(\mathbf{b}_{ik}, \mathbf{b}_{ik+1})$ 4b: • by HKZ-reducing L_i
5: **if** $\|\mathbf{b}_i^*\|$ can lose a factor δ **then** 6: • perform the changes on B 7: • $i \leftarrow \max(i-1, 1)$ 8: **else** $i \leftarrow i+1$	5: **if** $\mathrm{vol}(S_i)$ can lose a factor $\frac{1}{(1+\varepsilon)}$ **then** 6: • perform the changes on B 7: • $i \leftarrow \max(i-1, 1)$ 8: **else** $i \leftarrow i+1$
9: **endwhile**	9: **endwhile**

where $1 \leq i \leq \frac{n}{k}$. The analogue of the 2-dimensional L_i in LLL are the $2k$-dimensional large blocks $L_i = [\pi_{ik-k+1}(\mathbf{b}_{ik-k+1}), \ldots, \pi_{ik-k+1}(\mathbf{b}_{ik+k})]$ where $1 \leq i \leq \frac{n}{k} - 1$. The link between the small blocks $S_1, \ldots, S_{n/k}$ and the large blocks $L_1, \ldots, L_{n/k-1}$ is that S_i consists of the first k vectors of L_i, while S_{i+1} is the projection of the last k vectors of L_i over $\mathrm{span}(S_i)^{\perp}$. As a result, $\mathrm{vol}(L_i) = \mathrm{vol}(S_i) \times \mathrm{vol}(S_{i+1})$.

Formally, a basis is semi-block-$2k$-reduced if the following three conditions hold for some small $\varepsilon > 0$:

$$B \text{ is LLL-reduced} \tag{E.2}$$

$$\text{For all } 1 \leq i \leq \frac{n}{k}, \quad S_i \text{ is HKZ-reduced} \tag{E.3}$$

$$\text{For all } 1 \leq i < \frac{n}{k}, \quad \left(\frac{\mathrm{vol}(S_i)}{\mathrm{vol}(S_{i+1})}\right)^2 \leq (1+\varepsilon)\beta_k^k \tag{E.4}$$

Like in LLL, the large block L_i is reduced at each loop iteration in order to decrease $\mathrm{vol}(S_i)$ by a geometric factor $1/(1+\varepsilon)$. Note that $\mathrm{vol}(S_i)/\mathrm{vol}(S_{i+1})$ decreases by $1/(1+\varepsilon)^2$. By definition of β_k, this ratio can be made smaller than $\beta_k^{k/2}$ if L_i is HKZ-reduced. For this reason, condition (E.4) is a block generalization of Siegel's condition which can be fulfilled by an HKZ-reduction of L_i.

E.3.2 Complexity analysis

Each time a large block L_i is reduced, $\mathrm{vol}(S_i)$ decreases by a geometric factor $1/(1+\varepsilon)$ and since $\mathrm{vol}(L_i) = \mathrm{vol}(S_i) \times \mathrm{vol}(S_{i+1})$ remains constant, $\mathrm{vol}(S_{i+1})$ increases by the same factor. So the integer quantity $\prod_{i=1}^{n/k} \mathrm{vol}(S_i)^{2\left(\frac{n}{k}-i\right)}$ decreases by $1/(1+\varepsilon)^2$. This can occur at most a polynomial number of times : hence the complexity of the reduction is Poly(size of basis)*HKZ($2k$) where HKZ($2k$) is the complexity of a $2k$-dimensional HKZ reduction as seen in Section E.2.2. In order to ensure a polynomial complexity, it is necessary to keep $k \leq \log n / \log\log n$ or $k \leq \log n$ if we use the probabilistic AKS algorithm.

TAB. E.2 – Comparison between LLL and Schnorr's semi block-$2k$ reduction.

Algorithm	LLL	Schnorr's Semi-$2k$ reduction
Upper bound on$\|\mathbf{b}_1\| / \mathrm{vol}(L)^{\frac{1}{n}}$	$\approx \left(\frac{4}{3}\right)^{\frac{n}{4}}$	$\approx \beta_k^{\frac{n}{4k}}$
Upper bound on$\|\mathbf{b}_1\| / \lambda_1(L)$	$\approx \left(\frac{4}{3}\right)^{\frac{n}{2}}$	$\approx \beta_k^{\frac{n}{2k}}$
time	Poly(size of basis)	Poly(size of basis)*HKZ($2k$)
small block S_i	$\pi_i(\mathbf{b}_i) = \mathbf{b}_i^*$	$[\pi_{ik-k+1}(\mathbf{b}_{ik-k+1}), \ldots, \pi_{ik-k+1}(\mathbf{b}_{ik})]$
large block L_i	$[\pi_{i-1}(\mathbf{b}_{i-1}), \pi_{i-1}(\mathbf{b}_i)]$	$[\pi_{ik-2k+1}(\mathbf{b}_{ik-2k+1}), \ldots, \pi_{ik-2k+1}(\mathbf{b}_{ik})]$
size of small block	1	k
size of large block	2	$2k$
Quantity to upper bound	$\|\mathbf{b}_i^*\| / \left\|\mathbf{b}_{i+1}^*\right\|$	$\mathrm{vol}(S_i)/\mathrm{vol}(S_{i+1})$
Method	Size-reduce L_i and swap	HKZ reduce L_i
Potential	$\prod_{i=1}^{n} \|\mathbf{b}_i^*\|^{2(n-i)}$	$\prod_{i=1}^{\frac{n}{k}} \mathrm{vol}(S_i)^{2(\frac{n}{k}-i)}$

E.3.3 The Hermite factor of Schnorr's reduction

The Hermite factor of a semi block-$2k$-reduced basis depends mostly on Condition (E.4), which implies that $\mathrm{vol}(S_1) \lessapprox \beta_k^{\frac{n}{4}} \mathrm{vol}(L)^{k/n}$ because $\mathrm{vol}(L) = \prod_{i=1}^{n/k} \mathrm{vol}(S_i)$. If the first vector $\mathbf{b}_1$ is the shortest vector of S_1 (which is implied by (E.3)), then $\|\mathbf{b}_1\| \leq \sqrt{\gamma_k}\mathrm{vol}(S_1)^{\frac{1}{k}}$ by definition of Hermite's constant, and therefore :

$$\|\mathbf{b}_1\| / \mathrm{vol}(L)^{1/n} \lessapprox \sqrt{\gamma_k}\beta_k^{\frac{n}{4k}}$$

E.3.4 The approximation factor of Schnorr's reduction

If only condition (E.4) holds, even if $\mathbf{b}_1$ is the shortest vector of S_1, its norm can be arbitrarily far from the first minimum of L. Indeed, consider for instance the 6-dimensional lattice generated by $\mathrm{Diag}(1, 1, 1, 1, \varepsilon, \frac{1}{\varepsilon})$, and a blocksize $k = 3$. Then the first block S_1 is the identity and is therefore HKZ-reduced. The volume of the two blocks S_1 and S_2 is 1, thus condition (E.4) holds. But the norm of the first vector ($\|\mathbf{b}_1\| = 1$) is arbitrarily far from the shortest vector $\|\mathbf{b}_5\| = \varepsilon$.

Compared to Hermite's factor, we require additionally that every block S_k is reduced (which follows from condition (E.2)) to bound the approximation factor.

Using (E.1), there exists an index p such that $\left\|\mathbf{b}_p^*\right\| \leq \lambda_1(L)$. Let $a = \lfloor (p-1)/k \rfloor$, (so that the position p is inside the block S_{a+1}). Since B is LLL-reduced, $\mathrm{vol}(S_a) \lessapprox \frac{4}{3}^{\frac{(3k-1)k}{4}} \lambda_1(L)^k$, so the approximation factor is bounded by :

$$\|\mathbf{b}_1\| / \lambda_1(L) \lessapprox \sqrt{\gamma_k} \frac{4}{3}^{\frac{(3k-1)}{4}} \beta_k^{\frac{n/k-2}{2}}$$

Note however that Schnorr proved in [302] that Condition (E.3) allows to decrease the term $(4/3)^{(3k-1)/4}$ to $O\left(k^{2+\ln k}\right)$.

E.4 Rankin's constant and Schnorr's algorithm

E.4.1 Rankin's constant

If L is a n-dimensional lattice and $1 \leq m \leq n$, the Rankin invariant $\gamma_{n,m}(L)$ is defined as (cf. [292]) :

$$\gamma_{n,m}(L) = \min_{\substack{\mathbf{x}_1,...,\mathbf{x}_m \in L \\ \mathrm{vol}(\mathbf{x}_1,...,\mathbf{x}_m) \neq 0}} \left(\frac{\mathrm{vol}(\mathbf{x}_1,...,\mathbf{x}_m)}{\mathrm{vol}(L)^{m/n}} \right)^2$$

which can be rewritten as :

$$\gamma_{n,m}(L) = \min_{\substack{S \text{ sublattice of } L \\ \dim S = m}} \left(\frac{\mathrm{vol}(S)}{\mathrm{vol}(L)^{m/n}} \right)^2$$

Rankin's constant is the maximum $\gamma_{n,m} = \max \gamma_{n,m}(L)$ over all n-dimensional lattices. Clearly, $\gamma_{n,n}(L) = 1$ and $\gamma_{n,1}(L) = \gamma_n(L)$, so $\gamma_{n,n} = 1$ and $\gamma_{n,1} = \gamma_n$. Rankin's constants satisfy the following three relations, which are proved in [220, 292] :

$$\forall n \in \mathbb{N},\ \gamma_{n,1} = \gamma_n \tag{E.5}$$

$$\forall n, m \text{ with } m < n\ \gamma_{n,m} = \gamma_{n,n-m} \tag{E.6}$$

$$\forall r \in [m+1, n-1],\ \gamma_{n,m} \leq \gamma_{r,m}(\gamma_{n,r})^{m/r} \tag{E.7}$$

The only known values of Rankin's constants are $\gamma_{4,2} = \frac{3}{2}$, which is reached for the $\mathbb{D}_4$ lattice, and those corresponding to the nine Hermite constants known. In the definition of $\gamma_{n,m}(L)$, the minimum is taken over sets of m linearly independent vectors of L, but we may restrict the definition to primitive sets of L or pure sublattices of L, since for any sublattice S of L, there exists a pure sublattice S_1 of L with $\mathrm{span}(S) = \mathrm{span}(S_1)$ and $\mathrm{vol}(S)/\mathrm{vol}(S_1) = [S : S_1]$. If $\mathrm{vol}(S)$ is minimal, then $[S : S_1] = 1$ so $S = S_1$ is pure.

E.4.2 Relation between Rankin's constant and Schnorr's constant

Theorem E.23 *For all $k \geq 1$, $(\gamma_{2k,k})^{2/k} \leq \beta_k$.*

Proof. Let $B = [\mathbf{b}_1, \ldots, \mathbf{b}_{2k}]$ be a HKZ-reduced basis of a lattice L, and let $h(B) = \left(\|\mathbf{b}_1^*\|^2 \times \cdots \times \|\mathbf{b}_k^*\|^2 \right) / \left(\|\mathbf{b}_{k+1}^*\|^2 \times \cdots \times \|\mathbf{b}_{2k}^*\|^2 \right)$. By definition of β_k, $h(B) \leq \beta_k^k$. On the other hand, $h(B)$ is also equal to $\left(\mathrm{vol}(\mathbf{b}_1, \ldots, \mathbf{b}_k)/\mathrm{vol}(L)^{1/2}\right)^4$ and therefore : $\gamma_{2k,k}^2(L) \leq h(B)$. Thus $(\gamma_{2k,k}(L))^{2/k} \leq \beta_k$, which completes the proof. □

E.4.3 Improving the upper bound on Schnorr's Constant

The key result of this section is :

Theorem E.24 *For all $k \geq 2$, Schnorr's constant β_k satisfies : $\beta_k \leq \left(1 + \frac{k}{2}\right)^{2\ln 2 + \frac{1}{k}}$. Asymptotically it satisfies $\beta_k \leq \frac{1}{10} k^{2\ln 2}$.*

Without any change to Schnorr's algorithm, we deduce a much better quality for the output basis than with the former bound $\beta_k \leq 4k^2$, because both the exponent $2\ln 2 \approx 1.386$ is much lower than 2, and the coefficient $1/2^{2\ln 2}$ is about 10 times lower than 4. The bounds on the approximation factor

and Hermite's factor of Schnorr's algorithm can be raised to the power $\ln 2 \approx 0.69$. The proof uses an easy bound mentioned by Schnorr in [302] :

$$\beta_k \leq \prod_{j=0}^{k-1} \gamma_{k+j+1}^{2/(k+j)} \tag{E.8}$$

Numerically, it can be verified that the product (E.8) is $\leq k^{1.1}$ for all $k \leq 100$ (see Figure E.2 page 204). The bound $\gamma_j \leq 1 + \frac{j}{4}$ combined with an upper bound of the total exponents prove Theorem E.24 for all k (see Section E.6.1 in the appendix).

Surprisingly, we do not know a better upper bound on $(\gamma_{2k,k})^{2/k}$ than that of Theorem E.24. The inequality (E.7) leads exactly to the same bound for Rankin's constant.

E.4.4 A lower bound on Rankin's constant

In [7], Ajtai showed that $\beta_k \geq k^{\varepsilon}$ for small size of blocks and for some $\varepsilon > 0$, and presented worst cases for Schnorr's algorithm, which implies that the reduction power of semi block $2k$-reduction is limited. The following result proves an explicit lower bound on Rankin's constant, which suggests (but does not prove) that the approximation factor of any block-reduction algorithm (including Schnorr's semi block $2k$-reduction) based on the LLL strategy is limited.

Theorem E.25 *Rankin's constant satisfies* $(\gamma_{2k,k})^{\frac{2}{k}} \geq \frac{k}{12}$ *for all* $k \geq 1$.

This lower bound also applies to Schnorr's constant β_k because of Theorem E.23. Theorem E.25 is mainly based on the following lower bound for Rankin's constant proved in [342, 39] as a generalization of Minkowski-Hlawka's theorem :

$$\gamma_{n,m} \geq \left(n \frac{\prod_{j=n-m+1}^{n} Z(j)}{\prod_{j=2}^{m} Z(j)} \right)^{\frac{2}{n}}$$

where $Z(j) = \zeta(j)\Gamma(\frac{j}{2})/\pi^{\frac{j}{2}}$, $\Gamma(x) = \int_0^{\infty} t^{x-1}e^{-t} \cdot dt$ and ζ is Riemann's zeta function : $\zeta(j) = \sum_{p=1}^{\infty} p^{-j}$. As an application, for $k < 100$, it can be verified numerically that $(\gamma_{2k,k})^{\frac{2}{k}} \geq \frac{k}{9}$. More generally, we first bound $\ln Z(j)$, and we compare it to an integral to get the expected lower bound. The full proof of Theorem E.25 is given in Section E.6.2 of the Appendix.

E.5 Improving Schnorr's algorithm

The main subroutine in Schnorr's algorithm tries to solve the following problem : given a $2k$-dimensional lattice L, find a basis $[\mathbf{b}_1, \ldots, \mathbf{b}_{2k}]$ of L such that the two k-dimensional blocks $S_1 = L(\mathbf{b}_1, \ldots, \mathbf{b}_k)$ and $S_2 = \pi_{k+1}(L)$ minimize $\mathrm{vol}(S_1)$, because $\mathrm{vol}(S_1)/\mathrm{vol}(S_2) = \mathrm{vol}(S_1)^2/\mathrm{vol}(L)$ where $\mathrm{vol}(L)$ does not change. In Schnorr's algorithm, the quality of the output basis (which was expressed as a function of β_k in Sections E.3.3 and E.3.4) essentially depends on the upper bound that can be achieved on the ratio $\mathrm{vol}(S_1)/\mathrm{vol}(S_2)$.

E.5.1 The smallest volume problem

Rankin's constant and Schnorr's algorithm suggest the *smallest volume problem* : given a n-dimensional lattice L and an integer m such that $1 \leq m \leq n$, find an m-dimensional sublattice S of L such that $\mathrm{vol}(S)$ is minimal, that is, $\mathrm{vol}(S)/\mathrm{vol}(L)^{m/n} = \sqrt{\gamma_{n,m}(L)}$.

If $m = 1$, the problem is simply the shortest vector problem (SVP). If $m = n - 1$, the problem is equivalent to the shortest vector problem in the dual lattice. When $(n, m) = (2k, k)$, we call this

problem the *half-volume problem*. For any $m \leq n$, the minimality of the volume implies that any solution to this problem is a pure sublattice of L, so one way to solve this problem is to find a basis $[\mathbf{b}_1, \ldots, \mathbf{b}_n]$ such that $\mathrm{vol}\,(L(\mathbf{b}_1, \ldots, \mathbf{b}_m))$ is minimal.

We say that a basis of a n-dimensional lattice L is *m-Rankin reduced* if its first m vectors solve the smallest volume problem. Note that this is not exactly a basis reduction problem, as any notion of reduction of the basis of S is irrelevant. The only thing that matters is to minimize the volume of S. If we apply the LLL algorithm on a Rankin-reduced basis, the volume of the first m vectors can never increase : this means that LLL swaps never involve the pair $(m, m+1)$, and therefore the output basis is both LLL-reduced and Rankin-reduced. We thus have proved the following lemma :

Lemma E.34 *Let L be a n-dimensional sublattice and $1 \leq m \leq n$. There exists an LLL-reduced basis of L which is m-Rankin-reduced.*

Since the number of LLL reduced bases can be bounded independently of the lattice (see [58] because LLL-reduction implies Siegel reduction), the smallest volume problem can be solved by a gigantic exhaustive search (which is constant in fixed dimension though).

E.5.2 Block-Rankin reduction

A basis is *$2k$-Block-Rankin reduced* with factor $\delta \in [\frac{1}{2}; 1[$ if it is LLL-reduced with factor $(\frac{1}{2}, \delta)$ and all the blocks S_i and L_i defined as in Section E.3 satisfy : $\mathrm{vol}(S_i)/\mathrm{vol}(S_{i+1}) \leq \frac{1}{\delta}\gamma_{2k,k}(L_i)$. Compared to Schnorr's semi block-$2k$ reduction, this reduction notion enables to replace β_k in the bounds of the approximation factor and Hermite's factor by $\gamma_{2k,k}^{2/k}$.

Assume that an algorithm to k-Rankin-reduce a $2k$-dimensional basis is available. Then it is easy to see that Algorithm 14, inspired from LLL and Schnorr's semi block-$2k$ reduction, achieves block-Rankin reduction using a polynomial number of calls to the Rankin subroutine.

Algorithm 14 2k-block-Rankin reduction

Input : A basis $B = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ of a lattice and $\delta \in [\frac{1}{2}; 1[$
Output : A semi block $2k$-reduced basis.

$i \leftarrow 1$;
while $i \leq n/k$ **do**
  LLL-reduce S_i with factor δ, do the transformations on the basis vectors, not just on their projections
  return B **if** $i = n/k$.
  $B_{\mathrm{tmp}} \leftarrow B$; k-Rankin reduce L_i in Btmp
  if $\mathrm{vol}(S_i)$ in $B_{\mathrm{tmp}} \leq \delta \mathrm{vol}(S_i)$ in B **then**
    $B \leftarrow B_{\mathrm{tmp}}$; $i \leftarrow i - 1$
  else
    $i \leftarrow i + 1$
  end if
end while

E.5.3 The 4-dimensional case

Here, we study Rankin-reduction for $(n, m) = (4, 2)$. We first notice that HKZ reduction does not necessarily solve the half-volume problem. Consider indeed the following HKZ-reduced row basis :

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1+\varepsilon & 0 \\ 0 & 0 & \frac{1+\varepsilon}{2} & \frac{\sqrt{3}}{2}(1+\varepsilon) \end{bmatrix}$$

The volume ratio $\frac{\|\mathbf{b}_1^*\|\|\mathbf{b}_2^*\|}{\|\mathbf{b}_3^*\|\|\mathbf{b}_4^*\|}$ is equal to $\sqrt{\frac{4}{3}}$. If we swap the two 2-blocks, the new basis is no longer HKZ-reduced, but the ratio decreases to almost $\sqrt{\frac{3}{4}}$. This example can easily be generalized to any even dimension, which gives an infinite family of HKZ bases which do not reach the minimal half-volume.

However, the following lemma shows that Algorithm 15 can efficiently solve the half-volume problem in dimension 4, given as input an HKZ basis :

Lemma E.35 *Let* $(\mathbf{b}_1, ..., \mathbf{b}_4)$ *be an HKZ-reduced basis of a lattice* L*. To simplify notations, let* λ_1 *and* λ_2 *denote respectively* $\lambda_1(L)$ *and* $\lambda_2(L)$*. For all* $\mathbf{c}_1$ *and* $\mathbf{c}_2$ *in* L *such that* $\mathrm{vol}(\mathbf{c}_1, \mathbf{c_2}) \leq \mathrm{vol}(\mathbf{b_1}, \mathbf{b_2})$ *and* $(\mathbf{c}_1, \mathbf{c_2})$ *is reduced :* $\|\mathbf{c}_1\| \leq \|\mathbf{c}_2\|$ *and* $\|\mathbf{c}_1^*\|^2 \leq \frac{4}{3}\|\mathbf{c}_2^*\|^2$*.*

1. *Then* $\mathbf{c}_1$ *satisfies :* $\lambda_1^2 \leq \|\mathbf{c}_1\|^2 \leq \frac{4}{3}\lambda_1^2$*.*
2. *If* $\lambda_2 > \sqrt{\frac{4}{3}}\lambda_1$*, then* $\mathrm{vol}(\mathbf{c}_1, \mathbf{c}_2) = \mathrm{vol}(\mathbf{b}_1, \mathbf{b}_2)$ *given by HKZ reduction.*
3. *Otherwise* $\mathbf{c}_2$ *satisfies*$\|\mathbf{c}_2\|^2 \leq (\frac{4}{3}\lambda_1)^2$*.*

Proof. Because $\mathbf{c}_1$ belongs to L, $\|\mathbf{c}_1\| \geq \lambda_1$. Since $(\mathbf{b}_1, ..., \mathbf{b}_4)$ is an HKZ basis, the first vector is a shortest vector : $\|\mathbf{b}_1\| = \lambda_1$ and the second vector satisfies $\|\mathbf{b}_2^*\| \leq \lambda_2$, so $\mathrm{vol}(\mathbf{b}_1, \mathbf{b}_2) \leq \lambda_1\lambda_2$. We also know that $\mathrm{vol}(\mathbf{c}_1, \mathbf{c}_2) = \|\mathbf{c}_1\| \cdot \|\mathbf{c}_2\| \cdot \sin(\mathbf{c}_1, \mathbf{c}_2) \geq \frac{\sqrt{3}}{2}\|\mathbf{c}_1\| \cdot \|\mathbf{c}_2\|$ because $[\mathbf{c}_1, \mathbf{c}_2]$ is reduced. Since we have chosen $\|\mathbf{c}_1\| \leq \|\mathbf{c}_2\|$, then $\|\mathbf{c}_2\| \geq \lambda_2$. Thus $\lambda_1\lambda_2 \geq \frac{\sqrt{3}}{2}\|\mathbf{c}_1\| \cdot \lambda_2$, and $\|\mathbf{c}_1\|^2 \leq \frac{4}{3}\lambda_1^2$. If furthermore $\lambda_2 > \sqrt{\frac{4}{3}}\lambda_1$, then necessarily $\mathbf{c_1} = \pm\mathbf{b_1}$, then the HKZ reduction implies the minimality of $\mathrm{vol}(\mathbf{b_1}, \mathbf{b_2})$. If $\lambda_2 \leq \sqrt{\frac{4}{3}}\lambda_1$, then $\mathrm{vol}(\mathbf{c_1}, \mathbf{c}_2)^2 = \|\mathbf{c}_1\|^2 \cdot \|\mathbf{c}_2^*\|^2 \leq (\lambda_1\lambda_2)^2$, so $\|\mathbf{c}_2^*\|^2 \leq \lambda_2^2 \leq \frac{4}{3}\lambda_1^2$. And we also have $\|\mathbf{c}_2^*\|^2 \geq \frac{3}{4}\|\mathbf{c}_1^*\|^2$. □

Algorithm 15 4-dimensional Rankin-reduction

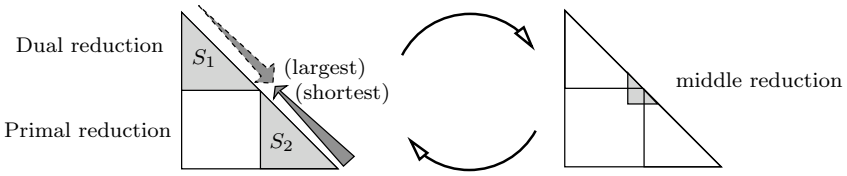
Input : An HKZ reduced basis $[\mathbf{b}_1, \ldots, \mathbf{b}_4]$ of a 4-dim lattice
Output : A Rankin-reduced basis $[\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{c}_4]$ minimizing $\mathrm{vol}(\mathbf{c}_1, \mathbf{c}_2)$

1: **if** $\|\mathbf{b}_2^*\| > \sqrt{\frac{4}{3}}\|\mathbf{b}_1\|$ **then**
2: **return** $(\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \mathbf{b}_4)$
3: **end if**
4: $(\mathbf{u}, \mathbf{v}) \leftarrow (\mathbf{b}_1, \mathbf{b}_2)$
5: **for** each lattice vector $\mathbf{c}_1$ shorter than $\sqrt{\frac{4}{3}}\|\mathbf{b}_1\|$ **do**
6: find the shortest vector $\mathbf{c}_2$ in the lattice projected over $\mathbf{c}_1^\perp$ (We can limit the enumeration to $\|\mathbf{c}_2\|$ lower than $\frac{\mathrm{vol}(\mathbf{b}_1, \mathbf{b}_2)}{\|\mathbf{c}_1\|}$).
7: **if** $\mathrm{vol}(\mathbf{c}_1, \mathbf{c}_2) < \mathrm{vol}(\mathbf{u}, \mathbf{v})$ **then** $(\mathbf{u}, \mathbf{v}) \leftarrow (\mathbf{c}_1, \mathbf{c}_2)$
8: **end for**
9: compute $\mathbf{c}_3$ and $\mathbf{c}_4$ a reduced basis of the lattice projected over $(\mathbf{u}, \mathbf{v})^\perp$
10: **return** $(\mathbf{u}, \mathbf{v}, \mathbf{c}_3, \mathbf{c}_4)$

Because the input basis is HKZ-reduced, it is easy to see that the number of vectors $\mathbf{c}_1$ enumerated in Algorithm 15 is bounded by a constant. It follows that the cost of Algorithm 15 is at most a constant times more expensive than a HKZ reduction of a 4-dimensional lattice.

If we plug Algorithm 15 into Algorithm 14, we obtain a polynomial-time reduction algorithm whose provable quality is a bit better than Schnorr's semi block-4 reduction : namely, the constant $\beta_2 \leq \gamma_4^{2/3}\gamma_3 = 2^{2/3} \approx 1.587$ in the approximation factor and the Hermite factor is replaced by the potentially smaller constant $\gamma_{4,2} = 3/2$. On the other hand, both algorithms only apply exhaustive search in dimension 4.

FIG. E.1 – Transference reduction



E.5.4 Higher blocksizes

The 4-dimensional case suggests two potential improvements over Schnorr's semi block $2k$-algorithm :

- If the half-volume problem can be solved in roughly the same time (or less) than a full $2k$-HKZ reduction, then Algorithm 14 would give potentially better approximation factors at the same cost.
- If one can approximate the half-volume problem in much less time than a full $2k$-HKZ reduction, we may still obtain good approximation factors in much less time than semi block $2k$-reduction, by plugging the approximation algorithm in place of Rankin reduction in Algorithm 14.

However, we do not know how to solve the half-volume problem exactly in dimension higher than 4, without using a gigantic exhaustive search. Perhaps a good approximation can be found in reasonable blocksize, by sampling short (but not necessarily shortest) lattice vectors, and testing random combinations of such vectors.

We now present an approximation algorithm for the half volume problem, which we call *transference reduction*. Transference reduction achieves a volume ratio lower than $\frac{1}{95}k^2$ in a $2k$-dimensional lattice by making only $O(k)$ calls to a k-dimensional exhaustive search, which is thus cheaper than a full $2k$-HKZ reduction. Note that $2k$-HKZ reduction achieves a smaller volume ratio using a $2k$-dimensional exhaustive search. Let $(\mathbf{b}_1, \ldots, \mathbf{b}_{2k})$ be a basis of a $2k$-dimensional lattice. The idea of the algorithm is to perform exhaustive searches in the two halves of the basis in order to find a pair of vectors which can be highly reduced. The reduction of this pair of vectors happens in the middle of the basis so that the first half-volume decreases.

As in the previous sections, we call $S_1 = L(\mathbf{b}_1, \ldots \mathbf{b}_k)$ and $S_2 = L(\pi_{k+1}(\mathbf{b}_{k+1}), \ldots, \pi_{k+1}(\mathbf{b}_{2k}))$. Using an exhaustive search, a shortest vector of S_2 is brought on the $k+1$-th position in order to make $\left\|\mathbf{b}^*_{k+1}\right\|^2 \leq \gamma_k \mathrm{vol}(S_2)^{2/k}$. The algorithm used to perform this exhaustive search in dimension k in a projected lattice is classical. Then a second exhaustive search brings a vector of S_1 maximizing $\|\mathbf{b}^*_k\|$ on the k-th position.

Lemma E.36 *Finding a basis $(\mathbf{b}_1, \ldots, \mathbf{b}_k)$ of a k-dimensional lattice S maximizing $\|\mathbf{b}^*_k\|$ reduces to finding a shortest vector of the dual lattice $S^\times$.*

Proof. The vector $\mathbf{u} = \mathbf{b}^*_k / \|\mathbf{b}^*_k\|^2$ is the last vector of the dual basis. Indeed, $\langle \mathbf{u}, \mathbf{b}_i \rangle = 0$ for $i = 1..k-1$ and $\langle \mathbf{u}, \mathbf{b}_k \rangle = 1$. If $\|\mathbf{b}^*_k\|$ is maximal, then $\mathbf{u}$ is minimal. So a simple reduction is to find a shortest vector $\mathbf{u}_k$ of the dual $S^\times$, extend it into a basis $U = (\mathbf{u}_1, \ldots, \mathbf{u}_k)$ of $S^\times$ and return the dual $U^{-t} = (\mathbf{b}_1, \ldots, \mathbf{b}_k)$. □

After maximizing $\|\mathbf{b}^*_k\|$, Hermite's inequality in the reversed dual of S_1 implies that $1/\|\mathbf{b}^*_k\|^2 \leq \gamma_k / \mathrm{vol}(S_1)^{2/k}$. At this point, the ratio $(\mathrm{vol}(S_1)/\mathrm{vol}(S_2))^{2/k}$ is lower than $\gamma_k^2 \|\mathbf{b}^*_k\|^2 / \|\mathbf{b}_{k+1}\|^2$. If the middle-vectors pair $(\pi_k(\mathbf{b}_k), \pi_k(\mathbf{b}_{k+1}))$ does not satisfy Lovász condition, then it is fully reduced and the algorithm starts over from the beginning. The only step which can change the ratio $\mathrm{vol}(S_1)/\mathrm{vol}(S_2)$ is the middle-vectors reduction, in which case it drops by a geometric factor. Hence the number of swaps in the middle is at most linear in the dimension and the size of the basis. In the end, the ratio $(\mathrm{vol}(S_1)/\mathrm{vol}(S_2))^{2/k}$ is lower than $\frac{4}{3}\gamma_k^2$ (or $\frac{1}{12}k^2$).

Tab. E.3 – Comparison between and Schnorr's semi block-$2k$ reduction and Transference reduction. (Here, SVP$(k+1)$ denotes the cost of finding the shortest lattice vector in dimension $k+1$).

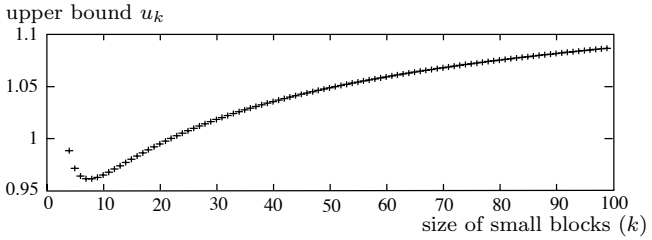
Algorithm	Semi-$2k$ reduction	Transference reduction
Upper bound on$\|\mathbf{b}_1\|/\mathrm{vol}(L)^{\frac{1}{n}}$	$\approx \beta_k^{\frac{n}{4k}} \lessapprox k^{n\ln 2/2k}$	$\approx \gamma_k^{\frac{n}{2k}} \lessapprox k^{n/2k}$
Upper bound on $\|\mathbf{b}_1\|/\lambda_1(L)$	$\approx \beta_k^{\frac{n}{2k}} \lessapprox k^{n\ln 2/k}$	$\approx \gamma_k^{\frac{n}{k}} \lessapprox k^{n/k}$
Cost	Poly(size of basis) $\times$ HKZ$(2k)$	Poly(size of basis) $\times k$*SVP$(k+1)$
Reduction of large blocks	HKZ-reduction	Transference reduction

The constant $4/3$ in this ratio can be reduced to almost 1 by adding further reduction conditions. Let $\hat{S}_1 = L(\mathbf{b}_1,\ldots,\mathbf{b}_{k+1})$ and $\hat{S}_2 = L(\pi_k(\mathbf{b}_k),\ldots,\pi_k(\mathbf{b}_{2k}))$ the widened blocks of S_1, S_2 and $\delta : \frac{1}{2} \leq \delta < 1$ a relaxing parameter. After minimizing $\left\|\mathbf{b}_{k+1}^*\right\|$ in S_2 and maximizing $\|\mathbf{b}_k^*\|$ in S_1 the following steps are performed : Using the third exhaustive search, a shortest vector of $\hat{S}_2$ is found. Only if the squared size of this shortest vector is smaller than $\delta\|\mathbf{b}_k^*\|^2$ this vector is brought on the k-th position and the algorithm starts over with minimizing $\left\|\mathbf{b}_{k+1}^*\right\|$ in S_2 and maximizing $\|\mathbf{b}_k^*\|$ in S_1. Otherwise the fourth exhaustive search in the dual of $\hat{S}_1$ checks if $\left\|\mathbf{b}_{k+1}^*\right\|^2$ approximates the maximized solution by the factor δ. If this condition does not hold, $\mathbf{b}_{k+1}$ is replaced by the maximized solution and the algorithm starts over from the beginning. Each of these two reduction steps decrease $\mathrm{vol}(S_1)^2$ by the factor δ, therefore the number of steps is still bounded by $O(k)$. In case both conditions hold the algorithm stops. For these new conditions we can again apply Hermite's inequality resulting in $\delta\cdot\|\mathbf{b}_k^*\|^2 \leq \gamma_{k+1}\mathrm{vol}(\hat{S}_2)^{2/(k+1)}$ and $\delta\cdot 1/\left\|\mathbf{b}_{k+1}^*\right\|^2 \leq \gamma_{k+1}/\mathrm{vol}(\hat{S}_1)^{2/(k+1)}$. It follows :

$$\left(\mathrm{vol}(\hat{S}_1)/\mathrm{vol}(\hat{S}_2)\right)^{2/(k+1)} \leq \gamma_{k+1}^2/\delta^2\cdot\left\|\mathbf{b}_{k+1}^*\right\|^2/\|\mathbf{b}_k^*\|^2 .$$

Because of $\mathrm{vol}(\hat{S}_1) = \mathrm{vol}(S_1)\cdot\left\|\mathbf{b}_{k+1}^*\right\|$, $\mathrm{vol}(\hat{S}_2) = \|\mathbf{b}_k^*\|\cdot\mathrm{vol}(S_2)$ this inequality can be transformed to $(\mathrm{vol}(S_1)/\mathrm{vol}(S_2))^{2/k} \leq (\frac{\gamma_{k+1}}{\delta})^{2\frac{k+1}{k}}\cdot\left\|\mathbf{b}_{k+1}^*\right\|^2/\|\mathbf{b}_k^*\|^2$. Combining this with inequality $(\mathrm{vol}(S_1)/\mathrm{vol}(S_2))^{2/k} \leq \gamma_k^2\|\mathbf{b}_k^*\|^2/\left\|\mathbf{b}_{k+1}^*\right\|^2$ obtained after the first two exhaustive searches, the ratio $(\mathrm{vol}(S_1)/\mathrm{vol}(S_2))^{2/k}$ is lower than $\gamma_k(\gamma_{k+1}/\delta)^{(k+1)/k}$ (or $\gamma_k^2(1+\varepsilon)$ with small ε if δ is near by 1). Asymptotically, Hermite's constants satisfy $\gamma_k \leq \frac{1.744k}{2\pi e}(1+\circ(1))$, so this extended transference reduction provides a ratio $(\mathrm{vol}(S_1)/\mathrm{vol}(S_2))^{2/k}$ lower than $\frac{1}{95}k^2$.

If we use transference reduction instead of Rankin reduction in Algorithm 14, we obtain a reduction algorithm making only $(k+1)$-dimensional exhaustive searches of a shortest vector, and providing an Hermite factor $\|\mathbf{b}_1\|/\mathrm{vol}(L)^{1/n} \lessapprox \sqrt{\gamma_k}\gamma_k^{n/2k}$ and an approximation factor $\|\mathbf{b}_1\|/\lambda_1(L) \lessapprox \gamma_k^{-\frac{3}{2}}\frac{4}{3}^{\frac{(3k-1)}{4}}\gamma_k^{\frac{n}{k}}$. These factors are asymptotically not as good as in the semi block-$2k$ reduction, but the exhaustive searches of transference reduction are much cheaper and thus allow to use a larger k. Interestingly the Hermite factor is essentially $\gamma_k^{n/2k}$, which means that the resulting algorithm may roughly be viewed as an algorithmic version of Mordell's inequality [220] : $\gamma_n \leq \gamma_k^{\frac{n-1}{k-1}}$. Similarly, LLL could be viewed as the algorithmic version of Hermite's inequality $\gamma_n \leq \gamma_2^{n-1}$, which is the particular case $k=2$ of Mordell's inequality.



This curve shows the numerical upper bound u_k of $\ln \prod_{j=0}^{k-1} \gamma_{k+j+1}^{2/(k+j)} / \ln k$, obtained by using the exact values of Hermite's constant γ_i for $1 \leq i \leq 8$ and $i = 24$, and the bound $\gamma_i \leq 1 + \frac{i}{4}$ elsewhere. Thus $u_k \leq 1.1$ for $1 \leq k \leq 100$.

FIG. E.2 – "Exponents" of the upper-bound on β_k

E.6 Appendix

E.6.1 Proof of Theorem E.24

The right-hand product (E.8) of Hermite's constants can be bounded using the absolute upper bound $\gamma_j \leq (1+j)/4$ by : $\beta_k^k \leq \prod_{j=0}^{k-1} (1 + \frac{k+j+1}{4})^{\frac{2k}{k+j}}$.

$$\beta_k^k \leq \left(1 + \frac{k}{2}\right)^{\sum_{j=0}^{k-1} \frac{2}{1+j/k}} \cdot \prod_{j=0}^{k-1} \left(\frac{1 + \frac{k+j+1}{4}}{1 + \frac{k}{2}}\right)^{\frac{k}{k+j}}.$$

The first sum can be compared with an integral :

$$\frac{1}{k} \sum_{j=0}^{k-1} \frac{2}{1 + j/k} \leq \frac{1}{k} + 2 \int_0^1 \frac{1}{1+x} dx,$$

and the last product is always smaller than 1 : more precisely, its asymptotical equivalent is

$$\exp\left(-k(\ln 2/2)^2\right) \approx 0.887^k.$$

Hence we obtain the absolute upper bound : $\beta_k^k \leq (1 + \frac{k}{2})^{1+2k \cdot \ln 2}$ or $\beta_k \leq (1 + \frac{k}{2})^{1/k + 2 \ln 2}$.

If we use the best asymptotical bound known for Hermite's constant $\gamma_j \leq \frac{1.744n}{2\pi e}(1 + \circ(1))$, we obtain using the same argument, the asymptotical upper bound :

$$\beta_k \leq \exp(-k(\ln 2/2)^2) \left(\frac{1.744k}{\pi e}\right)^{1/k + 2 \ln 2} \leq \frac{1}{10} k^{2 \ln 2}.$$

E.6.2 Proof of Theorem E.25

The Stirling equivalent of Γ satisfies $0 \leq \ln(\Gamma(x+1)) - \ln\left((\frac{x}{e})^x \sqrt{2\pi x}\right) \leq \frac{K}{x}$ where $K < 0.0835$ is a constant. Since the function $k \to k^{-j}$ is decreasing for $j \geq 2$, we may compare its integral with ζ, and we deduce the following bound :

$$\zeta(j) \leq 1 + \frac{1}{2^j} + \frac{1}{(j+1)2^{j+1}}$$

Combining these two relations, we obtain the following upper bound for $Z(j)$:

$$\ln(Z(j)) \quad \leq \quad \frac{j}{2} \ln \frac{j}{2} - \frac{j}{2}(\ln \pi + 1) + \rho(j)$$

where $\rho(j) = \left(1 - \ln\left(\frac{j}{2} - 1\right) + \frac{1}{2}\ln\left(2\pi\left(\frac{j}{2} - 1\right)\right) + \frac{K}{(\frac{j}{2}-1)}\right) + \left(\frac{1}{2^j} + \frac{1}{(j+1)2^{j+1}}\right)$. For $j > 13$, we have $\rho(j) < 0$, therefore it can be removed from the upper bound.

$$\forall j \geq 13,\ \ln(Z(j)) \leq \frac{j}{2}\ln\frac{j}{2} - \frac{j}{2}(\ln\pi + 1)$$

The lower bound is a consequence of Stirling's formula and the relation $1 \leq \zeta(j)$.

$$\forall j \geq 13,\ \left(\frac{j}{2} - 1\right)\ln\left(\frac{j}{2} - 1\right) - \frac{j}{2}(\ln(\pi) + 1) \leq \ln(Z(j)) \leq \frac{j}{2}\ln\frac{j}{2} - \frac{j}{2}(\ln\pi + 1)$$

We now use the upper bound and an integral to bound the denominator $\prod_{j=2}^{k} Z(j)$:

$$\begin{aligned}\sum_{j=2}^{k}\ln(Z(j)) &\leq \sum_{j=2}^{13}\ln(Z(j)) + \int_{14}^{k+1}\left(\frac{t}{2}\ln\frac{t}{2} - \frac{t}{2}(\ln\pi + 1)\right)dt \\ &\leq \frac{(k+1)^2}{4}\left(\ln(k+1) - \ln\pi - \frac{1}{2} - \ln 2\right) + c\end{aligned}$$

where $c = \sum_{j=2}^{13}\ln(Z(j)) - \frac{225}{4}(3\ln 2 - \ln\pi - \frac{1}{2} - \ln 2)$. And we apply the lower bound on the numerator $\prod_{j=n-m+1}^{n} Z(j)$:

$$\begin{aligned}\sum_{j=k+1}^{2k}\ln(Z(j)) &\geq \int_{k}^{2k}\left(\left(\frac{t}{2} - 1\right)\ln\left(\frac{t}{2} - 1\right) - \frac{t}{2}(\ln(\pi) + 1)\right)dt \\ &\geq k^2\left(\ln(k-1) - \frac{1}{4}\ln(k-2) + \frac{\ln 2}{4} - \frac{9}{8} - \frac{3}{4}\ln\pi\right) + r(k)\end{aligned}$$

where $r(k) = k\ln(k-2) - 2\ln(k-1) - \ln 2 + \frac{1}{2}k + \ln 2 - \ln(k-2) + \ln(k-1)$ is equivalent to $r(k) \sim -k \cdot \ln k$. Finally, we obtain a lower bound for $\gamma_{2k,k}$:

$$\ln\gamma_{2k,k}^{k} \geq \frac{k^2}{2}\ln(k) + \left(\frac{\ln 2}{2} - 1 - \frac{\ln\pi}{2}\right)k^2 + s(k)$$

where $s(k) = r(k) - k^2\left(\ln(\frac{k-1}{k}) + \frac{1}{4}\ln(\frac{k-2}{k})\right) - (\frac{2k+1}{4})\left(\ln(k+1) - \ln\pi - \frac{1}{2} - \ln 2\right) - \frac{1}{4}k^2\ln(\frac{k-1}{k}) + (-49\ln 7 + \frac{195}{4}\ln\pi - \ln 2/4 + \frac{585}{8} - \sum_{j=2}^{13}\ln(Z(j)))$. We can show that this function is equivalent to $-\frac{3}{2}k\ln k$, and that for $k > 100$, $\left|s(k)/k^2\right| \leq 0.06$. As a final step, we multiply the result by $2/k^2$ and apply exponentiation to obtain the bound $(\gamma_{2k,k})^{\frac{2}{k}} \geq \frac{k}{12}$ for all $k > 100$. Note that we already had the bound $\frac{k}{9}$ for $k \leq 100$ using numerical computation of $Z(j)$. Asymptotically, we have obtained the following lower bound : $(\gamma_{2k,k})^{\frac{2}{k}} \geq \frac{2k}{\pi e^2}$.

Réductions polynomiales

Cette partie montre que le problème calculatoire sur lequel repose la sécurité d'un système cryptographique n'est pas toujours celui qu'on croit. On peut parfois, à l'aide de réductions très efficaces, se ramener à des problèmes bien connus, ce qui influence le choix des tailles des clefs ou des paramètres. Cette partie se compose de deux articles :

Page 209 : version complète de [36].

> **Noisy Polynomial Interpolation and Noisy Chinese Remaindering**, EUROCRYPT 2000
> Daniel Bleichenbacher et Phong Q. Nguyen
>
> *Cet article présente des réductions très efficaces de problèmes calculatoires introduits récemment dans plusieurs systèmes cryptographiques, au problème du plus court vecteur dans un réseau. Il en résulte que soit les paramètres doivent être augmentés, soit le système doit être modifié pour changer d'hypothèse calculatoire.*

Page 227 : référence [268].

> **Adapting Density Attacks to Low-Weight Knapsacks**, ASIACRYPT 2005
> Phong Q. Nguyen et Jacques Stern
>
> *Cet article montre que le problème dit du sac à dos creux peut se réduire efficacement à des problèmes de plus court vecteur ou de plus proche vecteur dans un réseau. Cette variante du problème du sac à dos avait été introduit par plusieurs cryptosystèmes, afin justement d'éviter les attaques (par réduction de réseau) dites à faible densité sur le problème du sac à dos.*

Annexe F

Noisy Polynomial Interpolation and Noisy Chinese Remaindering

EUROCRYPT 2000

Version complète de [36] avec Daniel Bleichenbacher (Lucent Bell Labs)

Abstract: *The noisy polynomial interpolation problem is a new intractability assumption introduced last year in oblivious polynomial evaluation. It also appeared independently in password identification schemes, due to its connection with secret sharing schemes based on Lagrange's polynomial interpolation. This paper presents new algorithms to solve the noisy polynomial interpolation problem. In particular, we prove a reduction from noisy polynomial interpolation to the lattice shortest vector problem, when the parameters satisfy a certain condition that we make explicit. Standard lattice reduction techniques appear to solve many instances of the problem. It follows that noisy polynomial interpolation is much easier than expected. We therefore suggest simple modifications to several cryptographic schemes recently proposed, in order to change the intractability assumption. We also discuss analogous methods for the related noisy Chinese remaindering problem arising from the well-known analogy between polynomials and integers.*

F.1 Introduction

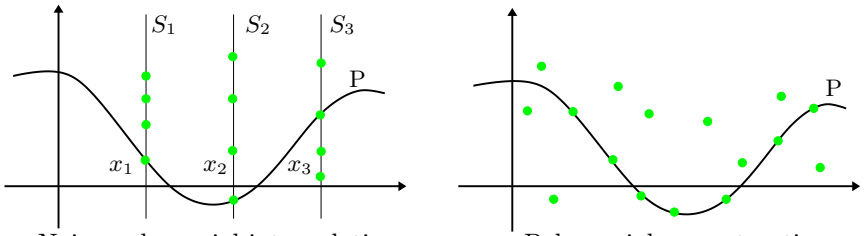
At STOC '99, Naor and Pinkas [247] introduced a new and useful primitive : oblivious evaluation of polynomials, where a polynomial P is known to Bob and he would like to let Alice compute the value $P(x)$ for an input x known to her in such a way that Bob does not learn x and Alice does not gain any additional information about P. The scheme they proposed is quite attractive, as it is much more efficient than traditional oblivious evaluation protocols, which leads to several applications. For instance, Gilboa [113] applied the scheme to two party RSA key generation. Naor and Pinkas mention other interesting applications in their paper [247], such as a method enabling two agencies each having a list of names, to find the common names on the lists without revealing other information.

Perhaps the only problem with the Naor-Pinkas scheme was a security issue, since the scheme used a new intractability assumption. The underlying computational problem, the so-called noisy polynomial interpolation problem, can be stated as follows :

Problem F.1 (Noisy polynomial interpolation) *Let P be a k-degree polynomial over a finite field $\mathbb{F}$. Given $n > k+1$ sets $S_1, \ldots, S_n$ and n distinct elements $x_1, \ldots, x_n \in \mathbb{F}$ such that each $S_i = \{y_{i,j}\}_{1 \leq j \leq m}$ contains $m-1$ random elements and $P(x_i)$, recover the polynomial P, provided that the solution is unique.*

A simple counting argument suggests that $m^n \ll |\mathbb{F}|^{n-(k+1)}$ should be satisfied to ensure the unicity of the solution. Several generalizations are possible : for instance, one can assume that the sets S_i's have different sizes instead of m. A related problem is the following :

Problem F.2 (Polynomial reconstruction) *Given as input integers k, t and n points (x_1, y_1), $\ldots$, $(x_n, y_n) \in \mathbb{F}^2$, output all univariate polynomials P of degree at most k such that $y_i = P(x_i)$ for at least t values of i.*



Noisy polynomial interpolation — Polynomial reconstruction

The polynomial reconstruction problem is well-known because the generalized Reed-Solomon list decoding problem reduces to it. The best algorithm known to solve this problem is the recent algorithm of Guruswami and Sudan [131] (GS), which was inspired by previous work of Ar *et al.* [13] on a related problem. Its running time is polynomial in n, and the algorithm succeeds provided $t > \sqrt{kn}$, for any field $\mathbb{F}$ of cardinality at most 2^n. Naor and Pinkas remarked the existence of a simple reduction from noisy polynomial interpolation to polynomial reconstruction, which led them to conjecture that the noisy polynomial interpolation problem was as hard as the polynomial reconstruction problem.

This paper provides evidence that the conjecture is likely to be false. More precisely, we present new methods to solve noisy polynomial interpolation which (apparently) do not apply to polynomial reconstruction. In particular, we prove that the noisy polynomial interpolation problem can be transformed into a lattice shortest vector problem with high probability, provided that the parameters satisfy a certain condition that we make explicit. This result is qualitatively similar to the well-known lattice-based methods [200, 75] to solve the subset sum problem : the subset sum problem can be transformed into a lattice shortest vector problem with high probability, provided that a so-called low-density condition is satisfied. As with subset sums, experimental evidence suggest that most practical instances of the noisy polynomial interpolation problem with small m can be solved. It follows that noisy polynomial interpolation is much easier than expected (despite known hardness results [5, 237] on the lattice shortest vector problem), and thus, should be used cautiously as an intractability assumption.

Interestingly, the noisy polynomial interpolation and the polynomial reconstruction problems also appeared in password authentication schemes [244, 93]. Both schemes use Shamir's secret sharing scheme based on Lagrange's polynomial interpolation, where the shares are encrypted with low entropy secrets. Shamir's scheme achieves perfect security, but here, additional information is available to the attacker. A closer inspection shows that [93] is based on the noisy polynomial interpolation problem, and is therefore insecure for many choices of the parameters. For instance, the authors propose to use $n = 22, k = 14$ and $m \approx 256$ to protect a 112-bit key. But this configuration can be broken using a meet-in-the-middle attack (see Section F.2.3) using $n' = 16$ in time 2^{64}. The solution described in [244] is much better as it is based on the hardness of the discrete log problem and a variant of the polynomial reconstruction problem.

We also discuss analogous methods for a related problem, the so-called noisy Chinese remaindering problem arising from the well-known analogy between polynomials and integers. Curiously, problems such as point counting on elliptic curves over finite fields and integer factorization of the form p^2q, can be viewed as generalized noisy Chinese remaindering problems. We explain why the lattice-based approach does not appear to be as useful in such settings.

The paper is organized as follows. In Section 2, we review simple methods for noisy polynomial interpolation. Section 3 is devoted to lattice-based methods. Cryptographic implications of these results are discussed in Section 4. In Section 5, we study analogous methods for the noisy Chinese remaindering problem. Due to lack of space, some details and proofs are omitted, but those can be found in the full version available on our webpages.

F.2 Simple methods for noisy polynomial interpolation

F.2.1 An error-correction method

When the noisy polynomial interpolation problem appeared in [247], the only known algorithm to solve it (apart from exhaustive search) was based on a simple reduction from noisy polynomial interpolation to polynomial reconstruction. More precisely, Naor and Pinkas noticed that by randomly choosing one element $y_{i,j}$ in S_i, one obtains an instance of the polynomial reconstruction problem with the n (randomly chosen) points $(x_i, y_{i,j})$. The solution P is of degree k, and we have $P(x_i) = y_{i,j}$ for approximately n/m values of i. Therefore the solution is expected to be outputted by the GS algorithm, provided that $\frac{n}{m} > \sqrt{kn}$, that is : $m < \sqrt{\frac{n}{k}}$. In fact, one can obtain a better reduction by taking all the points, which was apparently unnoticed. Indeed, if one picks all the nm points $(x_i, y_{i,j})$, then the solution P of degree k satisfies $P(x_i) = y_{i,j}$ for at least n values of (i, j). Hence, the GS algorithm will output P if $n > \sqrt{knm}$, that is :

$$m < \frac{n}{k}.$$

It is worth noting that this condition does not depend on the size of the finite field. The previous reductions do not use the specificity of the noisy polynomial interpolation instances. It is not known whether one can improve GS algorithm when applied to those particular instances, although [41] describes a simple algorithm achieving the same bound $m < n/k$. We now present methods to solve the problem when the condition $m < n/k$ is not satisfied.

F.2.2 A Gröbner basis method

A natural way to solve the noisy polynomial interpolation problem is reducing the problem to solving a system of polynomial multivariate equations. Write the unknown polynomial P as $P(X) = \sum_{i=0}^{k} a_i X^i$. For all i, there exists j such that $P(x_i) = y_{i,j}$, therefore :

$$\prod_{j=1}^{m} (P(x_i) - y_{i,j}) = 0.$$

One thus obtains n polynomial equations in the $k+1$ unknowns $a_0, \ldots, a_k$, in the field $\mathbb{F}$.

Gröbner basis is the usual way to solve such systems. However, the complexity of such techniques is super-exponential in k : in practice, it is likely that the method would be impractical if k is not very small (for instance, larger than 20). Theoretically, one could also apply the relinearization technique recently introduced by Kipnis and Shamir [181], at least in the case $m = 2$ (that is, a system of quadratic equations). At the moment, the behaviour of this new method is not completely understood, however latest results [76] suggest that the method is impractical for sufficiently large k, such as $k \geq 50$.

F.2.3 A meet-in-the-middle method

A meet-in-the-middle approach can be used to solve the noisy polynomial interpolation problem. Let $n' \leq n$ be the smallest integer for which we expect the solution to be unique. Define the Lagrange

interpolation polynomials in $\mathbb{F}[X]$:

$$L_i(X) = \prod_{\substack{1 \le j \le n' \\ j \neq i}} \frac{X - x_j}{x_i - x_j}.$$

The degree of L_i is $n' - 1$. We are looking for coefficients c_i, such that

$$\deg\left(\sum_{i=1}^{n'} y_{i,c_i} L_i(X)\right) \le k.$$

For all $\mathbf{c} = (c_1, \ldots, c_{\lfloor n'/2 \rfloor}) \in \{1, \ldots, m\}^{\lfloor n'/2 \rfloor}$ and $\tilde{\mathbf{c}} = (c_{\lfloor n'/2 \rfloor + 1}, \ldots, c'_n) \in \{1, \ldots, m\}^{\lceil n'/2 \rceil}$ we compute the polynomials $U_{\mathbf{c}}(X) = \sum_{i=1}^{\lfloor n'/2 \rfloor} y_{i,c_i} L_i(X)$ and $V_{\tilde{\mathbf{c}}}(X) = -\sum_{i=\lfloor n'/2 \rfloor + 1}^{n'} y_{i,c_i} L_i(X)$. We compare the two lists : If some $U_{\mathbf{c}}(X)$ and $V_{\tilde{\mathbf{c}}}(X)$ have identical coefficients for the terms $X^{k+1}, \ldots, X^{n'}$ then $U_{\mathbf{c}}(X) - V_{\tilde{\mathbf{c}}}(X)$ has degree at most k, and therefore, solves the problem.

The method requires the computation of $O(m^{\lceil n'/2 \rceil})$ polynomials $U_{\mathbf{c}}(X)$ and $V_{\tilde{\mathbf{c}}}(X)$. Since the values for $y_{i,j} L_i(X)$ can be precomputed and partial sums can be reused, the time complexity of this attack is $O(c(n' - k)m^{\lceil n'/2 \rceil})$, where c is the time for an addition in $\mathbb{F}$. The memory requirement of this algorithm is $O((\log q)m^{\lfloor n'/2 \rfloor})$, but an improved algorithm needing $O((\log q)m^{\lceil n'/4 \rceil})$ exists.

It is worth noting that the meet-in-the-middle method does not apply to the polynomial reconstruction problem. This is because the Lagrange polynomials $L_i(X)$ in this problem depend on the selection of the values $y_{i,j}$ used for the interpolation. Different $y_{i,j}$'s correspond to different x_i's and therefore different Lagrange polynomials. The meet-in-the-middle method takes advantage of the fact that the x_i's are known in advance.

Note that the meet-in-the-middle method can still be used if we have to compute $g^{f(x_0)}$ for some public x_0 and g, when given the $g^{y_{i,j}}$'s rather than the $y_{i,j}$'s. This is because polynomial interpolation is a linear function of the inputs $y_{i,j}$.

F.3 Lattice-based methods for noisy polynomial interpolation

We now describe lattice-based methods to solve noisy polynomial interpolation. To simplify the presentation, we assume in the whole section that the finite field $\mathbb{F}$ is a prime field $\mathbb{Z}_q$ (q being a prime number). The results extend to the general case by viewing $\mathbb{F}$ as a finite dimensional vector space over its prime field.

In this paper, we will call *lattice* any integer lattice, that is, any subgroup of $(\mathbb{Z}^n, +)$ for some n. Background on lattice theory can be found in several textbooks, such as [130, 326]. For lattice-based cryptanalysis, we refer to [171].

Our lattice-based methods build in polynomial time a lattice from a given instance of noisy polynomial interpolation. In this lattice, there is a particular lattice point, the so-called *target vector*, which is both unusually short and closely related to the solution of our problem. We will first give heuristic arguments suggesting that the target vector is the lattice shortest vector. Then we will modify our lattice to prove that the target vector is with high probability the shortest vector of the modified lattice, when the parameters satisfy a certain condition that we make explicit. The proofs are somewhat technical, but the underlying idea is similar to the one used to show that the low-density subset sum problem can be reduced with high probability to a lattice shortest vector problem [200, 75]. More precisely, we will estimate the probability that a fixed vector belongs to the lattice built from a randomly chosen instance of the problem. By enumerating all possible short vectors, we can then upper bound the probability that there exists a nonzero lattice point shorter than the target vector for a randomly chosen instance. From a practical point of view, one hopes

to solve the problem by using standard lattice reductions algorithms [205, 302, 307, 308] as lattice shortest vector oracles.

F.3.1 Linearization of noisy polynomial interpolation

Let $L_i(X)$ be the Lagrange interpolation polynomial defined as

$$L_i(X) = \prod_{j \neq i} \frac{X - x_j}{x_i - x_j}.$$

The solution P satisfies : $P(X) = \sum_{i=1}^{n} P(x_i) L_i(X)$. We linearize the problem : letting $\delta_{i,j}$ equal to 1 if $P(x_i) = y_{i,j}$, and 0 otherwise, one obtains $P(x_i) = \sum_{j=1}^{m} \delta_{i,j} y_{i,j}$, hence :

$$P(X) = \sum_{i=1}^{n} \sum_{j=1}^{m} \delta_{i,j} y_{i,j} L_i(X).$$

Since $P(X)$ has degree k, while L_i has degree $n-1$, we obtain $n-1-k$ linear equations in the nm unknowns $\delta_{i,j}$. As a linear system in the field $\mathbb{F}$, it is underdefined. However, one can also view the problem as a lattice problem for which lattice reduction might apply.[6]

The set L of vectors $(d_{1,1}, d_{1,2}, \ldots, d_{n,m}) \in \mathbb{Z}^{nm}$ such that the polynomial $\sum_{i=1}^{n} \sum_{j=1}^{m} d_{i,j} y_{i,j} L_i(X)$ has degree at most k is clearly a lattice in $\mathbb{Z}^{nm}$. The vector $(\delta_{1,1}, \delta_{1,2}, \ldots, \delta_{n,m})$ belongs to L, we call it the *target vector*. Its Euclidean norm is $\sqrt{n}$. To see how short this vector is compared to other lattice vectors, we need to analyze the lattice L. We wish to obtain results of the flavour of lattice-based algorithms to solve low-density subset sums [200, 75] : with high probability over a certain distribution of the inputs, and under specific conditions on the parameters, the target vector is the lattice shortest vector.

F.3.2 Volume of the lattice

The previous lattice is related to the lattices used by Ajtai [4] in his celebrated worst-case/average-case equivalence for certain lattice problems. More precisely, let A be a $n \times e$ matrix in $\mathbb{Z}_q$ where q is any integer. Let $L(A)$ be the set of n-dimensional integer row vectors $\mathbf{x}$ such that $\mathbf{x}A \equiv 0 \pmod{q}$. We call $L(A)$ the Ajtai lattice associated to A. It is easy to see that $L(A)$ is a n-dimensional lattice in $\mathbb{Z}^n$, from which one derives :

Lemma F.1 *Let $A \in \mathcal{M}_{n,e}(\mathbb{Z}_q)$. Then the volume of $L(A)$ divides q^e. It is exactly q^e if and only if $\{\mathbf{x}A \ : \mathbf{x} \in \mathbb{Z}_q^n\}$ is entirely $\mathbb{Z}_q^e$.*

Proof. By definition, $L(A)$ is the kernel of the group homomorphism φ that maps any $\mathbf{x} \in \mathbb{Z}^n$ to $(\mathbf{x}A \bmod q) \in \mathbb{Z}_q^e$. Therefore the group quotient $\mathbb{Z}^n / L(A)$ is isomorphic to the image of φ. But since $L(A)$ is a full-dimensional lattice in $\mathbb{Z}^n$, its volume is simply the index $[\mathbb{Z}^n : L(A)]$ of $L(A)$ in $\mathbb{Z}^n$, from which both statements follow. □

Letting $L_i(x) = \sum_{w=0}^{n-1} \ell_{i,w} x^w$, the lattice L of Section F.3.1 is equal to $L(A)$, where $\mathbb{F} = \mathbb{Z}_q$ and A is the following matrix of dimension $nm \times n-1-k$.

$$A = \begin{pmatrix} y_{1,1}\ell_{1,k+1} & \cdots & y_{1,1}\ell_{1,n-1} \\ \vdots & & \vdots \\ y_{i,j}\ell_{i,k+1} & \cdots & y_{i,j}\ell_{i,n-1} \\ \vdots & & \vdots \\ y_{n,m}\ell_{n,k+1} & \cdots & y_{n,m}\ell_{n,n-1} \end{pmatrix}$$

[6] If we used the field $\mathrm{GF}(q^a)$ rather than $\mathbb{Z}_q$ we would have $a(n-1-k)$ equations in nm unknowns over $\mathbb{Z}_q$ and the linear system might be solvable directly.

Lemma F.2 *Assume that for all $1 \leq i \leq n$ there exists $1 \leq w_i \leq m$, such that $y_{i,w_i} \neq 0$. Then* $\mathrm{rank}(A) = n - 1 - k$.

Proof. Remember, that for all $c_1, \ldots, c_n \in \mathbb{F}^n$ and $f(x) = \sum_{i=1}^n c_i L_i(x)$ we have $f(x_i) = c_i$. Hence, $\sum_{i=1}^n c_i L_i(x) = 0$ implies $c_1 = \cdots = c_n = 0$. This shows that the $n \times n$ matrix $(\ell_{i,j})_{1 \leq i \leq n; 0 \leq j \leq n-1}$ is nonsingular. In particular, the last $n - 1 - k$ columns are linearly independent and thus the matrix $(\ell_{i,j})_{1 \leq i \leq n; k+1 \leq j \leq n-1}$ has rank $n - 1 - k$. We assumed that $y_{i,w_i} \neq 0$ and therefore the matrix $A_0 = (y_{i,w_i} \ell_{i,j})_{1 \leq i \leq n; k+1 \leq j \leq n-1}$ has rank $n - 1 - k$ too. Since A_0 is a submatrix of A it follows that A has rank $n - 1 - k$ too. □

A consequence of this lemma is that the set $\{0, \ldots, q-1\}^{nm}$ contains exactly $q^{nm-n+1+k}$ lattice points and hence the volume of $L(A)$ is q^{n-1-k}. Therefore, if γ_d denotes Hermite's constant of order d, we have :

$$\lambda_1(L) \leq \sqrt{\gamma_{nm}} q^{\frac{n-1-k}{nm}},$$

where $\lambda_1(L)$ is the first minimum of L (the length of a shortest non-zero lattice point). The best asymptotic estimate known of Hermite's constant is the following (see [68]) :

$$\frac{d}{2\pi e} + \frac{\log(\pi d)}{2\pi e} + o(1) \leq \gamma_d \leq \frac{1.744 d}{2\pi e}(1 + o(1)).$$

It follows that one expects the target vector to be the shortest lattice vector if

$$\sqrt{n} \ll \sqrt{\frac{nm}{2\pi e}} q^{\frac{n-1-k}{nm}}.$$

This condition is very heuristic, as the lattice L cannot be considered as a "random" lattice.

F.3.3 Structure of the lattice

We now give a different heuristic argument to guess when the target vector is the shortest vector. The argument is inspired by lattice-based attacks against low-density subset sums (see [200, 75]). If we denote by $N(n, r)$ the number of integer points in the n-dimensional sphere of radius $\sqrt{r}$ centered at the origin, we have the following elementary result :

Lemma F.3 *Let A be a $nm \times e$ matrix in $\mathbb{Z}_q$ (q prime) chosen at random with uniform distribution. Then :*

$$\Pr\left(\lambda_1(L(A)) < \sqrt{n}\right) \leq \frac{N(nm, n)}{q^e}.$$

Proof. Let $\mathbf{x} = (x_1, \ldots, x_{nm}) \in \mathbb{Z}_q^{nm}$ be a non-zero vector. The probability that $\mathbf{x}A \equiv 0 \pmod q$ for a uniformly chosen matrix $A = (a_{i,j})_{1 \leq i \leq nm, 1 \leq j \leq e}$ is q^{-e}. Indeed, there exists $i_0 \in \{1, \ldots, nm\}$ such that $x_{i_0} \neq 0$. Then, for any choice of $(a_{i,j})_{i \neq i_0, 1 \leq j \leq e}$, there exists a unique choice of $(a_{i_0,j})_{1 \leq j \leq e}$ such that $\mathbf{x}A \equiv 0 \pmod q$, which gives the expected probability. Since the number of possible $\mathbf{x}$ is less than $N(nm, n)$, the result follows. □

It follows that one expects the target vector to be the shortest lattice vector when $N(nm, n) \ll q^{n-1-k}$. Numerical values of $N(nm, m)$ can be computed by recursion. And sharp theoretical estimates of $N(nm, m)$ can be obtained using the power series $h(x) = 1 + 2\sum_{k=1}^{\infty} x^{k^2}$ (see [227, Lemma 1]). However, the condition is still heuristic, since in our case, the matrix A cannot be considered as uniformly distributed. In particular, it does not seem easy to compute the probability that a fixed vector belongs to the lattice $L(A)$ for a randomly chosen instance of noisy polynomial interpolation.

F.3.4 Reduction by lattice improvement

To achieve a reduction from noisy polynomial interpolation to the lattice shortest vector problem, we consider a certain sublattice. The improvement is based on a property of the target vector which has not been used so far : for all i_1 and i_2, $\sum_{j=1}^{m} \delta_{i_1,j} = \sum_{j=1}^{m} \delta_{i_2,j} = 1$. This leads us to define the lattice Λ as the set of lattice points $(d_{1,1}, d_{1,2}, \cdots, d_{n,m}) \in L$ such that for all i_1 and i_2 :

$$\sum_{j=1}^{m} d_{i_1,j} = \sum_{j=1}^{m} d_{i_2,j}. \tag{F.1}$$

Since Λ is the intersection of the full-dimensional lattice L (in $\mathbb{Z}^{nm}$) with a $(nm-n+1)$-dimensional vector subspace, Λ is a $(nm-n+1)$-dimensional lattice in $\mathbb{Z}^{nm}$, which can be computed in polynomial time.

We will be able to compute the probability that a (fixed) short vector satisfying (F.1) belongs to Λ, which was apparently not possible for L. The probability is with respect to the natural distribution induced by the definition of noisy polynomial interpolation, which is the following :

- Let $x_1, \ldots, x_n$ be distinct elements of $\mathbb{F} = \mathbb{Z}_q$, and g be a function from $\{1, \ldots, n\}$ to $\{1, \ldots, m\}$.
- Choose uniformly at random a k-degree polynomial P in $\mathbb{F}[X]$.
- For all $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\} \setminus g(i)$, choose uniformly at random an element $y_{i,j}$ in $\mathbb{F}$, and let $y_{i,g(i)} = P(x_i)$.

Recall that the noisy polynomial interpolation problem is to recover either g or P, given k, the $y_{i,j}$'s and the x_i's. The (secret) function g indicates which $y_{i,j}$ is equal to $P(x_i)$.

Let $\mathbf{d} = (d_{1,1}, \ldots, d_{n,m}) \in \mathbb{Z}^{nm}$ be a vector satisfying (F.1). We define $p(\mathbf{d})$ as the probability that $\mathbf{d}$ belongs to the lattice Λ, that is, the probability that $\deg(\sum_{i=1}^{n} \sum_{j=1}^{m} d_{i,j} y_{i,j} L_i(X)) \leq k$, with respect to the previous distribution. Let $t(\mathbf{d})$ be the number of indices i for which there exists at least one nonzero $d_{i,j}$ modulo q with $j \neq g(i)$:

$$t(\mathbf{d}) = \big| \{1 \leq i \leq n : \exists j \in \{1, \ldots, m\} \setminus g(i) \text{ such that } d_{i,j} \not\equiv 0 \bmod q\} \big|.$$

The following technical lemma gives a formula for $p(\mathbf{d})$. It shows that the heuristic assumptions made in Section F.3.2 and Section F.3.3 are correct for all vectors $\mathbf{d}$ where $t(\mathbf{d}) \geq n-k-1$, but $p(\mathbf{d})$ is larger than expected when $t(\mathbf{d}) < n-k-1$. As we will see later the effect of those vectors is often negligible. A proof can be found in appendix A.

Lemma F.4 *Let $\mathbf{d} \in \mathbb{Z}^{nm}$ satisfying* (F.1)*. Then :*

$$p(\mathbf{d}) = q^{-\min(t(\mathbf{d}), n-k-1)}.$$

It follows that $p(\mathbf{d}) > \frac{1}{q}$ if and only if $t(\mathbf{d}) = 0$ (recall that $n > k+1$). But if $\mathbf{d}$ satisfies (F.1) and $t(\mathbf{d}) = 0$, then either $\mathbf{d}$ is a multiple (possibly zero) of the target vector, or at least one of $\mathbf{d}$'s entries is a nonzero multiple of q, implying $\|\mathbf{d}\| \geq q$. By enumerating all possible $\mathbf{d}$'s, we finally obtain a reduction :

Theorem F.1 *Let $\sqrt{r} < q$. Let a noisy polynomial interpolation instance be chosen uniformly at random as described above and let Λ be the sublattice built from the instance. Then the expected number of nonzero vectors $E(r, n, m)$ contained in Λ not equal to the target vector or a multiple of it with norm $\leq \sqrt{r}$ is :*

$$E(r, n, m) = \sum_{\lambda=-\lfloor r/n \rfloor}^{\lfloor r/n \rfloor} \sum_{w=1}^{n} R(w, r, \lambda, n, m)\ q^{-\min(w, n-k-1)},$$

where $R(w, r, \lambda, n, m)$ denotes the number of vectors $\mathbf{d} = (d_{1,1}, \ldots, d_{n,m}) \in \mathbb{Z}^{mn}$ such that $t(\mathbf{d}) = w$, $\|\mathbf{d}\| \leq \sqrt{r}$ and $\sum_{j=1}^{m} d_{i,j} = \lambda$.

If $E(n,n,m) < 1$ then $E(n,n,m)$ is a nontrivial upper bound on the probability that Λ contains a nonzero vector shorter than the target vector. The proof of Theorem F.1 and numerical methods to compute $E(r,n,m)$ are given in appendix B. The results are more complicated than low-density subset sum attacks for the following reasons. In low-density subset sum attacks, one can compute fairly easily an upper bound of the probability that a fixed nonzero short vector (different from the target vector) belongs to a certain lattice built from the subset sum instance (see [200, 75]). And the bound obtained is independent of the vector. It then remains to estimate the number of possible short vectors, by bounding the number of integer points in high-dimensional spheres (using techniques of [227]). Here, we have an exact formula for the probability instead of an upper bound, but the formula depends on the vector, for it involves $t(\mathbf{d})$. This leads to more complicated enumerations and asymptotic formulas. Hence, we cannot give a criterion as "simple" as the low-density criterion for subset sum, to indicate when the reduction is expected to hold. However, for some special cases we have some preliminary results :

Lemma F.5 *Let $n \geq 2$, $m \geq 2$ and $n^2 < q$. Let $0 < x < 1$ and $h(x) = 1 + 2\sum_{k=1}^{\infty} x^{k^2}$. Then :*

$$\frac{N(n, \lfloor n/2 \rfloor) + 2^{n+1} - 3}{q^{n-1-k}} \leq E(n,n,2) \leq \frac{N(n, \lfloor n/2 \rfloor) + 2^{n+1}}{q^{n-1-k}} + 2n^2/q + 4n/q \tag{F.2}$$

$$E(n,n,m) \leq \frac{N(nm,n)}{q^{n-1-k}} + 3x^{-n}\left(\left(1 + \frac{h(x)^m}{q}\right)^n - 1\right) \tag{F.3}$$

The proof of Lemma F.5 can be found in Appendix D. Note that $h(x)$ can be approximated numerically. The result for the case $m = 2$ are much stronger than the result for a general m. From a practical point of view, we can alternatively compute the upper bound $E(r,n,m)$ numerically for any given choice of the parameters. And the bound seems to be sharp in practice.

The following table shows for some values of m, n, q the largest k, such that the expected number of vectors with norm shorter or equal to $\sqrt{n}$ is smaller than 1. We compare this to the largest $\tilde{k}$ for which we would expect the target vector to be the shortest vector in the original lattice without improvement.

A missing entry in the column k says that for this particular choice of m and n the problem is very likely not solvable with the lattice based method for any k. We have chosen m and n such that the meet-in-the middle method has a time complexity of 2^{80}. We have chosen $q > 2^{80}$, so that elements of $\mathbb{Z}_q$ can be used to represent 80 bit keys for symmetric ciphers.

m	n	$\log_2(q)$	k	$\tilde{k}$
2	160	80	155	152
3	115	80	110	108
4	105	80	100	98
16	44	80	40	39
256	20	80	–	–

F.3.5 Non-prime fields

When $\mathbb{F}$ is a field of the form $\mathrm{GF}(q^a)$ with $a > 1$, Lemma F.4 still holds if one replaces q by q^a, with the same definition of $t(\mathbf{d})$ (that is, the number of indices i for which there exists at least one nonzero $d_{i,j}$ modulo q with $j \neq g(i)$), so that $p(\mathbf{d}) = q^{-a\min(t(\mathbf{d}), n-k-1)}$. Theorem F.1 and Lemma F.5 need to be modified accordingly. It follows that the lattice-based approach is useful only when the characteristic of $\mathbb{F}$ is sufficiently high ($q > \sqrt{n}$), so that any vector $\mathbf{d}$ satisfying $t(\mathbf{d}) = 0$ is strictly longer than the target vector.

F.3.6 Experiments

We implemented the improved lattice-based method on a 500 MHz 64-bit DEC Alpha using Victor Shoup's NTL library [319]. For a randomly chosen instance, we built the corresponding sublattice Λ. For lattice reduction, we successively applied three different types of reduction : plain LLL [205], Schnorr's BKZ reduction [302, 307] with block size 20, and when necessary, Schnorr-Hörner's pruned BKZ reduction [308] with block size 54 and pruning factor 14. We stopped the reduction as soon as the reduced basis contained the target vector.

To fix ideas on the efficiency of the lattice-based method, we chose $n = 160$ and $m = 2$, with a prime field of size 80 bits. The error-correction method succeeds only if $k \leq 80$. The meet-in-the-middle method requires at least $2^{k/2}$ operations. And the Gröbner basis approaches are very unlikely to be practical. Numerical values given by theorem F.1 (see Section F.3.4) suggest that the noisy polynomial interpolation problem can be reduced to a lattice shortest vector problem, as while as $k \leq 155$. The lattice dimension is then 160. Our implementation was able to solve noisy polynomial interpolation up to $k = 154$. For $k \leq 152$, only BKZ-20 reduction was necessary, and the total running time was less than 4 hours. For $153 \leq k \leq 154$, an additional Schnorr-Hörner pruned BKZ reduction was necessary : 1 hour for $k = 153$, and 8 hours for $k = 154$. We do not know if the theoretical value of $k = 155$ can be reached in practice : the corresponding lattice problem is hard because there are many lattice points almost as short as the target vector. The situation might be similar to lattice-based subset sum attacks : when the subset sum density is very close to the critical density, and the lattice dimension is large, the lattice problem is hard. It is worth noting that to ensure the unicity of the solution, one should have $k \leq 156$. This suggests that the lattice-based method is likely to solve most instances of practical interest for small m. We also made a few experiments with $m > 2$. A BKZ-20 reduction can solve in one day the problem with $n = 115$, $k = 101$, $m = 3$ and $n = 105$, $k = 80$, $m = 4$. For such parameters, the meet-in-the-middle method requires at least 2^{80} operations.

F.4 Cryptographic implications

We showed that when the parameters satisfy a certain relation, there exists a provable reduction from noisy polynomial interpolation to the lattice shortest vector problem. This results in an attack which is much more effective than previously known methods based on list decoding algorithms, due to the strength of current lattice reduction algorithms. We could not apply the same method to the polynomial reconstruction problem. This suggests (but does not prove) that the polynomial reconstruction problem is harder than the noisy polynomial interpolation problem, so that Conjecture 3.1 in [247] about the hardness equivalence[7] of the two problems does not hold.

It follows that cryptographic protocols should – if possible – be based on the polynomial reconstruction problem rather than the noisy polynomial interpolation problem. Such a change is possible for the oblivious polynomial evaluation of Naor and Pinkas [247]. There are two players Alice and Bob. Bob's secret input is a polynomial $P(x)$, which he hides in a bivariate polynomial $Q(x, y)$, such that $Q(0, y) = P(y)$. Alice has a secret value α and would like to learn $P(\alpha)$. Alice chooses a polynomial $S(x)$ with $S(0) = \alpha$. In a crucial step of the protocol Alice would like to learn $Q(x_i, S(x_i))$ without revealing $S(x)$. This is done by sending x_i and a list of random values $y_{i,j}$, except that one value $S(x_i)$. Bob computes $Q(x_i, y_{i,j})$ for all these values and A retrieves the answer she is interested in using a 1-out-of-m oblivious transfer. The privacy of Alice depends on the difficulty to find $S(x)$ given x_i and $y_{i,j}$, *i.e.* the noisy polynomial interpolation problem. However, the protocol can be changed by using the values $Q(x_{i,j}, y_{i,j})$ for distinct $x_{i,j}$'s rather than $Q(x_i, y_{i,j})$ [284].

Another way to prevent lattice-based attacks is to use a field where computing discrete logarithms

[7] In fact, Conjecture 3.1 relates the hardness of polynomial reconstruction and an easier version of noisy polynomial interpolation.

is intractable, and to publish the powers $g^{y_{i,j}}$ rather than the values $y_{i,j}$. It is then still possible to perform a polynomial interpolation, that is to compute $g^{f(x_0)}$, given sufficiently many values $g^{f(x_i)}$. In fact, the meet-in-the middle method is the only algorithm known to us that is applicable in this case and it can only be used for the noisy polynomial interpolation problem but not for the polynomial reconstruction problem. A protocol using the polynomial interpolation problem combined with the discrete logarithm problem is described in [244].

F.5 Noisy Chinese remaindering

There is a well-known analogy between polynomials and integers : the polynomial degree corresponds to the integer size ; Lagrange's interpolation corresponds to Chinese remainders ; and polynomial evaluation corresponds to the modulo operation (in fact, a polynomial P evaluated at x_0 can also be viewed as the remainder of $P(x)$ modulo the linear polynomial $x - x_0$). We refer to [120] for some examples. The noisy polynomial interpolation and polynomial reconstruction problems then become the following ones :

Problem F.3 (Noisy Chinese remaindering) *Let $0 \leq N \leq B$, and $p_1, \ldots, p_n$ be coprime integers. Given n sets $S_1, \ldots, S_n$ where each $S_i = \{r_{i,j}\}_{1 \leq j \leq m}$ contains $m-1$ random elements in $\mathbb{Z}_{p_i}$ and $N \bmod p_i$, recover the integer N, provided that the solution is unique (*e.g.*, $m^n B \ll \prod_{i=1}^{n} p_i$).*

Problem F.4 (Chinese remaindering with errors) *Given as input integers t, B and n points (r_1, p_1), $\ldots$, $(r_n, p_n) \in \mathbb{N}^2$ where the p_i's are coprime, output all numbers $0 \leq N < B$ such that $N \equiv r_i \pmod{p_i}$ for at least t values of i.*

We refer to [120] for a history of the latter problem, which is beyond the scope of this article. We will only mention that the best decoding algorithm known for the problem is the recent lattice-based work of Boneh [41], which improves previous work of Goldreich *et al.* [120]. The algorithm works in polynomial time and solves the problem provided that a certain condition is satisfied. The exact condition is analogous to the bound obtained by GS algorithm for polynomial reconstruction.

We note that there are two well-known problems for which the general noisy Chinese remaindering problem (in which one allows different sizes for the sets S_i's) arises. The first problem is point counting on elliptic curves over finite fields. The best general algorithm for this problem is the Schoof-Elkies-Atkin (SEA) algorithm [313, 92, 15, 16] (see [212] for implementation issues). Let E be an elliptic curve over a finite field of cardinality q. Hasse's theorem states that the cardinality of E is of the form $q + 1 - t$ where $|t| \leq 2\sqrt{q}$. The SEA algorithm tries to determine this t, using Chinese remainders. However, in practice, it turns out to be too expensive to compute the exact value of t modulo sufficiently many coprime numbers. Therefore, one actually determines many coprime numbers of two kinds : for the first kind of numbers, t modulo such numbers is exactly known ; for the second kind of numbers, the value of t modulo such numbers is constrained to a small number of values. This is exactly a noisy Chinese remaindering problem. To solve this problem, current versions of SEA apply a meet-in-the-middle strategy. The second problem is integer factorization of numbers of the form $N = p^2 q$. It has been noticed for some time (see for instance [281]) that for any number r, the Jacobi symbol $\left(\frac{r}{N}\right)$ is equal to the Legendre symbol $\left(\frac{r}{q}\right)$. It follows that for any number r, $q \bmod r$ is limited to half of $\mathbb{Z}_r$, and such a half can be determined. The problem of computing q can thus be viewed as a noisy Chinese remaindering problem. However, the S_i's are so dense that this formulation is likely to be useless.

We briefly review methods for noisy Chinese remaindering, analogous to the ones we described for noisy polynomial interpolation. One can first use the analog of the meet-in-the-middle method of Section F.2.3. One can also use the reduction to Chinese remaindering with errors and the algorithm of [41], in a way analogous to Section F.2.1. But the following simpler method achieves the same results.

F.5.1 Coppersmith's method

We obtain an analogous method to the Gröbner basis approach by translating the problem in terms of polynomial equations. The solution N satisfies for each i the following equation :

$$\prod_{j=1}^{m} (N - r_{i,j}) \equiv 0 \pmod{p_i}.$$

Using Chinese remainders and collecting all equations, one obtains a univariate polynomial equation of degree m in the unknown N modulo $\prod_{i=1}^{n} p_i$. We then apply the following lattice-based result by Coppersmith [70] :

Theorem F.2 *Let $P(x)$ be a polynomial of degree δ in one variable modulo an integer M of possibly unknown factorization. In time polynomial in $(\log M, 2^{\delta})$, one can find all integers x_0 such that $P(x_0) \equiv 0 \ (mod\, M)$ and $|x_0| \leq M^{1/\delta}$.*

In time polynomial in $(\sum_{i=1}^{n} \log p_i, 2^m)$, we can thus find the solution N to noisy Chinese remaindering, provided that : $B^m \leq \prod_{i=1}^{n} p_i$. This condition is analogous to the condition $m < n/k$ we obtained by applying GS algorithm to the noisy polynomial interpolation problem. The method is mentioned in [41].

F.5.2 Lattice-based methods

Let $P = \prod_{i=1}^{n} p_i$. By analogy to the lattice-based method of section F.3, we define interpolation numbers L_i in $\{0, \ldots, P-1\}$ by : $L_i \equiv 1 \pmod{p_i}$ and $L_i \equiv 0 \pmod{\prod_{j \neq i} p_j}$. The solution N of noisy Chinese remaindering satisfies :

$$N \equiv \sum_{i=1}^{n} (N \bmod p_i) L_i \pmod{P}.$$

We linearize the problem : letting $\delta_{i,j}$ equal to 1 if $N \equiv r_i \pmod{p_i}$, and 0 otherwise, one obtains

$$N \equiv \sum_{i=1}^{n} \sum_{j=1}^{m} \delta_{i,j} r_{i,j} L_i \pmod{P}.$$

This equation basically says that N is a small subset sum of the $r_{i,j}L_i$'s modulo P. It is thus natural to consider the $(nm+1)$-dimensional lattice L spanned by the rows of the following matrix :

$$\begin{pmatrix} P & 0 & \ldots & \ldots & 0 \\ r_{1,1}L_1 & B & 0 & \ldots & 0 \\ r_{1,2}L_1 & 0 & B & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ r_{n,m}L_n & 0 & \ldots & 0 & B \end{pmatrix}$$

The lattice L is the set of integer row vectors $(M, d_{1,1}B, d_{1,2}B, \ldots, d_{n,m}B) \in \mathbb{Z}^{nm+1}$ such that $M \equiv \sum_{i=1}^{n} \sum_{j=1}^{m} d_{i,j} r_{i,j} L_i \pmod{P}$. It contains the *target vector* $(N, \delta_{1,1}B, \delta_{1,2}B, \ldots, \delta_{n,m}B)$, which has norm $\sqrt{N^2 + nB^2} \leq B\sqrt{n+1}$. Since the previous matrix is triangular, the volume of L is simply $P \times B^{nm}$. It follows that the target vector is expected to be the shortest vector of L when

$$B\sqrt{n+1} \ll (PB^{nm})^{1/(nm+1)} \approx P^{1/(nm+1)} B,$$

that is $\sqrt{n} \ll P^{1/(nm+1)}$. The condition should however be taken with care, as the lattice L cannot be considered as random. For instance, note that any sufficiently short linear relation between $r_{i,1}, r_{i,2}, \ldots, r_{i,m}$ gives rise to a shorter lattice point. It can be proved that such a case occurs when one of the p_i's is small or one of the $|S_i|$'s is big (using the notion of orthogonal lattice [251], see Appendix C). As with noisy polynomial interpolation, one can improve the lattice L by considering the sublattice Λ of points $(M, d_{1,1}B, d_{1,2}B, \ldots, d_{n,m}B) \in L$ such that, for all i_1 and i_2, $\sum_{j=1}^{m} d_{i_1,j} = \sum_{j=1}^{m} d_{i_2,j}$. However, the previous obstruction still holds (see details in Appendix C). Thus, the lattice-based approach is unlikely to be useful for elliptic curve point counting or integer factorization. Still, the reduction can be proved for certain choices of the parameters, for we have the following analog of lemma F.4.

Lemma F.6 *Let* $\mathbf{d} = (M, d_{1,1}B, d_{1,2}B, \ldots, d_{n,m}B) \in \mathbb{Z}^{nm+1}$ *satisfying* (F.1) *and shorter than the target vector. Assume that* $B(m+1)\sqrt{n+1} < P/2$. *Then :*

$$p(\mathbf{d}) \leq q^{-\min(t(\mathbf{d}), n-k)},$$

where $q = \min p_i$, *k is the least positive integer such that* $B(m+1)\sqrt{n+1} < \frac{q^k}{2}$, *and* $t(\mathbf{d}) = \big|\{1 \leq i \leq n : \exists j \in \{1, \ldots, m\} \setminus g(i) \text{ such that } d_{i,j} \not\equiv 0 \bmod p_i\}\big|$.

This lemma is useful, when none of the $|S_i|$'s are big and none of the p_i's are small (which is not the case arising in elliptic curve point counting or integer factorization) in which case one can obtain a provable reduction to the lattice shortest vector problem roughly similar to Theorem F.1 since one can upper bound the probability that there exists a nonzero vector strictly shorter than the target vector. In particular, by taking all the p_i's of the same size (such as 32 bits), it is easy to build instances for which the lattice-based approach can experimentally solve noisy Chinese remaindering with a bound B much larger than with Coppersmith's method.

F.6 Conclusion

We presented various methods to solve the noisy polynomial interpolation problem. In particular, we proved the existence of a reduction from the noisy polynomial interpolation problem to the lattice shortest vector problem, for many choices of the parameters. This reduction appears to be very efficient in practice : experimental evidence suggest that many instances can be solved using standard lattice reduction algorithms. We therefore suggested simple modifications to several cryptographic schemes for which the security assumption relied on the computational hardness of noisy polynomial interpolation. We also briefly discussed analogous methods to solve the related noisy Chinese remaindering problem. The lattice-based approach is the best known method for certain choices of the parameters, but unfortunately not in applications such as elliptic curve point counting or integer factorization. There are several open problems, such as :

- Is there a better[8] reduction from noisy polynomial interpolation or Chinese remaindering to the lattice shortest vector problem ?
- Is there a lattice-based method to solve the polynomial reconstruction problem ?

Acknowledgments

We thank Amin Shokrollahi and Karin Melnick for their help and ideas with error correcting algorithms. We are grateful to Dan Boneh for several enlightening discussions, and for informing us on references [13, 120, 41]. We also thank Louis Granboulan and Andrew Odlyzko for their comments.

[8] holding for more or all choices of the parameters.

Appendix A : Proof of lemma F.4

To simplify the notations of the proof, we assume without loss of generality that $g(i) = 1$ for all i. Since $\mathbf{d}$ satisfies (F.1), there exists $\lambda \in \mathbb{Z}$ such that, for all i :

$$\sum_{j=1}^{m} d_{i,j} = \lambda.$$

Let $\tilde{y}_{i,j} = y_{i,j} - f(x_i)$ for $2 \leq j \leq m$, and $h(X) = \sum_{i=1}^{n} \sum_{j=2}^{m} d_{ij} \tilde{y}_{ij} L_i(X)$. Since the $y_{i,j}$'s, for $j \geq 2$, are independent and uniformly distributed over $\mathbb{F}$, the same holds for the $\tilde{y}_{i,j}$'s, for all i and j. We have :

$$\begin{aligned}
\sum_{i=1}^{n}\sum_{j=1}^{m} d_{i,j} y_{i,j} L_i(X) &= \sum_{i=1}^{n}\sum_{j=2}^{m} d_{i,j} y_{i,j} L_i(X) + \sum_{i=1}^{n} d_{i,1} P(x_i) L_i(X) \\
&= \sum_{i=1}^{n}\sum_{j=2}^{m} d_{i,j} y_{i,j} L_i(X) + \sum_{i=1}^{n} \left(\lambda - \sum_{j=2}^{m} d_{i,j}\right) P(x_i) L_i(X) \\
&= \sum_{i=1}^{n}\sum_{j=2}^{m} d_{i,j} \left(y_{i,j} - P(x_i)\right) L_i(X) + \lambda \sum_{i=1}^{n} P(x_i) L_i(X) \\
&= h(X) + \lambda P(X)
\end{aligned}$$

Since $P(X)$ has degree k, it follows that :

$$p(\mathbf{d}) = \Pr\left(\deg\left(\sum_{i=1}^{n}\sum_{j=1}^{m} d_{i,j} y_{i,j} L_i(X)\right) \leq k\right) = \Pr\left(\deg h(X) \leq k\right).$$

This new expression of $p(\mathbf{d})$ will be useful because it is independent of the polynomial $P(X)$. Note that $h(x_i) = \sum_{j=2}^{m} d_{ij} \tilde{y}_{ij}$. We will use two basic remarks :

1. Any nonzero polynomial of degree k has at most k roots over $\mathbb{F}$.
2. Any polynomial of degree at most $n-1$ can be uniquely expressed as a linear combination of the $L_i(X)$, since the L_i's are linearly independent.

We divide our analysis into two cases :

- Case 1 : $t(\mathbf{d}) \leq n-k-1$. From the expression of $h(x_i)$, it follows that $h(x_i) = 0$ for at least $k+1$ values x_i. Therefore, by remark 1, $\deg h(X) \leq k$ if and only if $h(X) = 0$. From remark 2, $h(X) = 0$ holds if and only if $\sum_{j=2}^{m} d_{ij} \tilde{y}_{ij} = 0$ for all $1 \leq i \leq n$. Among these n independent equations, only $t(\mathbf{d})$ are non trivial, each of which being satisfied with probability $1/q$. Hence, we proved that $p(\mathbf{d}) = q^{-t(\mathbf{d})}$.
- Case 2 : $t(\mathbf{d}) \geq n-k$. There exists a set $S = \{(i_1, j_1), \ldots, (i_{n-k-1}, j_{n-k-1})\}$, such that the i_r's are pairwise distinct and $d_{i_r,j_r} \neq 0$ for $1 \leq r \leq n-k-1$. Let the $\tilde{y}_{i,j}$'s be fixed for all $(i,j) \notin S$. Now consider the function φ that maps any $(\tilde{y}_{i,j})_{(i,j)\in S} \in \mathbb{F}^{n-k-1}$ to the $(n-k-1)$-tuple formed by the coefficients of degree $k+1, k+2, \ldots, n-1$ of the polynomial $h(X) = \sum_{i=1}^{n} \sum_{j=2}^{m} d_{ij} \tilde{y}_{ij} L_i(X)$. We notice that φ is one-to-one, by showing that φ is injective. Suppose indeed there at two tuples $(\tilde{y}_{i,j})_{(i,j)\in S}$ and $(\hat{y}_{i,j})_{(i,j)\in S}$ leading to two polynomials $\tilde{h}(X)$ and $\hat{h}(X)$ sharing the same $n-k-1$ high-order coefficients. Then $\deg \Delta(X) \leq k$, where :

$$\Delta(X) = \tilde{h}(X) - \hat{h}(X) = \sum_{(i,j)\in S} d_{i,j} (\tilde{y}_{i,j} - \hat{y}_{i,j}) L_i(X).$$

The polynomial $\Delta(X)$ is a linear combination of $n-k-1$ polynomials L_i's : there are $k+1$ distinct x_i's such that $\Delta(x_i)=0$. By remark 1, this implies $\Delta(X)=0$, and thus, by remark 2, $\tilde{y}_{i,j}-\hat{y}_{i,j}=0$ for all $(i,j)\in S$, since $d_{i,j}\not\equiv 0 \pmod q$ for $(i,j)\in S$. Hence, we proved that φ is one-to-one. In particular, there is a unique assignment of $(\hat{y}_{i,j})_{(i,j)\in S}$ such that $\deg h(X)\leq k$. We therefore proved $p(\mathbf{d})=q^{-(n-k-1)}$.

Appendix B : Estimating the number of short vectors

In order to compute the expected number of short vectors, we have to find the number of vectors $\mathbf{d}$ for a given $t(\mathbf{d})$ and given maximal norm. We do not have an asymptotic formula for this number, but we have found recurrence relations that allowed us to find the exact number for the necessary cases.

Let w,r,λ,n,m be non-negative integers. Then we denote by $T(r,\lambda,m)$ the number of vectors $\mathbf{d}=(d_1,\ldots,d_m)\in\mathbb{Z}^m$ satisfying $\sum_{j=1}^m d_j=\lambda$ and $\|\mathbf{d}\|=\sqrt{r}$. Denote by $R(w,r,\lambda,n,m)$ the number of vectors $\mathbf{d}=(d_{1,1},\ldots,d_{n,m})\in\mathbb{Z}^{mn}$ with $t(\mathbf{d})=w$ and norm $\|\mathbf{d}\|\leq\sqrt{r}$ and $\sum_{j=1}^m d_{i,j}=\lambda$.

We used the following recurrence relations for computing T and R.

$$T(r,\lambda,m)=\begin{cases}1 & \text{if } m=0 \text{ and } r=0 \text{ and } \lambda=0,\\ 0 & \text{if } m=0 \text{ and } (r\neq 0 \text{ or } \lambda\neq 0),\\ \sum_{j=-\lfloor\sqrt{r}\rfloor}^{\lfloor\sqrt{r}\rfloor} T(r-j^2,\lambda-j,m-1) & \text{if } m>0.\end{cases}$$

$$S(w,r,\lambda,m)=\begin{cases}1 & \text{if } w=0 \text{ and } r\geq 0,\\ 0 & \text{if } w=0 \text{ and } r<0,\\ \sum_{j=1}^{r}(T(j,\lambda,m)-\epsilon_{j,\lambda})S(w-1,r-j,\lambda,m) & \text{if } w>0,\end{cases}$$

where $\epsilon_{j,\lambda}=1$ if $j^2=\lambda$ and 0 otherwise. Here, $S(w,r,\lambda,m)$ denotes the number of vectors in $\mathbf{d}=(d_{1,1},\ldots,d_{w,m}),\in\mathbb{Z}^{wm}$, such that $\sum_{j=1}^m d_{i,j}=\lambda$ for all $1\leq i\leq w$, $\|\mathbf{d}\|\leq\sqrt{r}$ and for every $1\leq i\leq w$ there exists j such that $d_{i,j}\neq 0$. The latest condition ensures that $t(\mathbf{d})=w$, it is also the reason for the correction $\epsilon_{j,\lambda}$ above.

In order to compute $R(w,r,\lambda,n,m)$ we note there must be exactly $(n-w)$ values for i, such that $d_{i,g(i)}=\lambda$ and $d_{i,j}=0$ for $j\in\{1,\ldots,m\}\setminus\{g(i)\}$, where $g(i)$ is a function defined by $y_{i,g(i)}=P(x_i)$. This leads to the equation

$$R(w,r,\lambda,n,m)=\binom{n}{w}S(w,r-(n-w)\lambda^2,\lambda,m).$$

Now, we can compute the expected number of non-target vectors (i.e. a vector with $t(\mathbf{d})>0$) with norm $\leq\sqrt{r}$ as

$$\begin{aligned}E(r,n,m) &= \sum_{\mathbf{d}:\|\mathbf{d}\|\leq\sqrt{r}} q^{-\min(t(\mathbf{d}),n-k-1)}\\ &= \sum_{\lambda=-\lfloor r/n\rfloor}^{\lfloor r/n\rfloor}\sum_{w=1}^{n} R(w,r,\lambda,n,m)q^{-\min(w,n-k-1)}.\end{aligned}$$

Similarly the exact number $N(n,r)$ of vectors of dimension n and norm at most $\sqrt{r}$ can, of course, be computed by

$$N(n,r)=\begin{cases}1 & \text{if } n=0 \text{ and } r\geq 0,\\ 0 & \text{if } n=0 \text{ and } r<0,\\ \sum_{j=-\lfloor\sqrt{r}\rfloor}^{\lfloor\sqrt{r}\rfloor} N(n-1,r-j^2) & \text{if } n>0.\end{cases}$$

Appendix C : The lattice-based method for noisy Chinese remaindering

In this appendix, we give some conditions for which the lattice-based method for noisy Chinese remaindering must fail, that is, for which the lattice contains at least one nonzero vector shorter than the target vector.

We first consider the simple case of the original lattice L. We note that any short linear relation between $r_{i,1}, r_{i,2}, \ldots, r_{i,m}$ give rise to a short lattice point, by adding zero entries and multiplying by B. To predict the size of those linear relations, we use the notion of orthogonal lattice [251]. For any lattice Λ in $\mathbb{Z}^m$, the lattice $\Lambda^\perp$ is the set of integer row vectors $\mathbf{x}$ such that $\mathbf{x}$ is orthogonal to $\mathbf{y}$ for all $\mathbf{y} \in \Lambda$. It is known that $\det(\Lambda^\perp) \leq \det(\Lambda)$ and $\dim \Lambda + \dim \Lambda^\perp = n$. If we let $\mathbf{r}_i = (r_{i,1}, r_{i,2}, \ldots, r_{i,m})$, then the one-dimensional lattice spanned by $\mathbf{r}_i$ has volume less than $p_i\sqrt{m}$. It follows that :

$$\lambda_1(\mathbf{r_i}^\perp) \leq \sqrt{\gamma_{m-1}} \left(p_i\sqrt{m}\right)^{1/(m-1)}.$$

Therefore, there is at least one nonzero lattice point of L shorter than the target vector if :

$$\sqrt{\gamma_{m-1}} \left(p_i\sqrt{m}\right)^{1/(m-1)} < \sqrt{n+1}.$$

One of these n inequalities is satisfied when one of the p_i's is sufficiently small, or m is sufficiently big (or equivalently for the general noisy Chinese remaindering problem, when one of the $|S_i|$'s is large).

We now consider the sublattice Λ of L. To see that the previous obstruction still holds, it suffices to consider vectors which are orthogonal to both one of the $\mathbf{r}_i$'s, and the vector $(1, 1, \ldots, 1) \in \mathbb{Z}^m$. By adding zero entries to those vectors, and multiplying by B, one obtains a lattice point of Λ. The lattice spanned by $\mathbf{r}_i$ and $(1, 1, \ldots, 1)$ has volume less than $p_i m$. It follows that there exists at least one nonzero lattice point of Λ shorter than the target vector if for some i :

$$\sqrt{\gamma_{m-2}} \left(p_i m\right)^{1/(m-2)} < \sqrt{n+1}.$$

Again, one of these n inequalities is satisfied when one of the p_i's is sufficiently small, or m is sufficiently big.

Appendix D : Proof of Lemma F.5.

Proof of Equation (F.2)

The number $N(n, r)$ of n-dimensional vectors of norm $\leq \sqrt{r}$ satisfies the following inequality[9]

$$N(n, r) \leq 2^n \binom{n+r}{r}.$$

Let $\mathbf{d}$ be a vector satisfying $\sum_{j=1}^m d_{i,j} = \lambda$ for some integer λ. Then the norm of $\mathbf{d}$ is at least $\sqrt{|\lambda n|}$ and hence $r = n$ implies $\lambda \in \{-1, 0, 1\}$. We will represent $E(n, n, 2)$ as a sum of 4 summands

[9] This is quite a bad estimate. I'll prove it if nothing better can be found. The estimate is however, sufficient for this proof.

depending on the value of λ and whether $t(\mathbf{d}) \leq n-1-k$. In particular, we let

$$\begin{aligned} E(n,n,2) &= A+B+C+D, \quad \text{where} \\ A &= \sum_{w=1}^{n-2-k} R(w,n,0,n,2)q^{-w} \\ B &= \sum_{w=n-1-k}^{n} R(w,n,0,n,2)q^{-n+1+k} \\ C &= \sum_{\lambda\in\{1,-1\}} \sum_{w=1}^{n-2-k} R(w,n,\lambda,n,2)q^{-w} \\ D &= \sum_{\lambda\in\{1,-1\}} \sum_{w=n-1-k}^{n} R(w,n,\lambda,n,2)q^{-n+1+k} \end{aligned}$$

If $\lambda = 0$ then $d_{i,2} = -d_{i,1}$ and

$$\|\mathbf{d}\| = \sqrt{2\sum_{i=1}^{n} d_{i,1}^2}.$$

Hence there is a one-to-one mapping between the $2n$-dimensional vectors $\mathbf{d} = (d_{1,1}, \ldots, d_{n,2})$ satisfying $d_{i,1} + d_{i,2} = 0$ of norm $\leq \sqrt{n}$ and the n-dimensional vectors $(d_{1,1}, \ldots, d_{n,1})$ of norm $\leq \sqrt{n/2}$. It follows that the number $R(w,n,0,n,2)$ of vectors $\mathbf{d}$ with $d_{i,1} + d_{i,2} = 0$ and $t(\mathbf{d}) = w$ and norm $\|\mathbf{d}\| \leq n$ satisfies

$$R(w,n,0,n,2) \leq \binom{n}{w} N(w, \lfloor n/2 \rfloor).$$

And, hence we get

$$\begin{aligned} A &= \sum_{w=1}^{n-k-2} R(w,n,0,n,2)q^{-w} \leq \sum_{w=1}^{n-k-2} \binom{n}{w} N(w,\lfloor n/2\rfloor)q^{-w} \\ &\leq \sum_{w=1}^{n-k-2} \binom{n}{w}\binom{\lfloor n/2\rfloor + w}{w} 2^w q^{-w} \leq \sum_{w=1}^{n-k-2} \frac{n^w}{w!}(\lfloor n/2\rfloor)^w 2^w q^{-w} \\ &\leq \sum_{w=1}^{\infty} \frac{(n^2/q)^w}{w!} = e^{n^2/q} - 1 \leq 2n^2/q. \end{aligned}$$

Moreover, we also have

$$\sum_{w=1}^{n} R(w,n,0,n,2) = N(n,\lfloor n/2\rfloor) - 1.$$

Thus it follows

$$B \leq N(n,\lfloor n/2\rfloor)q^{-n+1+k}.$$

Finally, we also note that

$$A + B \geq \sum_{w=1}^{n} R(w,n,0,n,2)q^{-n+1+k} = (N(n,\lfloor n/2\rfloor) - 1)q^{-n+1+k}.$$

If $\lambda = \pm 1$ then for all $1 \leq i \leq n$ either $d_{i,1} = \lambda$ and $d_{i,2} = 0$ or $d_{i,1} = \lambda$ and $d_{i,2} = 0$. Hence, we have

$$R(w,n,\lambda,n,2) = \binom{n}{w}.$$

Therefore,

$$C = 2\sum_{w=1}^{n-2-k}\binom{n}{w}q^{-w} \leq 2\left(1+q^{-1}\right)^n - 2 \leq 2e^{n/q} - 2 \leq 4n/q.$$

Also, we have

$$D \leq 2\sum_{w=1}^{n} R(w,n,\lambda,n,2)q^{n-1-k} \leq 2^{n+1}q^{n-1-k}.$$

And finally, we get

$$C + D \geq 2\sum_{w=1}^{n} R(w,n,\lambda,n,2)q^{n-1-k} = (2^{n+1}-2)q^{n-1-k}.$$

Adding the results for $\lambda = 0$ and $\lambda = \pm 1$ completes the proof.

Proof of Equation (F.3).

According to [227, Lemma 1] (with $x = e^{-s}$ and $\alpha = 1$) we have

$$N(n,r) \leq x^{-r}h(x)^n, \text{ for all } 0 < x < 1$$

where $h(x) = 1 + 2\sum_{k=1}^{\infty} x^{k^2}$.

$N(nm,n)q^{-n+1+k}$ is a trivial upper bound for the expected number of short vectors with $t(\mathbf{d}) \geq n-1-k$. Hence, it remains to approximate the expected number of vectors with $t(\mathbf{d}) < n-1-k$. For any $s > 0$ this number is given by

$$\begin{aligned}
&\sum_{w=1}^{n-1-k}\sum_{\lambda\in\{-1,0,1\}} R(w,n,\lambda,n,m)q^{-w} \\
\leq\ & 3\sum_{w=1}^{n-1-k}\binom{n}{w}N(wm,n)q^{-w} \\
\leq\ & 3\sum_{w=1}^{n-1-k}\binom{n}{w}e^{sn}h(s)^{wm}q^{-w} \\
\leq\ & 3e^{sn}\left((1+h(s)^m/q)^n - 1\right)
\end{aligned}$$

Appendix E : Proof of Lemma F.6

The proof is analogous to the one of Lemma F.4, with some slight technical differences. To simplify the notations of the proof, we assume without loss of generality that $g(i) = 1$ for all i. Since $\mathbf{d}$ satisfies (F.1), there exists $\lambda \in \mathbb{Z}$ such that, for all i :

$$\sum_{j=1}^{m} d_{i,j} = \lambda.$$

Let $\tilde{r}_{i,j} = (r_{i,j} - N) \bmod p_i$ for $2 \leq j \leq m$, and h be the smallest residue (in absolute value) of $\sum_{i=1}^{n}\sum_{j=2}^{m} d_{ij}\tilde{r}_{ij}L_i) \bmod P$. Since the $r_{i,j}$'s, for $j \geq 2$, are independent and uniformly distributed

over $\mathbb{Z}_{p_i}$, the same holds for the $\tilde{r}_{i,j}$'s, for all i and j. We have :

$$\begin{aligned}
\sum_{i=1}^{n}\sum_{j=1}^{m} d_{i,j}r_{i,j}L_i &= \sum_{i=1}^{n}\sum_{j=2}^{m} d_{i,j}r_{i,j}L_i + \sum_{i=1}^{n} d_{i,1}(N \bmod p_i)L_i \\
&= \sum_{i=1}^{n}\sum_{j=2}^{m} d_{i,j}r_{i,j}L_i + \sum_{i=1}^{n}\left(\lambda - \sum_{j=2}^{m} d_{i,j}\right)(N \bmod p_i)L_i \\
&= \sum_{i=1}^{n}\sum_{j=2}^{m} d_{i,j}\left(r_{i,j} - N \bmod p_i\right)L_i + \lambda\sum_{i=1}^{n}(N \bmod p_i)L_i \\
&\equiv h + \lambda N \pmod{P}
\end{aligned}$$

Since $\mathbf{d}$ is shorter than the target vector : $\|\mathbf{d}\| \leq B\sqrt{n+1}$. Therefore $M \leq B\sqrt{n+1}$ and $|d_{i,j}| \leq \sqrt{n+1}$, so that $|\lambda| \leq m\sqrt{n+1}$. If $\mathbf{d}$ is in the sublattice then $M \equiv \sum_{i=1}^{n}\sum_{j=1}^{m} d_{i,j}r_{i,j}L_i \pmod{P}$ so that $M \equiv h + \lambda N \pmod{P}$. Notice that $-P/2 < h \leq P/2$ and $-P/2 < M - \lambda N \leq P/2$ as $|M - \lambda N| \leq B(m+1)\sqrt{n+1} < P/2$. Therefore $M = h + \lambda N$ holds over $\mathbb{Z}$, which implies $|h| \leq B(m+1)\sqrt{n+1}$. We therefore proved :

$$p(\mathbf{d}) \leq \Pr\left(|h| \leq B(m+1)\sqrt{n+1}\right).$$

Note that $h \equiv \sum_{j=2}^{m} d_{ij}\tilde{r}_{ij} \pmod{p_i}$. We will use one basic remark : If $0 \leq |u| < q^k$ and $u \equiv 0 \pmod{p_i}$ for at least k values of i, then u is zero. We divide our analysis into two cases :

- Case 1 : $t(\mathbf{d}) \leq n - k$. From the expression of $h \bmod p_i$, it follows that $h \equiv 0 \bmod p_i$ for at least k values of i. Since $|h| \leq B(m+1)\sqrt{n+1} < q^k$, we therefore have $h = 0$, thus $\sum_{j=2}^{m} d_{ij}\tilde{r}_{ij} \equiv 0 \pmod{p_i}$ for all $1 \leq i \leq n$. Among these n independent equations, only $t(\mathbf{d})$ are non trivial, each of which being satisfied with probability less than $1/q$. Hence, we proved that $p(\mathbf{d}) \leq q^{-t(\mathbf{d})}$.
- Case 2 : $t(\mathbf{d}) > n - k$. There exists a set $S = \{(i_1, j_1), \ldots, (i_{n-k}, j_{n-k})\}$, such that the i_r's are pairwise distinct and $d_{i_r,j_r} \neq 0$ for $1 \leq r \leq n - k$. Let the $\tilde{r}_{i,j}$'s be fixed for all $(i,j) \notin S$. Now suppose there at two tuples $(\tilde{r}_{i,j})_{(i,j)\in S}$ and $(\hat{r}_{i,j})_{(i,j)\in S}$ leading to two numbers $\tilde{h}$ and $\hat{h}$ less than $B(m+1)\sqrt{n+1}$ in absolute value. Then $|\Delta| < q^k$ where :

  $$\Delta = \tilde{h} - \hat{h} \equiv \sum_{(i,j)\in S} d_{i,j}(\tilde{r}_{i,j} - \hat{r}_{i,j})L_i \pmod{P}.$$

  There are at least k distinct i's such that $\Delta \equiv 0 \pmod{p_i}$, which implies $\Delta = 0$, and thus, $\tilde{r}_{i,j} - \hat{r}_{i,j} \equiv 0 \pmod{p_i}$ for all $(i,j) \in S$, since $d_{i,j} \not\equiv 0 \pmod{p_i}$ for $(i,j) \in S$. Hence, we proved that there is a unique assignment of $(\hat{r}_{i,j})_{(i,j)\in S}$ such that $|h| \leq B(m+1)\sqrt{n+1}$. We therefore proved $p(\mathbf{d}) \leq q^{-(n-k)}$.

Annexe G

Adapting Density Attacks to Low-Weight Knapsacks

ASIACRYPT 2005
[268] avec Jacques Stern (ENS)

Abstract: *Cryptosystems based on the knapsack problem were among the first public-key systems to be invented. Their high encryption/decryption rate attracted considerable interest until it was noticed that the underlying knapsacks often had a low density, which made them vulnerable to lattice attacks, both in theory and practice. To prevent low-density attacks, several designers found a subtle way to increase the density beyond the critical density by decreasing the weight of the knapsack, and possibly allowing non-binary coefficients. This approach is actually a bit misleading : we show that low-weight knapsacks do not prevent efficient reductions to lattice problems like the shortest vector problem, they even make reductions more likely. To measure the resistance of low-weight knapsacks, we introduce the novel notion of pseudo-density, and we apply the new notion to the Okamoto-Tanaka-Uchiyama (OTU) cryptosystem from Crypto '00. We do not claim to break OTU and we actually believe that this system may be secure with an appropriate choice of the parameters. However, our research indicates that, in its current form, OTU cannot be supported by an argument based on density. Our results also explain why Schnorr and Hörner were able to solve at Eurocrypt '95 certain high-density knapsacks related to the Chor-Rivest cryptosystem, using lattice reduction.*
Keywords : *Knapsack, Subset Sum, Lattices, Public-Key Cryptanalysis.*

G.1 Introduction

The knapsack (or subset sum) problem is the following : given a set $\{a_1, a_2, \ldots, a_n\}$ of positive integers and a sum $s = \sum_{i=1}^{n} m_i a_i$, where each $m_i \in \{0, 1\}$, recover the m_i's. On the one hand, it is well-known that this problem is NP-hard, and accordingly it is considered to be hard in the worst case. On the other hand, some knapsacks are very easy to solve, such as when the a_i's are the successive powers of two, in which case the problem is to find the binary decomposition of s. This inspired many public-key cryptosystems in the eighties, following the seminal work of Merkle and Hellman [234] :

The Public Key : a set of positive integers $\{a_1, a_2, \ldots, a_n\}$.

The Private Key : a method to transform the presumed hard public knapsack into an easy knapsack.

Encryption : a message $m = (m_1, m_2, \ldots, m_n) \in \{0,1\}^n$ is enciphered into $s = \sum_{i=1}^n m_i a_i$.

However, with the noticeable exception of the Okamoto-Tanaka-Uchiyama (OTU) quantum knapsack cryptosystem from Crypto '00 [277], all proposed knapsack schemes have been broken (see the survey by Odlyzko [274]), either because of the special structure of the public key (like in [251, 346]) leading to key-recovery attacks, or because of the so-called low-density attacks [200, 75] which allow to decrypt ciphertexts.

The *density* of the knapsack is defined as $d = n/\log_2 A$ where $A = \max_{1\leq i\leq n} a_i$. The density cannot be too high, otherwise encryption would not be injective. Indeed, any subset sum $s = \sum_{i=1}^n m_i a_i$ lies in $[0, nA]$, while there are 2^n ways to select the m_i's : if $2^n > nA$, that is, $d > n/(n - \log_2 n)$, there must be a collision $\sum_{i=1}^n m_i a_i = \sum_{i=1}^n m'_i a_i$, On the other hand, when the density is too low, there is a very efficient reduction from the knapsack problem to the lattice shortest vector problem (SVP) : namely, Coster *et al.* [75] showed that if $d < 0.9408\ldots$ (improving the earlier bound $0.6463\ldots$ by Lagarias-Odlyzko [200]), and if the a_i's are chosen uniformly at random over $[0, A]$, then the knapsack problem can be solved with high probability with a single call to a SVP-oracle in dimension n. In practical terms, this means that n must be rather large to avoid lattice attacks (see the survey [267]) : despite their NP-hardness, SVP and other lattice problems seem to be experimentally solvable up to moderate dimension. This is why several articles (e.g. [200, 75, 36, 261]) study efficient provable reductions from problems of cryptographic interest to lattice problems such as SVP or the lattice closest vector problem (CVP).

To thwart low-density attacks, several knapsack cryptosytems like Chor-Rivest [64], Qu-Vanstone [251], Okamoto-Tanaka-Uchiyama [277] use in their encryption process a *low-weight* knapsack instead of a random knapsack : $r = \sum_{i=1}^n m_i^2$ is much smaller than $n/2$, namely sublinear in n. This means that the message space is no longer $\{0,1\}^n$, but a subset with a special structure, such as the elements of $\{0,1\}^n$ with Hamming weight k, in the case of Chor-Rivest [64] or OTU [277]. Alternatively, it was noticed by Lenstra in [210] that such schemes still work with more general knapsacks where the coefficients are not necessarily 0 or 1 : this leads to the *powerline encoding* where the plaintexts are the elements $(m_1, \ldots, m_n) \in \mathbb{N}^n$ such that $\sum_{i=1}^n m_i = k$, where again k is much less than $n/2$. With such choices, it becomes possible to decrease the bit-length of the a_i's so as to increase the density d beyond the critical density : a general subset sum $s = \sum_{i=1}^n m_i a_i$ may then have several solutions, but one is able to detect the correct one because of its special structure. It was claimed that such knapsack schemes would resist lattice attacks.

Our Results. In this article, we show that low-weight knapsacks are still prone to lattice attacks in theory. Extending earlier work of [200, 75, 278], we provide a general framework to study provable reductions from the knapsack problem to two well-known lattice problems : the shortest vector problem (SVP) and the closest vector problem (CVP). The framework relates in a simple manner the success probability of the reductions to the number of integer points in certain high-dimensional spheres, so that the existence of reductions can be assessed based only on combinatorial arguments, without playing directly with lattices. We notice that this number of integer points can be computed numerically for any realistic choice of knapsacks, which makes it possible to analyze the resistance of any concrete choice of parameters for low-weight knapsack cryptosystems, which we illustrate on the Chor-Rivest cryptosystem. We also provide a simple asymptotic bound on the number of integer points to analyze the theoretical resistance of low-weight knapsack cryptosystems. Mazo and Odlyzko [227] earlier gave sharp bounds in certain cases which are well-suited to usual knapsacks, but not to low-weight knapsacks. As a result, we introduce the so-called *pseudo-density* $\kappa = r \log_2 n/\log_2 A$ (where $r = \sum_{i=1}^n m_i^2$) to measure the resistance of low-weight knapsacks to lattice attacks : if κ is sufficiently low, we establish provable reductions to SVP and CVP. This shows that the security of the Okamoto-Tanaka-Uchiyama cryptosystem [277] from Crypto '00 cannot be based on a density argument because its pseudo-density is too low : like NTRU [149], the security requires the hardness of lattice problems. However, we do not claim to break OTU, and we actually believe

that this system may be secure with an appropriate choice of the parameters, due to the gap between lattice oracles and existing lattice reduction algorithms, when the lattice dimension is sufficiently high. Our work shows that the density alone is not sufficient to measure the resistance to lattice attacks : one must also take into account the weight of the solution, which is what the pseudo-density does.

Related Work. Omura and Tanaka [278] showed that the Lagarias-Odlyzko reduction [200] could still apply to practical instantiations of the Chor-Rivest and Okamoto-Tanaka-Uchiyama schemes with binary encoding. However, they relied on the counting techniques of Mazo and Odlyzko [227] which are not tailored to low-weight knapsacks. Hence, they could analyze numerically the resistance of any concrete choice of the parameters, but the asymptotical behaviour was not clear. As a result, it was left open to define an analogue of density to low-weight knapsacks, and it was unknown whether or not the reduction could still work when plaintexts were non-binary strings such as in the powerline encoding. Our work shows that more general encodings like the powerline encoding do not rule out lattice attacks either.

Road map. The paper is organized as follows. In Section G.2 we provide necessary background on lattices and the number of integer points in high-dimensional spheres. We study reductions from knapsacks to the closest lattice vector problem (CVP) in Section G.3, in the case of binary knapsacks and low-weight knapsacks. We then extend those reductions to the shortest lattice vector problem (SVP) in Section G.4. We apply our results to the OTU cryptosystem in Section G.5, and to the Chor-Rivest cryptosystem in Section G.6. Finally, we discuss the significance of our results on the security of low-weight knapsack cryptosystems in Section G.7.

Acknowledgements. This work grew out of investigations carried out by the authors under a contract with NTT. We are grateful to NTT for requesting this research and allowing us to publish our results. The preparation of the paper has in part been supported by the Commission of the European Communities through the IST program under contract IST-2002-507932 ECRYPT. We thank Damien Stehlé and the anonymous referees for their helpful comments.

G.2 Background

G.2.1 Lattices

Let $\|.\|$ and $\langle .,. \rangle$ be the Euclidean norm and inner product of $\mathbb{R}^n$. We refer to the survey [267] for a bibliography on lattices. In this paper, by the term lattice, we actually mean an integral lattice. An integral lattice is a subgroup of $(\mathbb{Z}^n, +)$, that is, a non-empty subset L of $\mathbb{Z}^n$ which is stable by subtraction : $\mathbf{x} - \mathbf{y} \in L$ whenever $(\mathbf{x}, \mathbf{y}) \in L^2$. The simplest lattice is $\mathbb{Z}^n$. It turns out that in any lattice L, not just $\mathbb{Z}^n$, there must exist linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_d \in L$ such that :

$$L = \left\{ \sum_{i=1}^{d} n_i \mathbf{b}_i \mid n_i \in \mathbb{Z} \right\}.$$

Any such d-tuple of vectors $\mathbf{b}_1, \ldots, \mathbf{b}_d$ is called a basis of L : a lattice can be represented by a basis, that is, a matrix. Conversely, if one considers d integral vectors $\mathbf{b}_1, \ldots, \mathbf{b}_d \in \mathbb{Z}^n$, the previous set of all integral linear combinations of the $\mathbf{b}_i$'s is a subgroup of $\mathbb{Z}^n$, and therefore a lattice.

The *dimension* of a lattice L is the dimension d of the linear span of L. Since our lattices are subsets of $\mathbb{Z}^n$, they must have a shortest nonzero vector : In any lattice $L \subseteq \mathbb{Z}^n$, there is at least one nonzero vector $\mathbf{v} \in L$ such that no other nonzero lattice vector has a Euclidean norm strictly smaller than that of $\mathbf{v}$. Finding such a vector $\mathbf{v}$ from an arbitrary basis of L is called the *shortest vector problem* (SVP). Another famous lattice problem is the *closest vector problem* (CVP) : given a basis of $L \subseteq \mathbb{Z}^n$ and a point $\mathbf{t} \in \mathbb{Q}^n$, find a lattice vector $\mathbf{w} \in L$ minimizing the Euclidean norm of $\mathbf{w} - \mathbf{t}$.

It is well-known that as the dimension increases, CVP is NP-hard and SVP is NP-hard under randomized reductions (see [267, 240] for a list of complexity references). However, in practice, the best lattice reduction algorithms give good results up to moderate dimension : we will discuss this issue in Section G.7. This is why it is interesting to study the solvability of various algorithmic problems, when one is given access to a SVP-oracle or a CVP-oracle in moderate dimension. We will call the oracles only once.

G.2.2 Lattice Points in High-Dimensional Spheres

Following [36, 227], we denote by $N(n, r)$ the number of integer points in the n-dimensional sphere of radius $\sqrt{r}$ centered at the origin : that is, $N(n, r)$ is the number of $(x_1, \ldots, x_n) \in \mathbb{Z}^n$ such that $\sum_{i=1}^n x_i^2 \leq r$. Clearly, we have the following induction formula (which was also given in the full version of [36]) :

$$N(n,r) = \begin{cases} 1 & \text{if } n = 0 \text{ and } r \geq 0, \\ 0 & \text{if } n = 0 \text{ and } r < 0, \\ \sum_{j=-\lfloor\sqrt{r}\rfloor}^{\lfloor\sqrt{r}\rfloor} N(n-1, r-j^2) & \text{if } n > 0. \end{cases}$$

This allows to compute $N(n, r)$ numerically when n and r are not too large, since the running time is clearly polynomial in (n, r).

When n grows to infinity, sharp estimates of $N(n, r)$ are known when r is proportional to n (see [227]), in which case $N(n, r)$ is exponential in n. Two particular cases are interesting for the knapsack problem : the techniques of Mazo and Odlyzko [227] show that $N(n, n/2) \leq 2^{c_0 n}$ and $N(n, n/4) \leq 2^{c_1 n}$ where $(c_0, c_1) = (1.54724\ldots, 1.0628\ldots)$. Note that $1/c_0 = 0.6463\ldots$ is the critical density of the Lagarias-Odlyzko attack [200], while $1/c_1 = 0.9409\ldots$ is the critical density of the attack of Coster *et al.* [75]. These techniques are very useful when the ratio r/n is fixed and known, but less so for more general choices of n and r.

For low-weight knapsacks, we need to upper bound $N(n, r)$ when r is sublinear in n, in which case the techniques of Mazo and Odlyzko [227] do not seem well-suited. We will use instead the following simple bound :

Lemma G.7 *For all $n, r \geq 0$:*

$$N(n,r) \leq 2^r \binom{n+r-1}{r}.$$

Proof. Any vector counted by $N(n, r)$ has at most r non-zero coordinates. Therefore, it suffices to bound the number of integer points with positive coordinates, and to multiply by 2^r to take sign into account. To conclude, the number of integer points with positive coordinates and norm less than $\sqrt{r}$ is clearly bounded by the number K_n^r of combinations of r elements among n with repetition. And it is well-known that $K_n^r = \binom{n+r-1}{r}$. □

Corollary G.1 *For all $n, r \geq 0$:*

$$N(n,r) \leq \frac{2^r e^{r(r-1)/(2n)} n^r}{r!}.$$

Proof. It suffices to prove that $r!\binom{n+r-1}{r}/n^r \leq e^{r(r-1)/(2n)}$. We have :

$$\begin{aligned} r!\binom{n+r-1}{r}/n^r &= \frac{(n+r-1)(n+r-2)\cdots(n-1)}{n^r} \\ &\leq \prod_{k=1}^{r-1}(1+\frac{k}{n}) \leq \prod_{k=1}^{r-1} e^{k/n} \leq e^{r(r-1)/(2n)} \end{aligned}$$

□

It follows that if both n and r grow to infinity with a sublinear $r = o(n)$, then $N(n, r) = o(n^r)$ by Stirling's estimate.

G.3 Reducing Knapsacks to the Closest Vector Problem

In this section, we provide a general framework to reduce the knapsack problem to the closest vector problem. This allows us to easily study the case of low-weight knapsacks, which arguably simplifies the approach of [278] based on [200]. The earlier work [200, 75] only considered reductions to the shortest vector problem, but we start with the closest vector problem because it is simpler to understand, and it gives slightly stronger reductions. We will later adapt those results to the shortest vector problem.

We will distinguish two types of knapsacks. The *binary knapsack* problem is the original knapsack problem : given a set $\{a_1, a_2, \ldots, a_n\}$ of positive integers and a sum $s = \sum_{i=1}^{n} m_i a_i$, where each $m_i \in \{0, 1\}$, recover the m_i's. Because of the powerline encoding, we will also be interested in a more general knapsack problem with non-binary coefficients, which we call the *low-weight knapsack* problem : given a set $\{a_1, a_2, \ldots, a_n\}$ of positive integers and a linear combination $s = \sum_{i=1}^{n} m_i a_i$, where each $m_i \in \mathbb{Z}$ and $r = \sum_{i=1}^{n} m_i^2$ is small, recover the m_i's. The case $r = o(n)$ is of particular interest.

G.3.1 A General Framework

Solving the knapsack problem amounts to finding a small solution of an inhomogeneous linear equation, which can be viewed as a closest vector problem in a natural way, by considering the corresponding homogeneous linear equation, together with an arbitrary solution of the inhomogeneous equation. Let $s = \sum_{i=1}^{n} m_i a_i$ be a subset sum, where each $m_i \in \{0, 1\}$.

The link between knapsacks and lattices comes from the homogeneous linear equation. Consider indeed the set L of all integer solutions to the homogeneous equation, that is, L is the set of vectors $(z_1, \ldots, z_n) \in \mathbb{Z}^n$ such that :

$$z_1 a_1 + \cdots + z_n a_n = 0. \tag{G.1}$$

The set L is clearly a subgroup of $\mathbb{Z}^n$ and is therefore a lattice. Its dimension is $n-1$. It is well-known that a basis of L can be computed in polynomial time from the a_i's (see *e. g.* [251] for one way to do so).

Using an extended gcd algorithm, one can compute in polynomial time integers $y_1, \ldots, y_n$ such that

$$s = \sum_{i=1}^{n} y_i a_i. \tag{G.2}$$

The y_i's form an arbitrary solution of the inhomogenous equation. Now the vector $\mathbf{v} = (y_1 - m_1, \ldots, y_n - m_n)$ belongs to L. And this lattice vector is fairly close to the vector $\mathbf{t}_1 = (y_1, \ldots, y_n)$ as the coordinates of the difference are the m_i's. The main idea is that by finding the closest vector to $\mathbf{t}_1$ in the lattice L, one may perhaps recover $\mathbf{v}$ and hence the m_i's. The success probability of our reductions will depend in a simple manner on the number of integer points in high-dimensional spheres.

G.3.2 Binary Knapsacks

In the case of binary knapsacks, the distance between $\mathbf{t}_1$ and $\mathbf{v}$ is roughly $\sqrt{n/2}$. But because $m_i \in \{0, 1\}$, the lattice vector $\mathbf{v}$ is even closer to the vector $\mathbf{t}_2 = (y_1 - 1/2, \ldots, y_n - 1/2)$ for which the distance is exactly $\sqrt{n/4}$. It is this simple fact which explains the difference of critical density

between the Lagarias-Odlyzko reduction [200] and the reduction by Coster *et al.* [75]. The following results are straightforward :

Lemma G.8 *In the case of binary knapsacks, we have :*

1. $\mathbf{v}$ *is a closest vector to* $\mathbf{t}_2$ *in the lattice* L.
2. *If* $\mathbf{v}'$ *is a closest vector to* $\mathbf{t}_2$ *in* L, *then* $\|\mathbf{v}' - \mathbf{t}_2\| = \sqrt{n/4}$ *and* $\mathbf{v}'$ *is of the form* $\mathbf{v}' = (y_1 - m'_1, \ldots, y_n - m'_n)$ *where* $s = \sum_{i=1}^n m'_i a_i$ *and* $m'_i \in \{0, 1\}$.

Proof. The key observation is that elements of the lattice have integer coordinates and that each coordinate contributes to the distance to $\mathbf{t}_2$ by at least $1/2$. □

This gives a deterministic polynomial-time reduction from the binary knapsack problem to the closest vector problem (CVP) in a lattice of dimension $n-1$: this reduction was sketched in the survey [267], and can be viewed as a variant of an earlier reduction by Micciancio [238], who used a different lattice whose dimension was n, instead of $n-1$ here.

Thus, a single call to a CVP-oracle in an $(n-1)$-dimensional lattice automatically gives us a solution to the binary knapsack problem, independently of the value of the knapsack density, but this solution may not be the one we are looking for, unless the unicity of the solution is guaranteed. One particular case for which the unicity is guaranteed is Merkle-Hellman : more generally, for any *traditional* knapsack cryptosystem such that the set of plaintexts is the whole $\{0, 1\}^n$ without decryption failures, a single call to a CVP-oracle is sufficient to decrypt.

It is nevertheless interesting to know when one can guarantee the unicity of the solution for general knapsacks. But if for instance some a_i is a subset sum of other a_j's where $j \in J$, then clearly, all knapsacks involving only a_i and a_ℓ's where $\ell \notin J$ may also be decomposed differently using the a_j's where $j \in J$. This means that to guarantee unicity of solutions in a general knapsack, we may only hope for probabilistic statements, by considering random knapsacks where the a_i's are assumed to be chosen uniformly at random in $[0, A]$:

Theorem G.3 *Let* $(m_1, \ldots, m_n) \in \{0, 1\}^n$. *Let* $a_1, \ldots, a_n$ *be chosen uniformly and independently at random in* $[0, A]$. *Let* $s = \sum_{i=1}^n m_i a_i$. *Let* L *and the* y_i*'s be defined by* (G.1) *and* (G.2). *Let* $\mathbf{c}$ *be a vector in* L *closest to the vector* $\mathbf{t}_2 = (y_1 - 1/2, \ldots, y_n - 1/2)$. *Then the probability that* $\mathbf{c}$ *is not equal to* $(y_1 - m_1, \ldots, y_n - m_n)$ *is less than* $(2^n - 1)/A$.

Proof. By Lemma G.8, $\mathbf{c}$ is of the form $\mathbf{c} = (y_1 - m'_1, \ldots, y_n - m'_n)$ where $s = \sum_{i=1}^n m'_i a_i$ and $m'_i \in \{0, 1\}$. If $\mathbf{c}$ is not equal to $(y_1 - m_1, \ldots, y_n - m_n)$, then $\mathbf{m}' = (m'_1, \ldots, m'_n) \neq \mathbf{m} = (m_1, \ldots, m_n)$. But :

$$\sum_{i=1}^{n} (m_i - m'_i) a_i = 0. \tag{G.3}$$

Since $\mathbf{m} \neq \mathbf{m}'$, there exists i_0 such that $m_{i_0} \neq m'_{i_0}$. For any choice of $(a_i)_{i \neq i_0}$, there exists a unique choice of a_{i_0} satisfying (G.3), since $m_{i_0} - m'_{i_0} = \pm 1$. It follows that for a given $\mathbf{m}' \neq \mathbf{m}$, the probability that $(y_1 - m'_1, \ldots, y_n - m'_n)$ is equal to $\mathbf{c}$ is less than $1/A$. We conclude since the number of $\mathbf{m}'$ is $2^n - 1$. □

This shows that when the density $d = n / \log_2 A$ is < 1, there is with high probability a unique solution, and this solution can be obtained by a single call to a CVP-oracle in dimension $n - 1$.

G.3.3 Low-Weight Knapsacks

We showed that the hidden vector $\mathbf{v} \in L$ related to the knapsack solution was relatively close to two target vectors $\mathbf{t}_1$ and $\mathbf{t}_2$. In fact, $\mathbf{v}$ was a lattice vector closest to $\mathbf{t}_2$: the distance was $\sqrt{n/4}$. In the general binary case, this was better than $\mathbf{t}_1$ for which the distance was expected to be $\sqrt{n/2}$, provided that the Hamming weight of the knapsack was roughly $n/2$. But if the Hamming weight

k is much smaller than $n/2$, then the distance between $\mathbf{m}$ and $\mathbf{t}_1$ is only $\sqrt{k}$, which is much less than $\sqrt{n/4}$. We obtain the following general result regarding low-weight knapsacks (not necessarily binary) :

Theorem G.4 *Let $\mathbf{m} = (m_1, \ldots, m_n) \in \mathbb{Z}^n$. Let $a_1, \ldots, a_n$ be chosen uniformly and independently at random in $[0, A]$. Let $s = \sum_{i=1}^n m_i a_i$. Let L and the y_i's be defined by* (G.1) *and*(G.2). *Let $\mathbf{c}$ be a vector in L closest to the vector $\mathbf{t}_1 = (y_1, \ldots, y_n)$. Then the probability that $\mathbf{c}$ is not equal to $(y_1 - m_1, \ldots, y_n - m_n)$ is less than $N(n, \|\mathbf{m}\|^2)/A$.*

Proof. By definition, $\mathbf{c}$ is of the form $\mathbf{c} = (y_1 - m'_1, \ldots, y_n - m'_n)$ where $s = \sum_{i=1}^n m'_i a_i$ and $m'_i \in \mathbb{Z}$. Let $\mathbf{m}' = (m'_1, \ldots, m'_n)$. Because $\mathbf{c}$ cannot be farther from $\mathbf{t}_1$ than $\mathbf{v}$, $\|\mathbf{m}'\| \leq \|\mathbf{m}\|$. If $\mathbf{c}$ is not equal to $(y_1 - m_1, \ldots, y_n - m_n)$, then $\mathbf{m}' \neq \mathbf{m} = (m_1, \ldots, m_n)$: there exists i_0 such that $m_{i_0} \neq m'_{i_0}$. For any choice of $(a_i)_{i \neq i_0}$, there exists at most one choice of a_{i_0} satisfying (G.3). It follows that for a given $\mathbf{m}' \neq \mathbf{m}$, the probability that $(y_1 - m'_1, \ldots, y_n - m'_n)$ is the closest vector is less than $1/A$. We conclude since the number of $\mathbf{m}'$ is less than $N(n, \|\mathbf{m}\|^2)$, as $\|\mathbf{m}'\| \leq \|\mathbf{m}\|$. □

Note that $N(n, \|\mathbf{m}\|^2)$ can be evaluated numerically from Section G.2.2, so that one can bound the failure probability for any given choice of the parameters.

We saw that $\mathbf{t}_1$ was better than $\mathbf{t}_2$ with low-weight knapsacks, but the choice $\mathbf{t}_1$ can be improved if $k = \sum_{i=1}^n m_i \neq 0$, which is the case of usual knapsacks where all the m_i's are positive. Consider indeed $\mathbf{t}_3 = (y_1 - k/n, y_2 - k/n, \ldots, y_n - k/n)$. Then $\|\mathbf{v} - \mathbf{t}_3\|^2 = \|\mathbf{m}\|^2 - k^2/n$ which is less than $\|\mathbf{v} - \mathbf{t}_1\|^2 = \|\mathbf{m}\|^2$. By replacing $\mathbf{t}_1$ with $\mathbf{t}_3$ in Theorem G.4, the result becomes :

Theorem G.5 *Let $\mathbf{m} = (m_1, \ldots, m_n) \in \mathbb{Z}^n$ and $k = \sum_{i=1}^n m_i$. Let $a_1, \ldots, a_n$ be chosen uniformly and independently at random in $[0, A]$. Let $s = \sum_{i=1}^n m_i a_i$. Let L and the y_i's be defined by* (G.1) *and*(G.2). *Let $\mathbf{c}$ be a vector in L closest to the vector $\mathbf{t}_3 = (y_1 - k/n, \ldots, y_n - k/n)$. Then the probability that $\mathbf{c}$ is not equal to $(y_1 - m_1, \ldots, y_n - m_n)$ is less than $N(n, \|\mathbf{m}\|^2 - k^2/n)/A$.*

If $k = \sum_{i=1}^n m_i$ is proportional to n, Theorem G.5 yields a significant improvement over Theorem G.4 : for instance, if we consider a binary random knapsack for which $k \approx n/2$, Theorem G.5 involves $N(n, n/4)$ instead of $N(n, n/2)$ for Theorem G.4, which is exactly the difference between the critical densities of the Lagarias-Odlyzko reduction [200] and the reduction by Coster *et al.* [75]. However, in the case of low-weight knapsacks where $k = o(n)$, the improvement becomes marginal, as k^2/n is then negligible with respect to $\|\mathbf{m}\|^2$. To simplify the presentation and the discussion, we will therefore rather consider Theorem G.4.

G.4 Reducing Knapsacks to the Shortest Vector Problem

In the previous section, we established reductions from knapsack problems (binary and low-weight) to the closest vector problem. The original lattice attacks [200, 75] on knapsacks only considered reductions to the shortest vector problem (SVP), not to CVP. In this section, we show that our reductions to CVP can be adapted to SVP, thanks to the well-known embedding or (homogenization) method introduced by Kannan (see [178, 240, 250]), which tries to transform an $(n-1)$-dimensional CVP to an n-dimensional SVP. In general, the embedding method is only heuristic, but it can be proved in the special case of knapsack lattices. This is interesting from a practical point of view, because CVP is often solved that way.

We adapt Theorem G.4 to SVP. Again, we let $s = \sum_{i=1}^n m_i a_i$. Let L be the lattice defined by (G.1), and let the $y'_i s$ be defined by (G.2). Let $(\mathbf{b}_1, \ldots, \mathbf{b}_{n-1})$ be a basis of L. We embed L into the n-dimensional lattice L' spanned by $(1, y_1, \ldots, y_n) \in \mathbb{Z}^{n+1}$ and the $n-1$ vectors of the form $(0, \mathbf{b}_i) \in \mathbb{Z}^{n+1}$. We let $\mathbf{m}' = (1, m_1, \ldots, m_n) \in \mathbb{Z}^{n+1}$. By definition, $\mathbf{m}' \in L'$ and its norm is relatively short. The following result lowers the probability that $\mathbf{m}'$ is the shortest vector of L'.

Theorem G.6 *Let* $\mathbf{m} = (m_1, \ldots, m_n) \in \mathbb{Z}^n$. *Let* $a_1, \ldots, a_n$ *be chosen uniformly and independently at random in* $[0, A]$. *Let* $s = \sum_{i=1}^n m_i a_i$. *Let* L', $\mathbf{m}'$ *and the* y_i*'s be defined as previously. Let* $\mathbf{s}$ *be a shortest non-zero vector in* L'. *Then the probability that* $\mathbf{s}$ *is not equal to* $\pm\mathbf{m}'$ *is less than*

$$(1 + 2(1 + \|\mathbf{m}\|^2)^{1/2})N(n, \|\mathbf{m}\|^2)/A.$$

Proof. By definition of L', $\mathbf{s}$ is of the form $\mathbf{s} = (r, ry_1 - z_1, \ldots, ry_n - z_n)$ where $r \in \mathbb{Z}$, and $(z_1, \ldots, z_n) \in L$. Since $\mathbf{s}$ is a shortest vector :

$$\|\mathbf{s}\|^2 \leq \|\mathbf{m}'\|^2 = 1 + \|\mathbf{m}\|^2. \tag{G.4}$$

It follows that $r^2 \leq 1 + \|\mathbf{m}\|^2$. Let $u_i = ry_i - z_i$ and $\mathbf{u} = (u_1, \ldots, u_n)$. We have $\|\mathbf{u}\| \leq \|\mathbf{s}\|$. Notice that :

$$\sum_{i=1}^n (u_i - rm_i)a_i = 0. \tag{G.5}$$

We distinguish two cases. If $r = 0$, then $\mathbf{u} \neq 0$, and it follows that the probability of (G.5) being satisfied for a given $\mathbf{u} \neq 0$ is less than $1/A$. And the number of possible $\mathbf{u}$ is bounded by $N(n, \|\mathbf{m}\|^2)$. Otherwise, $r \neq 0$, and there are at most $2(1 + \|\mathbf{m}\|^2)^{1/2}$ possible values for r. If $\mathbf{s} \neq \pm\mathbf{m}'$, we claim that there exists i_0 such that $u_{i_0} - rm_{i_0} \neq 0$, in which case the probability that (G.5) is satisfied is less than $1/A$. Otherwise, $\mathbf{u} = r\mathbf{m}$: if $|r| > 1$, this would imply that $\|\mathbf{u}\| \geq \|\mathbf{m}\|$, and $\mathbf{s}$ would not be shorter than $\mathbf{m}'$; else $r = \pm 1$, and $\mathbf{u} = \pm\mathbf{m}$ which contradicts $\mathbf{s} \neq \pm\mathbf{m}'$. This concludes the proof. □

Theorem G.6 provides essentially the same bound on the success probability as Theorem G.4, because $\|\mathbf{m}\|$ is negligible with respect to $N(n, \|\mathbf{m}\|^2)$. This means that in the case of low-weight knapsacks, there is no significant difference between the CVP and SVP cases.

Theorem G.6 can be viewed as a generalization of the Lagarias-Odlyzko result [200]. Indeed, if we consider a binary knapsack of Hamming weight $\leq n/2$ (which we may assume without loss of generality), then the failure probability is less than

$$(1 + 2(1 + n/2)^{1/2})N(n, n/2)/A.$$

Since $N(n, n/2) \leq 2^{c_0 n}$ where $c_0 = 1.54724\ldots$ (see Section G.2), it follows that the failure probability of the reduction to SVP is negligible provided that the density $d = n/\log_2 A$ is strictly less than $1/c_0 = 0.6463\ldots$, which matches the Lagarias-Odlyzko result [200].

We omit the details but naturally, the improvement of Theorem G.5 over Theorem G.4 can be adapted to Theorem G.6 as well : $N(n, \|m\|^2)$ would decrease to $N(n, \|m\|^2 - k^2/n)$ where $k = \sum_{i=1}^n m_i$, provided that one subtracts k/n to both y_i and m_i in the definition of L' and $\mathbf{m}'$. In the particular case of binary knapsacks, this matches the result of Coster *et al.* [75] : because $N(n, n/4) \leq 2^{c_1 n}$ where $c_1 = 1.0628\ldots$, the failure probability would be negligible provided that the knapsack density is less than $1/c_1 = 0.9409\ldots$ Whereas there was almost no difference between the CVP reduction and the SVP reduction for low-weight knapsacks, there is a difference in the case for binary knapsacks : in Theorem G.3, the critical density was 1 and not $1/c_1$. And that would not have changed if we had transformed the CVP-reduction of Theorem G.3 (instead of that of Theorem G.5) into a probabilistic reduction to SVP. This is because Lemma G.8 used in Theorem G.3 (but not in Theorem G.5) has no analogue in the SVP setting, which explains why the result with a CVP-oracle is a bit stronger than with a SVP-oracle : there are more parasites with SVP.

In other words, the framework given in Section G.3 revisits the SVP reductions of Lagarias-Odlyzko [200] and Coster et al. [75]. By applying the embedding technique, we obtain the same critical densities when transforming our CVP reductions of Theorem G.4 and G.5 into SVP reductions.

G.5 Application to the OTU Cryptosystem

In this section, we apply the results of Sections G.2, G.3 and G.4 to the Okamoto-Tanaka-Uchiyama cryptosystem [277] from Crypto 2000.

G.5.1 Description of OTU

The OTU cryptosystem is a knapsack cryptosystem where the knapsack has a hidden structure based on discrete logarithms like the Chor-Rivest scheme [64], but where no information on the DL group leaks, thwarting attacks like [346]. The key generation of OTU requires the extraction of discrete logarithms : if quantum computers are available, one can apply Shor's quantum algorithm, otherwise one uses groups with a special structure (e.g. groups of smooth order) so that DL is tractable.

The knapsack $(a_1, \ldots, a_n)$ used by OTU has a special structure. Let $A = \max_{1 \leq i \leq n} a_i$. To allow decryption, it turns out that A is such that $A \geq p^k$ for some integers $p, k > 1$, and p is such that there are at least n coprime numbers $\leq p$, which implies that $p \geq n$, and therefore $A \geq n^k$, and $\log_2 A$ is at least linear in k. The OTU scheme allows two kinds of encoding :

- The binary encoding, where the plaintexts are all $(m_1, \ldots, m_n) \in \{0,1\}^n$ such that $\sum_{i=1}^n m_i = k$.
- The powerline encoding [210], where the plaintexts are all $(m_1, \ldots, m_n) \in \mathbb{N}^n$ such that $\sum_{i=1}^n m_i = k$.

There is no concrete choice of parameters proposed in [277]. However, it was pointed out on page 156 of [277] that the choice $k = 2^{(\log n)^c}$ where c is a constant < 1 would have interesting properties. We will pay special attention to that case since it is the only asymptotical choice of k given in [277], but we note from the discussion in [277, Section 3.4] that the scheme could tolerate larger values of k, up to maybe a constant times $n/\log n$. Perhaps the main drawback with larger values of k is the keysize, as the storage of the knapsack is $\Omega(nk)$ bits, which is then essentially quadratic if $k = n/\log n$. What is clear is that k is at most $O(n/\log n)$: indeed the density in OTU is $O(n/(k \log n))$, and the density must be lower bounded by a constant > 0 to ensure the hardness of the knapsack, which implies that $k = O(n/\log n)$. This means that we should study two cases : the suggested case $k = 2^{(\log n)^c}$ where c is a constant < 1, and the extreme case $k = O(n/\log n)$.

G.5.2 Resistance to Low-Density Attacks

The parameter A can be chosen as small as $O(p^k)$ and p can be as small as $n \log n$. For the suggested case $k = 2^{(\log n)^c}$, we have $\log A = O(k \log p) = o(n)$. It follows that the usual density $d = n/\log_2 A$ grows to infinity, which is why it was claimed in [277] that OTU prevents usual lattice attacks [200, 75]. However, this density argument is misleading because the weight k is sublinear in n.

Let $\mathbf{m} = (m_1, \ldots, m_n)$ and $s = \sum_{i=1}^n m_i a_i$. Theorems G.6 and G.4 provide efficient reductions from knapsacks to SVP and CVP, provided that $N(n, \|\mathbf{m}\|^2)$ is negligible with respect to A.

With the binary encoding, we have $\|\mathbf{m}\|^2 = k$, and therefore $N(n, \|\mathbf{m}\|^2) = N(n, k)$. We know that due to the choice of k in OTU (even in the extreme case), we have $k = o(n)$ with k growing to infinity. Corollary G.1 then implies that $N(n, k) = o(n^k)$, and therefore $N(n, k)/A = o(1)$ since $A \geq n^k$. Hence Theorems G.6 and G.4 provide efficient reductions (with success probability asymptotically close to 1) to SVP and CVP in dimension n, provided that $k = o(n)$, which is a necessary requirement for OTU.

We now show that the powerline encoding does not significantly improve the situation, even though a plaintext $\mathbf{m}$ with the powerline encoding only satisfies $k \leq \|\mathbf{m}\|^2 \leq k^2$. If $\|\mathbf{m}\|^2$ was close to k^2, rather than k, Corollary G.1 on $N(n, \|\mathbf{m}\|^2)$ would not allow us to conclude, because n^{k^2} would

dominate A. The following result shows that $\|\mathbf{m}\|^2$ is on the average much closer to k, as in the binary encoding :

Theorem G.7 *There exists a computable constant $\alpha > 0$ such that the following holds. Let $1 \leq k \leq n$ and $y = (k-1)/n$. Let $\mathbf{m} = (m_1, \ldots, m_n) \in \mathbb{N}^n$ be chosen uniformly at random such that $\sum_{i=1}^n m_i = k$. Then the expected value of $\|\mathbf{m}\|^2$ satisfies :*

$$E(\|\mathbf{m}\|^2) \leq k(1 + \alpha y).$$

Proof. As in the proof of Lemma G.7, let K_n^k denote the number of combinations of k elements among n with repetition : $K_n^k = \binom{n+k-1}{k} = \binom{n+k-1}{n-1}$. We have :

$$\begin{aligned} E(\|\mathbf{m}\|^2) &= nE(m_i^2) = n\sum_{x=1}^{k} x^2 \frac{K_{n-1}^{k-x}}{K_n^k} \\ &= n\sum_{x=1}^{k} x^2 \frac{k(k-1)\cdots(k-x+1)\times(n-1)}{(n+k-1)(n+k-2)\cdots(n+k-x-1)}. \end{aligned}$$

Let :

$$s(n,x,k) = n(n-1)x^2 \frac{k(k-1)\cdots(k-x+1)}{(n+k-1)(n+k-2)\cdots(n+k-x-1)},$$

so that $E(\|\mathbf{m}\|^2) = \sum_{x=1}^{k} s(n,x,k)$. We will see that the first term dominates in this sum :

$$s(n,1,k) = \frac{n(n-1)k}{(n+k-1)(n+k-2)} \leq k.$$

We now bound $s(n,x,k)$ for all $2 \leq x \leq k$:

$$\begin{aligned} s(n,x,k) &\leq kx^2 \frac{(k-1)(k-2)\cdots(k-x+1)}{(n+k-1)(n+k-2)\cdots(n+k-x+1)} \\ &= kx^2 \prod_{u=k-x+1}^{k-1} \frac{u}{n+u} \leq kx^2 \left(\frac{k-1}{n+k-1}\right)^{x-1} \\ &\leq kx^2 \left(\frac{y}{1+y}\right)^{x-1} \text{ with } y = \frac{k-1}{n}. \end{aligned}$$

Hence, by separating the first two terms in the sum :

$$E(\|\mathbf{m}\|^2) \leq k\left(1 + \frac{4y}{1+y} + \sum_{x=3}^{k} x^2 \left(\frac{y}{1+y}\right)^{x-1}\right).$$

Because $1 \leq k \leq n$, we have $0 \leq y < 1$ and $0 \leq y/(1+y) < 1/2$. Thus, we only need to bound the series :

$$f(y) = \sum_{x=3}^{\infty} x^2 \left(\frac{y}{1+y}\right)^{x-1}.$$

A short derivative computation shows that for any $0 \leq z < 1/2$, the function $x \mapsto x^2 z^{x-1}$ decreases over $x \geq 3$, because $2 + 3\ln(1/2) < 0$. Therefore, letting $z = y/(1+y)$, we obtain for all $k > 1$:

$$f(y) \leq \int_2^{\infty} x^2 z^{x-1} dx = \left[\frac{z^{x-1}}{\ln z}\left(x^2 - \frac{2x}{\ln z} + \frac{2}{\ln^2 z}\right)\right]_2^{\infty} = \frac{-z}{\ln z}\left(4 - \frac{4}{\ln z} + \frac{2}{\ln^2 z}\right).$$

Since $z \leq 1/2$, it follows that one can compute an absolute constant $\beta > 0$ such that for all $k > 1$, $f(y) \leq \beta z$, which in fact also holds when $k = 1$, that is, $z = 0$. Hence for all $1 \leq k \leq n$:

$$E(\|\mathbf{m}\|^2) \leq k\left(1 + \frac{4y}{1+y} + \beta z\right) \leq k(1 + (4+\beta)y).$$

This concludes the proof with $\alpha = 4 + \beta$. □

When $k = o(n)$, we have $y = o(1)$ and the upper bound becomes $k(1 + \alpha y) = k(1 + o(1))$, which already shows that with the powerline encoding, the expected value of $\|\mathbf{m}\|^2$ is essentially k, rather than k^2. This suggests that $N(n, \|\mathbf{m}\|^2)$ will on the average still be negligible with respect to A. But Theorem G.7 allows us to give a sharper estimate. In the extreme case of OTU, we have $k = O(n/\log n)$ growing to infinity, so $y = O(1/\log n)$ and the upper bound becomes $r = k(1 + O(1/\log n))$. By Corollary G.1 :

$$N(n, r)/A \leq \frac{2^r e^{r(r-1)/(2n)} n^r}{r! n^k}.$$

Here, $r^2/n = kO(n/\log n)(1 + O(1/\log n))/n = O(k/\log n)$ therefore :

$$2^r e^{r(r-1)/(2n)} = O(1)^k.$$

And $n^r = n^{k(1+O(1/\log n))} = n^k \times (n^{O(1/\log n)})^k \leq n^k \times O(1)^k$. Hence :

$$N(n, r)/A \leq \frac{O(1)^k}{r!} = o(1).$$

Thus, the reductions of Theorems G.6 and G.4 succeed with overwhelming probability even with the powerline encoding, even if the extreme choice of k in OTU is considered. This question was left open in [278].

Although we believe that the OTU cryptosystem may be secure with an appropriate choice of the parameters, our results indicate that in its current form, it cannot be supported by an argument based on density that would protect the system against a single call to an SVP oracle or a CVP oracle.

G.5.3 The Pseudo-Density

We now explain why in the case of low-weight knapsacks, Theorems G.6 and G.4 suggest to replace the usual density $d = n/\log_2 A$ by a pseudo-density defined by $\kappa = r \log_2 n/\log_2 A$, where r is an upper bound on $\|\mathbf{m}\|^2$, $\mathbf{m}$ being the knapsack solution.

Theorems G.6 and G.4 showed that a low-weight knapsack could be solved with high probability by a single call to a SVP-oracle or a CVP-oracle, provided that $N(n, r)/A$ was small. Corollary G.7 shows that :

$$N(n, r)/A \leq \frac{2^r e^{r(r-1)/(2n)}}{r!} \times \frac{n^r}{A}.$$

The left-hand term $2^r e^{r(r-1)/(2n)}/r!$ tends to 0 as r grows to ∞, provided that $r = O(n)$. The right-hand term n^r/A is $2^{r \log_2 n - \log_2 A}$. This shows that if the pseudo-density κ is ≤ 1, then the right-hand term will be bounded, and therefore the low-weight knapsack can be solved with high probability by a single call to either a SVP-oracle or a CVP-oracle. On the other hand, if the pseudo-density κ is larger than 1, it will not necessarily mean that the previous upper bound does not tend to zero, as there might be some compensation between the left-hand term and the right-hand term.

Consider for instance the case of OTU with binary encoding. For any choice of k, the pseudo-density $\kappa = k \log_2 n/\log_2 A$ is ≤ 1 because $A \geq n^k$ due to decryption requirements. Therefore there is a reduction to SVP and CVP with probability asymptotically close to 1. On the other

hand, if we consider the powerline encoding with an extreme case of k, the pseudo-density becomes $\kappa = k(1 + O(1/\log n)) \log_2 n / \log_2 A \leq 1 + O(1/\log n)$ which could perhaps be slightly larger than 1. Nevertheless, the computation of the previous section showed that $N(n, r)/A$ was still $o(1)$. Thus, the pseudo-density is a good indicator, but it may not suffice to decide in critical cases.

G.6 Application to the Chor-Rivest Cryptosystem

The Chor-Rivest cryptosystem [64] is another low-weight knapsack cryptosystem, which survived for a long time until Vaudenay [346] broke it, for all the parameter choices proposed by the authors in [64]. Vaudenay used algebraic techniques specific to the Chor-Rivest scheme, which do not apply to OTU. His attack recovers the private key from the public key. Schnorr and Hörner [308] earlier tried to decrypt Chor-Rivest ciphertexts by solving the underlying low-weight knapsack using an improved lattice reduction method which they introduced. They succeeded for certain choices of moderate parameters, but failed for the parameter choices proposed in [64]. Despite the fact that the Chor-Rivest scheme is broken, it is an interesting case with respect to lattice attacks, and this is why we apply our results to this scheme.

G.6.1 Description

We give a brief description of the Chor-Rivest cryptosystem [64]. One selects a small prime q and an integer k such that one can compute discrete logarithms in $\mathrm{GF}(q^k)$. One computes the discrete logarithms $b_1, \ldots, b_q \in \mathbb{Z}_{q^k-1}$ of certain well-chosen elements in $\mathrm{GF}(q^k)$, to ensure decryption. The elements of the knapsack are $a_i = b_i + d$ where d is an integer chosen uniformly at random in $\mathbb{Z}_{q^k-1}$. The set of plaintexts is the subset of all $(m_1, \ldots, m_q) \in \{0, 1\}^q$ having Hamming weight k, and the encryption of $(m_1, \ldots, m_q)$ is :

$$s = \sum_{i=1}^{q} a_i m_i \pmod{q^k - 1}.$$

The public key consists of the q, k and the a_i's.

Strictly speaking, Chor-Rivest involves a modular knapsack problem (modulo $q^k - 1$), rather than the initial knapsack problem. The density of the Chor-Rivest knapsack is $d = q/(k \log q)$, which can therefore be rather high for appropriate choices of q and k. But all our results on the knapsack problem we have discussed can be adapted to the modular knapsack problem. First of all, notice that a modular knapsack can be transformed into a basic knapsack if one can guess the hidden multiple of $q^k - 1$ involved, that is, if one knows the integer ℓ such that :

$$s + \ell(q^k - 1) = \left(\sum_{i=1}^{q} a_i m_i \right).$$

Clearly, ℓ can be exhaustively searched, and it is very close to k. In the worst-case for our reductions to lattice problems, the number of oracle calls will increase very slightly.

Alternatively, one can adapt the lattice used in our framework. Consider a modular knapsack $s = \sum_{i=1}^{n} a_i\, m_i \pmod{A}$. We replace the lattice L defined by (G.1) by the set L of vectors $(z_1, \ldots, z_n) \in \mathbb{Z}^n$ such that :

$$z_1 a_1 + \cdots + z_n a_n \equiv 0 \pmod{A}. \tag{G.6}$$

The set L is a subgroup of $\mathbb{Z}^n$ and is therefore a lattice. Its dimension is n, rather than $n - 1$. It is again well-known that a basis of L can be computed in polynomial time. This time, we compute in polynomial time integers $y_1, \ldots, y_n$ such that

$$s \equiv \sum_{i=1}^{n} y_i a_i \pmod{A}. \tag{G.7}$$

Tab. G.1 – Application to the Chor-Rivest parameters proposed in [64].

Value of (q, k)	(197,24)	(211,24)	(256,25)	(243,24)
Value of $N(q,k)/q^k$	2^{-57}	2^{-57}	2^{-60}	2^{-57}

Tab. G.2 – Application to the Chor-Rivest parameters attacked in [308].

Value of (q, k)	(103,12)	(151,16)
Value of $N(q,k)/q^k$	2^{-18}	2^{-29}

All of our results, such as Theorems G.3–G.6, can then be adapted to modular knapsacks provided some obvious minor changes, which we omit. For instance, in the statements of Theorems G.3–G.6, the uniform distribution must be over $[0, A[$, and we let $s = \sum_{i=1}^{n} a_i \; m_i \pmod A$. Naturally, equations (G.1) and (G.2) must be replaced respectively by equations (G.6) and (G.7).

G.6.2 Application

By definition, the pseudo-density of the Chor-Rivest knapsack (with binary encoding) is $\kappa = k \log_2 q / \log_2(q^k) = 1$. We thus conclude that the low-weight knapsack problems arising from the Chor-Rivest cryptosystem can be efficiently reduced to SVP and CVP with probability close to 1. In retrospect, it is therefore not surprising that Schnorr and Hörner [308] were able to solve certain Chor-Rivest knapsacks using lattice reduction.

Concretely, we can even compute upper bounds on the failure probability of the reduction for the parameters proposed in [64] and the ones used in [308], using numerical values of $N(n, r)$, as explained in Section G.2.2. The numerical results are summarized in Tables G.1 and G.2. Thus, if one had access to SVP-oracles or CVP-oracles in dimension roughly 200–250, one could decrypt Chor-Rivest ciphertexts with overwhelming probability for its proposed parameters.

G.7 Impact on the Security of Low-Weight Knapsack Cryptosystems

We have established efficient provable reductions from the low-weight knapsack problem to two well-known lattice problems : SVP and CVP. However, we do not claim to break low-weight knapsack cryptosystems like OTU. This is because there is an experimental and theoretical gap between lattice oracles for SVP/CVP and existing lattice reduction algorithms (see [267] for a list of references), as the lattice dimension increases. The state-of-the-art in lattice reduction suggests that exact SVP and CVP can only be solved up to moderate dimension, unless the lattice has exceptional properties (such as having one extremely short non-zero vector compared to all the other vectors).

To roughly estimate the hardness of SVP/CVP in a m-dimensional lattice of volume V, lattice practitioners usually compare $V^{1/m}\sqrt{m}$ with a natural quantity related to the expected solution : for SVP, the quantity is the norm of the expected shortest vector, while for CVP, it is the distance between the target vector and the lattice. If the ratio is not large, it means that the solution is not exceptionally small : SVP and CVP become intractable in practice if the dimension is sufficiently high. In the case of a knapsack defined by integers $a_1, \ldots, a_n$, the work of [251] on the so-called orthogonal lattices show as a simple particular case that the lattice L defined by (G.1) has volume $V = (\sum_{i=1}^{n} a_i^2)^{1/2} / \gcd(a_1, \ldots, a_n)$. Thus, with overwhelming probability, $V \approx A = \max_i a_i$. Since

the dimension of L is $n-1$, we need to consider $V^{1/(n-1)} \approx 2^{(\log_2 A)/(n-1)} \approx 2^{1/d}$ where d is the usual knapsack density. The quantity is thus $V^{1/(n-1)}\ \sqrt{n-1} \approx 2^{1/d}\sqrt{n}$. When dealing with a low-weight knapsack of weight $r = \sum_{i=1}^{n} m_i^2$, this quantity is not particularly large compared to the quantity $\sqrt{r}$ corresponding ot the solution, unless r is extremely small. This indicates that by taking a sufficiently high dimension n and a not too small r (which is also important to avoid simple dimension reduction methods like [225]), the corresponding lattice problems should be hard.

One may wonder how to select the lattice dimension to guarantee the hardness of SVP and CVP in practice. Current experimental records in lattice computations seem to depend on the type of lattices. For instance, Schnorr and Hörner [308], using what is still the best lattice reduction algorithm known in practice, failed to decrypt Chor-Rivest ciphertexts for its suggested parameters, which correspond to a lattice dimension around 200–250. Bleichenbacher and Nguyen [36] reported similar problems with a dense 160-dimensional lattice. On the other hand, Nguyen [250] broke the GGH-challenge in dimension 350, but not in dimension 400. The record computation for breaking the NTRU cryptosystem [149] is a SVP computation in dimension 214 by May (see [225]), while the smallest NTRU parameter currently proposed corresponds to a 502-dimensional lattice. Thus, in order to propose concrete parameters for OTU, it would be useful to gather experimental data with the best reduction algorithms known (keeping track of recent development such as [264]). Besides, SVP and CVP instances arising from knapsack problems could serve as a useful benchmark to test and design new lattice reduction algorithms.

Failles de conception

Cette partie illustre des failles de conception, en prenant pour exemple le cryptosystème NTRU et des variantes de RSA. Elle se compose de quatre articles :

Page 243 : version complète de [259] incorporant [256].

> **Learning a Parallelepiped : Cryptanalysis of GGH and NTRU Signatures**, EUROCRYPT 2006 *Best Paper Award*
> Phong Q. Nguyen et Oded Regev
>
> *Cet article présente la première attaque pratique contre la version de base du schéma de signature de Goldreich-Goldwasser-Halevi (GGH). Cette attaque est particulièrement efficace dans le cas particulier de la signature NTRU : 400 signatures suffisent à retrouver la clef secrète. Bien que les signatures GGH et NTRU s'appuient sur des problèmes de réseaux euclidiens, l'attaque ne fait curieusement pas appel aux réseaux : elle repose sur des méthodes statistiques et des problèmes d'optimisation à plusieurs variables utilisés notamment en traitement du signal.*

Page 259 : référence [158].

> **The Impact of Decryption Failures on the Security of NTRU Encryption**, CRYPTO 2003
> Nick Howgrave-Graham, Phong Q. Nguyen, David Pointcheval, John Proos, Joseph Silverman, Ari Singer et William Whyte
>
> *Le chiffrement NTRU a une propriété exceptionnelle : selon le choix des paramètres, l'algorithme de déchiffrement peut échouer à retrouver le message. L'article présente des attaques à chiffré choisi exploitant ces erreurs de déchiffrement. Depuis, la génération de paramètres de NTRU a été modifiée de façon à empêcher toute erreur de déchiffrement.*

Page 275 : référence [260].

> **On the Insecurity of a Server-Aided RSA Protocol**, ASIACRYPT 2001
> Phong Q. Nguyen et Igor E. Shparlinski
>
> *Cet article présente une attaque polynomiale prouvée contre un schéma de signature RSA assistée pour lequel était annoncée une preuve de sécurité dans le modèle dit générique.*

Page 287 : référence [87].

> **Cryptanalysis of the RSA Schemes with Short Secret Exponent from Asiacrypt '99**, ASIACRYPT 2000
> Glenn Durfee et Phong Q. Nguyen
>
> *Cet article montre comment attaquer des variantes de RSA qui modifiaient la génération usuelle de clefs afin d'accélérer le déchiffrement, tout en évitant les attaques usuelles. L'attaque adapte les méthodes de recherche de petites racines d'équations polynomiales initiées par Coppersmith.*

Annexe H

Learning a Parallelepiped : Cryptanalysis of GGH and NTRU Signatures

EUROCRYPT 2006

Best Paper Award

Version complète de [259] incorporant [256], avec Oded Regev (Univ. Tel-Aviv)

Abstract: *Lattice-based signature schemes following the Goldreich-Goldwasser-Halevi (GGH) design have the unusual property that each signature leaks information on the signer's secret key, but this does not necessarily imply that such schemes are insecure. At Eurocrypt '03, Szydlo proposed a potential attack by showing that the leakage reduces the key-recovery problem to that of distinguishing integral quadratic forms. He proposed a heuristic method to solve the latter problem, but it was unclear whether his method could attack real-life parameters of GGH and* NTRUSign. *Here, we propose an alternative method to attack signature schemes à la GGH, by studying the following learning problem : given many random points uniformly distributed over an unknown n-dimensional parallelepiped, recover the parallelepiped or an approximation thereof. We transform this problem into a multivariate optimization problem that can provably be solved by a gradient descent. Our approach is very effective in practice : we present the first successful key-recovery experiments on* NTRUSign-*251 without perturbation, as proposed in half of the parameter choices in NTRU standards under consideration by IEEE P1363.1. Experimentally, 400 signatures are sufficient to recover the* NTRUSign-*251 secret key, thanks to symmetries in NTRU lattices. We are also able to recover the secret key in the signature analogue of all the GGH encryption challenges.*

H.1 Introduction

Inspired by the seminal work of Ajtai [4], Goldreich, Goldwasser and Halevi (GGH) proposed at Crypto '97 [118] a lattice analogue of the coding-theory-based public-key cryptosystem of McEliece [228]. The security of GGH is related to the hardness of approximating the closest vector problem (CVP) in a lattice. The GGH article [118] focused on encryption, and five encryption challenges were issued on the Internet [117]. Two years later, Nguyen [250] found a flaw in the original GGH encryption scheme, which allowed to solve four out of the five GGH challenges, and obtain partial information on the last one. Although GGH might still be secure with an appropriate choice of the

parameters, its efficiency compared to traditional public-key cryptosystems is perhaps debatable : it seems that a very high lattice dimension is required, while the keysize grows roughly quadratically in the dimension (even when using the improvement suggested by Micciancio [239]). The only lattice-based scheme known that can cope with very high dimension is NTRU [149] (see the survey [267]), which can be viewed as a very special instantiation of GGH with a "compact" lattice and different encryption/decryption procedures (see [239, 240]).

In [118], Goldreich *et al.* described how the underlying principle of their encryption scheme could also provide a signature scheme. The resulting GGH signature scheme did not attract much interest in the research literature until the company NTRU Cryptosystems proposed a relatively efficient signature scheme called NTRUSign [146], based exactly on the GGH design but using the compact NTRU lattices. NTRUSign had a predecessor NSS [150] less connected to the GGH design, and which was broken in [111, 112]. Gentry and Szydlo [112] observed that the GGH signature scheme has an unusual property (compared to traditional signature schemes) : each signature released leaks information on the secret key, and once sufficiently many signatures have been obtained, a certain Gram matrix related to the secret key can be approximated. The fact that GGH signatures are not zero-knowledge can be explained intuitively as follows : for a given message, many valid signatures are possible, and the one selected by the secret key says something about the secret key itself.

This information leakage does not necessarily prove that such schemes are insecure. Szydlo [338] proposed a potential attack on GGH based on this leakage (provided that the exact Gram matrix could be obtained), by reducing the key-recovery problem to that of distinguishing integral quadratic forms. It is however unknown if the latter problem is easy or not, although Szydlo proposed a heuristic method based on existing lattice reduction algorithms applied to quadratic forms. As a result, it was unclear if Szydlo's approach could actually work on real-life instantiations of GGH and NTRUSign. The paper [148] claims that, for NTRUSign without perturbation, significant information about the secret key is leaked after 10,000 signatures. However, it does not identify any attack that would require less than 100 million signatures (see [146, Sect. 4.5] and [148, Sect. 7.2 and App. C]).

Our Results. In this article, we present a new key-recovery attack on lattice-based signature schemes following the GGH design, including NTRUSign. The basic observation is that a list of known pairs $(message, signature)$ gives rise to the following learning problem, which we call the hidden parallelepiped problem (HPP) : given many random points uniformly distributed over an unknown n-dimensional parallelepiped, recover the parallelepiped or an approximation thereof (see Fig. H.1). We transform the HPP into a multivariate optimization problem based on the fourth moment (also known as *kurtosis*) of one-dimensional projections. This problem can be solved by a gradient descent. Our approach is very effective in practice : we present the first successful key-recovery experiments on NTRUSign-251 without perturbation, as proposed in half of the parameter choices in the NTRU standards [63] being considered by IEEE P1363.1 [164] ; experimentally, 400 signatures are enough to disclose the NTRUSign-251 secret key. We have also been able to recover the secret key in the signature analogue of all five GGH encryption challenges ; the GGH case requires significantly more signatures because NTRU lattices have special properties which can be exploited by the attack. When the number of signatures is sufficiently high, the running time of the attack is only a fraction of the time required to generate all the signatures. From the theoretical side, we are able to show that under a natural assumption on the distribution of signatures, an attacker can recover the secret key of NTRUSign and the GGH challenges in polynomial time, given a polynomial number of signatures of random messages.

Related Work. Interestingly, it turns out that the HPP (as well as related problems) have already been looked at by people dealing with what is known as *Independent Component Analysis* (ICA) (see, e.g., the book by Hyvärinen *et al.* [160]). ICA is a statistical method whose goal is to find directions of independent components, which in our case translates to the n vectors that define the parallelepiped. It has many applications in statistics, signal processing, and neural network research. To the best of

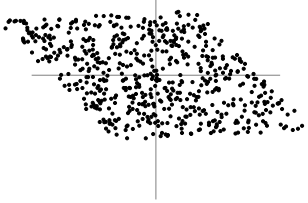


FIG. H.1 – The Hidden Parallelepiped Problem in dimension two.

our knowledge, this is the first time ICA is used in cryptanalysis.

There are several known algorithms for ICA, and most are based on a gradient method such as the one we use in our algorithm. Our algorithm is closest in nature to the FastICA algorithm proposed in [161], who also considered the fourth moment as a goal function. We are not aware of any rigorous analysis of these algorithms ; the proofs we have seen often ignore the effect of errors in approximations. Finally, we remark that the ICA literature offers other, more general, goal functions that are supposed to offer better robustness against noise etc. We have not tried to experiment with these other functions, since the fourth moment seems sufficient for our purposes.

Another closely related result is that by Frieze *et al.* [100], who proposed a polynomial-time algorithm to solve the HPP (and generalizations thereof). Technically, their algorithm is slightly different from those present in the ICA literature as it involves the Hessian, in addition to the usual gradient method. They also claim to have a fully rigorous analysis of their algorithm, taking into account the effect of errors in approximations. Unfortunately, most of the analysis is missing from the preliminary version, and to the best of our knowledge, a full version of the paper has never appeared.

OPEN PROBLEMS. It would be interesting to study natural countermeasures against our attack. To date, the most efficient countermeasures known are perturbation techniques (as suggested by [148, 63, 147]), which modify the signature generation process in such a way that the hidden parallelepiped is replaced by a more complicated set. For instance, the second half of parameter choices in NTRU standards [63] involves exactly a single perturbation. In this case, the attacker has to solve an extension of the hidden parallelepiped problem in which the parallelepiped is replaced by the Minkowski sum of two hidden parallelepipeds : the lattice spanned by one of the parallelepipeds is public, but not the other one. The drawbacks of perturbations is that they slow down signature generation, increase both the size of the secret key, and the distance between the signature and the message.

ROAD MAP. The paper is organized as follows. In Section H.2, we provide notation and necessary background on lattices, GGH and NTRUSIGN. In Section H.3, we introduce the hidden parallelepiped problem, and explain its relationship to GGH-type signature schemes. In Section H.4, we present a method to solve the hidden parallelepiped problem. In Section H.5, we present experimental results obtained with the attack on real-life instantiations of GGH and NTRUSIGN. In Section H.6, we provide a theoretical analysis of the main parts of our attack.

H.2 Background and Notation

Vectors of $\mathbb{R}^n$ will be row vectors denoted by bold lowercase letters such as $\mathbf{b}$, and we will use row representation for matrices. For any ring $\mathcal{R}$, $\mathcal{M}_n(\mathcal{R})$ will denote the ring of $n \times n$ matrices with entries in $\mathcal{R}$. The group of $n \times n$ invertible matrices with real coefficients will be denoted by $GL_n(\mathbb{R})$ and $O_n(\mathbb{R})$ will denote the subgroup of orthogonal matrices. The transpose of a matrix M will be denoted by M^t, so M^{-t} will mean the inverse of the transpose. The notation $\lceil x \rfloor$ denotes a closest integer to x. Naturally, $\lceil \mathbf{b} \rfloor$ will denote the operation applied to all the coordinates of $\mathbf{b}$. If X is a random variable, we will denote by $\mathrm{Exp}[X]$ its expectation. The gradient of a function f from $\mathbb{R}^n$ to $\mathbb{R}$ will be denoted by $\nabla f = (\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n})$.

H.2.1 Lattices

Let $\|\cdot\|$ and $\langle\cdot,\cdot\rangle$ be the Euclidean norm and inner product of $\mathbb{R}^n$. We refer to the survey [267] for a bibliography on lattices. In this paper, by the term lattice, we mean a full-rank discrete subgroup of $\mathbb{R}^n$. The simplest lattice is $\mathbb{Z}^n$. It turns out that in any lattice L, not just $\mathbb{Z}^n$, there must exist linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n \in L$ such that :

$$L = \left\{ \sum_{i=1}^{n} n_i \mathbf{b}_i \mid n_i \in \mathbb{Z} \right\}.$$

Any such n-tuple of vectors $[\mathbf{b}_1, \ldots, \mathbf{b}_n]$ is called a basis of L : an n-dimensional lattice can be represented by a basis, that is, a matrix of $GL_n(\mathbb{R})$. Reciprocally, any matrix $B \in GL_n(\mathbb{R})$ spans a lattice : the set of all integer linear combinations of its rows, that is, $\mathbf{m}B$ where $\mathbf{m} \in \mathbb{Z}^n$. The *closest vector problem* (CVP) is the following : given a basis of $L \subseteq \mathbb{Z}^n$ and a target $\mathbf{t} \in \mathbb{Q}^n$, find a lattice vector $\mathbf{v} \in L$ minimizing the distance $\|\mathbf{v} - \mathbf{t}\|$. If we denote by d that minimal distance, then approximating CVP to a factor k means finding $\mathbf{v} \in L$ such that $\|\mathbf{v} - \mathbf{t}\| \leq kd$. A measurable part D of $\mathbb{R}^n$ is said to be a *fundamental domain* of a lattice $L \subseteq \mathbb{R}^n$ if the sets $\mathbf{b} + D$, where $\mathbf{b}$ runs over L, cover $\mathbb{R}^n$ and have pairwise disjoint interiors. If B is a basis of L, then the parallelepiped $\mathcal{P}_{1/2}(B) = \{\mathbf{x}B : \mathbf{x} \in [-1/2, 1/2]^n\}$ is a fundamental domain of L. All fundamental domains of L have the same measure : the volume $\mathrm{vol}(L)$ of the lattice L.

H.2.2 The GGH Signature Scheme

The GGH scheme [118] works with a lattice L in $\mathbb{Z}^n$. The secret key is a non-singular matrix $R \in \mathcal{M}_n(\mathbb{Z})$, with very short row vectors (their entries are polynomial in n). In the GGH challenges [117], R was chosen as a perturbation of a multiple of the identity matrix, so that its vectors were almost orthogonal : more precisely, $R = kI_n + E$ where $k = 4\lceil\sqrt{n}+1\rfloor+1$ and each entry of the $n \times n$ matrix E is chosen uniformly at random in $\{-4, \ldots, +3\}$. Micciancio [239] noticed that this distribution has the weakness that it discloses the rough directions of the secret vectors. The lattice L is the lattice in $\mathbb{Z}^n$ spanned by the rows of R : the knowledge of R enables the signer to approximate CVP rather well in L. The basis R is then transformed to a non-reduced basis B, which will be public. In the original scheme [118], B is the multiplication of R by sufficiently many small unimodular matrices. Micciancio [239] suggested to use the Hermite normal form (HNF) of L instead. As shown in [239], the HNF gives an attacker the least advantage (in a certain precise sense) and it is therefore a good choice for the public basis. The messages are hashed onto a "large enough" subset of $\mathbb{Z}^n$, for instance a large hypercube. Let $\mathbf{m} \in \mathbb{Z}^n$ be the hash of the message to be signed. The signer applies Babai's round-off CVP approximation algorithm [17] to get a lattice vector close to $\mathbf{m}$:

$$\mathbf{s} = \lfloor \mathbf{m}R^{-1} \rceil R,$$

so that $\mathbf{s} - \mathbf{m} \in \mathcal{P}_{1/2}(R) = \{\mathbf{x}R : \mathbf{x} \in [-1/2, 1/2]^n\}$. Of course, any other CVP approximation algorithm could alternatively be applied, for instance Babai's nearest plane algorithm [17]. To verify the signature $\mathbf{s}$ of $\mathbf{m}$, one would first check that $\mathbf{s} \in L$ using the public basis B, and compute the distance $\|\mathbf{s} - \mathbf{m}\|$ to check that it is sufficiently small.

H.2.3 NTRUSign

NTRUSign [146] is a special instantiation of GGH with the compact lattices from the NTRU encryption scheme [149], which we briefly recall : we refer to [146, 63] for more details. In the NTRU standards [63] being considered by IEEE P1363.1 [164], one selects $N = 251$, $q = 128$. Let $\mathcal{R}$ be the ring $\mathbb{Z}[X]/(X^N - 1)$ whose multiplication is denoted by $*$. Using resultants, one computes a

quadruplet $(f, g, F, G) \in \mathcal{R}^4$ such that $f * G - g * F = q$ in $\mathcal{R}$ and f is invertible mod q, where f and g have 0–1 coefficients (with a prescribed number of 1), while F and G have slightly larger coefficients, yet much smaller than q. This quadruplet is the NTRU secret key. Then the secret basis is the following $(2N) \times (2N)$ matrix :

$$R = \begin{bmatrix} f_0 & f_1 & \cdots & f_{N-1} & g_0 & g_1 & \cdots & g_{N-1} \\ f_{N-1} & f_0 & \cdots & f_{N-2} & g_{N-1} & g_0 & \cdots & g_{N-2} \\ \vdots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots \\ f_1 & \cdots & f_{N-1} & f_0 & g_1 & \cdots & g_{N-1} & g_0 \\ F_0 & F_1 & \cdots & F_{N-1} & G_0 & G_1 & \cdots & G_{N-1} \\ F_{N-1} & F_0 & \cdots & F_{N-2} & G_{N-1} & G_0 & \cdots & G_{N-2} \\ \vdots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots \\ F_1 & \cdots & F_{N-1} & F_0 & G_1 & \cdots & G_{N-1} & G_0 \end{bmatrix},$$

where f_i denotes the coefficient of X^i of the polynomial f. Thus, the lattice dimension is $n = 2N$. Due to the special structure of R, it turns out that a single row of R is sufficient to recover the whole secret key. Because f is chosen invertible mod q, the polynomial $h = g/f \pmod q$ is well-defined in $\mathcal{R}$: this is the NTRU public key. Its fundamental property is that $f * h \equiv g \pmod q$ in $\mathcal{R}$. The polynomial h defines a natural public basis of L, which we omit (see [148]).

The messages are assumed to be hashed in $\{0, \ldots, q-1\}^{2N}$. Let $\mathbf{m} \in \{0, \ldots, q-1\}^{2N}$ be such a hash. We write $\mathbf{m} = (\mathbf{m}_1, \mathbf{m}_2)$ with $\mathbf{m}_i \in \{0, \ldots, q-1\}^N$. It is shown in [148] that the vector $(\mathbf{s}, \mathbf{t}) \in \mathbb{Z}^{2N}$ which we would obtain by applying Babai's round-off CVP approximation algorithm to $\mathbf{m}$ using the secret basis R can be alternatively computed using convolution products involving $\mathbf{m}_1$, $\mathbf{m}_2$ and the NTRU secret key (f, g, F, G). In practice, the signature is simply $\mathbf{s}$ and not $(\mathbf{s}, \mathbf{t})$, as $\mathbf{t}$ can be recovered from $\mathbf{s}$ thanks to $\mathbf{h}$. Besides, $\mathbf{s}$ might be further reduced mod q, but its initial value can still be recovered because it is such that $\mathbf{s} - \mathbf{m}_1$ ranges over a small interval (this is the same trick used in NTRU decryption). This gives rise for standard parameter choices to a signature length of $251 \times 7 = 1757$ bits. While this signature length is much smaller than other lattice-based signature schemes such as GGH, it is still significantly larger than more traditional signature schemes such as DSA.

This is the basic NTRUSign scheme [146]. In order to strengthen the security of NTRUSign, perturbation techniques have been proposed in [148, 63, 147]. Roughly speaking, such techniques perturb the hashed message $\mathbf{m}$ before signing with the NTRU secret basis. However, it is worth noting that there is no perturbation in half of the parameter choices recommended in NTRU standards [63] under consideration by IEEE P1363.1. Namely, this is the case for the parameter choices `ees251sp2`, `ees251sp3`, `ees251sp4` and `ees251sp5` in [63]. For the other half, only a single perturbation is recommended. But NTRU has stated that the parameter sets presented in [147] are intended to supersede these parameter sets.

H.3 The Hidden Parallelepiped Problem

Consider the signature generation in the GGH scheme described in Section H.2. Let $R \in \mathcal{M}_n(\mathbb{Z})$ be the secret basis used to approximate CVP in the lattice L. Let $\mathbf{m} \in \mathbb{Z}^n$ be the message digest. Babai's round-off CVP approximation algorithm [17] computes the signature $\mathbf{s} = \lfloor \mathbf{m} R^{-1} \rceil R$, so that $\mathbf{s} - \mathbf{m}$ belongs to the parallelepiped $\mathcal{P}_{1/2}(R) = \{\mathbf{x}R : \mathbf{x} \in [-1/2, 1/2]^n\}$, which is a fundamental domain of L. In other words, the signature generation is simply a reduction of the message $\mathbf{m}$ modulo the parallelepiped spanned by the secret basis R. If we were using Babai's nearest plane CVP approximation algorithm [17], we would have another fundamental parallelepiped (spanned by the Gram-Schmidt vectors of the secret basis) instead : we will not further discuss this case in this paper, since it does not create any significant difference and since this is not the procedure chosen in NTRUSign.

GGH [118] suggested to hash messages into a set much bigger than the fundamental domain of L. This is for instance the case in NTRUSign where the cardinality of $\{0, \ldots, q-1\}^{2N}$ is much bigger than the lattice volume q^N. Whatever the distribution of the message digest $\mathbf{m}$ might be, it would be reasonable to assume that the distribution $\mathbf{s} - \mathbf{m}$ is uniform (or very close to uniform) in the secret parallelepiped $\mathcal{P}_{1/2}(R)$. More precisely, it seems reasonable to make the following assumption :

Assumption H.1 (The Uniformity Assumption) *Let R be the secret basis of the lattice $L \subseteq \mathbb{Z}^n$. When the GGH scheme signs polynomially many "randomly chosen" message digests $\mathbf{m}_1, \ldots, \mathbf{m}_k \in \mathbb{Z}^n$ using Babai's round-off algorithm, the signatures $\mathbf{s}_1, \ldots, \mathbf{s}_k$ are such that the vectors $\mathbf{s}_i - \mathbf{m}_i$ are independent and uniformly distributed over $\mathcal{P}_{1/2}(R) = \{\mathbf{x}R : \mathbf{x} \in [-1/2, 1/2]^n\}$.*

Note that this is only an idealized assumption : in practice, the signatures and the message digests are integer vectors, so the distribution of $\mathbf{s}_i - \mathbf{m}_i$ is discrete rather than continuous, but this should not be a problem if the lattice volume is sufficiently large, as is the case in NTRUSign. Similar assumptions have been used in previous attacks [112, 338] on lattice-based signature schemes. We emphasize that all our experiments on NTRUSign do not use this assumption and work with real-life signatures.

We thus arrive at the following geometric learning problem (see Fig. H.1) :

Problem H.1 (The Hidden Parallelepiped Problem or HPP) *Let $V = [\mathbf{v}_1, \ldots, \mathbf{v}_n] \in GL_n(\mathbb{R})$ and let $\mathcal{P}(V) = \{\sum_{i=1}^n x_i \mathbf{v}_i : x_i \in [-1, 1]\}$ be the parallelepiped spanned by V. The input to the HPP is a sequence of* poly(n) *independent samples from $U(\mathcal{P}(V))$, the uniform distribution over $\mathcal{P}(V)$. The goal is to find a good approximation of the rows of $\pm V$.*

In the definition of the HPP, we chose $[-1, 1]$ rather than $[-1/2, 1/2]$ like in Assumption H.1 to simplify subsequent calculations.

Clearly, if one could solve the HPP, then one would be able to approximate the secret basis in GGH by collecting random pairs $(message, signature)$. To complete the attack, we need to show how to obtain the actual secret basis given a good enough approximation of it. One simple way to achieve this is by rounding the coordinates of the approximation to the closest integer. This approach should work well in cases where the entries in the secret basis are small in absolute value. Alternatively, one can use approximate-CVP algorithms to try to recover the secret basis, since one knows a lattice basis from the GGH public key. The success of this approach depends on whether the approximation is sufficiently good for existing lattice reduction algorithms. Previous experiments of [250] on the GGH-challenges [117] seem to suggest that in practice, even a moderately good approximation of a lattice vector is sufficient to recover the closest lattice vector, even in high dimension.

The Special Case of NTRUSign.

Following the announcement of our result in [259], Whyte observed [355] that symmetries in the NTRU lattices might lead to attacks that require far less signatures. Namely, Whyte noticed that in the particular case of NTRUSign, the hidden parallelepiped $\mathcal{P}(R)$ has the following property : for each $\mathbf{x} \in \mathcal{P}(R)$ the block-rotation $\sigma(\mathbf{x})$ also belongs to $\mathcal{P}(R)$, where σ is the function that maps any $(x_1, \ldots, x_N, y_1, \ldots, y_N) \in \mathbb{R}^{2N}$ to $(x_N, x_1, \ldots, x_{N-1}, y_N, y_1, \ldots, y_{N-1})$. This is because σ is a linear operation that permutes the rows of R and hence leaves $\mathcal{P}(R)$ invariant. As a result, by using the N possible rotations, each signature actually gives rise to N samples in the parallelepiped $\mathcal{P}(R)$ (as opposed to just one in the general case of GGH). For instance, 400 NTRUSign-251 signatures give rise to 100,400 samples in the NTRU parallelepiped. Notice that these samples are no longer independent and hence Assumption H.1 does not hold. Nevertheless, as we will describe later, this technique leads in practice to attacks using a significantly smaller number of signatures.

H.4 Learning a Parallelepiped

In this section, we describe our solution to the Hidden Parallelepiped Problem (HPP), based on the following steps. First, we approximate the covariance matrix of the given distribution. This covariance matrix is essentially V^tV (where V defines the given parallelepiped). We then exploit this approximation in order to transform our hidden parallelepiped $\mathcal{P}(V)$ into a unit hypercube : in other words, we reduce the HPP to the case where the hidden parallelepiped is a hypercube. Finally, we show how hypercubic instances of the HPP are related to a multivariate optimization problem based on the fourth moment, which we solve by a gradient descent. The algorithm is summarized in Algorithms 16 and 17, and in described in more detail in the following.

Algorithm 16 Solving the Hidden Parallelepiped Problem

Input: A polynomial number of samples uniformly distributed over a parallelepiped $\mathcal{P}(V)$.
Output: Approximations of rows of $\pm V$.

1: Compute an approximation G of the Gram matrix V^tV of V^t (see Section H.4.1).
2: Compute the Cholesky factor L of G^{-1}, so that $G^{-1} = LL^t$.
3: Multiply the samples of $\mathcal{P}(V)$ by L to the right to obtain samples of $\mathcal{P}(C)$ where $C = VL$.
4: Compute approximations of rows of $\pm C$ by Algorithm 17 from Section H.4.3.
5: Multiply each approximation by L^{-1} to the right to derive an approximation of a row of $\pm V$.

H.4.1 The Covariance Matrix Leakage

The first step in our algorithm is based on the idea of approximating the covariance matrix, which was already present in the work of Gentry and Szydlo [112, 338] (after this basic step, our strategy differs completely from theirs). Namely, Gentry and Szydlo observed that from GGH signatures one can easily obtain an approximation of V^tV, the Gram matrix of the transpose of the secret basis. Here, we simply translate this observation to the HPP setting.

Lemma H.1 (Covariance Matrix Leakage) *Let $V \in GL_n(\mathbb{R})$. Let $\mathbf{v}$ be chosen from the uniform distribution over the parallelepiped $\mathcal{P}(V)$. Then*

$$\mathrm{Exp}[\mathbf{v}^t\mathbf{v}] = V^tV/3.$$

In other words, the covariance matrix of the distribution $U(\mathcal{P}(V))$ is $V^tV/3$.

Proof. We can write $\mathbf{v} = \mathbf{x}V$ where $\mathbf{x}$ has uniform distribution over $[-1, 1]^n$. Hence,

$$\mathbf{v}^t\mathbf{v} = V^t\mathbf{x}^t\mathbf{x}V.$$

An elementary computation shows that $\mathrm{Exp}[\mathbf{x}^t\mathbf{x}] = I_n/3$ where I_n is the $n \times n$ identity matrix, and the lemma follows. □

Hence, by taking the average of $\mathbf{v}^t\mathbf{v}$ over all our samples $\mathbf{v}$ from $U(\mathcal{P}(V))$, and multiplying the result by 3, we can obtain an approximation of V^tV.

H.4.2 Morphing a Parallelepiped into a Hypercube

The second stage is explained by the following lemma.

Lemma H.2 (Hypercube Transformation) *Let $V \in GL_n(\mathbb{R})$. Denote by $G \in GL_n(\mathbb{R})$ the symmetric positive definite matrix V^tV. Denote by $L \in GL_n(\mathbb{R})$ the Cholesky factor[10] of G^{-1}, that is, L is the unique lower-triangular matrix such that $G^{-1} = LL^t$. Then the matrix $C = VL \in GL_n(\mathbb{R})$ satisfies the following :*

[10] Instead of the Cholesky factor, one can take any matrix L such that $G^{-1} = LL^t$. We work with Cholesky factorization as this turns out to be more convenient in our experiments.

1. *The rows of C are pairwise orthogonal unit vectors. In other words, C is an orthogonal matrix in $O_n(\mathbb{R})$ and $\mathcal{P}(C)$ is a unit hypercube.*
2. *If $\mathbf{v}$ is uniformly distributed over the parallelepiped $\mathcal{P}(V)$, then $\mathbf{c} = \mathbf{v}L$ is uniformly distributed over the hypercube $\mathcal{P}(C)$.*

Proof. The Gram matrix $G = V^tV$ is clearly symmetric positive definite. Hence $G^{-1} = V^{-1}V^{-t}$ is also symmetric positive definite, and has a Cholesky factorization $G^{-1} = LL^t$ where L is lower-triangular matrix. Therefore, $V^{-1}V^{-t} = LL^t$. Let $C = VL \in GL_n(\mathbb{R})$. Then

$$CC^t = VLL^tV^t = VV^{-1}V^{-t}V^t = I.$$

For the second claim, let $\mathbf{v}$ be uniformly distributed over $\mathcal{P}(V)$. Then we can write $\mathbf{v} = \mathbf{x}V$ where $\mathbf{x}$ is uniformly distributed over $[-1, 1]^n$. It follows that $\mathbf{v}L = \mathbf{x}VL = \mathbf{x}C$ has the uniform distribution over $\mathcal{P}(C)$. □

Lemma H.2 says that by applying the transformation L, we can map our samples from the parallelepiped $\mathcal{P}(V)$ into samples from the hypercube $\mathcal{P}(C)$. Then, if we could approximate the rows of $\pm C$, we would also obtain an approximation of the rows of $\pm V$ by applying L^{-1}. In other words, we have reduced the Hidden Parallelepiped Problem into what one might call the Hidden Hypercube Problem (see Fig. H.2). From an implementation point of view, we note that the Cholesky

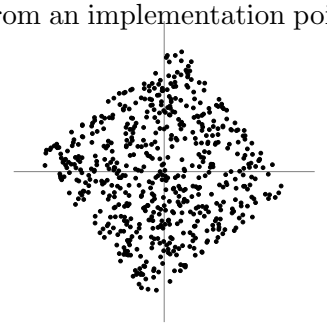


FIG. H.2 – The Hidden Hypercube Problem in dimension two.

factorization (required for obtaining L) can be easily computed by a process close to the Gram-Schmidt orthogonalization process (see [122]). Lemma H.2 assumes that we know $G = V^tV$ exactly. If we only have an approximation of G, then C will only be close to some orthogonal matrix in $O_n(\mathbb{R})$: the Gram matrix CC^t of C will be close to the identity matrix, and the image under L of our parallelepiped samples will have a distribution close to the uniform distribution of some unit hypercube.

H.4.3 Learning a Hypercube

For any $V = [\mathbf{v}_1, \ldots, \mathbf{v}_n] \in GL_n(\mathbb{R})$ and any integer $k \geq 1$, we define the k-th moment of $\mathcal{P}(V)$ over a vector $\mathbf{w} \in \mathbb{R}^n$ as

$$\mathrm{mom}_{V,k}(\mathbf{w}) = \mathrm{Exp}[\langle \mathbf{u}, \mathbf{w} \rangle^k],$$

where $\mathbf{u}$ is uniformly distributed over the parallelepiped $\mathcal{P}(V)$.[11] Clearly, $\mathrm{mom}_{V,k}(\mathbf{w})$ can be approximated by using the given sample from $U(\mathcal{P}(V))$. We are interested in the second and fourth moments. A straightforward calculation shows that for any $\mathbf{w} \in \mathbb{R}^n$, they are given by

$$\begin{aligned} \mathrm{mom}_{V,2}(\mathbf{w}) &= \frac{1}{3}\sum_{i=1}^{n}\langle \mathbf{v}_i, \mathbf{w} \rangle^2 = \frac{1}{3}\mathbf{w}V^tV\mathbf{w}^t, \\ \mathrm{mom}_{V,4}(\mathbf{w}) &= \frac{1}{5}\sum_{i=1}^{n}\langle \mathbf{v}_i, \mathbf{w} \rangle^4 + \frac{1}{3}\sum_{i \neq j}\langle \mathbf{v}_i, \mathbf{w} \rangle^2 \langle \mathbf{v}_j, \mathbf{w} \rangle^2. \end{aligned}$$

[11] This should not be confused with an unrelated notion of moment considered in [146, 147].

Note that the second moment is given by the covariance matrix mentioned in Section H.4.1. When $V \in O_n(\mathbb{R})$, the second moment becomes $\|\mathbf{w}\|^2/3$ while the fourth moment becomes

$$\mathrm{mom}_{V,4}(\mathbf{w}) = \frac{1}{3}\|\mathbf{w}\|^4 - \frac{2}{15}\sum_{i=1}^{n}\langle \mathbf{v}_i, \mathbf{w}\rangle^4.$$

The gradient of the latter is therefore

$$\nabla \mathrm{mom}_{V,4}(\mathbf{w}) \quad = \quad \sum_{i=1}^{n}\left(\frac{4}{3}\left(\sum_{j=1}^{n}\langle \mathbf{v}_j, \mathbf{w}\rangle^2\right)\langle \mathbf{v}_i, \mathbf{w}\rangle - \frac{8}{15}\langle \mathbf{v}_i, \mathbf{w}\rangle^3\right)\mathbf{v}_i.$$

For $\mathbf{w}$ on the unit sphere the second moment is constantly $1/3$, and

$$\begin{aligned} \mathrm{mom}_{V,4}(\mathbf{w}) &= \frac{1}{3} - \frac{2}{15}\sum_{i=1}^{n}\langle \mathbf{v}_i, \mathbf{w}\rangle^4, \\ \nabla \mathrm{mom}_{V,4}(\mathbf{w}) &= \frac{4}{3}\mathbf{w} - \frac{8}{15}\sum_{i=1}^{n}\langle \mathbf{v}_i, \mathbf{w}\rangle^3 \mathbf{v}_i. \end{aligned} \tag{H.1}$$

Lemma H.3 *Let $V = [\mathbf{v}_1, \ldots, \mathbf{v}_n] \in O_n(\mathbb{R})$. Then the global minimum of $\mathrm{mom}_{V,4}(\mathbf{w})$ over the unit sphere of $\mathbb{R}^n$ is $1/5$ and this minimum is obtained at $\pm\mathbf{v}_1, \ldots, \pm\mathbf{v}_n$. There are no other local minima.*

Proof. The method of Lagrange multipliers shows that for $\mathbf{w}$ to be an extremum point of $\mathrm{mom}_{V,4}$ on the unit sphere, it must be proportional to $\nabla \mathrm{mom}_{V,4}(\mathbf{w})$. By writing $\mathbf{w} = \sum_{i=1}^{n}\langle \mathbf{v}_i, \mathbf{w}\rangle \mathbf{v}_i$ and using Eq. (H.1), we see that there must exist some α such that $\langle \mathbf{v}_i, \mathbf{w}\rangle^3 = \alpha\langle \mathbf{v}_i, \mathbf{w}\rangle$ for $i = 1, \ldots, n$. In other words, each $\langle \mathbf{v}_i, \mathbf{w}\rangle$ is either zero or $\pm\sqrt{\alpha}$. It is easy to check that among all such points, only $\pm\mathbf{v}_1, \ldots, \pm\mathbf{v}_n$ form local minima. □

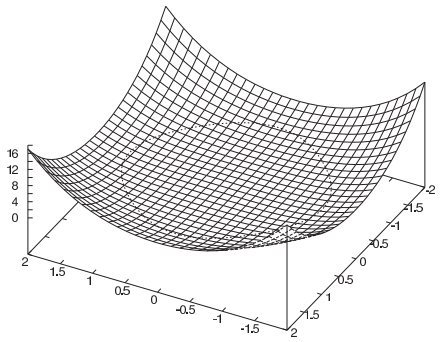


FIG. H.3 – The fourth moment for $n = 2$. The dotted line shows the restriction to the unit circle.

In other words, the hidden hypercube problem can be reduced to a minimization problem of the fourth moment over the unit sphere. A classical technique to solve such minimization problems is the gradient descent described in Algorithm 17.

Algorithm 17 Solving the Hidden Hypercube Problem by Gradient Descent

Parameters: A descent parameter δ.
Input: A polynomial number of samples uniformly distributed over a unit hypercube $\mathcal{P}(V)$.
Output: An approximation of some row of $\pm V$.

1: Let $\mathbf{w}$ be chosen uniformly at random from the unit sphere of $\mathbb{R}^n$.
2: Compute an approximation $\mathbf{g}$ of the gradient $\nabla \text{mom}_4(\mathbf{w})$ (see Section H.4.3).
3: Let $\mathbf{w}_{new} = \mathbf{w} - \delta \mathbf{g}$.
4: Divide $\mathbf{w}_{new}$ by its Euclidean norm $\|\mathbf{w}_{new}\|$.
5: **if** $\text{mom}_{V,4}(\mathbf{w}_{new}) \geq \text{mom}_{V,4}(\mathbf{w})$ where the moments are approximated by sampling **then**
6: **return** the vector $\mathbf{w}$.
7: **else**
8: Replace $\mathbf{w}$ by $\mathbf{w}_{new}$ and go back to Step 2.
9: **end if**

The gradient descent typically depends on a parameter δ, which has to be carefully chosen. Since we want to minimize the function here, we go in the opposite direction of the gradient. To approximate the gradient in Step 2 of Algorithm 17, we notice that

$$\nabla \text{mom}_{V,4}(\mathbf{w}) = \text{Exp}[\nabla(\langle \mathbf{u}, \mathbf{w} \rangle^4)] = 4\text{Exp}[\langle \mathbf{u}, \mathbf{w} \rangle^3 \mathbf{u}].$$

This allows to approximate the gradient $\nabla \text{mom}_{V,4}(\mathbf{w})$ using averages over samples, like for the fourth moment itself.

H.5 Experimental Results

As usual in cryptanalysis, perhaps the most important question is whether or not the attack works in practice. We therefore implemented the attack in C++ and ran it on a 2GHz PC/Opteron. The critical parts of the code were written in plain C++ using double arithmetic, while the rest used Shoup's NTL library version 5.4 [319]. Based on trial-and-error, we chose $\delta = 0.7$ in the gradient descent (Algorithm 17), for all the experiments mentioned here. The choice of δ has a big impact on the behavior of the gradient descent : the choice $\delta = 0.7$ works well, but we do not claim that it is optimal. When doing several descents in a row, it is useful to relax the halting condition 5 in Algorithm 17 to abort descents which seem to make very little progress.

H.5.1 NTRUSign

We performed two kinds of experiments against NTRUSign, depending on whether the symmetries of NTRU lattices explained in Section H.3.0.0 were used or not. All the experiments make it clear that perturbation techniques are really mandatory for the security of NTRUSign, though it is currently unknown if such techniques are sufficient to prevent this kind of attacks.

Without exploiting the symmetries of NTRU lattices.

We applied Algorithm 16 to real-life parameters of NTRUSign. More precisely, we ran the attack on NTRUSign-251 without perturbation, corresponding to the parameter choices `ees251sp2`, `ees251sp3`, `ees251sp4` and `ees251sp5` in the NTRU standards [63] under consideration by IEEE P1363.1 [164]. This corresponds to a lattice dimension of 502. We did not rely on the uniformity assumption : we generated genuine NTRUSign signatures of messages generated uniformly at random over $\{0, \ldots, q-1\}^n$.

The results of the experiments are summarized in Figure H.4. For each given number of signatures

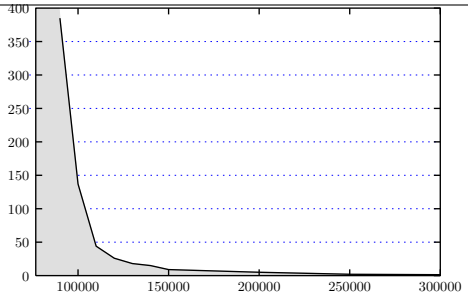


FIG. H.4 – Experiments on NTRUSIGN-251 without perturbation, and without using NTRU symmetries. The curve shows the average number of random descents required to recover the secret key, depending on the number of signatures, which is in the range 80,000–300,000.

TAB. H.1 – Experiments on NTRUSIGN-251 without perturbation, using NTRU symmetries.

Number of signatures	Expected number of descents to recover the secret key
1,000	2
500	40
400	100

in the range 80,000–300,000, we generated a set of signatures, and applied Algorithm 16 to it : from the set of samples we derived an approximation of the Gram matrix, used it to transform the parallelepiped into a hypercube, and finally, we ran a series of about a thousand descents, starting with random points. We regard a descent as successful if, when rounded to the nearest integer vector, the output of the descent gives *exactly* one of the vectors of the secret basis (which is sufficient to recover the whole secret basis in the case of NTRUSIGN). We did not notice any improvement with Babai's nearest plane algorithm [17] (with a BKZ-20 reduced basis [307]) as a CVP approximation. The curve shows the average number of random descents needed for a successful descent as a function of the number of signatures.

Typically, a single random descent does not take much time : for instance, a usual descent for 150,000 signatures takes roughly ten minutes. When successful, a descent may take as little as a few seconds. The minimal number of signatures to make the attack successful in our experiments was 90,000, in which case the required number of random descents was about 400. With 80,000 signatures, we tried 5,000 descents without any success. The curve given in Fig. H.4 may vary a little bit, depending on the secret basis : for instance, for the basis used in the experiments of Fig. H.4, the average number of random descents was 15 with 140,000 signatures, but it was 23 for another basis generated with the same NTRU parameters. It seems that the exact geometry of the secret basis has an influence, as will be seen in the analysis of Section H.6.

Exploiting the symmetries of NTRU lattices.

Based on Whyte's observation described in Section H.3.0.0, one might hope that the number of signatures required by the attack can be shrunk by a factor of roughly N. Surprisingly, this is indeed the case in practice (see Table H.1) : as few as 400 signatures are enough in practice to recover the secret key, though the corresponding 100,400 parallelepiped samples are not independent. This means that the previous number of 90,000 signatures required by the attack can be roughly divided by $N = 251$. Hence, NTRUSIGN without perturbation should be considered totally insecure.

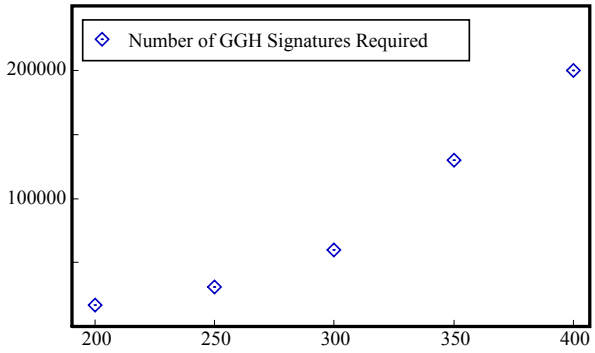


FIG. H.5 – Average number of GGH signatures required so that ten random descents coupled with Babai's nearest plane algorithm disclose with high probability a secret vector, depending on the dimension of the GGH challenge.

H.5.2 The GGH Challenges

We also did experiments on the GGH-challenges [117], which range from dimension 200 to 400. Because there is actually no GGH signature challenge, we simply generated secret bases like in the GGH encryption challenges. To decrease the cost of sample generation, and because there was no concrete proposal of a GGH signature scheme, we relied on the uniformity assumption : we created samples uniformly distributed over the secret parallelepiped, and tried to recover the secret basis.

When the number of samples becomes sufficiently high, the approximation obtained by a random descent is sufficiently good to disclose one of the vectors of the secret basis by simple rounding, just as in the NTRUSIGN case : however, the number of required samples is significantly higher than for NTRUSIGN ; for instance, with 200,000 samples in dimension 200, three descents are enough to disclose a secret vector by rounding ; whereas three descents are also enough with 250,000 samples for NTRUSIGN, but that corresponds to a dimension of 502 which is much bigger than 200. This is perhaps because the secret vectors of the GGH challenges are significantly longer than those of NTRUSIGN.

However, one can significantly improve the result by using a different rounding procedure. Namely, instead of rounding the approximation to an integer vector, one can apply a CVP approximation algorithm such as Babai's nearest plane algorithm [17] : such algorithms will succeed if the approximation is sufficiently close to the lattice ; and one can improve the chances of the algorithm by computing a lattice basis as reduced as possible. For instance, with only 20,000 samples in dimension 200, it was impossible to recover a secret vector by rounding, but it became easy with Babai's nearest plane algorithm on a BKZ-20 reduced basis [307] (obtained by BKZ reduction of the public HNF basis) : more precisely, three random descents sufficed on the average. More generally, Figure H.5 shows the average number of samples required so that ten random descents disclose a secret vector with high probability, depending on the dimension of the GGH challenge, using Babai's nearest plane algorithm on a BKZ-20 reduced basis. Figure H.5 should not be interpreted as the minimal number of signatures required for the success of the attack : it only gives an upper bound for that number. Indeed, there are several ways to decrease the number of signatures :

- One can run much more than ten random descents.
- One can take advantage of the structure of the GGH challenges. When starting a descent, rather than starting with a random point on the unit sphere, we may exploit the fact that we know the rough directions of the secret vectors.
- One can use better CVP approximation algorithms.

H.6 Theoretical Analysis

Our goal in this section is to give a rigorous theoretical justification to the success of the attack. Namely, we will show that given a large enough polynomial number of samples, Algorithm 16 succeeds in finding a good approximation to a row of V with some constant probability. For sake of clarity and simplicity, we will not make any attempt to optimize this polynomial bound on the number of samples. Let us remark that it is possible that a rigorous analysis already exists in the ICA literature, although we were unable to find any (an analysis under some simplifying assumptions can be found in [161]). Also, Frieze et al. [100] sketch a rigorous analysis of a similar algorithm.

In order to approximate the covariance matrix, the fourth moment, and its gradient, our attack computes averages over samples. Because the samples are independent and identically distributed, we can use known bounds on large deviations such as the Chernoff bound (see, e.g., [12]) to obtain that with extremely high probability the approximations are very close to the true values. In our analysis below we omit the explicit calculations, as these are relatively standard.

H.6.1 Analysis of Algorithm 17

We start by analyzing Algorithm 17. For simplicity, we consider only the case in which the descent parameter δ equals $3/4$. A similar analysis holds for $0 < \delta < 3/4$. Another simplifying assumption we make is that instead of the stopping rule in Step 5 we simply repeat the descent step some small number r of times (which will be specified later).

For now, let us assume that the matrix V is an orthogonal matrix, so our samples are drawn from a unit hypercube $\mathcal{P}(V)$. We will later show that the actual matrix V, as obtained from Algorithm 16, is very close to orthogonal, and that this approximation does not affect the success of Algorithm 17.

Theorem H.1 *For any $c_0 > 0$ there exists a $c_1 > 0$ such that given n^{c_1} samples uniformly distributed over some unit hypercube $\mathcal{P}(V)$, $V = [\mathbf{v}_1, \ldots, \mathbf{v}_n] \in O_n(\mathbb{R})$, Algorithm 17 with $\delta = 3/4$ and $r = O(\log \log n)$ descent steps outputs with constant probability a vector that is within ℓ_2 distance n^{-c_0} of $\pm\mathbf{v}_i$ for some i.*

Proof. We first analyze the behavior of Algorithm 17 under the assumption that all gradients are computed exactly without any error. We write any vector $\mathbf{w} \in \mathbb{R}^n$ as $\mathbf{w} = \sum_{i=1}^n w_i \mathbf{v_i}$. Then, using Eq. (H.1), we see that for $\mathbf{w}$ on the unit sphere,

$$\nabla \mathrm{mom}_{V,4}(\mathbf{w}) = \frac{4}{3}\mathbf{w} - \frac{8}{15}\sum_{i=1}^{n} w_i^3 \mathbf{v}_i.$$

Since we took $\delta = 3/4$, Step 3 in Algorithm 17 performs

$$\mathbf{w}_{new} = \frac{2}{5}\sum_{i=1}^{n} w_i^3 \mathbf{v}_i.$$

The vector is then normalized in Step 4. So we see that each step in the gradient descent takes a vector $(w_1, \ldots, w_n)$ to the vector $\alpha \cdot (w_1^3, \ldots, w_n^3)$ for some normalization factor α (where both vectors are written in the $\mathbf{v}_i$ basis). Hence, after r iterations, a vector $(w_1, \ldots, w_n)$ is transformed to the vector

$$\alpha \cdot (w_1^{3^r}, \ldots, w_n^{3^r})$$

for some normalization factor α.

Recall now that the original vector $(w_1, \ldots, w_n)$ is chosen uniformly from the unit sphere. It can be shown that with some constant probability, one of its coordinates is greater in absolute value than

all other coordinates by a factor of at least $1 + \Omega(1/\log n)$ (first prove this for a vector distributed according to the standard multivariate Gaussian distribution, and then note that by normalizing we obtain a uniform vector from the unit sphere). For such a vector, after only $r = O(\log \log n)$ iterations, this gap is amplified to more than, say, $n^{\log n}$, which means that we have one coordinate very close to ± 1 and all others are at most $n^{-\log n}$ in absolute value. This establishes that if all gradients are known precisely, Algorithm 17 succeeds with some constant probability.

To complete the analysis of Algorithm 17, we now argue that it succeeds with good probability even in the presence of noise in the approximation of the gradients. First, it can be shown that for any $c > 0$, given a large enough polynomial number of samples, with very high probability all our gradient approximations are accurate to within an additive error of n^{-c} in the ℓ_2 norm (we have r such approximations during the course of the algorithm). This follows by a standard application of the Chernoff bound followed by a union bound. Now let $\mathbf{w} = (w_1, \ldots, w_n)$ be a unit vector in which one coordinate, say the jth, is greater in absolute value than all other coordinates by at least a factor of $1 + \Omega(1/\log n)$. Since $\mathbf{w}$ is a unit vector, this in particular means that $w_j > 1/\sqrt{n}$. Let $\tilde{\mathbf{w}}_{new} = \mathbf{w} - \delta \nabla \mathrm{mom}_4(\mathbf{w})$. Recall that for each i, $\tilde{\mathbf{w}}_{new,i} = \frac{2}{5} w_i^3$ which in particular implies that $\tilde{\mathbf{w}}_{new,j} > \frac{2}{5} n^{-1.5} > n^{-2}$. By our assumption on the approximation $\mathbf{g}$, we have that for each i, $|\tilde{\mathbf{w}}_{new,i} - \mathbf{w}_{new,i}| \leq n^{-c}$. So for any $k \neq j$,

$$\frac{|\mathbf{w}_{new,j}|}{|\mathbf{w}_{new,k}|} \geq \frac{|\tilde{\mathbf{w}}_{new,j}| - n^{-c}}{|\tilde{\mathbf{w}}_{new,k}| + n^{-c}} \geq \frac{|\tilde{\mathbf{w}}_{new,j}|(1 - n^{-(c-2)})}{|\tilde{\mathbf{w}}_{new,k}| + n^{-c}}.$$

If $|\tilde{\mathbf{w}}_{new,k}| > n^{-(c-1)}$, then the above is at least $(1 - O(1/n))(w_j/w_k)^3$. Otherwise, the above is at least $\Omega(n^{c-3})$. Hence, after $O(\log \log n)$ steps, the gap w_j/w_k becomes $\Omega(n^{c-3})$. Therefore, for any $c_0 > 0$ we can make the distance between the output vector and one of the $\pm \mathbf{v}_i$s less than n^{-c_0} by choosing a large enough c. □

H.6.2 Analysis of Algorithm 16

The following theorem completes the analysis of the attack. In particular, it implies that if V is an integer matrix all of whose entries are bounded in absolute value by some polynomial, then running Algorithm 16 with a large enough polynomial number of samples from the uniform distribution on $\mathcal{P}(V)$ gives (with constant probability) an approximation to a row of $\pm V$ whose error is less than $1/2$ in each coordinate, and therefore leads to an *exact* row of $\pm V$ simply by rounding each coordinate to the nearest integer. Hence we have a rigorous proof that our attack can efficiently recover the secret key in both NTRUSign and the GGH challenges.

Theorem H.2 *For any $c_0 > 0$ there exists a $c_1 > 0$ such that given n^{c_1} samples uniformly distributed over some parallelepiped $\mathcal{P}(V)$, $V = [\mathbf{v}_1, \ldots, \mathbf{v}_n] \in GL_n(\mathbb{R})$, Algorithm 16 outputs with constant probability a vector $\tilde{\mathbf{e}}V$ where $\tilde{\mathbf{e}}$ is within ℓ_2 distance n^{-c_0} of some standard basis vector $\mathbf{e}_i$.*

Proof. Recall that a sample $\mathbf{v}$ from $U(\mathcal{P}(V))$ can be written as $\mathbf{x}V$ where $\mathbf{x}$ is chosen uniformly from $[-1, 1]^n$. So let $\mathbf{v}_i = \mathbf{x}_i V$ for $i = 1, \ldots, N$ be the input samples. Then our approximation G to the Gram matrix $V^t V$ is given by $G = V^t \tilde{I} V$ where $\tilde{I} = \frac{3}{N} \sum \mathbf{x}_i^t \mathbf{x}_i$. We claim that with high probability, $\tilde{I}$ is very close to the identity matrix. Indeed, for $\mathbf{x}$ chosen randomly from $[-1, 1]^n$, each diagonal entry of $\mathbf{x}^t \mathbf{x}$ has expectation $1/3$ and each off-diagonal entry has expectation 0. Moreover, these entries take values in $[-1, 1]$. By the Chernoff bound we obtain that for any approximation parameter $c > 0$, if we choose, say, $N = n^{2c+1}$ then with very high probability each entry in $\tilde{I} - I$ is at most n^{-c} in absolute value. In particular, this implies that all eigenvalues of the symmetric matrix $\tilde{I}$ are in the range $1 \pm n^{-c+1}$.

Recall that we define L to be the Cholesky factor of $G^{-1} = V^{-1} \tilde{I}^{-1} V^{-t}$ and that $C = VL$. Now $CC^t = VLL^t V^t = \tilde{I}^{-1}$, which implies that C is close to an orthogonal matrix. Let us make

this precise. Consider the singular value decomposition of C, given by $C = U_1 D U_2$ where U_1, U_2 are orthogonal matrices and D is diagonal. Then $CC^t = U_1 D^2 U_1^t$ and hence $D^2 = U_1^t \tilde{I}^{-1} U_1$. From this it follows that the diagonal of D consists of the square roots of the reciprocals of the eigenvalues of $\tilde{I}$, which in particular means that all values on the diagonal of D are also in the range $1 \pm n^{-c+1}$.

Consider the orthogonal matrix $\tilde{C} = U_1 U_2$. We claim that for large enough c, samples from $\mathcal{P}(C)$ 'look like' samples from $\mathcal{P}(\tilde{C})$. More precisely, assume that c is chosen so that the number of samples required by Algorithm 17 is less than, say, n^{c-4}. Then, it follows from Lemma H.4 below that the statistical distance between a set of n^{c-4} samples from $\mathcal{P}(C)$ and a set of n^{c-4} samples from $\mathcal{P}(\tilde{C})$ is at most $O(n^{-1})$. By Theorem H.1, we know that when given samples from $\mathcal{P}(\tilde{C})$, Algorithm 17 outputs an approximation of a row of $\pm\tilde{C}$ with some constant probability. Hence, when given samples from $\mathcal{P}(C)$, it must still output an equally good approximation of a row of $\pm\tilde{C}$ with a probability that is smaller by at most $O(n^{-1})$ and in particular, constant.

To complete the proof, let $\tilde{\mathbf{c}}$ be the vector obtained in Step 4. The output of Algorithm 16 is then

$$\tilde{\mathbf{c}} L^{-1} = \tilde{\mathbf{c}} C^{-1} V = (\tilde{\mathbf{c}} \tilde{C}^{-1})(\tilde{C} C^{-1}) V = (\tilde{\mathbf{c}} \tilde{C}^{-1})(U_1 D^{-1} U_1^t) V.$$

As we have seen before, all eigenvalues of $U_1 D^{-1} U_1^t$ are close to 1. It therefore follows that the above is a good approximation to a row of $\pm V$, and it is not hard to verify that the quality of this approximation satisfies the requirements stated in the theorem. □

Lemma H.4 *The statistical distance between the uniform distribution on $\mathcal{P}(C)$ and that on $\mathcal{P}(\tilde{C})$ is at most $O(n^{-c+3})$.*

Proof. We first show that the parallelepiped $\mathcal{P}(C)$ is almost contained and almost contains the cube $\mathcal{P}(\tilde{C})$:

$$(1 - n^{-c+2})\mathcal{P}(\tilde{C}) \subseteq \mathcal{P}(C) \subseteq (1 + n^{-c+2})\mathcal{P}(\tilde{C}).$$

To show this, take any vector $\mathbf{y} \in [-1, 1]^n$. The second containment is equivalent to showing that all the coordinates of $\mathbf{y} U_1 D U_1^t$ are at most $1 + n^{-c+2}$ in absolute value. Indeed, by the triangle inequality,

$$\begin{aligned} \|\mathbf{y} U_1 D U_1^t\|_\infty \leq \|\mathbf{y}\|_\infty + \|\mathbf{y} U_1 (D - I) U_1^t\|_\infty &\leq 1 + \|\mathbf{y} U_1 (D - I) U_1^t\|_2 \\ &\leq 1 + n^{-c+1}\sqrt{n} < 1 + n^{-c+2}. \end{aligned}$$

The first containment is proved similarly. On the other hand, the ratio of volumes between the two cubes is $((1 + n^{-c+2})/(1 - n^{-c+2}))^n = 1 + O(n^{-c+3})$. From this it follows that the statistical distance between the uniform distribution on $\mathcal{P}(C)$ and that on $\mathcal{P}(\tilde{C})$ is at most $O(n^{-c+3})$. □

Acknowledgements. We thank William Whyte for helpful discussions.

Annexe I

The Impact of Decryption Failures on the Security of NTRU Encryption

CRYPTO 2003

[158] avec Nick Howgrave-Graham (NTRU Cryptosystems), David Pointcheval (ENS), John Proos (Univ. Waterloo), Joseph Silverman, Ari Singer, et William Whyte (NTRU Cryptosystems)

Abstract: NTRUEncrypt *is unusual among public-key cryptosystems in that, with standard parameters, validly generated ciphertexts can fail to decrypt. This affects the provable security properties of a cryptosystem, as it limits the ability to build a simulator in the random oracle model without knowledge of the private key. We demonstrate attacks which use decryption failures to recover the private key. Such attacks work for all standard parameter sets, and one of them applies to any padding. The appropriate countermeasure is to change the parameter sets and possibly the decryption process so that decryption failures are vanishingly unlikely, and to adopt a padding scheme that prevents an attacker from directly controlling any part of the input to the encryption primitive. We outline one such candidate padding scheme.*

I.1 Introduction

An unusual property of the NTRU public-key cryptosystem is the presence of *decryption failures* : with standard parameters, validly generated ciphertexts may fail to decrypt. In this paper, we show the importance of decryption failures with respect to the security of the NTRU public-key cryptosystem. We believe this fact has been much overlooked in past research on NTRU.

First, we notice that decryption failures cannot be ignored, as they happen much more frequently than one would have expected. If one strictly follows the recommendations of the EESS standard [61], decryption failures happen as often as every 2^{12} messages with $N = 139$, and every 2^{25} messages with $N = 251$. It turns out that the probability is somewhat lower (around 2^{-40}) with NTRU products, as the key generation implemented in NTRU products surprisingly differs from the one recommended in [61]. In any case, decryption failures happen sufficiently often that one cannot dismiss them, even in NTRU products.

One may *a priori* think that the only drawback with decryption failures is that the receiver will not be able to decrypt. However, decryption failures have a big impact on the confidence we can have in the security level, because they limit the ability to build a simulator in the random oracle model without knowledge of the private key. This limitation is independent of the padding used. This implies that all the security proofs known (see [152, 258]) for various NTRU paddings may not be valid after all, because decryption failures have been ignored in such proofs. This also means that existing generic

padding schemes (such as REACT [276]) may not apply to NTRU as, to our knowledge, no existing padding scheme takes into account the possibility of decryption failures, perhaps because the only competitive cryptosystem that experiences decryption failures is NTRUEncrypt.

From a security point of view, the situation is even worse. It turns out that decryption failures not only influence the validity of a security proof, they leak information on the private key. We demonstrate this fact by presenting new efficient chosen-ciphertext attacks on NTRUEncrypt (with or without padding) that recover the private key. These chosen-ciphertext attacks are very different from chosen-ciphertext attacks [168, 154] formerly known against NTRU : they only use valid ciphertexts, while the attacks of [168, 154] use fake ciphertexts and can therefore be easily thwarted. Moreover, these chosen-ciphertext attacks do not use the full power of chosen-ciphertext attacks, but reaction attacks only [137] : Here, the attacker selects various messages, encrypts them, and checks whether the receiver is able to correctly decrypt the ciphertexts : eventually, the attacker has gathered sufficiently many decryption failures to be able to recover the private key. The only way to avoid such chosen-ciphertext attacks is to make sure that it is computationally infeasible to find decryption failures. This requires different parameter sets and certain implementation changes in NTRUEncrypt, which we hint in the final section of this paper.

The rest of the paper is organized as follows. In Sections I.2 and I.3, we recall NTRUEncrypt and the padding used in EESS [62]. In Section I.4, we explain decryption failures, and their impact on security proofs for NTRU paddings. In Section I.5, we present several efficient attacks based on decryption failures : some are tailored to certain paddings, while the most powerful one applies to any padding. In Appendix I.7, we give additional information on NTRUEncrypt, pointing out the difference between the specification of the EESS standard and the implementation in NTRU products. In Appendix I.8, we describe an alternative attack based on decryption failures that work against certain NTRU paddings.

I.2 The NTRU Encryption Scheme

I.2.1 Definitions and Notation

NTRUEncrypt operations take place in the quotient ring of polynomials $\mathcal{P} = \mathbb{Z}[X]/(X^N - 1)$. In this ring, addition of two polynomials is defined as pairwise addition of the coefficients of the same degree, and multiplication is defined as convolution multiplication. The convolution product $\mathsf{h} = \mathsf{f} * \mathsf{g}$ of two polynomials f and g is given by taking the coefficient of X^k to equal $\mathsf{h}_k = \sum_{i+j \equiv k \bmod N} \mathsf{f}_i \cdot \mathsf{g}_j$. Several different measures of the size of a polynomial turn out to be useful. We define the *norm* of a polynomial f in the usual way, as the square root of the sum of the squares of its coefficients. We define the *width* of a polynomial f as the difference between its largest coefficient and its smallest coefficient.

The fundamental parameter in NTRUEncrypt is N, the ring dimension. The parameter N is taken to be prime to prevent attacks due to Gentry [110], and sufficiently large to prevent lattice attacks. We also use two other parameters, p and q, which are relatively prime. Standard practice is to take q to be the power of 2 between $N/2$ and N, and p to be either the integer 3 or the polynomial $2 + X$. We thus denote p as a polynomial p in the following, and we focus on the case $\mathsf{p} = 2 + X$ as $p = 3$ is no longer recommended in NTRU standards [61, 62].

I.2.2 Overview

The basic NTRUEncrypt key generation, encryption, and decryption primitives are as follows.
Key Generation — Requires a source of (pseudo-)random bits, and subspaces $\mathcal{L}_\mathsf{f}, \mathcal{L}_\mathsf{g} \subseteq \mathcal{P}$ from which the polynomials f and g are drawn. These subspaces have the property that all polynomials

in them have small width – for example, they are now commonly taken to be the space of *all binary polynomials* with d_f, d_g 1s respectively.

- INPUT : Values N, p, q.
  - Randomly choose $\mathsf{f} \in \mathcal{L}_\mathsf{f}$ and $\mathsf{g} \in \mathcal{L}_\mathsf{g}$ in such a way that both f and g are invertible mod$q$;
  - Set $\mathsf{h} = \mathsf{p} * \mathsf{g} * \mathsf{f}^{-1} \bmod q$.
- PUBLIC OUTPUT : The input parameters and h.
- PRIVATE OUTPUT : The public output, f, and $\mathsf{f_p} \equiv \mathsf{f}^{-1} \bmod \mathsf{p}$.

Encryption — Requires a source of (pseudo-)random bits, subspaces $\mathcal{L}_\mathsf{r}$ and $\mathcal{L}_\mathsf{m}$ from which the polynomials r and m are to be drawn, and an invertible means $\mathcal{M}$ of converting a binary message m to a message representative $\mathsf{m} \in \mathcal{L}_\mathsf{m}$. The subspaces $\mathcal{L}_\mathsf{r}, \mathcal{L}_\mathsf{m}$ also have the property that all polynomials in them have low width.

- INPUT : A message m and the public key h.
  - Convert m to the message representative $\mathsf{m} \in \mathcal{L}_\mathsf{m} : \mathsf{m} = \mathcal{M}(m)$;
  - Generate random $\mathsf{r} \in \mathcal{L}_\mathsf{r}$.
- OUTPUT : The ciphertext $\mathsf{e} = \mathsf{h} * \mathsf{r} + \mathsf{m} \bmod q$.

Decryption —

- INPUT : The ciphertext e, the private key f, $\mathsf{f_p}$ and the parameters p and q.
  - Calculate $\mathsf{a} = \mathsf{f} * \mathsf{e} \bmod q$. Here, "mod q" denotes reduction of a into the range $[A, A+q-1]$, where the "centering value" A is calculated by a method to be discussed later;
  - Calculate $\mathsf{m} = \mathsf{f_p} * \mathsf{a} \bmod \mathsf{p}$, where the reduction is to the range $[0, 1]$.
- OUTPUT : The plaintext m, which is m converted from a polynomial in $\mathcal{L}_\mathsf{m}$ to a message.

I.2.3 Decryption Failures

In calculating $\mathsf{a} = \mathsf{f} * \mathsf{e} \bmod q$, one actually calculates

$$\mathsf{a} = \mathsf{f} * \mathsf{e} = \mathsf{f} * (\mathsf{r} * \mathsf{h} + \mathsf{m}) = \mathsf{p} * \mathsf{r} * \mathsf{g} + \mathsf{f} * \mathsf{m} \bmod q. \tag{I.1}$$

The polynomials p, r, g, m, and f are chosen to be small in $\mathcal{P}$, and so the polynomial $\mathsf{p} * \mathsf{r} * \mathsf{g} + \mathsf{f} * \mathsf{m}$ will, with "very high probability", have width less than q. If this is the case, it is possible to reduce a into a range $[A, A+q-1]$ such that the mod q equality in the Equation (I.1) is an exact equality in $\mathbb{Z}$. If this equality is exact, the second convolution gives

$$\mathsf{f_p} * \mathsf{a} = \mathsf{p} * (\mathsf{f_p} * \mathsf{r} * \mathsf{g}) + \mathsf{f_p} * \mathsf{f} * \mathsf{m} = 0 + 1 * \mathsf{m} = \mathsf{m} \bmod \mathsf{p}, \text{ recovering } \mathsf{m}.$$

Decryption only works if the equality mod$q$ in Equation (I.1) is also an equality in $\mathbb{Z}$. This condition will not hold if A has been incorrectly chosen so that some coefficients of $\mathsf{p} * \mathsf{r} * \mathsf{g} + \mathsf{f} * \mathsf{m}$ lie outside the centering range, or if $\mathsf{p} * \mathsf{r} * \mathsf{g} + \mathsf{f} * \mathsf{m}$ happens to have a width greater than q (so that there is no mod q range that makes the Equation (I.1) an exact equality). In this case, the recovered m will differ from the encrypted m by some multiple of q mod p. These events are the *decryption failures* at the core of this paper.

Before NTRUENCRYPT can be used, the subspaces $\mathcal{L}_\mathsf{f}, \mathcal{L}_\mathsf{g}, \mathcal{L}_\mathsf{r}$ must be specified. See appendix I.7 for details of the precise form of the polynomials $\mathsf{f}, \mathsf{g}, \mathsf{r}$ used in the standard [61] and the (slightly) different one deployed in NTRU Cryptosystems' products.

I.2.4 The NTRU Assumption

Among all the assumptions introduced in [258], the most important one is the one-wayness of the NTRU primitive, namely that the following problem is asymptotically hard to solve :

Definition I.1 (The NTRU Inversion Problem) *For a given security parameter k, which specifies N, p, q and the spaces $\mathcal{L}_\mathsf{f}$, $\mathcal{L}_\mathsf{g}$, $\mathcal{L}_\mathsf{m}$, and $\mathcal{L}_\mathsf{r}$, as well as a random public key h and $\mathsf{e} = \mathsf{h} * \mathsf{r} + \mathsf{m}$,*

where $\mathsf{m} \in \mathcal{L}_\mathsf{m}$ *and* $\mathsf{r} \in \mathcal{L}_\mathsf{r}$*, find* m*. We denote by* $\mathsf{Succ}^{\mathsf{ow}}_{\mathsf{ntru}}(\mathcal{A})$ *the success probability of any adversary* $\mathcal{A}$*.*

$$\mathsf{Succ}^{\mathsf{ow}}_{\mathsf{ntru}}(\mathcal{A}) = \Pr\left[\begin{array}{l}(\mathsf{h},\star) \leftarrow \mathcal{K}(1^k), \mathsf{m} \in \mathsf{Messages}, \mathsf{r} \in \mathsf{Random},\\ \mathsf{e} = \mathsf{h} * \mathsf{r} + \mathsf{m} \bmod q : \mathcal{A}(\mathsf{e},\mathsf{h}) = \mathsf{m}\end{array}\right].$$

I.3 The NTRU Paddings

For clarity reasons, in the description of the paddings, we consider the encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, which is an improvement of the plain NTRU cryptosystem that includes the two public encodings $\mathcal{M}$ and $\mathcal{R}$:

$$\left\{\begin{array}{rcll}\mathcal{K}(1^k) & = & (\mathsf{pk} = \mathsf{h}, \mathsf{sk} = (\mathsf{f}, \mathsf{f_p})), & \text{with,}\\ \mathcal{E}_{\mathsf{pk}}(m; r) & = & \mathcal{M}(m) + \mathcal{R}(r) * \mathsf{h} \bmod q, & \mathcal{M} : \mathsf{Messages} = \{0,1\}^{\mathsf{mLen}} \rightarrow \mathcal{L}_\mathsf{m}\\ \mathcal{D}_{\mathsf{sk}}(\mathsf{e}) & = & \mathcal{M}^{-1}((\mathsf{e} * \mathsf{f} \bmod q)_A * \mathsf{f_p}) & \mathcal{R} : \mathsf{Random} = \{0,1\}^{\mathsf{rLen}} \rightarrow \mathcal{L}_\mathsf{r}\end{array}\right.$$

Because of the encodings, without any assumption, recovering the bit-string m is as hard as recovering the polynomial $\mathsf{m} = \mathcal{M}(m)$. However, recovering ℓ bits of m does not necessarily provide ℓ bits of the polynomial $\mathsf{m} = \mathcal{M}(m)$, which is the reason why stronger assumptions were introduced in [258].

I.3.1 Description

Following the publication of [168], several padding schemes were proposed in [152, 151] to protect NTRUEncrypt against adaptive chosen-ciphertext attacks. The resulting schemes were studied in [258], but under the assumption that the probability of decryption failures was negligible. We briefly review one of these schemes, known as SVES-1 and standardized in [61].

Let m be the original plaintext represented by a k_1-bit string. For each encryption, one generates a random string b, whose bit-length k_2 is between 40 and 80 [152]. However, $k_1 + k_2 \leq \mathsf{mLen}$. Let $\|$ denote bit-string concatenation.

To encrypt, first split each of m and b into equal size pieces $m = \overline{m} \,\|\, \underline{m}$ and $b = \overline{r} \,\|\, \underline{r}$. Then, use two hash functions F and G that map $\{0,1\}^{k_1/2+k_2/2}$ into itself, to compute :

$$m_1 = (\overline{m} \,\|\, \overline{r}) \oplus F(\underline{m} \,\|\, \underline{r}) \text{ and } m_2 = (\underline{m} \,\|\, \underline{r}) \oplus G(m_1).$$

A third hash function, $H : \{0,1\}^{\mathsf{mLen}} \rightarrow \{0,1\}^{\mathsf{rLen}}$, is applied to yield the ciphertext :

$$\mathcal{E}^3_{\mathsf{pk}}(m; b) = \mathcal{E}_{\mathsf{pk}}(m_1 \,\|\, m_2; H(m \,\|\, b)).$$

The decryption algorithm consists of recovering m, b from the plain NTRU decryption, and re-encrypting to check whether one obtains the given ciphertext. This is equivalent to extracting, from m and e, the alleged r, and check whether it is equal to $\mathcal{R}(H(m \,\|\, b))$. We denote by Π^3 the corresponding encryption scheme.

I.3.2 Chosen-Ciphertext Attacks

First of all, one may note that because of the random polynomial r that is generated from $H(m \,\|\, b)$, nobody can generate a valid ciphertext without knowing both the plaintext m and the random b, except with negligible probability, for well-chosen conversion $\mathcal{R}$ (which is not necessarily injective, according to [61, 62]). Indeed, for a given ciphertext e, at most one r is acceptable. Without having asked $H(m \,\|\, b)$, the probability for $\mathcal{R}(H(m \,\|\, b))$ to be equal to r is less than ε_r :

$$\varepsilon_r = \max_{\mathsf{r}}\{\Pr_{x \in \{0,1\}^{\mathsf{rLen}}}[\mathsf{r} = \mathcal{R}(x)]\}.$$

However, to lift any security level from the CPA scenario to the CCA-one, one needs a good simulation of the decryption oracle : since any valid ciphertext has been correctly constructed, one looks at the list of the query-answers to the H-oracle, and can re-encrypt each possible plaintext to check which one is the good one. This perfectly simulates the decryption of validly generated ciphertexts, unless a *decryption failure* occurs (Fail event). In the latter case, the output decrypted plaintext is not the encrypted one, whereas the output simulated plaintext is : the probability of bad simulation is finally less than $\varepsilon_r + p(\mathsf{m}, \mathsf{r}, \mathsf{h})$, where

$$p(\mathsf{m}, \mathsf{r}, \mathsf{h}) = \Pr_{\mathsf{f},\mathsf{g}}[\mathcal{D}_{\mathsf{f},\mathsf{f}_\mathsf{p}}(\mathcal{E}_\mathsf{h}(\mathsf{m}, \mathsf{r})) \ \mathsf{Fail} \ | \mathsf{h} = \mathsf{p} * \mathsf{g} * \mathsf{f}^{-1} \ (\mathrm{mod}\, q)].$$

The security analyses in [258] were performed assuming that $p(\mathsf{m}, \mathsf{r}, \mathsf{h})$ is negligible (and even 0). But we shall see later that this is unfortunately not the case. We thus refine the security analyses and even show that the parameters have to be chosen differently.

I.4 Decryption Failures and Provable Security

I.4.1 Wrap Failures and Gap Failures

When decrypting, the recipient must place the coefficients of $\mathsf{a} = \mathsf{f} * \mathsf{e} \bmod q$ into the correct range $[A, A + q - 1]$. To calculate A, we use the fact that convolution multiplication respects the homomorphism $(\mathsf{a} * \mathsf{b})(1) = \mathsf{a}(1) \cdot \mathsf{b}(1)$, where $\mathsf{a}(1)$ is the sum of the coefficients of the polynomial a. The decrypter knows $\mathsf{r}(1)$ and $\mathsf{h}(1)$, and so he can calculate $I = \mathsf{m}(1) = \mathsf{e}(1) - \mathsf{r}(1) \cdot \mathsf{h}(1) \bmod q$. Assuming $\mathsf{m}(1)$ lies in the range $[N/2 - q/2, N/2 + q/2]$, we can calculate the average value of a coefficient of $\mathsf{p} * \mathsf{r} * \mathsf{g} + \mathsf{f} * \mathsf{m}$ and set

$$A = \left\lfloor \frac{\mathsf{p}(1) \cdot \mathsf{r}(1) \cdot \mathsf{g}(1) + \mathsf{f}(1) \cdot I}{N} \right\rceil - \frac{q}{2}.$$

The assumption about the form of m is a reasonable one, because for randomly chosen m with $N = 251$ there is a chance of less than 2^{-56} that $\mathsf{m}(1)$ will be less than $N/2 - q/2$ or greater than $N/2 + q/2$. In any case, the value of $\mathsf{m}(1)$ is known to the encrypter, so they do not learn anything from a decryption failure based on m being too thick or too thin.

Having obtained A, we reduce a into the range $[A, A + q - 1]$. If the actual values of any of the coefficients of $\mathsf{p} * \mathsf{r} * \mathsf{g} + \mathsf{f} * \mathsf{m}$ lie outside this range, decryption will produce the wrong message and this will be picked up in the re-encryption stage of decryption. However, if $\mathsf{p} * \mathsf{r} * \mathsf{g} + \mathsf{f} * \mathsf{m}$ has a width less than q, there is still an A for which decryption is possible. This case, where the initial choice of A does not work but there is a choice of A which could work, is referred to as a "*wrap failure*".

Wrap failures are more common than "*gap failures*", where the width of $\mathsf{p} * \mathsf{r} * \mathsf{g} + \mathsf{f} * \mathsf{m}$ is strictly greater than q. The standard [61] therefore recommends that, on the occurrence of a decryption failure, the decrypter adjusts the decryption range by setting A' successively equal to $A \pm 1, \pm 2, \ldots$, placing the coefficients of a into the new range $[A', A' + q - 1]$, and performing the mod p reduction. This is to be carried out until A' differs from A by some set T, the "*wrapping tolerance*" ; if decryption has not succeeded at that point, the decryption function outputs the invalid symbol $\perp$.

This method increases the chance that a ciphertext will eventually decrypt ; however, an attacker with access to timing information can tell when this recentering has occurred. For standard $N = 251$ parameters and NTRUEncrypt implemented as in NTRU products, a wrap failure on random m occurs once every 2^{21} messages, while a gap failure occurs about once every 2^{43} messages. The centering method above will therefore leak information at least once every million or so decryptions, and possibly more often if the attacker can carry out some precomputation as in [235]. For NTRUEncrypt as implemented following the EESS standard, the number of messages is much lower : for $N = 251$, a gap failure occurs once every 2^{25} message.

I.4.2 Provable Security

In order to deal with any padding, one needs more precise probability informations than just $p(\mathsf{m},\mathsf{r},\mathsf{h})$:

$$\begin{array}{rcl} p(\mathsf{m},\mathsf{h}) & = & \Pr_{\mathsf{r}}[p(\mathsf{m},\mathsf{r},\mathsf{h})] \\ p(\mathsf{h}) & = & \Pr_{\mathsf{m}}[p(\mathsf{m},\mathsf{h})] = \Pr_{\mathsf{m},\mathsf{r}}[p(\mathsf{m},\mathsf{r},\mathsf{h})] \end{array} \qquad \begin{array}{rcl} p_0 & = & E_{\mathsf{h}}[\max_{\mathsf{m},\mathsf{r}}\{p(\mathsf{m},\mathsf{r},\mathsf{h})\}]; \\ p_1 & = & E_{\mathsf{h}}[\max_{\mathsf{m}}\{p(\mathsf{m},\mathsf{h})\}]; \\ p_2 & = & E_{\mathsf{h}}[p(\mathsf{h})]. \end{array}$$

Note that all of these probabilities are averages over the whole space of h. Implicit in these definitions is the assumption that, even if some keys (f,g) are more likely than others to experience decryption failures, an adversary cannot tell from the public key h which private key is more failure-prone.

Clearly, one cannot ensure that $p(\mathsf{m},\mathsf{r},\mathsf{h})$ is small, so p_0 is likely to be non-negligible. However, in several paddings, $\mathsf{r} = \mathcal{R}(H(m \,\|\, b))$, where H is a random oracle, therefore the probability of bad simulation involves at least p_1, or even p_2. As discussed above, with recommended parameters, p_2 can be as small as 2^{-43}. Even this is not negligible, but better parameters may hopefully make these values negligible. However, there is a gap between the existence of such a pair (m,r) that makes a *decryption failure*, and the feasibility, for an adversary, to find/build some :

$$\mathsf{Succ}^{\mathcal{A}}_{\mathsf{fail}}(\mathsf{h}) = \Pr_{\mathsf{f},\mathsf{g}}[\mathcal{D}_{\mathsf{f},\mathsf{f}_\mathsf{p}}(\mathcal{E}_{\mathsf{h}}(\mathsf{m},\mathsf{r}))\ \mathsf{Fail} \,|\, \mathsf{h} = \mathsf{p} * \mathsf{g} * \mathsf{f}_\mathsf{p} \bmod q, (\mathsf{m},\mathsf{r}) \leftarrow \mathcal{A}(\mathsf{h})].$$

As above, one needs to study the probabilities over some classes of adversaries, or when the adversary does not have the entire control over m or r :

$$\begin{array}{rcl} \tilde{p}_0(t) & = & \max\{E_{\mathsf{h}}[\mathsf{Succ}^{\mathcal{A}}_{\mathsf{fail}}(\mathsf{h})], |\mathcal{A}| \le t\} \\ \tilde{p}_1(t,Q) & = & \textit{idem where } (\mathsf{m},y) \leftarrow \mathcal{A}(\mathsf{h}), \mathsf{r} = G(\mathsf{m},y) \\ \tilde{p}_2(t,Q) & = & \textit{idem where } (x,y) \leftarrow \mathcal{A}(\mathsf{h}), \mathsf{m} = F(x,y), \mathsf{r} = G(x,y). \end{array}$$

In the above bounds, for $\tilde{p}_0(t)$, we consider any adversary whose running time is bounded by t. For $\tilde{p}_1(t,Q)$ and $\tilde{p}_2(t,Q)$, F and G are furthermore assumed to be random oracles, to which the adversary can ask up to Q queries. Clearly, for any t and any Q,

$$\tilde{p}_0(t) \le p_0 \quad \tilde{p}_1(t,Q) \le Q \times p_1 \quad \tilde{p}_2(t,Q) \le Q \times p_2.$$

We now reconsider the SVES-1 padding scheme, keeping these probabilities in mind. We note that the adversary controls m, by deriving m and b from the required m_1 and m_2. However, r is out of the adversary's control. Therefore after q_D queries to the decryption oracle and q_H queries to the random oracle H, the probability of a decryption failure is less than $q_D \times \tilde{p}_1(t, q_H)$, where t is the running time of the adversary. Denoting by $T_{\mathcal{E}}$ the time for one encryption, one gets the following improvement for a CCA attacker over a CPA one :

$$\mathsf{Adv}^{\mathsf{ind-cca}}_{\Pi^3}(t) \le \mathsf{Adv}^{\mathsf{ind-cpa}}_{\Pi^3}(t + q_H T_{\mathcal{E}}) + 2q_D \times (\varepsilon_r + \tilde{p}_1(t, q_H)) \,.$$

I.4.3 Improved Paddings

In [258], new paddings have been suggested, with better provable security (based on the NTRU inversion problem only). But decryption failures have been ignored again.

The OAEP-based Scheme — The first suggestion was similar to the SVES-1 padding, also using two more hash functions

$$F : \{0,1\}^{k_1} \to \{0,1\}^{k_2} \text{ and } G : \{0,1\}^{k_2} \to \{0,1\}^{k_1}.$$

One first computes $s = m \oplus G(b)$ and $t = b \oplus F(s)$. The ciphertext consists of $\mathcal{E}_{\mathsf{pk}}(s \,\|\, t; H(m \,\|\, b))$. Of course, the decryption checks the validity of r, relatively to $H(m \,\|\, b)$. The OAEP construction

provides semantic security, while the H function strengthens it to chosen-ciphertext security (as already explained).

Here, the adversary can choose s, t directly, then reverse the OAEP construction to obtain m, b. However, they cannot control r. Therefore

$$\mathsf{Adv}^{\mathsf{ind-cca}}_{\mathsf{oaep}'}(t) \leq 2\mathsf{Succ}^{\mathsf{ow}}_{\mathsf{ntru}}(t + QT_{\mathcal{E}}) + 2q_D \times (\varepsilon_r + \tilde{p}_1(t, q_H)) + \frac{4q_H}{2^{k_1}} + \frac{2q_G}{2^{k_2}}.$$

where $Q = q_F q_G + q_H$. But this makes a quadratic reduction, as for any OAEP-based cryptosystem [25, 102]. The particular above construction admits a better reduction, but under a stronger computational assumption.

SVES-2 —

The successor to SVES-1 proposed for standardization is a minor variant of the above [62], designed to handle variable length messages. In SVES-2, one uses two hash functions F and G, and form $M_1 = b \,\|\, \mathrm{len}(m) \,\|\, m_1$, $M_2 = m_2 \,\|\, 000\ldots$. In SVES-2, the message length is restricted to be an integer number of bytes, and the length is encoded in a single byte. The final N mod 8 bits of M_2 will always be zeroes (we use N_8 to denote N mod 8). We form $s = M_2 \oplus F(M_1)$, $t = M_1 \oplus G(s)$, and calculate the ciphertext as $\mathcal{E}_{\mathsf{pk}}(t \,\|\, s; H(m \,\|\, b))$. On decryption, the usual checks are performed, and in addition the decrypter checks that the length is valid and that M_2 consists of 0s from the end of the message onwards.

In this case, an attacker who chooses s, t and reverses the OAEP construction must get the correct length of m and the correct 0 bits at the end. However, an attacker can choose s, then select t such that $t \oplus G(s)$ has the correct form to be M_1, that is, such that $\mathrm{len}(m)$ is its maximum value. The reverse OAEP operation on s, t will then yield a valid M_1, M_2 if the last N_8 bits of $s \oplus F(M_1)$ are 0. Therefore an attacker can control all the bits of s, t with probability 2^{8+N_8}, or all but 8 of the bits of s, t with probability 2^{N_8}. Therefore

$$\mathsf{Adv}^{\mathsf{ind-cca}}_{\mathrm{SVES-2}}(t) \leq 2\mathsf{Succ}^{\mathsf{ow}}_{\mathsf{ntru}}(t + QT_{\mathcal{E}}) + 2q_D \times \left(\varepsilon_r + \frac{\tilde{p}_1(t, q_H)}{2^{N_8}}\right) + \frac{4q_H}{2^{k_1}} + \frac{2q_G}{2^{k_2}}.$$

where $Q = q_F q_G + q_H$.

NTRU-REACT — Thanks to the OW-PCA–security level of the NTRU primitive (granted the pseudo-inverse of h), one can directly use the REACT construction [276], in which the decryption algorithm of course checks the validity of c, but also the validity of r. The semantic security is clear, since the adversary has no advantage without having asked some crucial queries to the hash functions. With chosen-ciphertext attacks, the adversary cannot produce a valid ciphertext without having built correctly the authentication tag, except with probability $1/2^{k_2}$. Therefore, the simulation of the decryption oracle is perfect, unless a decryption failure occurs : the adversary knows b and thus m, but makes a decryption failure, that is not detected by the simulation. Since the adversary has the control over both m and r, the security against chosen-ciphertext attacks is not very high :

$$\mathsf{Adv}^{\mathsf{ind-cca}}_{\mathsf{react}}(t) \leq 2\mathsf{Succ}^{\mathsf{ow}}_{\mathsf{ntru}}(t + (q_G + q_H)T_{\mathcal{E}}) + 2q_D \times \left(\frac{1}{2^{k_2}} + \tilde{p}_0(t)\right).$$

In the *Improved NTRU-REACT*, the adversary completely loses control over r, which improves the security level, but not enough, since $\tilde{p}_0(t)$ is replaced by $\tilde{p}_1(t)$ only.

I.4.4 Comments

It is clear that decryption failures on valid ciphertexts mean that NTRUEncrypt with the parameter sets given cannot have provable security to the level claimed. In the presence of decryption failures, it is impossible to correctly simulate the decryption oracle : the simulator will output a valid

decryption on certain ciphertexts which the genuine decryption engine will fail to decrypt. Without knowledge of the private key, it is impossible to build a simulator ; and if the simulator requires knowledge of the private key, it is impossible to have provable security. In the next section we will see how to use this information to recover the private key with considerably less effort than a standard lattice attack would take.

I.5 Some Attacks Based on Decryption Failures

In this section we present some attacks against NTRUEncrypt as implemented in NTRU Cryptosystems' products for the $N = 251$ parameter set. See [62] for details of this parameter set, and Appendix I.7 for information about the precise structure of f, g and r used. We stress that the attacks work even better on the EESS standard [61, 62], because decryption failures arise much more frequently there. Although no paddings can prevent decryption failures, it turns out that some paddings are more prone to attacks based on decryption failures : this is because the attacker has more or less flexibility on the choice of the actual message and random nonce actually given as input to the encryption primitive, depending on the padding used.

I.5.1 Review : The Reversal of a Polynomial

Before outlining the attacks, we review the notion of the *reversal* $\bar{\mathsf{c}}(X) \equiv \mathsf{c}(X^{-1})$ of a polynomial c. If we represent c as the array $\mathsf{c} = [\mathsf{c}_0, \mathsf{c}_1, \mathsf{c}_2, \ldots, \mathsf{c}_{N-1}]$ then its reversal is $\bar{\mathsf{c}} = [\mathsf{c}_0, \mathsf{c}_{N-1}, \mathsf{c}_{N-2}, \ldots, \mathsf{c}_1]$: this is a ring automorphism. We denote by $\hat{\mathsf{c}}$ the product of c and $\bar{\mathsf{c}}$. This product has the property that $\hat{\mathsf{c}}_i = \mathsf{c} \cdot (X^i * \mathsf{c})$, or in other words that the successive terms of $\hat{\mathsf{c}}$ are obtained by taking the dot product of c with successive rotations of itself. The significance of this is that we know that $\hat{\mathsf{c}}_0 = \sum_i \mathsf{c}_i^2 = \|\mathsf{c}\|^2$, while the other terms of $\hat{\mathsf{c}}$ will be $\mathcal{O}(\|\mathsf{c}\|)$ in size.

We therefore know that $\mathsf{f} * \bar{\mathsf{f}}$ has one term of size d_f, and the others are of size about $\sqrt{d_f}$. The polynomial $\mathsf{f} * \bar{\mathsf{f}}$ is therefore of great width compared to a product of two arbitrary polynomials of the same norm as f. We therefore assume that whenever $\mathsf{p} * \mathsf{r} * \mathsf{g} + \mathsf{f} * \mathsf{m}$ is of great width, it means that r is correlated significantly with $\bar{\mathsf{g}}$ and m is correlated significantly with $\bar{\mathsf{f}}$.

I.5.2 A General Attack

We derive a powerful chosen-ciphertext attack that works independently of the padding used. Indeed, assume that an attacker is able to collect many triplets $(\mathsf{m}, \mathsf{r}, \mathsf{e})$ such that e is an encryption of m with random nonce r, which cannot be correctly decrypted. If the probability of decryption failure is sufficiently high, an attacker could obtain such triplets by mounting a weak chosen-ciphertext attack, independently of the padding, by simply selecting random messages, encrypting them until decryption failures occur (which can be checked thanks to the decryption oracle). Interestingly, such an attack only uses valid ciphertexts.

For such a triplet $(\mathsf{m}, \mathsf{r}, \mathsf{e})$, we know that $\mathsf{p} * \mathsf{r} * \mathsf{g} + \mathsf{f} * \mathsf{m}$ is of great width, and the previous section suggests that there is an integer i such that r somehow looks like $X^i * \bar{\mathsf{g}}$ and m somehow looks like $X^i * \bar{\mathsf{f}}$. Unfortunately, we do not know the value of i, otherwise it would be trivial to recover $\bar{\mathsf{f}}$ by simply taking the average of $X^{-i} * \mathsf{m}$. However, we can get rid of the unknown i by using the reversal : if m and r look like respectively $X^i * \bar{\mathsf{f}}$ and $X^i * \bar{\mathsf{g}}$, then $\hat{\mathsf{m}}$ and $\hat{\mathsf{r}}$ must look like respectively $\hat{\mathsf{f}}$ and $\hat{\mathsf{g}}$. Once $\hat{\mathsf{f}}$ and $\hat{\mathsf{g}}$ are derived by averaging methods, f and g themselves may be recovered in polynomial time using an algorithm due to Gentry and Szydlo [112]. Strictly speaking, to apply [112], one also needs to determine the ideal spanned by f which can be derived from $\hat{\mathsf{f}} = \mathsf{f} * \bar{\mathsf{f}}$ and $\mathsf{f} * \bar{\mathsf{g}}$ (which is itself obtained by multiplying from $\bar{\mathsf{H}}$ and $\hat{\mathsf{f}} = \mathsf{f} * \bar{\mathsf{f}}$).

To check the validity of this attack, we would need to find a lot of decryption failures, which is relatively time-consuming, depending on the parameters. Instead, we checked that the attack worked,

B	Messages
18	100,000

binary f

B	Messages	Norm (guess - $\hat{\mathsf{f}}$)
36	500,000	15.5
36	1,400,000	7.38

$\mathsf{f} = 1 + \mathsf{p} * \mathsf{F}$

TAB. I.1 – Messages to recover $\hat{\mathsf{f}}$ for various values of B using $\mathcal{O}_B$.

by experimenting with a weaker attack based on the following oracle $\mathcal{O}_B$: When the oracle $\mathcal{O}_B$ is queried on a valid ciphertext e, it indicates whether or not the width of $\mathsf{p} * \mathsf{r} * \mathsf{g} + \mathsf{f} * \mathsf{m}$ is greater than B. Thus, the oracle $\mathcal{O}_q$ simply detects gap failures. By using values of B much smaller than q, we are able to verify the behavior of our attack in a reasonable time, with the following algorithm :

1. Set $\mathsf{u}, \mathsf{v} = 0$.
2. Generate a large number of valid ciphertexts $\mathsf{e} = \mathsf{r} * \mathsf{h} + \mathsf{m} \bmod q$. For each ciphertext e :
   (a) Call $\mathcal{O}_B(\mathsf{e})$.
   (b) If $\mathcal{O}_B(\mathsf{e})$ shows the width of a as being greater than B, set $\mathsf{u} = \mathsf{u} + \hat{\mathsf{m}}$; set $\mathsf{v} = \mathsf{v} + \hat{\mathsf{r}}$.
3. Divide u and v by the number of valid ciphertexts used.

Over a long enough transcript, u and v should converge to $\hat{\mathsf{f}}$ and $\hat{\mathsf{g}}$. We investigated this for f binary and for $\mathsf{f} = 1 + \mathsf{p} * \mathsf{F}$, to see how many messages with width greater than B were necessary to recover $\hat{\mathsf{f}}$ exactly. The results are shown in tables I.1. Experimentally, we find that we approximate $\hat{\mathsf{f}}$ best by $\hat{\mathsf{f}}_i = (\langle \hat{\mathsf{m}} \rangle_i - 61.24747)/0.007882857$.

When the distance from the guess to $\hat{\mathsf{f}}$ is about $\sqrt{N}/2 \approx 7.9$, we can essentially recover $\hat{\mathsf{f}}$ by rounding. We can conclude that from a real decryption oracle ($k_2 \leq 128$) no more than a million decryption failures, and perhaps considerably fewer, are necessary to recover $\hat{\mathsf{f}}$ and $\hat{\mathsf{g}}$. This validates our general chosen-ciphertext attack which applies to all paddings, and shows that the security of NTRU encryption in the EESS standard [61, 62] clearly falls far short of the hoped-for level of 2^{80}.

We note that one could imagine an even more powerful attack, where the attacker would simply average $\hat{\mathsf{e}}$ for those e that cause wraps. For a sufficiently long transcript, this will converge to $A + B\hat{\mathsf{f}} + C\hat{\mathsf{g}}$. We have not investigated this idea in full – it will undoubtedly involve longer convergence times than the other attacks outlined above – but it is interesting that a successful attack may be mounted even by an entirely passive attacker.

I.5.3 A Specific Attack Based on Controlling m

The previous attack works against any padding and already emphasizes the importance of decryption failures on the security of NTRU encryption. Here, we describe a slightly more efficient chosen-ciphertext attack tailored to the SVES-1 padding scheme, based on the fact that an attacker essentially controls m directly (see above). This attack shows that certain paddings are weaker than others with respect to attacks based on decryption failures. We denote by r_m the value of r obtained from the m and b obtained from a given m in valid encryption. The strategy will be to try and cause wrap failures. We introduce the notation

$$\mathcal{B}_i = \{\text{binary polynomials with } i \text{ 1s and } N - i \text{ zeroes}\},$$

and denote by $\mathsf{Flat}(\mathsf{c})$ the operation of taking c and setting all terms that are 1 or more to be exactly equal to 1. Experimentally, the average width of the various polynomials is :

$$\langle \mathsf{Width}(\mathsf{p} * \mathsf{r} * \mathsf{g}) \rangle \approx 41; \quad \langle \mathsf{Width}(\mathsf{f} * \mathsf{m}) \rangle \approx 47; \quad \langle \mathsf{Width}(\mathsf{p} * \mathsf{r} * \mathsf{g} + \mathsf{f} * \mathsf{m}) \rangle \approx 62.$$

Overlap	Pr[Overlap] (Theoretical)	Pr[Overlap]) (Experimental)	Pr[Wrap]	No. Left from 2^{30} F_{15}s
15	$2^{-20.7}$	–	$2^{-14.4}$	10
14	$2^{-15.135}$	$2^{-14.7}$	$2^{-15.5}$	200
13	$2^{-10.698}$	$2^{-10.70}$	$2^{-16.6}$	2,000
12	$2^{-6.981}$	$2^{-7.116}$	$2^{-17.5}$	20,000
11	$2^{-3.857}$	$2^{-4.030}$	$2^{-18.5}$	80,000
10	$2^{-1.423}$	$2^{-1.693}$	2^{-22}	40,000
9	–	$2^{-0.91}$	2^{-23}	25,000
8	–	$2^{-3.49}$	2^{-24}	2,000
7	–	$2^{-14.5}$	–	–

TAB. I.2 – Effects of first cleaning on a set of 2^{30} F_{15}

If the attacker can increase the width of $\mathsf{p} * \mathsf{r} * \mathsf{g} + \mathsf{f} * \mathsf{m}$ to 128, he will cause a gap failure ; alternatively, if he can add about 33 to the largest term in $\mathsf{p} * \mathsf{r} * \mathsf{g} + \mathsf{f} * \mathsf{m}$ while leaving the others essentially unchanged, this will cause a wrap failure. The following attack exploits this observation

Step 1 : first cleaning of random strings.

The attacker picks a random $\mathsf{F}_{15} \in \mathcal{B}_{15}$, $\mathsf{D} \in \mathcal{B}_5$, and forms

$$\mathsf{m} = \mathsf{Flat}((1 + X) * \mathsf{D} * \mathsf{F}_{15}).$$

On decryption (with $\mathsf{p} = 2 + X$), a will include a term approximately equal to

$$(1 + X) * (2 + X) * \mathsf{f} * \mathsf{D} * \mathsf{F}_{15} = (2 + 3X + X^2) * \mathsf{f} * \mathsf{D} * \mathsf{F}_{15}.$$

If the 1s in F_{15} match a set of 15 1s in $\bar{\mathsf{f}}$, then we know that at least one term in $\mathsf{p} * \mathsf{F} * \mathsf{m}$ will be 45 or more because of the 3 in $(1 + X) * (2 + X)$. With high odds, this will mean that $\mathsf{p} * \mathsf{F} * \mathsf{m}$ has greater than average width, and so greater than average chance of causing a decryption failure. This lets the attacker attempt to identify substrings of $\hat{\mathsf{F}}$:
Attack : Step 1. The attacker picks a random $\mathsf{F}_{15} \in \mathcal{B}_{15}$, $\mathsf{D} \in \mathcal{B}_5$ and forms $\mathsf{m} = \mathsf{Flat}((1+X) * \mathsf{D} * \mathsf{F}_{15})$. For all rotations of m, he submits $\mathsf{e} = \mathsf{r}_\mathsf{m} * \mathsf{h} + \mathsf{m} \bmod q$ to the decryption oracle. If any rotation of m causes a wrap, he stores F_{15} ; otherwise, he discards it.

Obviously, there will be a large number of false positives in this step. An m might cause a decryption failure purely through luck ; alternatively, an F_{15} which has an overlap of 14 rather than 15 with $\bar{\mathsf{F}}$ will have a good chance of causing a wrap failure. We cannot distinguish between the cases immediately, so the strategy is to take an initial set of random F_{15}s, and use the decryption oracle to "clean" them so that the resulting set has a greater proportion of strings of high overlap with f.

Table I.2 shows the effect of the first cleaning on a set of 2^{30} random F_{15}s. The wrap probabilities were determined experimentally, using our knowledge of f, by generating strings of a specific overlap and testing to see if they caused wraps. The final column is given by $2^{30} \cdot \Pr[\text{overlap}] \cdot \Pr[\text{wrap}] \cdot N$, as we try all the rotations of each of the 2^{30} ms. The total number of queries to the decryption oracle is $N \cdot 2^{30} \approx 2^{38}$.

From table I.2 we see that even the first cleaning is very effective : from a random set of size 2^{30} where most F_{15}s have an overlap of 9 with f, we have created a set of size about 2^{17} where the most commonly occurring overlap is 11. We now want to further improve the quality of our set of F_{15}s.

Overlap	No. Tested	No. Left
15	10	3
14	200	40
13	2,000	120
12	20,000	800
11	80,000	1500
10	40,000	2
9	25,000	1
8	2,000	0

TAB. I.3 – Effects of second cleaning on the set of F_{15}

Step 2 : second cleaning.

The attacker now queries each surviving F_{15} by using different values of D to create ms, and observing whether or not these cause wraps.

Attack : Step 2. For each F_{15} that survived the first cleaning step, the attacker picks several $\mathsf{D} \in \mathcal{B}_5$. For each D, he forms $\mathsf{m} = \mathsf{Flat}((1+X) * \mathsf{D} * \mathsf{F}_{15})$. For all rotations of m, he submits $\mathsf{e} = \mathsf{r_m} * \mathsf{h} + \mathsf{m} \bmod q$ to the decryption oracle. If any rotation of m causes a wrap, he stores F_{15} ; otherwise, he discards it.

Table I.3 shows the results of performing this step, choosing 2^8 Ds for each F_{15}. After this cleaning, there are almost no F_{15}s left with an overlap of less than 11. The total work in this stage is $2^8 \cdot 2^{17} \cdot N \approx 2^{33}$, and there are about 2^{12} F_{15}s left.

Now that the attacker has a relatively good set of F_{15}, he can try and assemble them to recover F.

Step 3 : find correct relative rotations.

Here the challenge is to find the correct rotations of the F_{15} relative to each other. One possibility would be to pick two F_{15}s and test the ms obtained by all rotations of the two against each other. However, it appears that we get better results by picking sets of three F_{15}s and trying all their relative rotations.

Attack : Step 3. Let $\mathsf{F}_1, \mathsf{F}_2, \mathsf{F}_3$ be any three of the F_{15} that survived the second cleaning step. For each $0 \leq i, j < N$, set

$$\mathsf{F}_{\sim 45} = \mathsf{Flat}(\mathsf{F}_1 + X^i \mathsf{F}_2 + X^j \mathsf{F}_3)$$

(Note that $\mathsf{F}_{\sim 45}$ will typically have slightly fewer than 45 1s). Set $\mathsf{m} = \mathsf{Flat}((1 + X) * \mathsf{F}_{\sim 45})$. For all rotations of m, submit $\mathsf{e} = \mathsf{r_m} * \mathsf{h} + \mathsf{m} \bmod q$ to the decryption oracle. Store the number of wraps caused by m. Once all i, j pairs have been exhausted, pick another three $\mathsf{F}_1, \mathsf{F}_2, \mathsf{F}_3$ and repeat. Continue until all the F_{15} have been used.

Figure I.1 shows the wrap probability for m obtained as specified above. If $\mathsf{F}_1, \mathsf{F}_2, \mathsf{F}_3$ are rotated correctly relative to each other, the overlap with $\bar{\mathsf{F}}$ will typically be 33 or more, leading to a significantly greater chance of a wrap. Note that, because of the $(2+X)$ in f, if $\mathsf{F}_1 + X^i \mathsf{F}_2 + X^j \mathsf{F}_3$ is the correct relative alignment of $\mathsf{F}_1, \mathsf{F}_2, \mathsf{F}_3$, a large number of wraps will also be caused by $\mathsf{F}_1 + X^{i\pm 1} \mathsf{F}_2 + X^{j\pm 1} \mathsf{F}_3$. This helps us to weed out freak events : rather than simply taking the relative rotation of $\mathsf{F}_1, \mathsf{F}_2, \mathsf{F}_3$ that gives the highest number of wraps, we look for the set of three consecutive rotations that give the highest total number of wraps and pick the rotation in the middle.

This step takes about $N^3 \cdot 2^{12} \approx 2^{36}$ work, and at the end of it we have about 2^{10} strings of length about 45, which will in general have an overlap of 33 or more with $\bar{\mathsf{F}}$. The remaining task is to rotate these strings correctly relative to each other and recover $\bar{\mathsf{F}}$ from them, but this is relatively trivial.

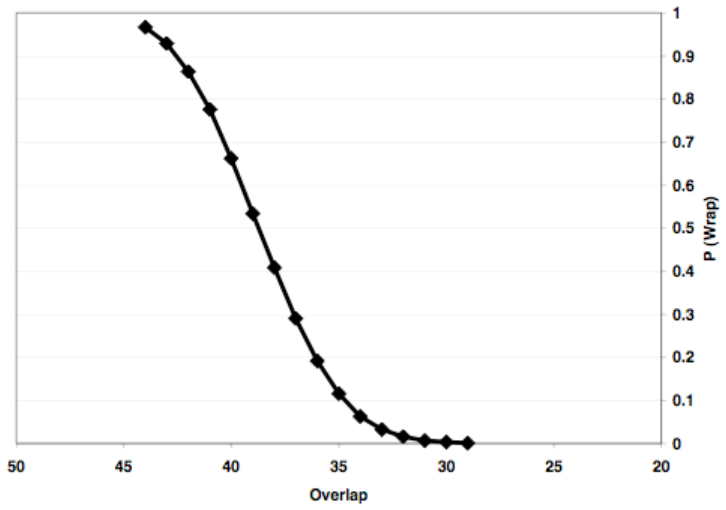


FIG. I.1 – Wrap probability on a single test for strings of length 45

Step 4 : from 45s to $\bar{\mathsf{F}}$.

We here use the fact that the $\mathsf{F}_{\sim 45}$ will be better correlated with each other when rotated correctly than in other rotations.
Attack : Step 4. Sort the $\mathsf{F}_{\sim 45}$ from the previous step in order by the number of wraps they caused. Set u equal to the $\mathsf{F}_{\sim 45}$ at the top of the list. For each other $\mathsf{F}_{\sim 45}$, find the X^i that maximizes the overlap of $X^i \cdot \mathsf{F}$ and the reference F. Set $\mathsf{u} = \mathsf{u} + X^i \cdot \mathsf{F}$. When all the Fs have been added, take the top d_f entries of u and set them equal to 1. Set the other entries to 0. This recovers F_{rev}.

Since there are 72 1s in $\bar{\mathsf{F}}$ and 179 0s, and since the $\mathsf{F}_{\sim 45}$ have typically 33 correct 1s and 12 incorrect 1s, we expect the entries in u corresponding to 1s in f_{rev} to have an average value of $33/72 \approx 0.46$, and the entries in u corresponding to 0s in f_{rev} to have an average value of $12/179 \approx 0.06$. This makes it easy to distinguish between the two. We have not implemented this part of the attack, but we do not anticipate any problems in its execution.

SVES-1 attack : Summary

We have presented an attack on the SVES-1 scheme that allows an attacker with access to decryption timing information to recover the private key in about 2^{40} queries to a decryption oracle with $N = 251$. This is a level of security that clearly falls far short of the hoped-for level of 2^{80}.

I.6 Countermeasures

NTRUENCRYPT as specified in [61] clearly falls short of the desired security levels, since it only involves the probability $\tilde{p}_1$. With the given parameters, even $\tilde{p}_2$ is likely to be non-negligible. One should thus recommend at least the following two countermeasures.

I.6.1 Changing the parameters

The parameters, and perhaps the form of f, g, and r, should be altered so that decryption failures occur no more often than the claimed security level of the parameter set, so that the probability $\tilde{p}_2$, or even p_2, is indeed negligible. (For example, for $N = 251$, an attacker should be required to carry

out 2^{80} work to find a single gap failure). Unfortunately, no efficient method is known to provably compute such a probability, though the paper [328] provides calculations under some simplifying assumptions.

I.6.2 Changing the padding

A padding scheme with the appropriate provable security properties should be adopted. We have presented both theoretical and experimental reasons for preferring an NTRUEncrypt padding scheme in which an attacker can control neither m nor r. Theoretically, only this padding scheme allows us to use p_2, the smallest of the expected decryption failure probabilities. Experimentally, we have demonstrated an attack which uses direct control of m to recover the private key faster than the attack which does not use control of m.

We therefore suggest the following padding scheme, which we call NAEP, as one that might be suitable for NTRUEncrypt. The construction uses the hash functions

$$G : \{0,1\}^{\mathsf{mLen}} \to \{0,1\}^{\mathsf{rLen}} \text{ and } H : \mathcal{P} \to \{0,1\}^{\mathsf{mLen}}.$$

As before, m is the plaintext of length k_1 bits, and b is a random string, unique for each message, of length $k_2 = \mathsf{mLen} - k_1$ bits. One computes $\mathsf{r} = \mathcal{R}(G(m \,\|\, b))$ and $\mathsf{R} = \mathsf{r} * \mathsf{h} \bmod q$. Then the ciphertext consists of $\mathcal{E}_{\mathsf{pk}}((m \,\|\, b) \oplus H(\mathsf{R}); G(m \,\|\, b))$. Of course, the decryption checks the validity of r, relatively to $G(m \,\|\, b)$.

We do not make any claim on the provable security of this scheme. An analysis of the properties of a variant of this scheme, with a specific instantiation of H, appears in [156] and claims a security result which depends on $\tilde{p}_2$ only (and of course the intractability of the basic NTRU primitive.)

Acknowledgments

We would like to thank Jeff Hoffstein and Jill Pipher for fruitful discussions and contributions.

I.7 Appendix : Standardized and Deployed Versions of NTRUEncrypt

I.7.1 Format of Objects

NTRUEncrypt as standardized in [61] uses special forms for f, g and r, and specifies a padding method which is claimed to give provable security. We review these details here.

Private key : f.

The private key f has two special features. First, f has the form $1 + \mathsf{p} * \mathsf{F}$, where F is a binary polynomial. This means that $\mathsf{f} = 1 \bmod \mathsf{p}$, and therefore that $\mathsf{f}_\mathsf{p} = 1$, eliminating the need for the second convolution on decryption [151]. Second, the binary polynomial F is of the form $\mathsf{f}_1 * \mathsf{f}_2 + \mathsf{f}_3$, where f_1, f_2 and f_3 are chosen so that :

- $\mathsf{f}_1, \mathsf{f}_2, \mathsf{f}_3$ are binary and have $d_{f_1}, d_{f_2}, d_{f_3}$ 1s respectively ;
- $\mathsf{f}_1 * \mathsf{f}_2$ is binary ;
- The 1s in f_3 are chosen such that they are not adjacent to any of the 1s in $\mathsf{f}_1 * \mathsf{f}_2$.

It should be pointed out that only the first of these restrictions is documented in [61] : the description above is of the form of private keys used in NTRU Cryptosystems' software and has the effect of decreasing the occurrence of decryption failures (but not to the point of making decryption failures sufficiently unlikely to avoid any security problems). For more details on the use of products of low Hamming weight polynomials in NTRU and other cryptosystems, see [151, 153].

Private key : g.

The private key g is chosen to be binary, to have d_g 1s, and to have no consecutive 1s. Note that the last of these restrictions is not documented in [61], but is used in NTRU Cryptosystems' software.

Message : m.

The message representative m is a binary polynomial of degree N. An algorithm for converting from octet strings to binary polynomials can be found in [61].

Blinding value : r.

The blinding value r is chosen to be of the form $\mathsf{r}_1 * \mathsf{r}_2 + \mathsf{r}_3$. Here, $\mathsf{r}_1, \mathsf{r}_2, \mathsf{r}_3$ are generated by setting them to 0, then selecting $d_{r_1}, d_{r_2}, d_{r_3}$ indices between 0 and $N-1$ and adding one to the coefficient at each index. The difference between this and taking $\mathsf{r}_1, \mathsf{r}_2, \mathsf{r}_3$ to be binary is that the indices used to generate them can repeat : for example, r_1 could consist of $d_{r_1} - 2$ 1s and one 2. A recent paper [235] uses this specific fact to recover g. The results presented here are more general and work for r of any form in the presence of decryption failures.

I.7.2 Encryption Schemes

There have been several encryption schemes associated with NTRU, but only two have been standardized in EESS #1. The first, SVES-1, proceeds as follows. It takes the number of random bits, d_b, as a parameter, and hash functions F, G, H.
Encryption – To encrypt a message m of length $|m| = N - d_b$ bits :

1. Generate the string b consisting of d_b random bits.
2. Set s equal to the first $|m|/2$ bits of m concatenated with the first $d_b/2$ bits of b. Set t equal to the last $|m|/2$ bits of m concatenated with the last $d_b/2$ bits of b.
3. Set $t' = t \oplus F(s)$. Set $s' = s \oplus G(t')$. Set $\mathsf{m}' = s'||t'$. Set $\mathsf{r} = H(m, b)$.
4. Output the ciphertext $\mathsf{e} = \mathsf{r} * \mathsf{h} + \mathsf{m}' \bmod q$.

Decryption – To decrypt the ciphertext e :

1. Recover m' from e using standard NTRUEncrypt decryption.
2. Recover m, b from m' by reversing the masking defined above.
3. Set $\mathsf{r} = H(m, b)$ and calculate $\mathsf{e}' = \mathsf{r} * \mathsf{h} + \mathsf{m}' \bmod q$. If the result is the same as the received e, output m. Otherwise, output "invalid ciphertext".

SVES-1 was shown in [258] to have inadequate provable IND-CPA properties, due to the decision to split b into two parts. EESS #1 therefore also specifies SVES-2, which is similar to SVES-1 with the exception that all of b is included in the first hash function call. There are other minor differences between the two schemes – SVES-2 is designed to allow variable-length messages more gracefully, for example – but these are more engineering than cryptographic decisions.

One interesting fact to note is that in both SVES-1 and SVES-2 the message is randomized by the mask generation functions, but an adversary is free to choose the value of m' directly and then reverse the masking operation to find the m and b that would have given that m'.

I.8 Appendix : Another chosen-ciphertext attack

Here we present a brief overview of a second chosen-ciphertext attack against NTRUEncrypt. The attack is based on decryption failures; however, unlike the other attack presented in this paper, does not rely on the secret polynomial f being of the form $1 + (2 + X)F$. In fact, the new attack is not specific to the case of $\mathsf{p} = 2 + X$ and can also be applied against the originally proposed version of NTRU [149] which had $\mathsf{p} \in \mathbb{Z}$.

The attack assumes that the attacker can detect wrap errors and that the r values used during encryption must be selected at random. For the basic version of the attack we also assume that a message polynomial m can be encrypted with many random r (as is the case for the proposed NTRU-REACT padding schemes). We will discuss below the effect on the attack if each m yields a unique r_m. The basic version of the attack consists of repeating the following three steps until the secret key is revealed.

Step 1 : finding a decryption failure.

The goal of step 1 is to determine a valid pair m, r which lead to a decryption failure. The most straight forward approach is to simply select random m and r until the ciphertext they generate causes a decryption failure.

Instead of a random search, it is also possible to perform a systematic search for the required decryption failure. Given an m and r an attacker can determine exactly the set, $I_{\mathsf{m},\mathsf{r}}$, of (f, g) pairs for which m, r will cause a decryption failure. Determining if m, r causes a decryption failure reveals whether or not (f, g) is in $I_{\mathsf{m},\mathsf{r}}$. So instead of simply selecting m and r at random an attacker could perform some precomputation and obtain a list of m, r pairs for which $\bigcup I_{\mathsf{m},\mathsf{r}}$ is larger than it would have been in a random search.

Step 2 : search for more r's.

For the majority of m, r pairs which lead to decryption failures, $\mathsf{m} * \mathsf{f}$ will have one coefficient, i, which is both abnormally far from its expected value and further from the expected value than any other coefficient. We shall refer to the difference in the distances of the two coefficients of $\mathsf{m} * \mathsf{f}$ furthest from their expected value as the *gap* of $\mathsf{m} * \mathsf{f}$. The true goal of step 1 is actually to find an m such that $\mathsf{m} * \mathsf{f}$ has both a coefficient which is far from its expected value and a large gap.
Attack : Step 2. By repeatedly picking random r' and determining if m, r' leads to a decryption failure, the attacker can determine a list $\mathsf{r}_0, \mathsf{r}_1, \ldots, \mathsf{r}_k$ of r values which cause decryption failures when used with m.

Suppose that step 1 found an m with the desired properties. The range $[A, A + q - 1]$ used during decryption is centered at the expected value of the coefficients of $\mathsf{p} * \mathsf{r}' * \mathsf{g} + \mathsf{f} * \mathsf{m}$. Thus, since the ith coefficient of $\mathsf{m} * \mathsf{f}$ is abnormally far from its expected value, the rate at which the m, r' cause decryption failures will be much higher than for random m, r. Furthermore, the expected value of every coefficient of $\mathsf{p} * \mathsf{r} * \mathsf{g}$ is $\mathsf{p}(1)\mathsf{g}(1)\mathsf{r}(1)/N$. Thus when an m, r' pair causes a decryption failure its most likely cause is the ith coefficient of $\mathsf{p} * \mathsf{r}' * \mathsf{g} + \mathsf{f} * \mathsf{m}$. The strength of this bias towards the i coefficient the will depend on the gap of $\mathsf{m} * \mathsf{f}$. This bias will cause a correlation between the $\mathsf{r}_0, \mathsf{r}_1, \ldots, \mathsf{r}_k$ found in step 2 and $\bar{\mathsf{g}}$.

Step 3 : recovering the secret key.

If k is sufficiently large then the value of $\bar{\mathsf{g}}$ (and thus g) can be determined directly from the polynomials $\mathsf{r}_0, \mathsf{r}_1, \ldots, \mathsf{r}_k$. However, it is possible to find the secret key with fewer r_j than would allow the direct recovery of $\bar{\mathsf{g}}$. This is accomplished by using the r_j to determine some of coefficients of g

and then using this partial knowledge of g in combination with the known lattice attacks on NTRU as in [225].

If the gap of $\mathsf{m} * \mathsf{f}$ is small then the bias towards the coefficient i may not be large enough to allow the recovery of the secret key. If this is the situation then the attack simply returns to step 1. Note that even if an iteration does not reveal the entire secret key some information may still have been determined.

Attack summary and variations.

Two important questions arise regarding this attack. First, how much work is involved in one iteration of the steps ? Second, how many iterations through the steps will be required ? The number of iterations required depends on the maximum work allowed to be done in step 2 of an iteration. The more effort put into finding r_j's in step 2 the more likely step 3 is to succeed. Details on the running time of the attack against the $\mathsf{p} = 3$ parameters suggested in [327] can be found in [290]. Below we include some details on the running time against the $N = 251$ parameter set of NTRUEncrypt as standardized in [62].

If, as with the SVES-1 padding scheme, each polynomial m only has one valid r then the basic attack described above can not be used. The problem arises in step 2, where if m is held constant then the r' used will also be constant. To overcome this problem the attacker can, instead of keeping m fixed, use the cyclic shifts of both m and m with minor changes applied to it. Care must be taken to record the shift amounts with the r_j found so that the shifts can be undone in step 3.

I.8.1 Implementation results

The attack was implemented against 100 instances of the $N = 251$ parameter set of NTRUEncrypt as standardized in [62] taking $\mathsf{f} = 1 + pF$, where F was a binary polynomial. Our implementation of the attack put a bound of 3 million on the number of r checked in step 2, checked to see if the secret key could be recover after every 25 decryption failures in step 2, and aborted iterations in step 2 if the rate at which the r_j were found was below a given threshold. The implementation assumed that the secret key could be recovered when the dimension of the lattice which would need to be reduced was less than one hundred. Of the 100 instances of the attacks the number of instances which found the secret key after $1, 2, 3, 4, 5, 6, 7$ and 8 iterations were $48, 23, 17, 4, 4, 2, 1$ and 1 respectively. Table I.4 shows the average number of m, r pairs tested and decryption failures required over the 100 instances of the attack and during the step 2's of the successful iterations.

	m, r Pairs Checked		Decryption Failures	
	Avg Number	Std Dev	Avg Number	Std Dev
Total Attack	1991909.11	1706591.03	170.65	130.56
Successful Step 2	842589.58	767601.34	118.74	43.77

TAB. I.4 – $N = 251$ attack details

Annexe J

On the Insecurity of a Server-Aided RSA Protocol

ASIACRYPT 2001

[260] avec Igor E. Shparlinski (Macquarie University)

Abstract: *At Crypto '88, Matsumoto, Kato and Imai proposed a protocol, known as RSA-S1, in which a smart card computes an RSA signature, with the help of an untrusted powerful server. There exist two kinds of attacks against such protocols : passive attacks (where the server does not deviate from the protocol) and active attacks (where the server may return false values). Pfitzmann and Waidner presented at Eurocrypt '92 a passive meet-in-the-middle attack and a few active attacks on RSA-S1. They discussed two simple countermeasures to thwart such attacks : renewing the decomposition of the RSA private exponent, and checking the signature (in which case a small public exponent must be used). We present a new lattice-based provable passive attack on RSA-S1 which recovers the factorization of the RSA modulus when a very small public exponent is used, for many choices of the parameters. The first countermeasure does not prevent this attack because the attack is a one-round attack, that is, only a single execution of the protocol is required. Interestingly, Merkle and Werchner recently provided a security proof of RSA-S1 against one-round passive attacks in some generic model, even for parameters to which our attack provably applies. Thus, our result throws doubt on the real significance of security proofs in the generic model, at least for server-aided RSA protocols. We also present a simple analysis of a multi-round lattice-based passive attack proposed last year by Merkle.*

Keywords : *Cryptanalysis, RSA signature, Server-aided protocol, Lattices.*

J.1 Introduction

Small units like chip cards or smart cards have the possibility of computing, storing and protecting data. Today, many of these cards include fast and secure coprocessors allowing to quickly perform the expensive operations needed by public key cryptosystems. However, a large proportion of the cards consists of cheap cards with too limited computing power for such tasks. To overcome this problem, extensive research has been conducted under the generic name "server-aided secret computations" (SASC). In the SASC protocol, the client (the smart card) wants to perform a secret computation (for example, RSA signature generation) by borrowing the computing power of an untrusted powerful server without revealing its secret information. One distinguishes two kinds of attacks against such protocols : attacks where the server follows rigorously the protocol are called *passive attacks*, while

attacks where the server may return false computations are called *active attacks*. Attacks are called multi-round when they require several executions of the protocol between the same parties.

Most of the SASC protocols proposed for RSA signatures have been shown to be either inefficient or insecure (see for instance the two recent examples [253, 232]), which explains why, to our knowledge, none of these protocols has ever been used in practice. Many of these protocols are variants of the protocols RSA-S1 and RSA-S2 proposed by Matsumoto, Kato and Imai [222] at Crypto '88, which use a random linear decomposition of the RSA private exponent. At Eurocrypt '92, Pfitzmann and Waidner [282] presented several natural meet-in-the-middle passive attacks and some efficient active attacks against RSA-S1 and RSA-S2. To prevent such attacks, they discussed two countermeasures which should be used together : one is to renew the decomposition of the private exponent at each signature, the other is to check the signature before the end of the protocol, which is a well-known countermeasure but requires a very small public exponent since the check is performed by the card.

The first countermeasure was effective against the original active attacks of [282], but Merkle [232] showed last year at ACM CCS '00 that the resulting scheme was still insecure. Indeed, he presented an efficient lattice-based multi-round passive attack, which was successful (in practice) against many choices of the parameters. Merkle's paper [232] included an analysis of the attack, inspired by well-known lattice-based methods [75] to solve the subset sum problem. However, the analysis was rather technical and not exactly correct (it assumed a distribution of the parameters which was not the one induced by the protocol). We present a simple analysis of a slight variant of Merkle's attack, which enables to explain experimental results, and to provide provable results for certain choices of the parameters.

The main contribution of this paper is a new lattice-based passive attack which recovers the private exponent (like Merkle's attack), but only in the case a very small public exponent is used (which is the second countermeasure). Interestingly, this attack is only one-round in the sense that a single execution of the protocol is sufficient, whereas Merkle's attack is multi-round, requiring many signatures produced by the card with the help of the same server. Consequently, the first countermeasure has no impact on this new attack. And these results point out the limits of the *generic model*, as applied to the security analysis of server-aided RSA protocols. Indeed, Merkle and Werchner [233] proved at PKC '98 that the RSA-S1 protocol was secure against one-round passive attacks in the generic model, in the sense that all generic attacks have complexity at least that of a square-root attack (better than the meet-in-the-middle attack presented by Pfitzmann and Waidner [282]). Roughly speaking, in this context, generic attacks (see [233] for a precise definition) do not take advantage of special properties of the group used. However, our attack shows that the RSA-S1 scheme is not even secure against one-round passive attacks in the standard model of computation. In particular, the attack provably works against certain choices of the parameters to which the square-root attack cannot apply. Thus, contrary to what Merkle and Werchner claimed in [233], the generic model is not appropriate for investigating the security of server-aided RSA protocols.

The rest of the paper is organized as follows. In Section J.2, we make a short description of the RSA-S1 server-aided protocol and review some useful background. We refer to [222, 282] for more details. In Section J.3, we present our variant of Merkle's lattice-based attack, together with an analysis. In Section J.4, we present our new lattice-based attack on low-exponent RSA-S1.

J.2 Background

J.2.1 The RSA-S1 Server-Aided Protocol

Let N be an RSA-modulus and let φ denote the Euler function. Let e and d be respectively the RSA public and private exponents :

$$ed \equiv 1 \pmod{\varphi(N)}.$$

For an integer s we denote by $[s]$ the set of integers of the interval $[0, s-1]$ and by $[s]_\pm$ the set of integers of the interval $[-s+1, s-1]$.

Let k, ℓ and m be positive integers and let $\mathcal{B}_{k,\ell,m}$ be the set of vectors

$$\mathbf{f} = (f_1, \ldots, f_m) \in \left[2^\ell\right]^m$$

with $\gcd(f_1, \ldots, f_m, \varphi(N)) = 1$ and with

$$\sum_{i=1}^{m} \mathrm{wt}(f_i) = k, \tag{J.1}$$

where $\mathrm{wt}(f)$ denotes the Hamming weight, that is, the sum of binary digits of an integer $f \geq 0$.

The RSA-S1 server-aided protocol from [222] computes an RSA signature $x^d \pmod N$ with the help of an (untrusted) server in the following way :

The RSA-S1 Protocol.

Step 1 The card selects a vector $\mathbf{f} = (f_1, \ldots, f_m) \in \mathcal{B}_{k,\ell,m}$ at random accordingly to any fixed probability distribution.

Step 2 The card sends a vector $\mathbf{d} = (d_1, \ldots, d_m) \in [\varphi(N)]^m$ chosen uniformly at random from the set of vectors satisfying the congruence

$$\sum_{i=1}^{m} f_i d_i \equiv d \pmod{\varphi(N)}, \tag{J.2}$$

if possible. Otherwise the card returns to Step 1.

Step 3 The card asks the server to compute and return $z_i \equiv x^{d_i} \pmod N$, $i = 1, \ldots, m$.

Step 4 The card computes

$$x^d \equiv \prod_{i=1}^{m} z_i^{f_i} \pmod N.$$

Our description follows the presentation of [232] rather than the one of the original paper [222]. For instance, [222] asks that $\sum_{i=1}^{m} \mathrm{wt}(f_i) \leq k$ instead of (J.1) but this difference is marginal as all our results can easily be adapted to this case.

For Step 4, the card mainly has two possibilities, due to memory restrictions. One is the square-and-multiply method, which requires at most $k\ell$ modular multiplications and very little memory. The other is the algorithm of [51], which enables to compute $\prod_{i=1}^{m} z_i^{f_i} \pmod N$ efficiently but requires more memory than the square-and-multiply method. When using this algorithm, to optimize the choice of the parameters, one should remove the restriction (J.1) and replace the choice $f_i \in \left[2^\ell\right]$ by $f_i \in [h]$ where h is some small integer, not necessarily a power of 2. The algorithm then requires at most $m + h - 3$ modular multiplications, and the temporary storage of either m or $h-1$ elements, according to whether the card stores all the m elements $z_1, \ldots, z_m$, or the $h-1$ elements $t_j = \prod_{f_i = j} z_i$, $1 \leq j < h$ (which must be computed upon reception of the z_i's). Other known tricks to speed-up the computation of products of exponentiations (see [126] and [231, Sect. 14.6]) do not seem to be useful in this context.

The protocol requires the transfer of approximately $2m \log N$ bits. Since the bandwidth of a cheap smartcard is typically 9600 bauds, this means that m must be restricted to low values. For instance, with a 1024-bit modulus, the value $m = 50$ already represents 10.7 seconds.

J.2.2 Passive attacks on RSA-S1

Notice that the protocol is broken as soon as the f_i's are disclosed. Indeed, the integer $\sum_{i=1}^{m} f_i d_i$ is congruent to the RSA private exponent modulo $\varphi(N)$, and therefore enables to sign any message (and this can be checked thanks to the public exponent e). And, of course, one may further recover the factorization of N in randomized polynomial time, from $e\sum_{i=1}^{m} f_i d_i - 1$ which is a non-zero multiple of $\varphi(N)$ (see for instance [231, Section 8.2.2]).

The authors of [222] claimed that the only possible passive attack was to exhaustive search the f_i's, which requires roughly C operations where :

$$C = \left(\begin{array}{c} m\ell \\ k \end{array} \right).$$

But obviously, one can devise simple meet-in-the-middle passive attacks. Pfitzmann and Waidner [282] noticed that one could split $(f_1, \ldots, f_m)$ as $(g_1, \ldots, g_m) + (h_1, \ldots, h_m)$ where $\sum \mathrm{wt}(g_i) \leq \sum \mathrm{wt}(h_i) = \lceil k/2 \rceil$, and deduced an attack with time and space complexity roughly :

$$\left(\begin{array}{c} m\ell \\ \lceil k/2 \rceil \end{array} \right).$$

The attack of [282] is however not optimal : the complexity can easily be improved using a trick used by Coppersmith [334] in a meet-in-the-middle attack against the discrete logarithm problem with low Hamming weight. By choosing random subsets of cardinality $\lceil m\ell/2 \rceil$ inside $\{1, \ldots, m\ell\}$, one obtains a randomized meet-in-middle-attack with time and space complexity roughly :

$$\sqrt{k} \left(\begin{array}{c} \lceil m\ell/2 \rceil \\ \lceil k/2 \rceil \end{array} \right).$$

Thus, we obtain an attack of complexity roughly the square root $\sqrt{C}$ of that of exhaustive search. Therefore in our numerical experiments we mainly consider sets of parameters for which $C \geq 2^{120}$. Note however that even with $C \approx 2^{100}$, the square-root attack is not much practical, due to memory constraints.

In [233], Merkle and Werchner proposed an adaptation of generic algorithms (see [320]) to server-aided RSA protocols, and showed that any one-round passive generic attack on RSA-S1 had complexity at least $\Omega(\sqrt{C})$.

In [282], Pfitzmann and Waidner also presented a few active attacks which cannot be avoided by increasing the parameters contrary to the passive attacks mentioned previously. They discussed two countermeasures to prevent their own active attacks :

- Renewing the decomposition of the private exponent d at each execution of the protocol, as described in Steps 1 and 2.
- Verifying the signature $x^d \pmod N$ before releasing it, by computing $(x^d)^e \pmod N$ and checking that it is equal to x. This countermeasure is well-known and requires a very small public exponent e (otherwise there is no computational advantage in using the server to compute $x^d \pmod N$).

The second countermeasure seems necessary but is not sufficient to prevent one of the active attacks of [282], and it creates the attack of Section J.4. The first countermeasure prevents all the active attacks of [282], but creates the passive attack of Merkle [232], which we analyze in Section J.3. Interestingly, it seems that the attacks of Section J.3 and J.4 do not apply to the RSA-S2 protocol, which is a CRT variant of RSA-S1 (see [222, 282]). The situation is reminiscent of that of RSA with small private exponent, in which the best attack known [43] fails if the private exponent is small modulo both $p - 1$ and $q - 1$.

J.2.3 Lattices

Our attacks are based on lattice basis reduction, a familiar tool in public-key cryptanalysis. We give a brief overview of lattice theory (see the survey [267] for a list of references). In this paper, we call a *lattice* any subgroup of $(\mathbb{Z}^n, +)$: in the literature, these are called integer lattices. For any set of vectors $\mathbf{b}_1, \ldots, \mathbf{b}_d \in \mathbb{Z}^n$, we define the set of all integral linear combinations :

$$L(\mathbf{b}_1, \ldots, \mathbf{b}_d) = \left\{ \sum_{i=1}^{d} n_i \mathbf{b}_i : n_i \in \mathbb{Z} \right\}.$$

By definition, $L(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ is a lattice, called the lattice spanned by the vectors $\mathbf{b}_1, \ldots, \mathbf{b}_d$. A *basis* of a lattice L is a set of linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_d$ such that :

$$L = L(\mathbf{b}_1, \ldots, \mathbf{b}_d).$$

In any lattice, there is always at least one basis, and in general, there are in fact infinitely many lattice bases. But all the bases of a lattice L have the same number of elements, called the *rank* or *dimension* of the lattice. All the bases also have the same d-dimensional volume, which is by definition the square root of the determinant $\det_{1 \leq i,j \leq d} \langle \mathbf{b}_i, \mathbf{b}_j \rangle$, where $\langle , \rangle$ denotes the Euclidean inner product. This volume $\mathrm{vol}(L)$ is called the volume or determinant of the lattice. When the lattice dimension d is equal to the space dimension n, this volume is simply the absolute value of the determinant of any lattice basis.

For a vector $\mathbf{a}$, we denote by $\|\mathbf{a}\|$ its Euclidean norm. A basic problem in lattice theory is the shortest vector problem (SVP) : given a basis of a lattice L, find a non-zero vector $\mathbf{v} \in L$ such that $\|\mathbf{v}\|$ is minimal among all non-zero lattice vectors. Any such vector is called a shortest lattice vector. It is well-known that the Euclidean norm of a shortest lattice vector is always less than $\sqrt{d}\mathrm{vol}(L)^{1/d}$, d denoting the lattice dimension. In "usual" lattices, one does not expect the norm of a shortest lattice vector to be much less than this upper bound.

Many attacks in public-key cryptanalysis work by reduction to SVP, or to approximating SVP (see the survey [267]). The shortest vector problem was recently shown to be NP-hard under randomized reductions [5], and therefore, it is now widely believed that there is no polynomial-time algorithm to solve SVP. However, there exist polynomial-time algorithms which can provably approximate SVP. The first algorithm of that kind was the celebrated LLL lattice basis reduction algorithm of Lenstra, Lenstra and Lovász [205]. We use the best deterministic polynomial-time algorithm currently known to approximate SVP, which is due to Schnorr [302] and is based on LLL :

Lemma J.5 *There exists a deterministic polynomial time algorithm which, given as input a basis of an s-dimensional lattice L, outputs a non-zero lattice vector $\mathbf{u} \in L$ such that :*

$$\|\mathbf{u}\| \leq 2^{O\left(s \log^2 \log s / \log s\right)} \min \{\|\mathbf{z}\| : \quad \mathbf{z} \in L, \mathbf{z} \neq 0\}.$$

Recently, Ajtai *et al.* [8] discovered a randomized algorithm which slightly improves the approximation factor $2^{O\left(s \log^2 \log s / \log s\right)}$ to $2^{O(s \log \log s / \log s)}$. In practice, the best algorithm to approximate SVP is a heuristic variant of Schnorr's algorithm [302]. Interestingly, these algorithms typically perform much better than theoretically expected : they often return a shortest lattice vector, provided that the lattice dimension is not too large. Hence, it is useful to predict what can be achieved efficiently if an SVP-oracle (that is, an algorithm which solves SVP) is available. For instance, this was done for the subset sum problem [75]. However, unless the lattice dimension is extremely small, it is hard to predict beforehand whether an SVP-instance is solvable in practice, which means that experiments are always necessary in this case.

J.3 An Analysis of Merkle's Multi-round Attack

J.3.1 Merkle's Attack

The attack of Merkle [232] is based on the following observation : Because for each $\mathbf{f} = (f_1, \ldots, f_m) \in \mathcal{B}_{k,\ell,m}$ and $\mathbf{d} = (d_1, \ldots, d_m) \in [\varphi(N)]^m$

$$0 < \sum_{i=1}^{m} f_i d_i < k2^{\ell}\varphi(N)$$

we have

$$\sum_{i=1}^{m} f_i d_i \equiv d + j\varphi(N)$$

with $j \in \left[k2^{\ell}\right]$, that is, j cannot take too many distinct values.

It is shown in [232] that regardless of the distribution of the vectors $\mathbf{f} \in \mathcal{B}_{k,\ell,m}$ with probability at least $1/k2^{\ell}$ for two pairs $\mathbf{f}_1 = (f_1, \ldots, f_m)$, $\mathbf{d}_1 = (d_1, \ldots, d_m)$, and $\mathbf{f}_2 = (f_{m+1}, \ldots, f_{2m})$, $\mathbf{d}_2 = (d_{m+1}, \ldots, d_{2m})$ of vectors produced by the above protocol we have the following equation (over the integers rather than modulo N) :

$$\sum_{i=1}^{m} f_i d_i = \sum_{i=m+1}^{2m} f_i d_i. \tag{J.3}$$

In fact, any rule to select the above vectors gives rise to a collision after at most $k2^{\ell}$ executions of the protocol. Besides, the "birthday paradox" suggests that a collision is likely to happen after roughly $k^{1/2}2^{\ell/2}$ executions of the protocol.

The linear equation (J.3) is unusual because each f_i is small (compared to the d_i's), and this can be interpreted in terms of lattices. More precisely, it is argued in [232] that $(\mathbf{f}_1, \mathbf{f}_2)$ is the shortest vector in a particular lattice related to the homogeneous equation (J.3) and the congruences

$$\sum_{i=1}^{m} f_i d_i \equiv \sum_{i=m+1}^{2m} f_i d_i \equiv d \pmod{\varphi(N)}. \tag{J.4}$$

However, the analysis presented by Merkle is not sufficient, because it assumes a distribution of the parameters which is not the one of the protocol (see [232, Theorem 2.1]). And no result is proposed without SVP-oracles. Hence, Merkle's attack, as presented in [232], is not a proved attack, even under the assumption of an SVP-oracle, which is not so unusual for a lattice-based attack. Nevertheless, the experiments conducted by Merkle (see [232]) showed that the attack was successful in practice against many choices of the parameters. Thus, it was interesting to see whether Merkle's attack could be proved, with or without SVP-oracles. Here, we provide a proof, for a slight variant of Merkle's attack. The analysis we present can in fact be extended to the original attack, but our variant is slightly simpler to describe and to analyze, while the difference of efficiency between the two attacks is marginal.

J.3.2 A Variant of Merkle's Attack

We work directly with the lattice corresponding to (J.3) : Let $\mathcal{L}(\mathbf{d}_1, \mathbf{d}_2)$ be the $(2m-1)$-dimensional lattice formed by all vectors $\mathbf{z} \in \mathbb{Z}^{2m}$ with

$$\sum_{i=1}^{m} z_i d_i = \sum_{i=m+1}^{2m} z_i d_i.$$

This lattice is the simplest case of an orthogonal lattice (as introduced in [251]), and one can compute a basis of such lattices in polynomial time. It can easily be showed that the volume of the lattice is given by :

$$\mathrm{vol}(\mathcal{L}(\mathbf{d}_1, \mathbf{d}_2)) = \frac{\left(d_1^2 + \ldots + d_{2m}^2\right)^{1/2}}{\gcd(d_1, \ldots, d_{2m})}.$$

Thus, one would expect its shortest non-zero vector to have a norm around :

$$(2m-1)^{1/2}\mathrm{vol}\left(\mathcal{L}(\mathbf{d}_1, \mathbf{d}_2)\right)^{1/(2m-1)} \approx (2m-1)^{1/2}\varphi(N)^{1/(2m-1)}.$$

On the other hand, the vector $\mathbf{f} = (f_1, \ldots, f_{2m})$ belongs to this lattice, and has a norm of at most $k^{1/2}2^\ell$. Hence, if $k^{1/2}2^\ell$ is much smaller than $(2m-1)^{1/2}\varphi(N)^{1/(2m-1)}$, we expect $\mathbf{f}$ to be the shortest vector of $\mathcal{L}(\mathbf{d}_1, \mathbf{d}_2)$, and if it is smaller enough, then the gap between $\mathbf{f}$ and the other lattice vectors guarantees that the algorithm of Lemma J.5 will find it. Once $\mathbf{f}$ is known, one can derive the value $\sum_{i=1}^m f_i d_i$, which is congruent to the RSA private exponent modulo $\varphi(N)$, and therefore enables to sign any message. And one may further recover the factorization of N in randomized polynomial time, from $e\sum_{i=1}^m f_i d_i - 1$ which is a non-zero multiple of $\varphi(N)$ (see for instance [231, Section 8.2.2]).

In [232], the original attack of Merkle worked with a slight variant of the lattice $\mathcal{L}(\mathbf{d}_1, \mathbf{d}_2)$, to take advantage of the fact that $f_i \in \left[2^\ell\right]$ and not $f_i \in \left[2^\ell\right]_\pm$. Such a trick was used for the subset sum problem [75]. However, this trick is not as useful here, because the distributions are different. This means that the difference between our variant and the original attack is marginal.

J.3.3 Theoretical results

The previous reasoning can in fact be made rigorous by a tight analysis, which gives rise to the following result :

Theorem J.3 *There is a deterministic algorithm $\mathcal{A}$ which, given as input an RSA modulus N, together with a public exponent e, and a set $\mathcal{D}$ of $k2^\ell$ vectors $\mathbf{d} \in [\varphi(N)]^m$ corresponding to a certain set $\mathcal{F}$ of vectors $\mathbf{f} \in \mathcal{B}_{k,\ell,m}$ generated by $k2^\ell$ independent executions of* RSA-S1*, outputs a value $\mathcal{A}(\mathcal{D})$ in time polynomial in $k, 2^\ell, m, \log N$ such that :*

$$\Pr_{\mathcal{D}}\left[\mathcal{A}(\mathcal{D}) \equiv d \pmod{\varphi(N)}\right] \geq 1 - \frac{k^{m+2}2^{2\ell(m+2)+O(m^2\log^2\log m/\log m)}}{\varphi(N)}$$

where the probability is taken over all random choices of $\mathcal{D}$ for the given $\mathcal{F}$.

Proof. Given a set $\mathcal{D}$ of $k2^\ell$ vectors $\mathbf{d}$ associated with the protocol RSA-S1, which corresponds to a certain set $\mathcal{F}$ of $k2^\ell$ (unknown) vectors $\mathbf{f} \in \mathcal{B}_{k,\ell,m}$, the algorithm $\mathcal{A}$ selects all possible pairs of such vectors $\mathbf{d}_1$ and $\mathbf{d}_2$ and uses the algorithm of Lemma J.5 to find a short vector $\mathbf{u}$ in the $(2m-1)$-dimensional lattice $\mathcal{L}(\mathbf{d}_1, \mathbf{d}_2)$ formed by all vectors $\mathbf{z} \in \mathbb{Z}^{2m}$ such that

$$\sum_{i=1}^{m} z_i d_i = \sum_{i=m+1}^{2m} z_i d_i.$$

We know that there is at least one pair $(\mathbf{d}_1, \mathbf{d}_2)$ such that the equation (J.3) holds. Notice that for any $\mathbf{f} \in \mathcal{B}_{k,\ell,m}$, we have

$$\|f\|^2 = \sum_{i=1}^{m} f_i^2 < 2^\ell \sum_{i=1}^{m} f_i \leq k2^{2\ell}. \tag{J.5}$$

Thus, if we apply the algorithm of Lemma J.5 to $\mathcal{L}(\mathbf{d}_1, \mathbf{d}_2)$, we obtain a vector $\mathbf{u} = (u_1, \ldots, u_{2m})$ such that :

$$\begin{aligned} \|\mathbf{u}\|^2 &\leq 2^{O(m \log^2 \log m / \log m)} \min\left\{\|\mathbf{z}\|^2, \quad \mathbf{z} \in \mathcal{L}(\mathbf{d}_1, \mathbf{d}_2)\right\} \\ &\leq 2^{O(m \log^2 \log m / \log m)} \left(\|\mathbf{f}_1\|^2 + \|\mathbf{f}_2\|^2\right) \\ &\leq k 2^{2\ell + O(m \log^2 \log m / \log m)}. \end{aligned}$$

Therefore, there exists some integer $U = k^{1/2} 2^{\ell + O(m \log^2 \log m / \log m)}$ such that $|u_i| < U$ for $i = 1, \ldots, 2m$, that is, $\mathbf{u} \in [U]_{\pm}^{2m}$.

We write $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2)$ where $\mathbf{u}_1, \mathbf{u}_2 \in [U]_{\pm}^{m}$ and say that $\mathbf{u}$ is *similar* to the concatenation $(\mathbf{f}_1, \mathbf{f}_2)$ if either $\mathbf{u}_1$ is non-zero and parallel to $\mathbf{f}_1$, or $\mathbf{u}_2$ is non-zero and parallel to $\mathbf{f}_2$. Notice that if one knows a vector $\mathbf{u} \neq 0$ similar to $\mathbf{f}_1, \mathbf{f}_2$, one obtains at most 2^ℓ possible values for either $\mathbf{f}_1$ or $\mathbf{f}_2$. And if $\mathbf{f}_1$ or $\mathbf{f}_2$ is correct, then $\langle \mathbf{f}_1, \mathbf{d}_1 \rangle$ or $\langle \mathbf{f}_2, \mathbf{d}_2 \rangle$ is congruent to d modulo $\varphi(N)$, which can be checked by signing a message. Hence it is enough to show that with probability at least $1 - k^{m+2} 2^{2\ell(m+2) + O(m^2 \log^2 \log m / \log m)} \varphi(N)^{-1}$ the vector $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2)$ returned by the algorithm of Lemma J.5 is similar to $(\mathbf{f}_1, \mathbf{f}_2)$.

First for $\mathbf{f}_1, \mathbf{f}_2 \in \mathcal{B}_{k,\ell,m}$ we estimate the size of the set $\mathcal{E}(\mathbf{f}_1, \mathbf{f}_2)$ of pairs of vectors $\mathbf{d}_1, \mathbf{d}_2 \in [\varphi(N)]^m$ such that for some $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2) \in [U]_{\pm}^{2m}$ which is not similar to $(\mathbf{f}_1, \mathbf{f}_2)$ we have the equation

$$\sum_{i=1}^{m} u_i d_i = \sum_{i=m+1}^{2m} u_i d_i. \tag{J.6}$$

Let us fix a nonzero vector $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2) \in [U]_{\pm}^{2m}$ and a vector $(\mathbf{f}_1, \mathbf{f}_2) \in \mathcal{B}_{k,\ell,m}$ which are not similar. Without loss of generality we may assume that $\mathbf{u}_2 \neq 0$ and is not parallel to $\mathbf{f}_2$ and that $f_{2m} \neq 0$. Then excluding d_{2m} from (J.6) using (J.3), we obtain an equation

$$\sum_{i=1}^{m} c_i d_i = \sum_{i=m+1}^{2m-1} c_i d_i \tag{J.7}$$

with $c_i = u_i - f_i u_{2m} / f_{2m}$, $i = 1, \ldots, 2m-1$. By our assumption, for at least one $i \geq m+1$, the coefficient $c_i \neq 0$. Without loss of generality we may assume that $c_{2m-1} \neq 0$. Then the first congruence in (J.4) gives us at most $2^\ell \varphi(N)^{m-1}$ possible values for $\mathbf{d}_1 = (d_1, \ldots, d_m)$. Indeed, assuming that $f_m \neq 0$ and selecting the integers $d_1, \ldots, d_{m-1} \in [\varphi(N)]$ arbitrarily, we obtain a congruence of the form $f_m d_m \equiv D \pmod{\varphi(N)}$ which has at most $\gcd(f_m, \varphi(N)) \leq f_m < 2^\ell$ solutions $d_m \in [\varphi(N)]$. Finally, for any of $\varphi(N)^{m-2}$ possible choices of $d_{m+1}, \ldots, d_{2m-2} \in [\varphi(N)]^{m-2}$ the equation (J.7) gives at most one value for d_{m-1} and then the second congruence in (J.4) gives us at most $\gcd(f_{2m}, \varphi(N)) \leq f_{2m} < 2^\ell$ possible values for d_{2m}. So the total number of solutions for such $\mathbf{u}$ is at most $2^{2\ell} \varphi(N)^{2m-3}$. The total number of such vectors is at most U^{2m}. Thus we finally derive

$$\begin{aligned} \#\mathcal{E}(\mathbf{f}_1, \mathbf{f}_2) &\leq (2U)^{2m} 2^{2\ell} \varphi(N)^{2m-3} \\ &\leq k^m 2^{2\ell(m+1) + O(m^2 \log^2 \log m / \log m)} \varphi(N)^{2m-3}. \end{aligned}$$

For each vector $\mathbf{f} \in \mathcal{B}_{k,\ell,m}$ there are exactly $\varphi(N)^{m-1}$ vectors $\mathbf{d} \in [\varphi(N)]^m$ satisfying the congruence (J.2). Therefore, the probability that there is a pair of vectors $\mathbf{f}_1, \mathbf{f}_2 \in \mathcal{F}$ such that the corresponding vectors $\mathbf{d}_1, \mathbf{d}_2 \in \mathcal{D}$ satisfy $\mathbf{d}_1, \mathbf{d}_2 \in \mathcal{E}(\mathbf{f}_1, \mathbf{f}_2)$ is at most

$$\begin{aligned} &\frac{(\#\mathcal{F})^2 k^m 2^{2\ell(m+1) + O(m^2 \log^2 \log m / \log m)} \varphi(N)^{2m-3}}{\varphi(N)^{2m-2}} \\ &\qquad = k^{m+2} 2^{2\ell(m+2) + O(m^2 \log^2 \log m / \log m)} \varphi(N)^{-1}, \end{aligned}$$

and the result follows. □

□

Assuming that an SVP-oracle is available, we derive much stronger estimates.

Theorem J.4 *There is a deterministic algorithm $\mathcal{A}$ which, given an access to an SVP-oracle and as input an RSA modulus N, together with a public exponent e, a set $\mathcal{D}$ of $k2^\ell$ vectors $\mathbf{d} \in [\varphi(N)]^m$ corresponding to a certain set $\mathcal{F}$ of vectors $\mathbf{f} \in \mathcal{B}_{k,\ell,m}$ generated by $k2^\ell$ independent executions of* RSA-S1*, outputs a value $\mathcal{A}(\mathcal{D})$ in time polynomial in $k, 2^\ell, m, \log N$ such that :*

$$\Pr_{\mathcal{D}}[\mathcal{A}(\mathcal{D}) \equiv d \pmod{\varphi(N)}] \geq 1 - \frac{k^{m+2}2^{2(\ell m+2\ell+m)}}{\varphi(N)}$$

where the probability is taken over all random choices of $\mathcal{D}$ for the given $\mathcal{F}$.

As in [232], instead of waiting for $k2^\ell$ executions of RSA-S1 one may also restrict to only two executions, which yields the following version of Theorems J.3 and J.4 :

Theorem J.5 *There is a deterministic algorithm $\mathcal{A}$ which, given as input an RSA modulus N, together with a public exponent e, a pair of vectors $\mathbf{d}_1, \mathbf{d}_2 \in [\varphi(N)]^m$ corresponding to a pair of vectors $\mathbf{f}_1, \mathbf{f}_2 \in \mathcal{B}_{k,\ell,m}$ generated by two independent executions of* RSA-S1*, outputs a value $\mathcal{A}(\mathbf{d}_1, \mathbf{d}_2)$ in time polynomial in $k, 2^\ell, m, \log N$ such that :*

$$\Pr_{\mathbf{d}_1,\mathbf{d}_2}[\mathcal{A}(\mathbf{d}_1, \mathbf{d}_2) \equiv d \pmod{\varphi(N)}] \geq \frac{1}{k2^\ell} - \frac{k^m 2^{2\ell(m+1)+O(m^2 \log^2 \log m/\log m)}}{\varphi(N)}$$

where the probability is taken over all random choices of $\mathbf{d}_1, \mathbf{d}_2$ for the given $\mathbf{f}_1, \mathbf{f}_2$.

Theorem J.6 *There is a deterministic algorithm $\mathcal{A}$ which, given access to an SVP-oracle and as input an RSA modulus N, together with a public exponent e, a pair of vectors $\mathbf{d}_1, \mathbf{d}_2 \in [\varphi(N)]^m$ corresponding to a pair of vectors $\mathbf{f}_1, \mathbf{f}_2 \in \mathcal{B}_{k,\ell,m}$ generated by two independent executions of* RSA-S1*, makes a single call to the SVP-oracle with the lattice $\mathcal{L}(\mathbf{d}_1, \mathbf{d}_2)$ and outputs a value $\mathcal{A}(\mathbf{d}_1, \mathbf{d}_2)$ in time polynomial in $k, 2^\ell, m, \log N$ such that :*

$$\Pr_{\mathbf{d}_1,\mathbf{d}_2}[\mathcal{A}(\mathbf{d}_1, \mathbf{d}_2) \equiv d \pmod{\varphi(N)}] \geq \frac{1}{k2^\ell} - \frac{k^m 2^{2(\ell m+\ell+m)}}{\varphi(N)}$$

where the probability is taken over all random choices of $\mathbf{d}_1, \mathbf{d}_2$ for the given $\mathbf{f}_1, \mathbf{f}_2$.

Notice that unless k (and thus $\ell \geq k/m$) is exponentially large compared to m, which is completely impractical, the terms k^{m+2} and k^m in the bounds of Theorems J.3 and J.5 respectively, can be included in the term $2^{O(m^2 \log^2 \log m/\log m)}$.

J.3.4 Experiments

In practice, the attack is as efficient as Merkle's original attack, due to the fact that strong lattice basis reduction algorithms behave like oracles for the shortest vector problem up to moderate dimension. In [232], Merkle reported the experimental results presented in Table J.1. Notice however that none of the sets of parameters of Table J.1 leads to an efficient protocol (for the card).

Tab. J.1 – Experiments with Merkle's attack

m	k	ℓ	Success (%)	Complexity of the sqrt attack
25	28	11	100	2^{62}
32	26	10	100	2^{62}
38	26	9	100	2^{63}
42	26	8	100	2^{63}
48	26	7	70	2^{63}
56	26	6	10	2^{63}

J.4 A New One-Round Attack on Low Exponent RSA-S1

J.4.1 Description of the Attack

We now assume that a very small public exponent e is used. We also assume that the secret primes p and q defining $N = pq$ have approximately the same length. Let $s = p + q = O(N^{1/2})$. We have $\varphi(N) = N - s + 1$. When the RSA-S1 protocol is performed once, we have :

$$\sum_{i=1}^{m} f_i d_i \equiv d \pmod{\varphi(N)},$$

and therefore,

$$\sum_{i=1}^{m} f_i e d_i \equiv 1 \pmod{\varphi(N)}.$$

From (J.5) we see that there exists $r \in \left[k2^{\ell}e\right]$ such that

$$\sum_{i=1}^{m} f_i e d_i = 1 + r\varphi(N) = 1 + r(N - s + 1).$$

Hence

$$\sum_{i=1}^{m} f_i e d_i = 1 + r - rs \pmod{N}, \tag{J.8}$$

where $|1 + r - rs| = O(k2^{\ell}eN^{1/2})$. We thus obtain a linear equation modulo N where the unknown coefficients f_i and $1 + r - rs$ are all relatively small. This suggests to define the $(m+1)$-dimensional lattice $\mathcal{L}_{e,N}(\mathbf{d})$ spanned by the rows of the following matrix :

$$\begin{pmatrix} N & 0 & 0 & \dots & 0 \\ ed_1 & eR & 0 & \dots & 0 \\ ed_2 & 0 & eR & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ ed_m & 0 & \dots & 0 & eR \end{pmatrix}$$

where $R = \left\lfloor N^{1/2} \right\rfloor$. Obviously, the volume of this lattice is $\mathrm{vol}(\mathcal{L}_{e,N}(\mathbf{d})) = e^m N R^{m/2}$. Therefore, one would expect its shortest vector to be of norm roughly $(m+1)^{1/2} e^{m/(m+1)} N^{(m+2)/(2m+2)}$. On the other hand, the lattice contains the target vector

$$\mathbf{t} = (1 + r - rs, f_1 eR, \dots, f_m eR),$$

whose norm is $\|\mathbf{t}\| = O\left(k2^{\ell}eN^{1/2}\right)$ because of (J.5). Hence, the target vector is likely to be the shortest vector in this lattice if $ke^{1/(m+1)}2^{\ell}$ is much smaller than $m^{1/2}N^{1/(2m+2)}$. Note that this condition is satisfied for sufficiently large N and that it is very similar to the heuristic condition we obtained in Section J.3.2, which suggests that the efficiency of the attacks of Section J.4 and J.3 should be comparable. In case the target vector is really much smaller than the other lattice vectors, then the algorithm of Lemma J.5 finds it. Once the target vector is known, we can recover a private exponent equivalent to d thanks to $\sum_{i=1}^{m} f_i d_i$, which enables to sign any message, as in Merkle's attack. Again, one may further derive a not too large multiple of $\varphi(N)$, which yields the factorization of N in randomized polynomial time.

J.4.2 Theoretical results

The previous attack can be proved, using the same counting arguments of the proof of Theorem J.3 :

Theorem J.7 *There is a deterministic algorithm $\mathcal{A}$ which, given as input an RSA modulus $N = pq$ such that $p+q = O(N^{1/2})$, together with a public exponent e, and a vector $\mathbf{d} \in [\varphi(N)]^m$ corresponding to a certain vector $\mathbf{f} \in \mathcal{B}_{k,\ell,m}$ generated by* RSA-S1*, outputs a value $\mathcal{A}(\mathbf{d})$ in time polynomial in $k, 2^{\ell}, m, \log N$ such that :*

$$\Pr_{\mathbf{d}}\left[\mathcal{A}(\mathbf{d}) \equiv d \pmod{\varphi(N)}\right] \geq 1 - \frac{k^{m+1}e^{m+1}2^{\ell(m+1)+O(m^2 \log^2 \log m/\log m)}}{N^{1/2}}$$

where the probability is taken over all random choices of $\mathbf{d}$ for the given $\mathbf{f}$.

Proof. The algorithm $\mathcal{A}$ starts by applying the algorithm of Lemma J.5 to find a short vector $\mathbf{w} \neq 0$ in the $(m+1)$-dimensional lattice $\mathcal{L}_{e,N}(\mathbf{d})$. Since $\mathbf{t}$ is a lattice vector and because $p + q = O(N^{1/2})$, we have :

$$\|\mathbf{w}\| \leq 2^{O(m \log^2 \log m/\log m)}\|\mathbf{t}\| = ke2^{\ell+O(m \log^2 \log m/\log m)}N^{1/2}.$$

By definition of the lattice, $\mathbf{w}$ is of the form :

$$\mathbf{w} = (u_0 N + \sum_{i=1}^{m} ed_i u_i, u_1 eR, \ldots, u_m eR),$$

where each u_i is an integer.

Therefore, there exists some integer $U = ke2^{\ell+O(m \log^2 \log m/\log m)}$ such that $|u_i| < U$ for $i = 1, \ldots, 2m$. Thus $\mathbf{u} = (u_1, \ldots, u_m) \in [U]_{\pm}^m$. We may assume that $\|\mathbf{w}\| < N$ otherwise the right hand side of the inequality of the theorem is negative, making the bound trivial. Then necessarily $\mathbf{u} \neq 0$. We also have

$$\sum_{i=1}^{m} ed_i u_i \equiv w_0 \pmod{N} \tag{J.9}$$

for some $w_0 \in [W]_{\pm}$ where $W = O\left(ke2^{\ell+O(m \log^2 \log m/\log m)}N^{1/2}\right)$.

Clearly, we may assume that $2^{\ell} \leq \min\{p, q\}$ otherwise the result is trivial. Thus for any $i = 1, \ldots, m$ with $f_i \neq 0$ we have $\gcd(f_i, N) = 1$. As before we see that for each w_0 and for each $\mathbf{u} \in [U]_{\pm}^m$ not parallel to $\mathbf{f}$ there are at most $\varphi(N)^{m-2}$ vectors $\mathbf{d} \in [\varphi(N)]^m$ satisfying both (J.8) and (J.9). Therefore the total number of vectors $\mathbf{d} \in [\varphi(N)]^m$ which satisfy (J.8) and at least one congruence (J.9), for some $w_0 \in [W]_{\pm}$ and some nonzero vector $\mathbf{u} \in [U]_{\pm}^m$ not parallel to $\mathbf{f}$, is at most

$$2^{m+1}WU^m\varphi(N)^{m-2} = k^{m+1}e^{m+1}2^{\ell(m+1)+O(m^2 \log^2 \log m/\log m)}N^{1/2}\varphi(N)^{m-2}.$$

Taking into account that $\varphi(N) \geq N/2$ we obtain the desired result. □

□

Of course, the same proof provides a stronger result if an SVP-oracle is available :

Theorem J.8 *There is a deterministic algorithm $\mathcal{A}$ which, given access to an SVP-oracle and as input an RSA modulus $N = pq$ such that $p + q = O(N^{1/2})$, together with a public exponent e, vector $\mathbf{d} \in [\varphi(N)]^m$ corresponding to a certain vector $\mathbf{f} \in \mathcal{B}_{k,\ell,m}$ generated by* RSA-S1*, makes a single call to the SVP-oracle with the lattice $\mathcal{L}_{e,N}(\mathbf{d})$ and outputs a value $\mathcal{A}(\mathbf{d})$ in time polynomial in $k, 2^\ell, m, \log N$ such that :*

$$\Pr_{\mathbf{d}}[\mathcal{A}(\mathbf{d}) \equiv d \pmod{\varphi(N)}] \geq 1 - \frac{k^{m+1} e^{m+1} 2^{\ell(m+1)+O(m)}}{N^{1/2}}$$

where the probability is taken over all random choices of $\mathbf{d}$ for the given $\mathbf{f}$.

Certainly one can obtain similar results when the primes p and q are not balanced, although the probability of success decreases.

J.4.3 Experiments

We made a few experiments with a (balanced) 1024-bit RSA modulus and a public exponent $e = 3$, using Victor Shoup's NTL library [319]. The experiments have confirmed the heuristic condition. By applying standard floating point LLL reduction, and improved reduction if necessary, we have been able to recover the private exponent for all the parameters considered by Merkle in his own experiments [232] (see Table J.1). The success rate has been 100%, except with the case $(m, k, \ell) = (56, 26, 6)$ where it is 65% (for this case, Merkle only achieved a 10% success rate). We also made some experiments on other (more realistic) sets of parameters. For instance, over 100 samples, we have always been able to recover the factorization with $(m, k, \ell) = (60, 30, 3)$, $(70, 30, 2)$ and $(80, 40, 1)$. The attack takes at most a couple of minutes, as the lattice dimension is only $m + 1$. These results show that no set of parameters for RSA-S1 provides sufficient security without being impractical for the card.

Annexe K

Cryptanalysis of the RSA Schemes with Short Secret Exponent from Asiacrypt '99

ASIACRYPT 2000
[87] avec Glenn Durfee (Stanford)

Abstract: *At Asiacrypt '99, Sun, Yang and Laih proposed three RSA variants with short secret exponent that resisted all known attacks, including the recent Boneh-Durfee attack from Eurocrypt '99 that improved Wiener's attack on RSA with short secret exponent. The resistance comes from the use of unbalanced primes p and q. In this paper, we extend the Boneh-Durfee attack to break two out of the three proposed variants. While the Boneh-Durfee attack was based on Coppersmith's lattice-based technique for finding small roots to bivariate modular polynomial equations, our attack is based on its generalization to trivariate modular polynomial equations. The attack is heuristic but works well in practice, as the Boneh-Durfee attack. In particular, we were able to break in a few minutes the numerical examples proposed by Sun, Yang and Laih. The results illustrate once again the fact that one should be very cautious when using short secret exponent with RSA.*

K.1 Introduction

The RSA [294] cryptosystem is the most widely used public-key cryptosystem. However, RSA is computationally expensive, as it requires exponentiations modulo N, where N is a large integer (at least 1024 bits due to recent progress in integer factorization [60]) product of two primes p and q. Consequently, speeding up RSA has been a stimulating area of research since the invention of RSA. Perhaps the simplest method to speed up RSA consists of shortening the exponents of the modular exponentiations. If e is the RSA public exponent and d is the RSA secret exponent, one can either choose a small e or a small d. The choice of a small d is especially interesting when the device performing secret operations (signature generation or decryption) has limited computed power, such as smartcards. Unfortunately, Wiener [356] showed over 10 years ago that if $d \leq N^{0.25}$, then one could (easily) recover d (and hence, the secret primes p and q) in polynomial time from e and N using the continued fractions algorithm. Verheul and van Tilborg [347] slightly improved the bound in 1997, by showing that Wiener's attack could be applied to larger d, provided an exhaustive search on about $2\log_2(d/N^{0.25})$ bits. At Eurocrypt '99, Boneh and Durfee [43] presented the first substantial improvement over Wiener's bound. Their attack can (heuristically) recover p and q in polynomial

time if $d \leq N^{0.292}$. The attack is heuristic because it is based on the seminal lattice-based work by Coppersmith [70] on finding small roots to low-degree modular polynomial equations, in the bivariate case.[12] However, it should be emphasized that the attack works very well in practice.

At Asiacrypt '99, Sun, Yang and Laih [337] noticed that all those attacks on RSA with short secret exponent required some (natural) assumptions on the public modulus N. For instance, the Wiener's bound $N^{0.25}$ only holds if $p + q = O(\sqrt{N})$, and e is not too large. Similar restrictions apply to the extension to Wiener's attack by Verheul-van Tilborg [347], and to the Boneh-Durfee attack [43]. This led Sun, Yang and Laih to propose in [337] simple variants of RSA using a short secret exponent that, *a priori*, foiled all such attacks due to the previous restrictions. More precisely, they proposed three RSA schemes, in which only the (usual) RSA key generation is modified. In the first scheme, one chooses p and q of greatly different size, and a small exponent d in such a way that the previous attacks cannot apply. In particular, d can even be smaller than $N^{0.25}$ if p and q are unbalanced enough. The second scheme consists of a tricky construction that selects slightly unbalanced p and q in such a way that both e and d are small, roughly around $\sqrt{N}$. The third scheme is a mix of the first two schemes, which allows a trade-off between the sizes of e and d. Sakai, Morii and Kasahara [298] earlier proposed a different key generation scheme which achieves similar results to the third scheme, but that scheme can easily been shown insecure (see [337]).

In this paper, we show that the first and third schemes of [337] are insecure, by extending the Boneh-Durfee attack. Our attack can also break the second scheme, but only if the parameters are carelessly chosen. Boneh and Durfee reduced the problem of recovering the factors p and q to finding small roots of a particular bivariate modular polynomial equation derived from the basic equation $ed \equiv 1 \pmod{\varphi(N)}$. Next, they applied an optimized version (for that particular equation) of Coppersmith's generic technique [70] for such problems. However, when p and q are unbalanced, the particular equation used by Boneh and Durfee is not enough, because it has no longer any "small" root. Our attack extends the Boneh-Durfee method by taking into account the equation $N = pq$. We work with a system of two modular equations with three unknowns ; interestingly, when p and q are imbalanced, this approach leads to an attack on systems with d even larger than the $N^{0.292}$ bound of Boneh and Durfee. The attack is extremely efficient in practice : for typical instances of two of the schemes of [337], this approach breaks the schemes within several minutes. Also, our "triviariate" version of Coppersmith's technique we use may be of independent interest.

The remainder of this paper is organized as follows. In Section K.2, we briefly review former attacks on RSA with short secret exponents, recalling necessary background on lattice theory and Coppersmith's method to find small roots of low-degree modular polynomial equations. This is useful to explain our attacks. In Section K.3, we describe the RSA schemes with short secret exponent of [337]. In Section K.4, we present the new attack using the trivariate approach. We discuss an implementation of the attack and its running time on typical instances of the RSA variants in Section K.5.

K.2 Former Attacks on RSA with Short Secret Exponent

All known attacks on RSA with short secret exponent focus on the equation $ed \equiv 1 \bmod \varphi(N)$ (where $\varphi(N) = N - (p + q) + 1$) rewritten as :

$$ed = 1 + k\left(\frac{N+1}{2} - s\right) \tag{K.1}$$

where k is an unknown integer and $s = (p + q)/2$. The primes p and q can be recovered from either d or s. Note that k and d are coprime.

[12] The bivariate case is only heuristic for now, as opposed to the (simpler) univariate case, for which the method can be proved rigorously. For more information, see [70, 40, 266].

K.2.1 The Wiener Attack

Wiener's attack [356] is based on the continued fractions algorithm. Recall that if two (unknown) coprime integers A and B satisfy $|x - \frac{B}{A}| < \frac{1}{2A^2}$ where x is a known rational, then $\frac{B}{A}$ can be obtained in polynomial time as a convergent of the continued fraction expansion of x. Here, (K.1) implies that

$$\left|\frac{2e}{N} - \frac{k}{d}\right| = \frac{|2 + k(1-2s)|}{Nd}.$$

Therefore, if $\frac{k(2s-1)-2}{N} < \frac{1}{2d}$, d can be recovered in polynomial time from e and N, as k/d is a convergent of the continued fraction expansion of $2e/N$. That condition can roughly be simplified to $ksd = O(N)$, and is therefore satisfied if k, s and d are all sufficiently small. In the usual RSA key generation, $s = O(\sqrt{N})$ and $k = O(d)$, which leads to the approximate condition $d = O(N^{0.25})$. But the condition gets worse if p and q are unbalanced, making s much larger than $\sqrt{N}$. For instance, if $p = O(N^{0.25})$, the condition becomes $d = O(N^{0.125})$.

The extension of Wiener's attack by Verheul and van Tilborg [347] applies to $d > N^{0.25}$ provided exhaustive search on $O(\log_2(d/N^{0.25}))$ bits if p and q are balanced. Naturally, the attack requires much more exhaustive search if p and q are unbalanced.

K.2.2 The Boneh-Durfee Attack

The Small Inverse Problem.

The Boneh-Durfee attack [43] looks at the equation (K.1) modulo e :

$$-k\left(\frac{N+1}{2} - s\right) \equiv 1 \pmod{e}. \tag{K.2}$$

Assume that the usual RSA key generation is used, so that $|s| < \sqrt{e}$ and $|k| < d$ (ignoring small constants). The problem of finding such a small root (s, k) of that bivariate modular equation was called the *small inverse problem* in [43], since one is looking for a number $(N+1)/2 - s$ close to $(N+1)/2$ such that its inverse $-k$ modulo e is rather small. Note that heuristically, the small inverse problem is expected to have a unique solution whenever $|k| < d \leq N^{0.5}$. This led Boneh and Durfee to conjecture that RSA with $d \leq N^{0.5}$ is insecure.

Coppersmith [70] devised a general lattice-based technique to find sufficiently small roots of low-degree modular polynomial equations, which we will review in the next subsections, as it is the core of our attacks. By optimizing that technique to the specific polynomial of (K.2), Boneh and Durfee showed that one could solve the small inverse problem (and hence, break RSA) when $d \leq N^{0.292}$. This bound corresponds to the usual case of balanced p and q. It gets worse as p and q are unbalanced (see [43, 337]), because s becomes larger.

Lattice theory.

Coppersmith's technique, like many public-key cryptanalyses, is based on lattice basis reduction. We only review what is strictly necessary for this paper. Additional information on lattice theory can be found in numerous textbooks, such as [130, 326]. For the important topic of lattice-based cryptanalysis, we refer to the recent survey [266].

We will call *lattice* any subgroup of some $(\mathbb{Z}^n, +)$, which corresponds to the case of integer lattices in the literature. Consequently, for any integer vectors $\mathbf{b}_1, \ldots, \mathbf{b}_r$, the set $L(\mathbf{b}_1, \ldots, \mathbf{b}_r) = \{\sum_{i=1}^r n_i \mathbf{b}_i \mid n_i \in \mathbb{Z}\}$ of all integer linear combinations of the $\mathbf{b}_i$'s is a lattice, called the lattice *spanned* by the $\mathbf{b}_i$'s. In fact, all lattices are of that form. When $L = L(\mathbf{b}_1, \ldots, \mathbf{b}_r)$ and the $\mathbf{b}_i$'s are further linearly independent (over $\mathbb{Z}$), then $(\mathbf{b}_1, \ldots, \mathbf{b}_r)$ is called a *basis* of L. Any lattice L has infinitely

many bases. However, any two bases share some things in common, notably the number of elements r and the Gram determinant $\det_{1\leq i,j\leq r}\langle \mathbf{b}_i, \mathbf{b}_j\rangle$ (where $\langle , \rangle$ denotes the Euclidean dot product). The parameter r is called the lattice *dimension* (or *rank*), while the square root of the Gram determinant is the lattice *volume* (or *determinant*), denoted by $\mathrm{vol}(L)$. The name volume comes from the fact that the volume matches the r-dimensional volume of the parallelepiped spanned by the $\mathbf{b}_i$'s. In the important case of full-dimensional lattices (r equal to n), the volume is also the absolute value of the determinant of any basis (hence the name determinant). In general, it is hard to give a "simple" expression for the lattice volume, and one contents oneself with the Hadamard's inequality to estimate the volume :

$$\mathrm{vol}(L) \leq \prod_{i=1}^{r} \|\mathbf{b}_i\|.$$

Fortunately, sometimes, the lattice is full-dimensional and we know a specific basis which is triangular, making the volume easy to compute.

The volume is important because it enables one to estimate the size of short lattice vectors. A well-known result by Minkowski shows that in any r-dimensional lattice L, there exists a non-zero $\mathbf{x} \in L$ such that $\|\mathbf{x}\| \leq \sqrt{r} \cdot \mathrm{vol}(L)^{1/r}$, where $\|.\|$ denotes the Euclidean norm. That bound is in some (natural) sense the best possible. The LLL algorithm [205] can be viewed, from a qualitative point of view, as a constructive version of Minkowski's result. Given any basis of some lattice L, the LLL algorithm outputs in polynomial time a so-called *LLL-reduced* basis of L. The exact definition of an LLL-reduced basis is beyond the scope of this paper, we only mention the properties that are of interest here :

Fact K.1 *Any LLL-reduced basis $(\mathbf{b}_1, \ldots, \mathbf{b}_r)$ of a lattice L in $\mathbb{Z}^n$ satisfies :*

$$\|\mathbf{b}_1\| \leq 2^{r/2}\mathrm{vol}(L)^{1/r} \quad \textit{and} \quad \|\mathbf{b}_2\| \leq 2^{(r-1)/2}\mathrm{vol}(L)^{1/(r-1)}.$$

Coppersmith's technique.

For a discussion and a general exposition of Coppersmith's technique [70], see the recent surveys [40, 266]. We describe the technique in the bivariate case, following a simplified approach due to Howgrave-Graham [155].

Let e be a large integer of possibly unknown factorization. Assume that one would like to find all small roots of $f(x, y) \equiv 0 \pmod{e}$, where $f(x, y)$ is an integer bivariate polynomial with at least one monomial of maximal total degree which is monic. If one could obtain two algebraically independent integral bivariate polynomial equations satisfied by all sufficiently small modular roots (x, y), then one could compute (by resultant) a univariate integral polynomial equation satisfied by x, and hence find efficiently all small (x, y). Coppersmith's method tries to obtain such equations from reasonably short vectors in a certain lattice. The lattice comes from the linearization of a set of equations of the form $x^u y^v f(x, y)^w \equiv 0 \pmod{e^w}$ for appropriate integral values of u, v and w. Such equations are satisfied by any solution of $f(x, y) \equiv 0 \pmod{e}$. Small solutions (x_0, y_0) give rise to unusually short solutions to the resulting linear system, hence short vectors in the lattice. To transform modular equations into integer equations, one uses the following elementary lemma, with the (natural) notation $\|h(x, y)\| = \sqrt{\sum_{i,j} a_{i,j}^2}$ for $h(x, y) = \sum_{i,j} a_{i,j} x^i y^j$:

Lemma K.6 *Let $h(x, y) \in \mathbb{Z}[x, y]$ be a polynomial which is a sum of at most r monomials. Suppose that $h(x_0, y_0) \equiv 0 \bmod e^m$ for some positive integer m where $|x_0| < X$ and $|y_0| < Y$, and $\|h(xX, yY)\| < e^m/\sqrt{r}$. Then $h(x_0, y_0) = 0$ holds over the integers.*

Now the trick is to, given a parameter m, consider the polynomials

$$h_{u_1,u_2,v}(x, y) = e^{m-v} x^{u_1} y^{u_2} f(x, y)^v.$$

where u_1, u_2 and v are integers. Notice that any root (x_0, y_0) of $f(x, y)$ modulo e is a root modulo e^m of $h_{u_1,u_2,v}(x, y)$, and therefore, of any integer linear combination $h(x, y)$ of the $h_{u_1,u_2,v}(x, y)$'s. If such a combination $h(x, y)$ further satisfies $\|h(xX, yY)\| < e^m/\sqrt{r}$, where r is the number of monomials of h, then by Lemma K.6, the integer equation $h(x, y) = 0$ is satisfied by all sufficiently small modular roots of h modulo e. Thus, it suffices to find two algebraically independent such equations $h_1(x, y)$ and $h_2(x, y)$.

The use of integer linear combination suggests that we represent the polynomials as vectors in a lattice, so that finding polynomials with small norm reduces to finding short vectors in a lattice. More precisely, let $\mathcal{S}$ be a set of indices (u_1, u_2, v), and choose a representation of the polynomials $h_{u_1,u_2,v}(x, y)$ with $(u_1, u_2, v) \in \mathcal{S}$ as n-dimensional integer vectors for some n. Let L be the lattice in $\mathbb{Z}^n$ spanned by the vectors corresponding to $h_{u_1,u_2,v}(xX, yY)$ with $(u_1, u_2, v) \in \mathcal{S}$. Apply the LLL algorithm on the lattice, and let $h_1(xX, yY)$ and $h_2(xX, yY)$ be the polynomials corresponding to the first two vectors of the reduced basis obtained. Denoting by r the dimension of L, one deduces from the LLL theoretical bounds that :

$$\|h_1(xX, yY)\| \leq 2^{r/2}\mathrm{vol}(L)^{1/r} \text{ and } \|h_2(xX, yY)\| \leq 2^{(r-1)/2}\mathrm{vol}(L)^{1/(r-1)}.$$

To apply Lemma K.6, we want both of these upper bounds to be less than $e^m/\sqrt{n}$; since the factor 2^r is negligible with respect to e^m, this amounts to saying

$$\mathrm{vol}(L) \ll e^{mr}. \tag{K.3}$$

There are two problems. The first problem is that even if this condition is satisfied, so that Lemma K.6 applies, we are not guaranteed that the integer equations $h_1(x, y) = 0$ and $h_2(x, y) = 0$ obtained are algebraically independent. In other words, h_2 will provide no additional information beyond h_1 if the two *linearly* independent short basis vectors do not also yield *algebraically* independent equations. It is still an open problem to state precisely when this can be guaranteed, although all experiments to date suggest this is an accurate heuristic assumption to make when inequality (K.3) holds. We note that a similar assumption is used in the work of Bleichenbacher [32] and Jutla [173].

The second problem is more down-to-earth : how can we make sure that $\mathrm{vol}(L)$ is small enough to satisfy inequality (K.3) ? Note that Hadamard's bound is unlikely to be useful. Indeed, in general, some of the coefficients of $f(x, y)$ are about the size of e, so that $\|h_{u_1,u_2,v}(xX, yY)\|$ is at least e^m. To address this problem, one must choose in a clever way the set of indices $\mathcal{S}$ to have a close estimate on $\mathrm{vol}(L)$. The simplest solution is to choose $\mathcal{S}$ so that L is full-dimensional (r equal to n) and the $h_{u_1,u_2,v}(xX, yY)$'s form a triangular matrix for some ordering on the polynomials and on the monomials (the vector coordinates). Since we want $\mathrm{vol}(L)$ to be small, each coefficient on the diagonal should be the smallest one of $h_{u_1,u_2,v}(xX, yY) = e^{m-v}(xX)^{u_1}(yY)^{u_2}f(xX, yY)^v$, which is likely to be the one corresponding to the monic monomial of maximal total degree of $f(x, y)$.

In the general case, $f(x, y)$ may have several monomials of maximal total degree, and the only simple choice of $\mathcal{S}$ is to cover all the monomials of total degree less than some parametrized bound. More precisely, if Δ is the total degree of $f(x, y)$, and $x^a y^{\Delta-a}$ is a monic monomial of $f(x, y)$, one defines $\mathcal{S}$ as the set of (u_1, u_2, v) such that $u_1 + u_2 + \Delta v \leq h\Delta$ and $u_1, u_2, v \geq 0$ with $u_1 < a$ or $u_2 < \Delta - a$. Then the volume of the corresponding lattice can be computed exactly, and it turns out that (K.3) is satisfied whenever $XY < e^{1/\Delta-\varepsilon}$ for and m is sufficiently large.

However, depending on the shape of $f(x, y)$ (represent each monomial $x^i y^j$ by the point (i, j)), other choices of $\mathcal{S}$ might lead to improved bounds. Boneh and Durfee applied such tricks to the polynomial (K.2). In [43], they discussed several choices of $\mathcal{S}$. Using certain sets $\mathcal{S}$ for which the lattice is full-dimensional and one knows a triangular lattice basis, they obtained a first bound $d \leq N^{0.284}$ for their attack. Next, they showed that using a slightly different $\mathcal{S}$ for which the lattice is no longer full-dimensional, one ends up with the improved bound $d \leq N^{0.292}$. The latter choice of $\mathcal{S}$ is much harder to analyze. For more details, see [43].

K.3 The Sun-Yang-Laih RSA Key Generation Schemes

K.3.1 Scheme (I)

The first scheme corresponds to a simple unbalanced RSA [317] in which the parameters are chosen to foil previously known attacks :

1. Select two random primes $p < q$ such that both p and $N = pq$ are sufficiently large to foil factorization algorithms such as ECM and NFS. The more unbalanced p and q are, the smaller d can be.
2. Randomly select the secret exponent d such that $\log_2 d + \log_2 p > \frac{1}{3}\log_2 N$ and $d > 2^\gamma \sqrt{p}$, where γ is the security parameter (larger than 64).
3. If the public exponent e defined by $ed \equiv 1 \pmod{\varphi(N)}$ is not larger than $\varphi(N)/2$, one restarts the previous step.

A choice of parameters suggested by the authors is : p is a 256-bit prime, q is a 768-bit prime, d is a 192-bit number. Note that 192 is far below Wiener's bound (256 bits) and Boneh-Durfee's bound (299 bits).

K.3.2 Scheme (II)

The second scheme selects one of the primes in such a way that one can select e and d to be small at the same time :

1. Fix the bit-length of N.
2. Select a random prime p of $\frac{1}{2}\log_2 N - 112$ bits, and a random k of 112 bits.
3. Select a random d of $\frac{1}{2}\log_2 N + 56$ bits coprime with $k(p-1)$.
4. Compute the two Bézout integers u and v such that $du - k(p-1)v = 1$, $0 < u < k(p-1)$ and $0 < v < d$.
5. Return to Step 3 if $v + 1$ is not coprime with d.
6. Select a random h of 56 bits until $q = v + hd + 1$ is prime.

The RSA parameters are p, q, $e = u + hk(p-1)$, d and $N = pq$. Notice that e and d satisfy the equation $ed = 1 + k\varphi(N)$. They both have approximate bit-length $\frac{1}{2}\log_2 N + 56$. The primes p and q have approximate bit-length $\frac{1}{2}\log_2 N - 112$ and $\frac{1}{2}\log_2 N + 112$ respectively.

A possible choice of parameters for Scheme (II) might be : p a 400-bit prime, q a 624-bit prime, and e and d are each 568-bit integers.

K.3.3 Scheme (III)

The third scheme is a mix of the first two schemes, allowing a trade-off between e and d such that $\log_2 e + \log_2 d \approx \log_2 N + \ell_k$ where ℓ_k is a predetermined constant. More precisely, the scheme is a parametrized version of scheme II : p, k, d and h have respective bit-length ℓ_p (less than $\frac{1}{2}\log_2 N$), ℓ_k, ℓ_d, and $\log_2 N - \ell_p - \ell_d$. To resist various attacks, the following is required :

1. $\ell_k \gg \ell_p - \ell_d + 1$.
2. $4\alpha(2\beta + \alpha - 1) \gg 3(1 - \beta - \alpha)^2$, where $\alpha = \frac{\log_2 N - \ell_p}{\log_2 N + \ell_k - \ell_d}$ and $\beta = \frac{\ell_k}{\log_2 N + \ell_k - \ell_d}$.
3. k must withstand an exhaustive search and $\ell_k + \ell_p > \frac{1}{3}\log_2 N$.

   A choice of parameters suggested by the authors is : p is a 256-bit prime, q is a 768-bit prime, e is an 880-bit number, and d is a 256-bit number.

K.4 The Attack Algorithm

In this section we demonstrate how to launch an attack on Schemes (I) and (III). The approach used here closely follows that taken by Boneh and Durfee [43], but differs in several crucial ways to allow it to work when the factors p and q of the public modulus N are unbalanced. Interestingly, our attack gets better (works for larger and larger d) the more unbalanced the factors of the modulus become.

Recall the RSA equation

$$ed = 1 + k\left(\frac{N+1}{2} - \frac{p+q}{2}\right).$$

We note that the Boneh-Durfee approach treats this as an equation modulo e with two "small" unknowns, k and $s = (p+q)/2$. This approach no longer works if p and q are unbalanced, since a good bound on s can no longer be established. For this reason, the authors of the schemes from Section K.3 hoped that these schemes would resist the lattice-based cryptanalysis outlined in Section K.2.2.0. However, we will see that a more careful analysis of the RSA equation, namely one that does not treat $p+q$ as a single unknown quantity but instead leaves p and q separately as unknowns, leads to a successful attack against two of these schemes.

Writing $A = N + 1$, the RSA equation implies

$$2 + k(A - p - q) \equiv 0 \pmod{e}.$$

The critical improvement of our attack is to view this as a modular equation with *three* unknowns, k, p, q, with the special property that the product pq of two of them is the known quantity N. We may view this problem as follows : given a polynomial $f(x, y, z) = x(A + y + z) - 2$, find (x_0, y_0, z_0) satisfying :

$$f(x_0, y_0, z_0) \equiv 0 \pmod{e},$$

where

$$|x_0| < X, \;\; |y_0| < Y, \;\; |z_0| < Z, \;\; \text{and} \;\; y_0 z_0 = N.$$

Note that the bounds $X \approx ed/N$, $Y \approx p$, and $Z \approx q$ can be estimated to within a power of 2 based on the security parameters chosen for the scheme.

Following Coppersmith's method, our approach is to pick r equations of the form $e^{m-v} x^{u_1} y^{u_2} z^{u_3} \cdot f^v(x, y, z)$ and to search for low-norm integer linear combinations of these polynomials. The basic idea is to start with a handful of equations of the form $y^{a+j} f^m(x, y, z)$ for $j = 0, \ldots, t$ for some integers a and t with $t \geq 0$. Knowing $N = pq$ allows us to replace all occurrences of the monomial yz with the constant N, reducing the number of variables in each of these equations to approximately m^2 instead of the expected $\frac{1}{3}m^3$. We will refer to these as the *primary polynomials*.

Since there are only $t+1$ of these equations, this will result in a lattice that is less than full rank ; we therefore include some additional equations to bring the lattice to full rank in order to compute its determinant. We refer to these as the *helper polynomials*. We have a great deal of choice in picking the helper polynomials ; naturally, some choices are better than others, and it is generally a tedious but straightforward optimization problem to choose the primary and helper polynomials that are optimal. The equations we work with are the following. Fix an integer m, and let a and $t > 0$ be integers which we will optimize later. We define

- $g_{k,i,b}(x, y, z) := e^{m-k} x^i y^a z^b f^k(x, y, z)$, for $k = 0..(m-1)$, $i = 1..(m-k)$, and $b = 0, 1$; and,
- $h_{k,j}(x, y, z) := e^{m-k} y^{a+j} f^k(x, y, z)$, for $k = 0..m$ and $j = 0..t$.

The primary polynomials are $h_{m,j}(x, y, z)$ for $j = 0, \ldots, t$, and the rest are the helper polynomials. Following Coppersmith's technique, we form a lattice L by representing $g_{k,i,b}(xX, yY, zZ)$ and $h_{k,j}(xX, yY, zZ)$ by their coefficients vectors, and use LLL to find low-norm integer linear combinations $h_1(xX, yY, zZ)$ and $h_2(xX, yY, zZ)$. The polynomials $h_1(x, y, z)$ and $h_2(x, y, z)$ have (k, p, q)

as a root over the integers ; to remove z as an unknown, we use the equality $z = N/y$, obtaining $H_1(x, y)$ and $H_2(x, y)$ which have (k, p) as a solution. Taking the resultant $\mathrm{Res}_x(H_1(x, y), H_2(x, y))$ yields a polynomial $H(y)$ which has p as a root. Using standard root-finding techniques allows us to recover the factor p of N efficiently, completing the attack.

The running time of this algorithm is dominated by the time to run LLL on the lattice L, which has dimension $(m + 1)(m + t + 1)$. So it would be ideal to keep the parameters m and t as low as possible, limiting to a reasonable number the polynomials used to construct L. Surprisingly, the attack is successful even if only a handful of polynomials are used. The example given by the original authors for schemes (I) succumbs easily to this attack with $m = 3$ and $t = 1$; with these parameters, our attack generates 20 polynomials. Scheme (III) can be cryptanalyzed with parameters $m = 2$ and $t = 2$, yielding 15 polynomials. This gives lattices of dimension 20 (see Figure K.1) and 15, respectively, which can be reduced *via* the LLL algorithm within a matter of seconds on a desktop computer. We discuss our implementation and the results of our experiments more in Section K.5.

K.4.1 Analysis of the Attack

In order to be sure that LLL returns vectors that are "short enough" to use Lemma K.6, we must derive sufficiently small bounds on the determinant of the lattice L formed from the polynomials $g_{k,i,b}(xX, yY, zZ)$ and $h_{k,j}(xX, yY, zZ)$. Fortunately, this choice of polynomials makes the computation of the determinant of L fairly straightforward, if somewhat tedious. We provide the details in the appendix.

Representing the Lattice as a Triangular Matrix. In order to compute the volume of the lattice L, we would like to list the polynomials $g_{k,i,b}(xX, yY, zZ)$ and $h_{k,j}(xX, yY, zZ)$ in a way that yields a triangular matrix. There is an ordering on these polynomials that leads to such a representation : we first list the $g_{k,i,b}(xX, yY, zZ)$ indexed outermost by $k = 0, \ldots, m - 1$, then $i = 0, \ldots, k$, then innermost by $b = 0, 1$. We then list $h_{k,j}(xX, yY, zZ)$ indexed outermost by $k = 0, \ldots, m$ then $j = 0, \ldots, t$. (See Figure K.1 for the case of $m = 2$, $t = 1$, $a = 1$.) Each new polynomial introduces exactly one new monomial $x^{u_1}y^{u_2}$ or $x^{u_1}z^{u_3}$. Note that no monomial involving the product yz appears, since yz can be eliminated[13] using the identity $N = yz$.

The determinant of this matrix is simply the product of the entries on the diagonal, which for $m = 3$, $t = 1$, $a = 1$ is

$$\mathrm{vol}(L) = \det(M) = e^{40}X^{40}Y^{34}Z^4. \tag{K.4}$$

We expect the LLL algorithm to return vectors short enough to use Lemma K.6 when

$$\mathrm{vol}(L) = e^{40}X^{40}Y^{34}Z^4 < e^{mr} = e^{60}.$$

The example given by the original authors for Scheme (I) is to use p of 256 bits, q of 768 bits, d of 192 bits, and e of 1024 bits. This gives bounds

$$X \approx ed/N \approx e^{3/16}, \;\; Y \approx e^{1/4}, \text{ and } Z \approx e^{3/4};$$

we may then confirm

$$\det(M) = e^{40}X^{40}Y^{34}Z^4 \approx e^{59} < e^{60} = e^{mr},$$

[13] Caution must be taken to ensure the polynomials remain monic in the terms $x^{u_1}y^{u_2}$ and $x^{u_1}z^{u_3}$ of highest degree ; if the substitution $yz \mapsto N$ causes a coefficient of such a term to be different from 1, then we multiply the polynomial by N^{-1} mod e^m (and reduce mod e^m as appropriate) before continuing.

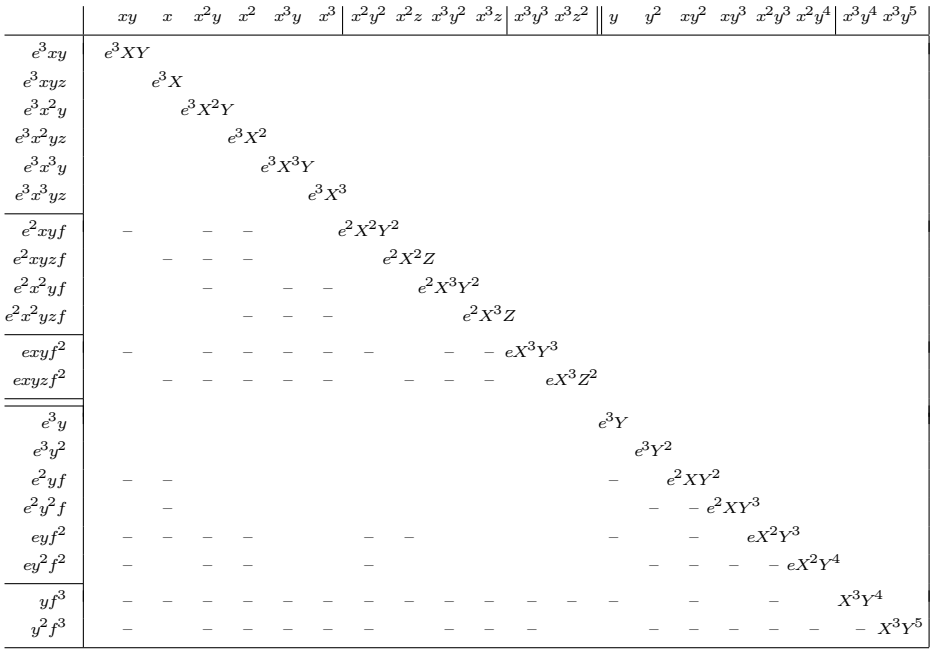


FIG. K.1 – *Example of the lattice formed by the vectors $g_{k,i,b}(xX, yY, zZ)$ and $h_{k,j}(xX, yY, zZ)$ when $m = 3$, $t = 1$, and $a = 1$. The matrix is lower triangular. Entries marked with "–" indicate off-diagonal quantities whose values do not affect the determinant calculation. The polynomials used are listed on the left, and the monomials they introduce are listed across the top. The double line break occurs between the $g_{k,i,b}$ and the $h_{k,j}$, while the single line breaks occur between increments of k. The last single line break separates the helper polynomials (top) from the two primary polynomials (bottom).*

so Lemma K.6 applies.[14] Therefore, when we run the LLL algorithm on this lattice, we will get two short vectors corresponding to polynomials $h_1(x, y, z)$, $h_2(x, y, z)$; by the bound on the determinant, we know that these polynomials will have norm that is low enough to use Lemma K.6. Therefore these polynomials will have (k, p, q) as a solution over the integers. To turn these into bivariate equations, we use the equality $z = N/y$ to get $H_1(x, y)$ and $H_2(x, y)$ which have (k, p) as a solution over the integers. We then take the resultant $Res_x(H_1(x, y), H_2(x, y))$ to obtain a univariate polynomial $H(y)$ that has p as a root.

More generally, if we pick optimal values for t and a take m sufficiently large, our attack will be successful for even larger bounds on d. The highest possible bound on d for which our attack can work depends on the parameters chosen for the scheme. Suppose the parameter $d \approx N^\delta$ is used. The table below summarizes the largest possible δ for which our attack can succeed. We point out the choices of parameters that give rise to the schemes of Section K.3.

For example, with the example for Scheme (I), where $e \approx N$ and $p \approx N^{0.25}$, our attack will be successful not only for the $\delta = 0.188$ suggested, but all the way up to $\delta < 0.364$ (assuming a large enough m is used.) Similarly, our attack works in Scheme (III) up to $d < N^{0.412}$. Notice that our attack comes close to, but cannot quite reach, the $d < N^{0.55}$ required to break Scheme (II).

[14]The reader may have noticed that we have suppressed the error term associated with the execution of the LLL algorithm. Interestingly, even if the LLL "fudge factor" is taken into account, this bound is still good enough. We require

$$\text{vol}(L) < 2^{r^2/2} e^{59} = 2^{200} e^{59} < e^{59+\frac{1}{5}} < e^{mr}/(\sqrt{r})^r \approx e^{60-\frac{1}{20}}.$$

Slightly larger parameters m and t are required to rigorously obtain the bound for norm of the second basis vector, although in practice the LLL algorithm works well enough so that the parameters chosen here are sufficient.

$\log_N(p)$ \ $\log_N(e)$	1.0	0.9	0.86	0.8	0.7	0.6	0.55
0.5	0.284	0.323	0.339	0.363	0.406	0.451	0.475
0.4	0.296	0.334	0.350	0.374	0.415	0.460	0.483_{II}
0.3	0.334	0.369	0.384	0.406	0.446	0.487	0.510
0.25	0.364_{I}	0.398	0.412_{III}	0.433	0.471	0.511	0.532
0.2	0.406	0.437	0.450	0.470	0.505	0.542	0.562
0.1	0.539	0.563	0.573	0.588	0.615	0.644	0.659

FIG. K.2 – *Largest δ (where $d < N^{\delta}$) for which our attack can succeed, as a function of the system parameters.*

K.4.2 Comparison with the Bivariate Approach

Alternatively, one can consider the system of two modular equations with three unknowns as a single bivariate equation by incorporating the equation $N = pq$ into the main trivariate equation. This was independently noticed by Willi Meier [229], who also addressed the problem of breaking Schemes (I) and (III), using a bivariate approach rather than our trivariate approach. One then obtains an equation of the form $f(x, y) = x^2y + Axy + Bx + Cy$ modulo e, where the unknowns are k and the smallest prime among p and q.

However, it turns out that the application of Coppersmith's technique to this particular bivariate equation yields worse bounds than with the trivariate approach previously described. For example, the bivariate approach allows one to break scheme (I) as long as $d < N^{0.135}$ (and perhaps slightly higher, if sublattices are considered as in [43]), but fails for larger d. One can view the bivariate approach a special case of our trivariate approach, in which one degree of freedom for optimization has been removed. One then sees that the bivariate approach constrains the choice of primary and helper polynomials in a suboptimal way, resulting in worse bounds on d.

K.5 Implementation

We implemented this attack using Victor Shoup's Number Theory Library [319] and the Maple Analytical Computation System [219]. The attack runs very efficiently, and in all instances of Schemes (I) and (III) we tested, it produced algebraically independent polynomials $H_1(x, y)$ and $H_2(x, y)$. These yielded a resultant $H(y) = (y - p)H_0(y)$, where $H_0(y)$ is irreducible, exposing the factor p of N in every instance. This strongly suggests that this "heuristic" assumption needed to complete the multivariate modular version of Coppersmith's technique is extremely reliable, and we conjecture that it always holds for suitably bounded lattices of this form. The running times of our attacks are given below.

Scheme	size of n	size of p	size of e	size of d	m	t	a	lattice rank	running time
I	1024	256	1024	192	3	1	1	20	40 seconds
III	1024	256	880	256	2	2	0	15	9 seconds

These tests were run on a 500MHz Pentium III running Solaris.

K.6 Conclusions and Open Problems

We showed that unbalanced RSA [317] actually improves the attacks on short secret exponent by allowing larger exponent. This enabled us to break most of the RSA schemes [337] with short secret exponent from Asiacrypt '99. The attack extends the Boneh-Durfee attack [43] by using a

"trivariate" version of Coppersmith's lattice-based technique for finding small roots of low-degree modular polynomial equations. Unfortunately, despite experimental evidence, the attack is for now only heuristic, as the Boneh-Durfee attack. It is becoming increasingly important to find sufficient conditions for which Coppersmith's technique on multivariate modular polynomials can be proved.

Our results illustrate once again the fact that one should be very cautious when using RSA with short secret exponent. To date, the best method to enjoy the computational advantage of short secret exponent is the following countermeasure proposed by Wiener [356]. When $N = pq$, the idea is to use a private exponent d such that both $d_p = d \bmod (p-1)$ and $d_q = d \bmod (q-1)$ are small. Such a d speeds up RSA signature generation since RSA signatures are often generated modulo p and q separately and then combined using the Chinese Remainder Theorem. Classical attacks do not work since d is likely to be close to $\varphi(N)$. It is an open problem whether there is an efficient attack on such secret exponents. The best known attack runs in time $\min(\sqrt{d_p}, \sqrt{d_q})$.

Acknowledgements

Part of this work was done while the second author was visiting Stanford University, whose hospitality is gratefully acknowledged. We thank Hyun-Soo Hong and Damien Stehlé for helpful comments on the proceedings version.

K.7 Appendix : General calculation of the determinant

The general formula for the determinant of the lattice we build in Section K.4 is

$$\mathrm{vol}(L) = \det(M) = e^{C_e} X^{C_x} Y^{C_y} Z^{C_z},$$

where

$$\begin{aligned}
C_e &= C_x = \frac{1}{6} m(m+1)(4m+3t+5), \\
C_y &= \begin{cases} \frac{1}{6}(m^3 + 3(a+t+1)m^2 + (3t^2+6at+3a^2+6a+6t+2)m \\ \qquad +(3t^2+6at+3a^2+4a+3t-a^3)) & \text{if } a \geq 0, \\ \frac{1}{6}(m^3 + 3(a+t+1)m^2 + (3t^2+6at+3a^2+6a+6t+2)m \\ \qquad +(3t^2+6at+3a^2+3a+3t)) & \text{if } a < 0, \end{cases} \\
C_z &= \begin{cases} \frac{1}{6}(m^3 - 3(a-1)m^2 + (3a^2-6a+2)m + (3a^2-2a-a^3)) & \text{if } a \geq 0, \\ \frac{1}{6}(m^3 - 3(a-1)m^2 + (3a^2-6a+2)m + (3a^2-3a)) & \text{if } a < 0. \end{cases}
\end{aligned}$$

We need $\det(M) < e^{mr} = e^{m(m+1)(m+t+1)}$. In order to optimize the choice of t and a, we write $t = \tau m$ and $a = \alpha m$, and observe

$$\begin{aligned}
C_e &= C_x = \frac{1}{6}(3\tau+4)m^3 + o(m^3), \\
C_y &= \begin{cases} \frac{1}{6}(3\tau^2+6\alpha\tau+3\alpha^2+3\alpha+3\tau+1-\alpha^3)m^3 + o(m^3) & \text{if } \alpha \geq 0, \\ \frac{1}{6}(3\tau^2+6\alpha\tau+3\alpha^2+3\alpha+3\tau+1)m^3 + o(m^3) & \text{if } \alpha < 0, \end{cases} \\
C_z &= \begin{cases} \frac{1}{6}(3\alpha^2-3\alpha+1-\alpha^3)m^3 + o(m^3) & \text{if } \alpha \geq 0, \\ \frac{1}{6}(3\alpha^2-3\alpha+1)m^3 + o(m^3) & \text{if } \alpha < 0. \end{cases}
\end{aligned}$$

Suppose we write $e = N^{\varepsilon}$, $d = N^{\delta}$, and $Y = N^{\beta}$, so $Z = N^{1-\beta}$. Then $X = N^{\varepsilon+\delta-1}$. So the requirement on $\det(M)$ now becomes

$$N^{\varepsilon C_e + (\varepsilon+\delta-1)C_x + \beta C_y + (1-\beta)C_z} < e^{m(m+1)(m+t+1)} = N^{\varepsilon(\tau+1)m^3 + o(m^3)}.$$

The above expression holds (for large enough m) when

$$\varepsilon C_e + (\varepsilon + \delta - 1)C_x + \beta C_y + (1-\beta)C_z - (\tau + 1) < 0. \tag{K.5}$$

The left-hand-side of this expression achieves its minimum at

$$\begin{aligned} \tau_0 &= (2\alpha_0\beta - \beta - \delta + 1)/(2\beta), \\ \alpha_0 &= \begin{cases} 1 - \beta - (1 - \beta - \delta + \beta^2)^{(1/2)} & \text{if } \beta < \delta, \\ (\beta - \delta)/(2\beta - 2) & \text{if } \beta \geq \delta. \end{cases} \end{aligned}$$

Using $\tau = \tau_0$ and $\alpha = \alpha_0$ will give us the minimum value on the left-hand-side of inequality K.5, affording us the largest possible X to give an attack on the largest possible $d < N^\delta$. The entries in Figure K.2 were generated by plugging in τ_0 and α_0 and solving for equality in Equation K.5.

It is interesting to note that formulation of the root-finding problem for RSA as a trivariate equation is strictly more powerful than its formulation as the small inverse problem. This is because the small inverse problem is not expected to have a unique solution once $\delta > 0.5$, while our attack works in many cases with $\delta > 0.5$. We note that when $\varepsilon = 1$ and $\beta = 0.5$ – as in standard RSA – our attack gives identical results to simpler Boneh-Durfee attack ($d < N^{0.284}$). Their optimization of using lattices of less than full rank to achieve the $d < N^{0.292}$ bound should also work with our approach, but we have not analyzed how much of an improvement it will provide.

Failles d'implémentation

> « Most security failures in its area of interest are due to failures in implementation, not failure in algorithms or protocols. »
>
> The NSA

Cette partie illustre les problèmes de sécurité qui peuvent se poser au niveau des implémentations cryptographiques, même lorsqu'on implémente des algorithmes cryptographiques bien connus. Elle se compose de trois articles :

Page 301 : référence [261].

The Insecurity of the Digital Signature Algorithm with Partially Known Nonces, Journal of Cryptology (2002)
Phong Q. Nguyen et Igor E. Shparlinski

Cet article présente une attaque polynomiale prouvée contre la norme de signature américaine DSA, sous l'hypothèse que soient révélés quelques bits des nombres pseudo-aléatoires utilisés lors de chaque génération de signature. L'algorithme DSA est probabiliste comme la plupart des algorithmes de signature à base de logarithme discret.

Page 323 : référence [246].

Experimenting with Faults, Lattices, and the DSA, PKC 2005
David Naccache, Phong Q. Nguyen, Michael Tunstall et Claire Whelan

Cet article montre que le scénario de l'attaque précédente est réaliste contre des implémentations typiques de DSA sur carte à puce, lorsque l'adversaire utilise des techniques d'injections de faute. En combinant l'attaque précédente avec des attaques matérielles par injection de faute, on peut donc extraire la clef secrète d'une implémentation typique de DSA sur carte à puce.

Page 333 : référence [255].

Can We Trust Cryptographic Software ? Cryptographic Flaws in GNU Privacy Guard v1.2.3, EUROCRYPT 2004
Phong Q. Nguyen

Cet article présente diverses faiblesses cryptographiques dans le logiciel libre GPG de sécurisation du courrier électronique, qui est inclus dans les principales distributions du système d'exploitation LINUX. Ces faiblesses ont été trouvées en examinant le code source de GPG. La faille découverte la plus grave a conduit à la suppression de la signature El Gamal dans GPG.

Annexe L

The Insecurity of the Digital Signature Algorithm with Partially Known Nonces

Journal of Cryptology (2002)
[261] avec Igor E. Shparlinski (Macquarie University)

Abstract: *We present a polynomial-time algorithm that provably recovers the signer's secret DSA key when a few consecutive bits of the random nonces k (used at each signature generation) are known for a number of DSA signatures at most linear in* $\log q$ *(q denoting as usual the small prime of DSA), under a reasonable assumption on the hash function used in DSA. For most significant or least significant bits, the number of required bits is about* $\log^{1/2} q$*, but can be decreased to* $\log\log q$ *with a running time* $q^{O(1/\log\log q)}$ *subexponential in* $\log q$*, and even further to 2 in polynomial time if one assumes access to ideal lattice basis reduction, namely an oracle for the lattice closest vector problem for the infinity norm. For arbitrary consecutive bits, the attack requires twice as many bits. All previously known results were only heuristic, including those of Howgrave-Graham and Smart who recently introduced that topic. Our attack is based on a connection with the* hidden number problem *(HNP) introduced at Crypto '96 by Boneh and Venkatesan in order to study the bit-security of the Diffie–Hellman key exchange. The HNP consists, given a prime number q, of recovering a number $\alpha \in \mathbb{F}_q$ such that for many known random $t \in \mathbb{F}_q$ a certain approximation of of $t\alpha$ is known. To handle the DSA case, we extend Boneh and Venkatesan's results on the HNP to the case where t has not necessarily perfectly uniform distribution, and establish uniformity statements on the DSA signatures, using exponential sum techniques. The efficiency of our attack has been validated experimentally, and illustrates once again the fact that one should be very cautious with the pseudo-random generation of the nonce within DSA.*

Keywords: *Cryptanalysis, DSA, lattices, LLL, closest vector problem, distribution, discrepancy, exponential sums.*

L.1 Introduction

L.1.1 The Digital Signature Algorithm (DSA)

Recall the *Digital Signature Algorithm* (see [231, 333]), or DSA, used in the American federal digital signature standard [248].

Let p and $q \geq 3$ be prime numbers with $q|p-1$. As usual $\mathbb{F}_p$ and $\mathbb{F}_q$ denote fields of p and q elements which we assume to be represented by the elements $\{0,\ldots,p-1\}$ and $\{0,\ldots,q-1\}$ respectively.

For a rational number z and $m \geq 1$ we denote by $\lfloor z \rfloor_m$ the unique integer a, $0 \leq a \leq m-1$ such that $a \equiv z \pmod{m}$ (provided that the denominator of z is relatively prime to m). We also use $\log z$ to denote the binary logarithm of $z > 0$.

Let $\mathcal{M}$ be the set of messages to be signed and let $h : \mathcal{M} \to \mathbb{F}_q$ be an arbitrary hash-function. The signer's secret key is an element $\alpha \in \mathbb{F}_q^*$.

Let $g \in \mathbb{F}_p$ be a fixed element of multiplicative order q, that is $g^q = 1$ and $q \neq 1$ which is *publicly* known. To sign a message $\mu \in \mathcal{M}$, one chooses a random integer $k \in \mathbb{F}_q^*$ usually called the *nonce*, and which must be kept secret. One then defines the following two elements of $\mathbb{F}_q$:

$$\begin{aligned} r(k) &= \left\lfloor \left\lfloor g^k \right\rfloor_p \right\rfloor_q \\ s(k,\mu) &= \left\lfloor k^{-1}\left(h(\mu) + \alpha r(k)\right) \right\rfloor_q . \end{aligned}$$

The pair $(r(k), s(k,\mu))$ is the *DSA signature* of the message μ with a nonce k. In general, q has bit-length 160 and p has bit-length between 512 and 1024.

L.1.2 Former results

The security of DSA relies on the hardness of the discrete logarithm problem in prime fields and it subgroups. Under slight modifications and the random oracle model [24], the security of DSA (with respect to adaptive chosen-message attacks) can be proved relative to the hardness of discrete logarithm (see [52]). The well-known random oracle model assumes that the hash function behaves as a random oracle, that is, its values are independent and uniformly distributed.

However, serious precautions must be taken when using DSA. It was noticed by Vaudenay [345] that the primes p and q need to be validated, for one could forge signature collisions otherwise. Special care must be taken with the nonce k. It is well-known that if k is disclosed, then the secret key α can easily be recovered. It was shown by Bellare *et al.* [23] that one can still recover α if the nonce k is produced by Knuth's linear congruential generator with known parameters, or variants. That attack is provable under the random oracle model, and relies on Babai's approximation algorithm [17] for the *closest vector problem* (CVP) in a lattice, which is based on the celebrated LLL algorithm [205]. The attack does not work if the parameters of the generator are unknown.

Recently, Howgrave-Graham and Smart [159] introduced a different scenario. Suppose that for a reasonable number of signatures, a small fraction of the corresponding nonce k is revealed. For instance, suppose that the ℓ least significant bits of k are known. Howgrave-Graham and Smart proposed in [159] several heuristic attacks to recover the secret key in such setting and variants (known bits in the middle, or split in several blocks) when ℓ is not too small. Like [23], the attacks are based on LLL-based Babai's CVP approximation algorithm [17]. However, the attacks of [23] and [159] are quite different. Howgrave-Graham and Smart followed an applied approach. The attack used several heuristic assumptions which did not allow precise statements on its theoretical behaviour It was assumed that the DSA signatures followed a perfectly uniform distribution, that some lattice enjoyed some natural however heuristic property, and that Babai's algorithm [17] behaves much better than theoretically guaranteed. Consequently, it was hard to guess what were the limitations of the attack such as how small could ℓ be in practice, and what could be proved.

L.1.3 Our results

In this paper, we present the first provable polynomial-time attack against DSA when the nonces are partially known, under two reasonable assumptions : the size of q should not be too small compared

to p, and the probability of collisions for the hash function h should not be too large compared to $1/q$. More precisely, under these conditions, we show that if for a certain (polynomially bounded) number of random messages $\mu \in \mathcal{M}$ and random nonces $k \in [1, q-1]$ about $\log^{1/2} q$ least significant bits of k are known, then in polynomial time one can recover the signer's secret key α. The same result holds for the most significant bits when one uses an appropriate definition of the most significant bits tailored to modular residues. With the usual definition of most significant bits, one needs one more bit than in the case of least significant bits, as q might be only marginally larger than a power of two (certainly this distinction is important only for our numerical results). The result is slightly worse for arbitrary windows of consecutive bits : in such a case, one requires twice as many bits (contrary to what the analysis of [159] suggested). For least significant bits (or appropriate most significant bits), the number of bits can be decreased to 2 if one further assumes access to ideal lattice reduction (namely, an oracle for the closest vector problem for the infinity norm). Such an assumption is realistic in low dimension despite NP-hardness results on lattice problems, due to the well-known experimental fact that state-of-the-art lattice basis reduction algorithms behave much better than theoretically guaranteed. Alternatively, the number of bits can be decreased to $\log \log q$ but with a running time $q^{O(1/\log \log q)}$ subexponential in $\log q$, using the closest vector approximation algorithm of [8, Corollary 16]. This subexponential running time is interesting, as the bit-length of q is usually chosen to be 160, in order to avoid square-root attacks.

Our attack has been validated experimentally. Using a standard workstation, we could most of the time recover in a few minutes the signer's DSA 160-bit secret key when only $\ell = 3$ least significant bits of the nonces were known for about 100 signatures. Interestingly, this improves the experimental results of [159], where the best experiment corresponded to $\ell = 8$, and where it was suggested that $\ell = 4$ was impossible.

It should be pointed out that the study of the security of DSA in such settings might have practical implications. Indeed, Bleichenbacher [34] recently noticed that in AT&T's CryptoLib version 1.2 (a widely distributed cryptographic library), the implementation of DSA suffers from the following flaw : the random nonce k is always odd, thus leaking its least significant bit. Apparently, this is because the same routine is used in the implementation of the El Gamal signature scheme, for which k must be coprime with $p-1$, and thus necessarily odd. Our results do not show that CryptoLib's DSA implementation can be broken, but they do not rule out such a possibility either, even with the same attack. In fact, they indicate a potential weakness in this implementation.

This has been confirmed by a very recent important result of Bleichenbacher [35], who has presented a heuristic attack on DSA with time complexity 2^{64} (and requiring memory 2^{40} and 2^{22} signatures), when the pseudo-random number generator suggested by the NIST to produce the nonces is used (see [248]). The NIST generator suffered from the following flaw : the outputs are biased in the sense that small values modulo q are more likely to occur than high values modulo q. This is because the output is some 160-bit pseudo-random number reduced modulo the 160-bit prime q. Bleichenbacher's heuristic attack also applies to the case when some of the bits of the nonces are known. The attack is based on clever meet-in-the-middle techniques, and not lattices. Currently, the best experimental result with this attack is that one can recover the secret key given a leakage of $\log 3 \approx 1.58$ bits for 2^{22} signatures, in about 3 days on a 450 MHz Ultrasparc using 500Mb of RAM. Thus, our experimental results are superseded by Bleichenbacher's results. Note however that the techniques used are completely different, and that our method remains the only one yielding provable results at the moment.

L.1.4 Overview of our attack

Our attack follows Nguyen's approach [254] that reduces the DSA problem to a variant of the *hidden number problem* (HNP) introduced in 1996 by Boneh and Venkatesan [48, 49]. The HNP can be stated as follows : recover a number $\alpha \in \mathbb{F}_q$ such that for many known random $t \in \mathbb{F}_q$, an

approximation $\mathrm{APP}_{\ell,q}(\alpha t)$ of αt is known. Here, for any rationals n and ℓ, the notation $\mathrm{APP}_{\ell,q}(n)$ denotes any rational r such that :

$$|n - r|_q \leq \frac{q}{2^{\ell+1}},$$

where the symbol $|.|_q$ is defined as $|z|_q = \min_{b\in\mathbb{Z}} |z - bq|$ for any real z.

The connection between the DSA problem and the HNP can easily be explained. Assume that we know the ℓ least significant bits of a nonce $k \in \mathbb{F}_q^*$. That is, we are given an integer a such that $0 \leq a \leq 2^\ell - 1$ and $k - a = 2^\ell b$ for some integer $b \geq 0$. Given a message μ signed with the nonce k, the congruence

$$\alpha r(k) \equiv s(k,\mu)k - h(\mu) \pmod q,$$

can be rewritten for $s(k,\mu) \neq 0$ as :

$$\alpha r(k) 2^{-\ell} s(k,\mu)^{-1} \equiv \left(a - s(k,\mu)^{-1} h(\mu)\right) 2^{-\ell} + b \pmod q. \tag{L.1}$$

Now define the following two elements

$$\begin{aligned} t(k,\mu) &= \left\lfloor 2^{-\ell} r(k) s(k,\mu)^{-1} \right\rfloor_q, \\ u(k,\mu) &= \left\lfloor 2^{-\ell} \left(a - s(k,\mu)^{-1} h(\mu)\right) \right\rfloor_q \end{aligned}$$

and remark that both $t(k,\mu)$ and $u(k,\mu)$ can easily be computed by the attacker from the publicly known information. Recalling that $0 \leq b \leq q/2^\ell$, we obtain

$$0 \leq \lfloor \alpha t(k,\mu) - u(k,\mu) \rfloor_q < q/2^\ell.$$

And therefore :

$$|\alpha t(k,\mu) - u(k,\mu) - q/2^{\ell+1}|_q \leq q/2^{\ell+1}. \tag{L.2}$$

Thus, an approximation $\mathrm{APP}_{\ell,q}(\alpha t(k,\mu))$ is known. Collecting several relations of this kind for several pairs (k,μ), the problem of recovering the secret key α is thus a HNP in which the distribution of the multiplier $t(k,\mu)$ is not necessarily perfectly uniform, and which at first sight seems hard to study. This problem of recovering will be called the DSA–HNP in the rest of the paper.

To solve the DSA–HNP, we apply a lattice-based algorithm proposed by Boneh and Venkatesan in [48], which relies on a simple reduction from the HNP to the CVP. This polynomial-time algorithm, which we will call BV, is again based on Babai's CVP approximation algorithm [17]. It provably solves the HNP when $\ell \geq \log^{1/2} q + \log\log q$. That result is often cited as the only positive application known of the LLL algorithm, because it enabled Boneh and Venkatesan to establish in [48] some results on the bit-security of the Diffie–Hellman key exchange and related cryptographic schemes. However, in the latter application, the distribution of the multipliers t is not perfectly uniform, making some of the statements of [48] incorrect. This led Gonzáles Vasco and Shparlinski [124] to extend results on the BV algorithm to the case where t is randomly selected from a subgroup of $\mathbb{F}_q^*$, to obtain rigorous statements on the bit-security of the Diffie–Hellman key exchange and related schemes (see also [125]).

In the DSA–HNP as well, the distribution of the multiplier $t(k,\mu)$ is not necessarily perfectly uniform. Hence, we present another extension of the results of [48] on the BV algorithm using the notion of discrepancy, in the spirit of that of [124, 125]. To achieve the proof of our attack, we show using exponential sum techniques that the DSA signatures follow some kind of uniform distribution.

L.1.5 Structure of the paper and notation

The paper is organised as follows. In Section L.2, we review a few facts on lattices and the HNP and we present three extensions of [48, Theorem 1] where the multipliers can have imperfect uniform distribution. In Section L.3, we obtain uniformity results on the distribution of DSA signatures, which might be of independent interest. Finally, in Section L.4, we collect the aforementioned results and apply it to DSA.

Throughout the paper the implied constants in symbols 'O' may occasionally, where obvious, depend on the small positive parameter ε and are absolute otherwise ; they all are effective and can be explicitly evaluated.

We use $[\alpha, \beta]$ and $]\alpha, \beta[$ to denote the closed and open intervals, respectively.

As usual, $\Pr(\mathcal{E})$ denotes the probability of an event $\mathcal{E}$.

For a real x, $\lfloor x \rfloor$ denotes the integer part of x, that is the integer n such that $n \leq x < n+1$. $\lceil x \rceil$ is the integer n such that $n \geq x > n-1$.

L.2 Lattices and the Hidden Number Problem

L.2.1 Background on lattices

As in [48], our results rely on rounding techniques in lattices. We briefly review a few results and definitions. For general references on lattice theory and its important cryptographic applications, we refer to the recent surveys [266, 267].

Let $\{\mathbf{b}_1, \ldots, \mathbf{b}_s\}$ be a set of linearly independent vectors in $\mathbb{R}^s$. The set of vectors

$$L = \left\{ \sum_{i=1}^{s} n_i \mathbf{b}_i \mid n_i \in \mathbb{Z} \right\},$$

is called an s-dimensional full rank lattice. The set $\{\mathbf{b}_1, \ldots, \mathbf{b}_s\}$ is called a *basis* of L, and L is said to be spanned by $\{\mathbf{b}_1, \ldots, \mathbf{b}_s\}$.

A basic lattice problem is the closest vector problem (CVP) : given a basis of a lattice L in $\mathbb{R}^s$ and a target $\mathbf{u} \in \mathbb{R}^s$, find a lattice vector $\mathbf{v} \in L$ which minimises the Euclidean norm $\|\mathbf{u} - \mathbf{v}\|$ among all lattice vectors. The CVP generally refers to the Euclidean norm, but of course, other norms are possible as well : we denote by CVP_∞ the problem corresponding to the infinity norm. Both the CVP and the CVP_∞ are NP-hard (see [266, 267] for references). We call CVP_∞-oracle any algorithm that solves the CVP exactly.

We use the best CVP approximation polynomial-time result known, which follows from the recent shortest vector algorithm of Ajtai *et al.* [8] and Kannan's reduction from approximating the CVP to approximating the shortest vector problem [177] :

Lemma L.1 *For any constant $\gamma > 0$, there exists a randomized polynomial-time algorithm which, given a lattice L and a vector $\mathbf{r} \in \mathbb{Q}^s$, finds a lattice vector $\mathbf{v}$ satisfying with probability exponentially close to 1 the inequality*

$$\|\mathbf{v} - \mathbf{r}\| \leq 2^{\gamma s \log\log s / \log s} \min \{\|\mathbf{z} - \mathbf{r}\|, \quad \mathbf{z} \in L\}.$$

Proof. By taking $k = \lceil c_1 \log n \rceil$ in [8, Corollary 15] where $c_1 > 0$ is a sufficiently large constant, we obtain a randomized polynomial-time algorithm which approximates the shortest vector within $2^{c_2 s \log\log s / \log s}$ for any constant $c_2 > 0$. Besides, Kannan proved in [177, Section 7] that any algorithm approximating the shortest vector problem to within a non-decreasing function $f(s)$ can be used to approximate CVP to within $s^{3/2} f(s)^2$. Since the number of calls of the algorithm remains polynomial, one obtains the desired statement. □

The best deterministic polynomial-time algorithm known for the problem has a slightly larger approximation factor $2^{\eta s \log^2 \log s / \log s}$, see for instance [236, Section 2.1], or [266, Section 2.4], or [267, Section 2.4]. This result is a combination of Schnorr's generalisation [302] of the lattice basis reduction algorithm of Lenstra, Lenstra and Lovász [205] with the aforementioned reduction of Kannan [177]. In the literature, one often finds a weaker and older result (due to Babai [17]) where the approximation factor is only $2^{s/2}$.

In Lemma L.1, the success probability is exponentially close to 1 : in the rest of the paper, we assume that the probability is at least $1 - 2^{-s^3}$, which we are allowed to because if the probability is at least $1 - 2^{-cs}$ for some constant $c > 0$, we can obtain the probability $1 - 2^{-s^3}$ by applying the algorithm a polynomial number of times.

L.2.2 The Hidden Number Problem

We sketch the Boneh and Venkatesan algorithm (BV) proposed in [48] to solve the HNP. Our presentation is slightly different from that of [48]. Consider an instance of the HNP : let $t_1, \ldots, t_d$ be chosen uniformly and independently at random in $\mathbb{F}_q^*$, and $u_i = \mathrm{APP}_{\ell,q}(\alpha t_i)$. Given $t_1, \ldots, t_d$, $u_1, \ldots, u_d$, ℓ, and q, we wish to find the hidden number α. Recall that by definition, $|u_i - \alpha t_i|_q \leq q/2^{\ell+1}$. The BV algorithm is based on a lattice interpretation of those d inequalities : a vector derived from the u_i's is exceptionally close to a particular lattice vector related to the hidden number α. This is done by considering the $(d+1)$-dimensional lattice $L(q, \ell, t_1, \ldots, t_d)$ spanned by the rows of the following matrix :

$$\begin{pmatrix} q & 0 & \cdots & 0 & 0 \\ 0 & q & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \ldots & 0 & q & 0 \\ t_1 & \ldots & \ldots & t_d & 1/2^{\ell+1} \end{pmatrix}. \tag{L.3}$$

The inequality $|u_i - \alpha t_i|_q \leq q/2^{\ell+1}$ implies the existence of an integer h_i such that :

$$|u_i - \alpha t_i - q h_i| \leq q/2^{\ell+1}. \tag{L.4}$$

Notice that the row vector $\mathbf{h} = (\alpha t_1 + q h_1, \ldots, \alpha t_d + q h_d, \alpha/2^{\ell+1})$ belongs to $L(q, \ell, t_1, \ldots, t_d)$, since it can be obtained by multiplying the last row vector by α and then subtracting appropriate multiples of the first d row vectors. Since the last coordinate of this vector discloses the hidden number α, we call $\mathbf{h}$ the *hidden vector*. The hidden vector is very close to the row vector $\mathbf{u} = (u_1, \ldots, u_d, 0)$. Indeed, by (L.4) and $0 \leq \alpha < q$, we have :

$$\|\mathbf{h} - \mathbf{u}\|_\infty \leq q/2^{\ell+1}.$$

The choice of the $(d+1) \times (d+1)$ entry in the matrix (L.3) was made to balance the size of the entries of $\mathbf{h} - \mathbf{u}$.

The BV algorithm applies Babai's nearest plane algorithm [17] to the lattice $L(q, \ell, t_1, \ldots, t_d)$ and the target vector $\mathbf{u}$, which of course can both be built from available information. This yields a lattice point $\mathbf{v}$ that must satisfy :

$$\|\mathbf{u} - \mathbf{v}\| \leq 2^{(d+1)/4} \|\mathbf{u} - \mathbf{h}\| \leq (d+1)^{1/2} 2^{(d+1)/4} q/2^{\ell+1}.$$

Thus, the lattice vector $\mathbf{h} - \mathbf{v}$ satisfies :

$$\|\mathbf{h} - \mathbf{v}\|_\infty \leq \|\mathbf{h} - \mathbf{u}\|_\infty + \|\mathbf{u} - \mathbf{v}\| \leq \frac{q\left(1 + (d+1)^{1/2} 2^{(d+1)/4}\right)}{2^{\ell+1}}.$$

This means, that if ℓ is not too small, $\mathbf{h} - \mathbf{v}$ is a very short lattice vector. Intuitively, only very particular lattice vectors should have infinity norm less than $q/2^{1+\eta}$. The following lemma (which is

actually the core of [48, Theorem 5]) formalises this intuition by characterizing all short vectors in $L(q, \ell, t_1, \ldots, t_d)$:

Lemma L.2 *Let α be a fixed integer in the range $[1, q-1]$ and let $\ell \geq \eta > 0$. Choose integers $t_1, \ldots, t_d$ uniformly and independently at random in the range $[1, q-1]$. Then with probability $P \geq 1 - q/2^{d\eta}$, all vectors $\mathbf{w}$ in $L(q, \ell, t_1, \ldots, t_d)$ such that $\|\mathbf{w}\|_\infty \leq q/2^{1+\eta}$ are of the form*

$$\mathbf{w} = (0, \ldots, 0, \beta/2^{\ell+1}),$$

where $\beta \equiv 0 \pmod{q}$.

Proof. Let $\mathbf{w} \in L(q, \ell, t_1, \ldots, t_d)$. By definition of the lattice, there exist integers $\beta, z_1, \ldots, z_d$ such that

$$\mathbf{w} = (\beta t_1 - z_1 q, \ldots, \beta t_d - z_d q, \beta/2^{\ell+1}). \tag{L.5}$$

If $\beta \equiv 0 \pmod{q}$, then each $\beta t_i - z_i q$ is a multiple of q, and therefore, $\|\mathbf{w}\|_\infty \leq q/2^{1+\eta}$ implies that each $\beta t_i - z_i q$ is zero, so that :

$$\mathbf{w} = (0, \ldots, 0, \beta/2^{\ell+1}).$$

Hence, to achieve the proof of the lemma, it suffices to prove that the probability P that there exists an integer $\beta \not\equiv 0 \pmod{q}$ and integers $z_1, \ldots, z_d$ such that $\|\mathbf{w}\|_\infty \leq q/2^{1+\eta}$ (where $\mathbf{w}$ is defined by (L.5)), is less than $q/2^{d\eta}$.

For any $\beta \not\equiv 0 \pmod{q}$, denote by $E(\beta)$ the event that there exist integers $z_1, \ldots, z_d$ such that $\|\mathbf{w}\|_\infty \leq q/2^{1+\eta}$. Obviously, if $|\beta t_i - z_i q| \leq q/2^{1+\eta}$ then $|\beta t_i|_q \leq q/2^{1+\eta}$ (recall the definition $|n|_q = \min\{\lfloor n \rfloor_q, q - \lfloor n \rfloor_q\}$ in Section L.1.4). Hence, the probability of $E(\beta)$ is less than the probability that for all i, $|\beta t_i|_q < q/2^{1+\eta}$. It follows by independence of the t_i's, that $\Pr(E(\beta)) \leq p(\beta, q)^d$, if $p(\beta, q)$ denotes the probability that $|\beta t|_q \leq q/2^{1+\eta}$ for t uniformly chosen in $[1, q-1]$. By definition,

$$p(\beta, q) = 1 - \Pr\left(q/2^{1+\eta} < \lfloor \beta t \rfloor_q < q - q/2^{1+\eta}\right).$$

Since $\beta \not\equiv 0 \pmod{q}$, and t is uniformly chosen in $[1, q-1]$, $\lfloor \beta t \rfloor_q$ is uniformly chosen in $[1, q-1]$, implying that :

$$\begin{aligned} p(\beta, q) &= 1 - \frac{\lfloor q - q/2^{1+\eta} \rfloor - \lceil q/2^{1+\eta} \rceil + 1}{q-1} \\ &\leq 1 - \frac{q - q/2^\eta + 1}{q-1} = \frac{q - 2^{1+\eta}}{2^\eta (q-1)} \leq \frac{1}{2^\eta}. \end{aligned}$$

Hence, the probability of $E(\beta)$ is less than $1/2^{d\eta}$. Finally, notice that $E(\beta)$ occurs only if $E(\lfloor \beta \rfloor_q)$ occurs, so that

$$P \leq \sum_{\beta=1}^{q-1} \Pr(E(\beta)),$$

from which the result follows. □

Now, if $\mathbf{h} - \mathbf{v}$ is sufficiently short to satisfy the condition of Lemma L.2, the hidden number α can easily be derived from the last coordinate of $\mathbf{v}$ because of $\mathbf{h}$. A straightforward computation shows that the condition is satisfied for all sufficiently large q, if $\ell = \lceil \log^{1/2} q \rceil + \lceil \log \log q \rceil$ and $d = 2\lceil \log^{1/2} q \rceil$, using Babai's CVP approximation algorithm [17], and not Lemma L.1. Thus, with these parameters, the polynomial-time BV algorithm recovers with high probability α, which is formalised by [48, Theorem 1]. Of course the value of ℓ can be slightly decreased if one uses Lemma L.1 instead of Babai's algorithm.

L.2.3 Extending the Hidden Number Problem

As we have seen, the correctness of the BV algorithm relies on Lemma L.2. We would like to generalise Lemma L.2 to cases where the multiplier t has not necessarily perfectly uniform distribution. A simple look at the proof of Lemma L.2 shows that the distribution of t only intervenes in the upper bounding of the probability $p(\beta, q)$ that $|\beta t|_q \leq q/2^{1+\eta}$. We need $p(\beta, q)$ to be less than a constant strictly less than 1. We rewrite $p(\beta, q)$ as :

$$\begin{aligned} p(\beta, q) &= 1 - \Pr\left(q/2^{1+\eta} < \lfloor \beta t \rfloor_q < q - q/2^{1+\eta}\right) \\ &= 1 - \Pr\left(\frac{\lfloor \beta t \rfloor_q}{q} \in \left]\frac{1}{2^{1+\eta}}, 1 - \frac{1}{2^{1+\eta}}\right[\right). \end{aligned}$$

This suggests to use the classical notion of discrepancy [85, 195, 272]. Recall that the *discrepancy* $\mathcal{D}(\Gamma)$ of a sequence $\Gamma = \{\gamma_1, \ldots, \gamma_N\}$ of N elements of the interval $[0, 1]$ is defined as

$$\mathcal{D}(\Gamma) = \sup_{J \subseteq [0,1]} \left| \frac{A(J, N)}{N} - |J| \right|,$$

where the supremum is extended over all subintervals J of $[0, 1]$, $|J|$ is the length of J, and $A(J, N)$ denotes the number of points γ_n in J for $0 \leq n \leq N-1$. The term $\lfloor \beta t \rfloor_q / q$ in our expression of $p(\beta, q)$ suggests the following definition. We say that a finite sequence $\mathcal{T}$ of integers is Δ-*homogeneously distributed modulo* q if for any integer a coprime with q the discrepancy of the sequence $\{\lfloor at \rfloor_q / q\}_{t \in \mathcal{T}}$ is at most Δ. Indeed, if t is now chosen uniformly at random from a Δ-homogeneously distributed modulo q sequence $\mathcal{T}$, then by definition :

$$\Pr\left(\frac{\lfloor \beta t \rfloor_q}{q} \in \left]\frac{1}{2^{1+\eta}}, 1 - \frac{1}{2^{1+\eta}}\right[\right) \geq \left]\left]\frac{1}{2^{1+\eta}}, 1 - \frac{1}{2^{1+\eta}}\right[\right[- \Delta = 1 - \frac{1}{2^{\eta}} - \Delta.$$

This obviously leads to the following generalization of Lemma L.2 :

Lemma L.3 *Let α be a fixed integer in the range $[1, q-1]$ and let $\ell \geq \eta > 0$. Choose integers $t_1, \ldots, t_d$ uniformly and independently at random from a Δ-homogeneously distributed modulo q sequence $\mathcal{T}$. Then with probability at least $1 - q(1/2^{\eta} + \Delta)^d$, all $\mathbf{w}$ in $L(q, \ell, t_1, \ldots, t_d)$ such that $\|\mathbf{w}\|_\infty \leq q/2^{1+\eta}$ are of the form*

$$\mathbf{w} = (0, \ldots, 0, \beta/2^{\ell+1}),$$

where $\beta \equiv 0 \pmod{q}$.

Using Lemma L.3, we easily obtain a generalization of [48, Theorem 1] :

Lemma L.4 *Let $\omega > 0$ be an arbitrary absolute constant. For a prime q, define*

$$\ell = \left\lceil \omega \left(\frac{\log q \log\log\log q}{\log\log q} \right)^{1/2} \right\rceil \qquad \text{and} \qquad d = \lceil 3 \log q / \ell \rceil .$$

Let $\mathcal{T}$ be a $2^{-\ell}$-homogeneously distributed modulo q sequence of integer numbers. There exists a probabilistic polynomial-time algorithm $\mathcal{A}$ such that for any fixed integer α in the interval $[0, q-1]$, given as input a prime q, d integers $t_1, \ldots, t_d$ and d rationals

$$u_i = APP_{\ell,q}(\alpha t_i), \qquad i = 1, \ldots, d,$$

its output satisfies for sufficiently large q

$$\Pr\left[\mathcal{A}\left(q, t_1, \ldots, t_d; u_1, \ldots, u_d\right) = \alpha\right] \geq 1 - q^{-1}$$

where the probability is taken over all $t_1, \ldots, t_d$ chosen uniformly and independently at random from the elements of $\mathcal{T}$ and all coin tosses of the algorithm $\mathcal{A}$.

Proof. We simply follow the sketch of Section L.2.2. The algorithm $\mathcal{A}$ applies the algorithm of Lemma L.1 with $s = d + 1$ and $\gamma = \omega/10$ to the lattice $L(q, \ell, t_1, \ldots, t_d)$ spanned by the rows of the matrix (L.3), and the target vector $\mathbf{u} = (u_1, \ldots, u_d, 0)$. The algorithm $\mathcal{A}$ outputs $\lfloor \beta \rfloor_q$ where $\beta/2^{\ell+1}$ is the last entry of the vector $\mathbf{v}$ yielded by the algorithm of Lemma L.1.

We now analyze the correctness of $\mathcal{A}$. Letting the lattice vector $\mathbf{h} = (\lfloor \alpha t_1 \rfloor_q, \ldots, \lfloor \alpha t_d \rfloor_q, \alpha/2^{\ell+1})$, we see from Lemma L.1 that the lattice vector $\mathbf{v}$ must satisfy with probability at least $1 - 2^{-d^3}$:

$$\|\mathbf{u} - \mathbf{v}\| \leq 2^{\gamma(d+1)\log\log(d+1)/\log(d+1)}\|\mathbf{u} - \mathbf{h}\| \leq 2^{\omega d \log\log d/9\log d}\|\mathbf{u} - \mathbf{h}\|.$$

Since $\|\mathbf{h} - \mathbf{u}\|_\infty < q/2^{\ell+1}$, we obtain :

$$\|\mathbf{h} - \mathbf{v}\|_\infty \leq q2^{-\ell-1+\omega d \log\log d/9\log d}.$$

One easily verifies that

$$\omega d \log\log d/9\log d \leq \ell/2$$

for sufficiently large q. Thus, $\mathbf{h} - \mathbf{v}$ satisfies the assumption of Lemma L.3 with $\eta = \ell/2 + 1$ provided that q is large enough. Therefore, $\mathcal{A}$ outputs the hidden number α with probability at least

$$1 - q(2^{-\eta} + 2^{-\ell})^d - 2^{-d^3} \geq 1 - q2^{-d(\eta-1)} - 2^{-d^3} \geq 1 - q^{-1}$$

and the result follows. □

Since our results apply lattice reduction, it is interesting to know how our results are affected if ideal lattice reduction is available, due to the well-known experimental fact that lattice basis reduction algorithms behave much better than theoretically guaranteed, despite NP-hardness results for most lattice problems (see [266, 267]). The methodology of Section L.2.2 is more adapted to the infinity norm than the Euclidean norm, so the following result is an improved version of Lemma L.4, when a CVP_∞-oracle is available :

Lemma L.5 *Let $\eta > 0$ be fixed. For a prime q, define $\ell = 1 + \eta$, and*

$$d = \left\lceil \frac{8}{3}\eta^{-1}\log q \right\rceil.$$

Let $\mathcal{T}$ be a $f(q)$-homogeneously distributed modulo q sequence of integer numbers, where $f(q)$ is any function with $f(q) \to 0$ as $q \to \infty$. There exists a deterministic polynomial-time algorithm $\mathcal{A}$ using a CVP_∞-oracle (in dimension $d + 1$) such that for any fixed integer α in the interval $[0, q - 1]$, given as input a prime q, d integers $t_1, \ldots, t_d$ and d rationals

$$u_i = APP_{\ell,q}(\alpha t_i), \qquad i = 1, \ldots, d,$$

its output satisfies for sufficiently large q

$$\Pr\left[\mathcal{A}\left(q, t_1, \ldots, t_d; u_1, \ldots, u_d\right) = \alpha\right] \geq 1 - \frac{1}{q}$$

where the probability is taken over all $t_1, \ldots, t_d$ chosen uniformly and independently at random from the elements of $\mathcal{T}$.

Proof. We follow the proof of Lemma L.4, and replace the algorithm of Lemma L.1 by a CVP_∞-oracle. This time, we have :

$$\|\mathbf{h} - \mathbf{v}\|_\infty \leq \frac{q}{2^\ell} = \frac{q}{2^{1+\eta}}.$$

Applying Lemma L.3, we obtain that the probability of success of the algorithm is at least $1-q(1/2^{\eta}+f(q))^d$. For sufficiently large q we have $1/2^{\eta}+f(q) \leq 1/2^{3\eta/4}$, so that :

$$(1/2^{\eta}+f(q))^d \leq 1/2^{3d\eta/4} \leq 1/2^{2\log q},$$

from which the result follows. □

It is worth noting that in Lemma L.5 the assumption on the distribution of $\mathcal{T}$ is quite weak, which explains why in practice, attacks based on variants of the HNP are likely to work (as illustrated in [159, 254]). In fact, only a non-trivial upper bound on the number of fractions $\lfloor at \rfloor_q/q$, $t \in \mathcal{T}$ in a given interval is really needed (rather than the much stronger property of homogeneous distribution modulo q).

We remark that the choice of parameters in DSA and ECDSA is based on the assumption that any attack should take time of order at least $q^{1/2}$. Thus any attack requiring significantly lesser time could still be a threat. Interestingly, one can obtain a combination of Lemma L.4 and Lemma L.5 which leads to such an attack

Lemma L.6 *For a prime q, define $\ell = \lfloor \log\log q \rfloor$, and*

$$d = \left\lceil 4\frac{\log q}{\log\log q} \right\rceil.$$

Let $\mathcal{T}$ be a $2^{-\ell}$-homogeneously distributed modulo q sequence of integer numbers. There exists a probabilistic algorithm $\mathcal{A}$ which runs in time $q^{O(1/\log\log q)}$ and such that for any fixed integer α in the interval $[0, q-1]$, given as input a prime q, d integers $t_1, \ldots, t_d$ and d rationals

$$u_i = APP_{\ell,q}(\alpha t_i), \qquad i = 1, \ldots, d,$$

its output satisfies for sufficiently large q

$$\Pr\left[\mathcal{A}\left(q, t_1, \ldots, t_d; u_1, \ldots, u_d\right) = \alpha\right] \geq 1 - \frac{1}{q}$$

where the probability is taken over all $t_1, \ldots, t_d$ chosen uniformly and independently at random from the elements of $\mathcal{T}$.

Proof. We repeat the arguments of the proof of Lemma L.4 however we use the closest vector approximation algorithm of [8, Corollary 16] in the corresponding place which runs in time at most $2^{O(d)} = q^{O(1/\log\log q)}$. Following the same calculations as in the proof of Lemma L.4, we obtain that the probability of failure does not exceed

$$q\left(d^{1/2}2^{-\ell+O(1)}\right)^d \leq q^{-1}$$

for sufficiently large q. □

L.3 Distribution of Signatures Modulo q

From the previous section, it remains to study the distribution of signatures. In this section, we obtain uniformity results on the distribution of $t(k,\mu)$ modulo q, which might be of independent interest.

L.3.1 Preliminaries on exponential sums

Let $\mathbf{e}_p(z) = \exp(2\pi iz/p)$ and $\mathbf{e}_q(z) = \exp(2\pi iz/q)$. One of our main tools is the *Weil bound* on exponential sums with rational functions which we present in the following form given by [245, Theorem 2].

Lemma L.7 *For any polynomials $g(X), f(X) \in \mathbb{F}_q[X]$ such that the rational function $F(X) = f(X)/g(X)$ is not constant on $\mathbb{F}_q$, the bound*

$$\left| \sum_{\lambda \in \mathbb{F}_q}^{*} \mathbf{e}_q\left(F(\lambda)\right) \right| \le \left(\max\{\deg g\,, \deg f\} + u - 2\right) q^{1/2} + \delta$$

holds, where $\sum^*$ means that the summation is taken over all $\lambda \in \mathbb{F}_q$ which are not poles of $F(X)$ and

$$(u, \delta) = \begin{cases} (v, 1), & \text{if } \deg f \le \deg g, \\ (v+1, 0), & \text{if } \deg f > \deg g, \end{cases}$$

and v is the number of distinct zeros of $g(X)$ in the algebraic closure of $\mathbb{F}_q$.

We also need some estimates from [189] of exponential sums with exponential functions. In fact we present them in the somewhat simplified forms similar to those given in [124].

Lemma L.8 *For any $\varepsilon > 0$ there exists $\delta > 0$ such that for any element $g \in \mathbb{F}_p$ of multiplicative order $T \ge p^{1/3+\varepsilon}$ the bound*

$$\max_{\gcd(c,p)=1} \left| \sum_{x=0}^{T-1} \mathbf{e}_p\left(cg^x\right) \right| \le T^{1-\delta}$$

holds.

Proof. The result follows immediately from the estimate

$$\max_{\gcd(c,p)=1} \left| \sum_{x=0}^{T-1} \mathbf{e}_p\left(cg^x\right) \right| = O\left(B(T,p)\right),$$

where

$$B(T,p) = \begin{cases} p^{1/2}, & \text{if } T \ge p^{2/3}; \\ p^{1/4}T^{3/8}, & \text{if } p^{2/3} > T \ge p^{1/2}; \\ p^{1/8}T^{5/8}, & \text{if } p^{1/2} > T \ge p^{1/3}; \end{cases}$$

which is essentially [189, Theorem 3.4]. □

Lemma L.9 *Let Q be a sufficiently large integer. For any $\varepsilon > 0$ there exists $\delta > 0$ such that for all primes $p \in [Q, 2Q]$, except at most $Q^{5/6+\varepsilon}$ of them, and any element $g_{p,T} \in \mathbb{F}_p$ of multiplicative order $T \ge p^{\varepsilon}$ the bound*

$$\max_{\gcd(c,p)=1} \left| \sum_{x=0}^{T-1} \mathbf{e}_p\left(cg_{p,T}^x\right) \right| \le T^{1-\delta}$$

holds.

Proof. For each integer $T \geq 1$ and for each prime $p \equiv 1 \pmod{T}$ we fix an element $g_{p,T}$ of multiplicative order T. Then [189, Theorem 5.5] claims that for any $U > 1$ and any integer $\nu \geq 2$, for all primes $p \equiv 1 \pmod{T}$ except at most $O(U/\log U)$ of them, the bound

$$\max_{\gcd(c,p)=1} \left| \sum_{x=0}^{T-1} \mathbf{e}_p\left(cg_{p,T}^x\right) \right| = O\left(Tp^{1/2\nu^2}\left(T^{-1/\nu} + U^{-1/\nu^2}\right)\right),$$

holds. We remark that the value of the above exponential sum does not depend on the particular choice of the element $g_{p,T}$.

Taking

$$\nu = \left\lfloor \frac{1}{\varepsilon} \right\rfloor + 1, \qquad U = Q^{1/2+\varepsilon/3}, \qquad V = Q^{1/3+\varepsilon/3}$$

after simple computation we obtain that there exists some $\delta > 0$, depending only on ε, such that for any fixed $T \geq Q^{\varepsilon}$ the bound

$$\max_{\gcd(c,p)=1} \left| \sum_{x=0}^{T-1} \mathbf{e}_p\left(cg_{p,T}^x\right) \right| \leq T^{1-\delta},$$

holds for all except $O(U/\log U)$ primes $p \equiv 1 \pmod{T}$ in the interval $p \in [Q, 2Q]$. As it follows from Lemma L.8, a similar bound also holds for $T \geq V$. So the total number of exceptional primes p for which the bound of the lemma does not hold for at least one $T \geq p^{\varepsilon} \geq Q^{\varepsilon}$ is $O(UV) = O\left(Q^{5/6+2\varepsilon/3}\right)$. Thus for sufficiently large Q we obtain the desired result. □

L.3.2 Distribution of $r(k)$

For any integer $\rho \in [0, q-1]$, we denote by $N(\rho)$ the number of solutions of the equation

$$r(k) = \rho, \qquad k \in [1, q-1].$$

Lemma L.10 *Let Q be a sufficiently large integer. The following statement holds with $\vartheta = 1/3$ for all primes $p \in [Q, 2Q]$, and with $\vartheta = 0$ for all primes $p \in [Q, 2Q]$ except at most $Q^{5/6+\varepsilon}$ of them. For any $\varepsilon > 0$ there exists $\delta > 0$ such that for any element $g \in \mathbb{F}_p$ of multiplicative order $q \geq p^{\vartheta+\varepsilon}$ the bound*

$$N(\rho) = O\left(q^{1-\delta}\right), \qquad \rho \in [0, q-1],$$

holds.

Proof. Let

$$L = \left\lfloor \frac{p-\rho-1}{q} \right\rfloor.$$

We remark that $N(\rho)$ is the number of solutions $k \in [1, q-1]$ of the congruence

$$g^k \equiv qx + \rho \pmod{p}, \qquad k \in [1, q-1],\ x \in [0, L].$$

Using the identity (see Exercise 11.a in [348, Chapter 3])

$$\sum_{c=0}^{p-1} \mathbf{e}_p(cu) = \begin{cases} 0, & \text{if } u \not\equiv 0 \pmod{p}; \\ p, & \text{if } u \equiv 0 \pmod{p}; \end{cases}$$

we obtain

$$\begin{aligned} N(\rho) &= \frac{1}{p} \sum_{k=1}^{q-1} \sum_{x=0}^{L} \sum_{c=0}^{p-1} \mathbf{e}_p\left(c\left(g^k - qx - \rho\right)\right) \\ &= \frac{1}{p} \sum_{c=0}^{p-1} \mathbf{e}_p(-c\rho) \sum_{k=1}^{q-1} \mathbf{e}_p\left(cg^k\right) \sum_{x=0}^{L} \mathbf{e}_p(-cqx). \end{aligned}$$

Separating the term

$$\frac{(q-1)(L+1)}{p} \le \frac{(q-1)(p/q+1)}{p} \le 2$$

corresponding to $c = 0$, we derive

$$\begin{aligned} N(\rho) &\le 2 + \frac{1}{p}\sum_{c=1}^{p-1}\left|\sum_{k=1}^{q-1}\mathbf{e}_p\left(cg^k\right)\right|\left|\sum_{x=0}^{L}\mathbf{e}_p\left(-cqx\right)\right| \\ &\le 2 + \frac{1}{p}\sum_{c=1}^{p-1}\left|\sum_{k=1}^{q-1}\mathbf{e}_p\left(cg^k\right)\right|\left|\sum_{x=0}^{L}\mathbf{e}_p\left(cqx\right)\right|. \end{aligned}$$

Combining Lemmas L.8 and L.9 to estimate the sum over $k \in [1, q-1]$ (certainly the missing term corresponding to $k = 0$ does not change the order of magnitude of this sum) with the estimate

$$\sum_{c=1}^{p-1}\left|\sum_{x=0}^{L}\mathbf{e}_p\left(cqx\right)\right| = \sum_{c=1}^{p-1}\left|\sum_{x=0}^{L}\mathbf{e}_p\left(cx\right)\right| = O(p\log p),$$

see Exercise 11.c in [348, Chapter 3], we obtain the desired result. □

In particular, denote by $\mathcal{S}$ the set of pairs $(k,\mu) \in [1, q-1] \times \mathcal{M}$ with $s(k,\mu) \ne 0$ (that is, the set of pairs (k,μ) for which the congruence (L.1) holds and thus $t(k,\mu)$ is defined). Then

$$|\mathcal{S}| = q|\mathcal{M}|\left(1 + O\left(q^{-\delta}\right)\right) \tag{L.6}$$

for all p and q satisfying the conditions of Lemma L.10.

L.3.3 Distribution of $t(k,\mu)$

For a hash function $h : \mathcal{M} \to \mathbb{F}_q$ we also denote by W the number of pairs $(\mu_1, \mu_2) \in \mathcal{M}^2$ with $h(\mu_1) = h(\mu_2)$. Thus, $W/|\mathcal{M}|^2$ is a probability of a *collision* and our results are nontrivial under a reasonable assumption that this probability is of order of magnitude close to $1/q$.

First of all, we need to estimate exponential sums with the multipliers $t(k,\mu)$:

Lemma L.11 *Let Q be a sufficiently large integer. The following statement holds with $\vartheta = 1/3$ for all primes $p \in [Q, 2Q]$, and with $\vartheta = 0$ for all primes $p \in [Q, 2Q]$ except at most $Q^{5/6+\varepsilon}$ of them. For any $\varepsilon > 0$ there exists $\delta > 0$ such that for any element $g \in \mathbb{F}_p$ of multiplicative order $q \ge p^{\vartheta+\varepsilon}$ the bound*

$$\max_{\gcd(c,q)=1}\left|\sum_{(k,\mu)\in\mathcal{S}}\mathbf{e}_q\left(ct(k,\mu)\right)\right| = O\left(W^{1/2}q^{3/2-\delta}\right)$$

holds.

Proof. For each $\mu \in \mathcal{M}$ we denote by $\mathcal{K}_\mu$ the set of $k \in [1, q-1]$ for which $(k,\mu) \in \mathcal{S}$.

We consider a $c_0 \in \mathbb{F}_q^*$ corresponding to the largest exponential sum of interest to us. We denote

$$\sigma = \left|\sum_{(k,\mu)\in\mathcal{S}}\mathbf{e}_q\left(c_0t(k,\mu)\right)\right| = \max_{\gcd(c,q)=1}\left|\sum_{(k,\mu)\in\mathcal{S}}\mathbf{e}_q\left(ct(k,\mu)\right)\right|.$$

We have

$$\sigma \le \sum_{\mu\in\mathcal{M}}\left|\sum_{k\in\mathcal{K}_\mu}\mathbf{e}_q\left(c_0t(k,\mu)\right)\right|.$$

For $\lambda \in \mathbb{F}_q$ we denote by $H(\lambda)$ the number of $\mu \in \mathcal{M}$ with $h(\mu) = \lambda$. We also define the integer $a \in [1, q-1]$ by the congruence $a \equiv 2^{-\ell} c_0 \pmod q$. Then

$$\sigma \le \sum_{\lambda \in \mathbb{F}_q} H(\lambda) \left| \sum_{\substack{k=1 \\ \alpha r(k) \not\equiv -\lambda \pmod q}}^{q-1} \mathbf{e}_q\left(a \frac{kr(k)}{\lambda + \alpha r(k)}\right) \right|.$$

Applying the Cauchy inequality we obtain

$$\sigma^2 \le \sum_{\lambda \in \mathbb{F}_q} H(\lambda)^2 \sum_{\lambda \in \mathbb{F}_q} \left| \sum_{\substack{k=1 \\ \alpha r(k) \not\equiv -\lambda \pmod q}}^{q-1} \mathbf{e}_q\left(a \frac{kr(k)}{\lambda + \alpha r(k)}\right) \right|^2. \tag{L.7}$$

First of all we remark that

$$\sum_{\lambda \in \mathbb{F}_q} H(\lambda)^2 = W. \tag{L.8}$$

Furthermore,

$$\begin{aligned}
&\sum_{\lambda \in \mathbb{F}_q} \left| \sum_{\substack{k=1 \\ \alpha r(k) \not\equiv -\lambda \pmod q}}^{q-1} \mathbf{e}_q\left(a \frac{kr(k)}{\lambda + \alpha r(k)}\right) \right|^2 \\
&\quad = \sum_{\lambda \in \mathbb{F}_q} \sum_{\substack{k=1 \\ \alpha r(k) \not\equiv -\lambda \pmod q}}^{q-1} \\
&\qquad\qquad \times \sum_{\substack{m=1 \\ \alpha r(m) \not\equiv -\lambda \pmod q}}^{q-1} \mathbf{e}_q\left(a\left(\frac{kr(k)}{\lambda + \alpha r(k)} - \frac{mr(m)}{\lambda + \alpha r(m)}\right)\right) \\
&\quad = \sum_{k,m=1}^{q-1} {\sum_{\lambda \in \mathbb{F}_q}}^{*} \mathbf{e}_q\left(a\left(\frac{kr(k)}{\lambda + \alpha r(k)} - \frac{mr(m)}{\lambda + \alpha r(m)}\right)\right),
\end{aligned}$$

where, as in Lemma L.7, the symbol $\sum^*$ means that the summation in the inner sum is taken over all $\lambda \in \mathbb{F}_q$ with

$$\lambda \not\equiv -\alpha r(k) \pmod q \qquad \text{and} \qquad \lambda \not\equiv -\alpha r(m) \pmod q.$$

It is easy to see that if $r(k) \ne r(m)$ then the rational function

$$F_{k,m}(X) = \frac{kr(k)}{X + \alpha r(k)} - \frac{mr(m)}{X + \alpha r(m)}$$

is not constant in $\mathbb{F}_q$. If $r(k) = r(m)$ then

$$F_{k,m}(X) = \frac{(k-m)r(k)}{X + \alpha r(k)}.$$

Thus it is constant only if $k = m$ or $r(k) = r(m) = 0$. From Lemma L.10 we see that the number of such pairs is $O\left(q^{2-2\delta} + q\right)$ for some $\delta > 0$ for which we estimate the sum over λ trivially as q. For other pairs $(k, m) \in [1, q-1]^2$ we use Lemma L.7 getting

$$\begin{aligned}
\sum_{\lambda \in \mathbb{F}_q} \left| \sum_{\substack{k=1 \\ \alpha r(k) \not\equiv -\lambda \pmod q}}^{q-1} \mathbf{e}_q\left(a \frac{kr(k)}{\lambda + \alpha r(k)}\right) \right|^2 &= O\left(\left(q^{2-2\delta} + q\right) q + q^{5/2}\right) \\
&= O\left(q^{3-2\delta}\right)
\end{aligned}$$

(without loss of generality we may assume that $\delta < 1/4$). Substituting this estimate and the identity (L.8) in (L.7), we obtain the desired statement. □

Lemma L.12 *Let Q be a sufficiently large integer. The following statement holds with $\vartheta = 1/3$ for all primes $p \in [Q, 2Q]$, and with $\vartheta = 0$ for all primes $p \in [Q, 2Q]$ except at most $Q^{5/6+\varepsilon}$ of them. For any $\varepsilon > 0$ there exists $\delta > 0$ such that for any element $g \in \mathbb{F}_p$ of multiplicative order $q \geq p^{\vartheta+\varepsilon}$ the sequence $t(k, \mu)$, $(k, \mu) \in \mathcal{S}$, is $2^{-\log^{1/2} q}$-homogeneously distributed modulo q provided that*

$$W \leq \frac{|\mathcal{M}|^2}{q^{1-\delta}}.$$

Proof. Let us fix an integer a coprime with q. According to a general discrepancy bound, given by [272, Corollary 3.11] for the discrepancy D of the set

$$\left\{ \frac{\lfloor at(k,\mu) \rfloor_q}{q} \ : \ (k, \mu) \in \mathcal{S} \right\}$$

we have

$$\begin{aligned} D &\leq \frac{\log q}{|\mathcal{S}|} \max_{\gcd(c,q)=1} \left| \sum_{(k,\mu)\in\mathcal{S}} \mathbf{e}_q \left(c \lfloor at(k,\mu) \rfloor_q \right) \right| \\ &\leq \frac{\log q}{|\mathcal{S}|} \max_{\gcd(c,q)=1} \left| \sum_{(k,\mu)\in\mathcal{S}} \mathbf{e}_q \left(cat(k,\mu) \right) \right| \\ &\leq \frac{\log q}{|\mathcal{S}|} \max_{\gcd(c,q)=1} \left| \sum_{(k,\mu)\in\mathcal{S}} \mathbf{e}_q \left(ct(k,\mu) \right) \right|. \end{aligned}$$

Applying Lemma L.11 and the bound (L.6) we obtain

$$D = O\left(W^{1/2} q^{1/2-\delta} |\mathcal{M}|^{-1} \right) = O(q^{-\delta/2}) = o(2^{-\log^{1/2} q})$$

and the desired result follows. □

L.4 Insecurity of the Digital Signature Algorithm

L.4.1 Theoretical results

It now suffices to collect the previous results. For an integer ℓ we define the oracle $\mathcal{O}_\ell$ which, for any given DSA signature $(r(k), s(k, \mu))$, $k \in [0, q-1]$, $\mu \in \mathcal{M}$, returns the ℓ least significant bits of k. Combining (L.2), Lemma L.4 and Lemma L.12, we obtain :

Theorem L.1 *Let $\omega > 0$ be an arbitrary absolute constant. Let Q be a sufficiently large integer. The following statement holds with $\vartheta = 1/3$ for all primes $p \in [Q, 2Q]$, and with $\vartheta = 0$ for all primes $p \in [Q, 2Q]$ except at most $Q^{5/6+\varepsilon}$ of them. For any $\varepsilon > 0$ there exists $\delta > 0$ such that for any element $g \in \mathbb{F}_p$ of multiplicative order q, where $q \geq p^{\vartheta+\varepsilon}$ is prime, and any hash function h with*

$$W \leq \frac{|\mathcal{M}|^2}{q^{1-\delta}},$$

given an oracle $\mathcal{O}_\ell$ *with*

$$\ell = \left\lceil \omega \left(\frac{\log q \log\log\log q}{\log\log q} \right)^{1/2} \right\rceil,$$

there exists a probabilistic polynomial-time algorithm to find the signer's DSA secret key α *from* $O\left((\log q \log\log q / \log\log\log q)^{1/2}\right)$ *signatures* $(r(k), s(k,\mu))$ *with* $k \in [0, q-1]$ *and* $\mu \in \mathcal{M}$ *selected independently and uniformly at random. The probability of success is at least* $1 - q^{-1}$.

Proof. We choose $k \in [0, q-1]$ and $\mu \in \mathcal{M}$ independently and uniformly at random and ignore pairs $(k,\mu) \notin \mathcal{S}$. It follows from (L.6) that the expected number of choices in order to get d pairs $(k,\mu) \in \mathcal{S}$ is $d + O\left(dq^{-\delta}\right)$ for some $\delta > 0$ depending only on $\varepsilon > 0$.

Now, combining the inequality (L.2), Lemma L.12 and Lemma L.4 we obtain our main result. □

In Section L.5.1, we extend this result to other consecutive bits, such as most significant bits or bits in the middle. The result is essentially the same for most significant bits, while one requires twice as many bits for arbitrary consecutive bits.

If a CVP_∞-oracle is available, the number ℓ of required bits can be decreased to 2 due to Lemma L.5. The dimension of the lattice used by the oracle is $d+1$, where the number d of required signatures is $O(\log q)$. More precisely we have :

Theorem L.2 *Let* Q *be a sufficiently large integer. The following statement holds with* $\vartheta = 1/3$ *for all primes* $p \in [Q, 2Q]$, *and with* $\vartheta = 0$ *for all primes* $p \in [Q, 2Q]$ *except at most* $Q^{5/6+\varepsilon}$ *of them. For any* $\varepsilon > 0$ *there exists* $\delta > 0$ *such that for any element* $g \in \mathbb{F}_p$ *of multiplicative order* q, *where* $q \geq p^{\vartheta+\varepsilon}$ *is prime, and any hash function* h *with*

$$W \leq \frac{|\mathcal{M}|^2}{q^{1-\delta}},$$

given an oracle $\mathcal{O}_\ell$ *with* $\ell = 2$ *and a* CVP_∞*-oracle for the dimension* $d+1$ *where*

$$d = \left\lceil \frac{8}{3} \log q \right\rceil,$$

there exists a probabilistic polynomial-time algorithm to find the signer's DSA secret key α *from* d *signatures* $(r(k), s(k,\mu))$ *with* $k \in [0, q-1]$ *and* $\mu \in \mathcal{M}$ *selected independently and uniformly at random. The probability of success is at least* $1 - q^{-1}$.

Accordingly, from Lemma L.6 we derive :

Theorem L.3 *Let* Q *be a sufficiently large integer. The following statement holds with* $\vartheta = 1/3$ *for all primes* $p \in [Q, 2Q]$, *and with* $\vartheta = 0$ *for all primes* $p \in [Q, 2Q]$ *except at most* $Q^{5/6+\varepsilon}$ *of them. For any* $\varepsilon > 0$ *there exists* $\delta > 0$ *such that for any element* $g \in \mathbb{F}_p$ *of multiplicative order* q, *where* $q \geq p^{\vartheta+\varepsilon}$ *is prime, and any hash function* h *with*

$$W \leq \frac{|\mathcal{M}|^2}{q^{1-\delta}},$$

given an oracle $\mathcal{O}_\ell$ *with*

$$\ell = \lceil \log\log q \rceil,$$

there exists a probabilistic algorithm to find the signer's DSA secret key α, *in time* $q^{O(1/\log\log q)}$, *from* $O(\log q / \log\log q)$ *signatures* $(r(k), s(k,\mu))$ *with* $k \in [0, q-1]$ *and* $\mu \in \mathcal{M}$ *selected independently and uniformly at random. The probability of success is at least* $1 - q^{-1}$.

L.4.2 Experimental results

We report experimental results on the attack obtained with the NTL library [319] (see also [254]). The running time is less than half an hour for a number of signatures d less than a hundred, on a 500 MHz DEC Alpha. We used a 160-bit prime q, and a 512-bit prime p. For each choice of parameters size, we run the attack several times on newly generated parameters (including the prime q and the multipliers of the DSA–HNP). Each trial is referred as a *sample*. Using Babai's nearest plane algorithm [17] and Schnorr's Korkine-Zolotarev reduction [302, 307] with blocksize 20, we could break DSA with ℓ as low as $\ell = 4$ and $d = 70$. More precisely, the method always worked for $\ell = 5$ (a hundred samples). For $\ell = 4$, it worked 90% of the time over 100 samples. For $\ell = 3$, it always failed on about 100 samples, even with $d = 100$.

We made additional experiments with the well-known embedding strategy (see [266, 267]) and Schnorr's improved lattice reduction [302, 307] to solve the CVP. The embedding strategy heuristically reduces the CVP to the lattice shortest vector problem. More precisely, if the CVP-instance is given by the vector $\mathbf{a} = (a_1, \ldots, a_d)$ and a d-dimensional lattice spanned by the row vectors $\mathbf{b}_i = (b_{i,1}, \ldots, b_{i,d})$ with $1 \leq i \leq d$, the embedding strategy builds the lattice L spanned by the rows of the following matrix :

$$\begin{pmatrix} b_{1,1} & \ldots & b_{1,d} & 0 \\ b_{2,1} & \ldots & b_{2,d} & 0 \\ \vdots & & \vdots & \vdots \\ b_{d,1} & \ldots & b_{d,d} & 0 \\ a_1 & \ldots & a_d & 1 \end{pmatrix}.$$

It is hoped that the shortest vector of L (or one of the vectors of the reduced basis) is of the form $(\mathbf{a} - \mathbf{u}, 1)$ where $\mathbf{u}$ is a sufficiently close lattice vector we are looking for. Using that strategy, we were always able to solve the DSA problem with $\ell = 3$ and $d = 100$ (on more than 50 samples). We always failed with $\ell = 2$ and $d = 150$. In our experiments, to balance the coefficients of the lattice, we replaced the coefficient 1 in the lowest right-hand entry by $q/2^{\ell+1}$. When the attack succeeded, the vector $(\mathbf{a} - \mathbf{u}, q/2^{\ell+1})$ (where $\mathbf{u}$ is a lattice point revealing the hidden number) was generally the second vector of the reduced basis.

Our experimental bound is very close to that of Lemma L.5. We believe it should be possible to reach $\ell = 2$ in practice using a lattice basis reduction algorithm more suited to the infinity norm (see for instance [293]), especially since the lattice dimension is reasonable. In fact, even $\ell = 1$ might be possible in practice : the proof of Lemma L.5 does not rule out such a possibility.

As previously mentioned in the introduction, Bleichenbacher [35] has recently discovered a new attack in this setting, which does not use lattices. This attack is heuristic, but gives better experimental results for currently recommended values of parameters, if one is only interested in minimizing ℓ. More precisely, the best experimental result of [35] shows that one can recover the DSA secret key given a leakage of $\log 3 \approx 1.58$ bits for 2^{22} signatures, in about 3 days on a 450 MHz Ultrasparc using 500Mb of RAM. The number of signatures required is much larger than with the lattice approach with 3 bits (which is close to the information theoretic bound), but it is still reasonably small.

L.5 Remarks

L.5.1 Other consecutive bits

A similar argument works if, more generally, we are given consecutive bits at a known position.

The simplest case is when the consecutive bits are the most significant bits. The definition of most significant bits may depend on the context, as opposed to least significant bits. Here, we study two possible definitions. The usual definition refers to the binary encoding of elements in $\mathbb{F}_q$, where each element is encoded with n bits where $n = 1 + \lfloor \log q \rfloor$ is the bit-length of q. Thus, we define the

ℓ most significant bits of an element $x \in \mathbb{F}_q$ as the unique positive integer $\mathrm{MSB}_{\ell,q}(x) \in \{0, \ldots, 2^{\ell-1}\}$ such that :

$$x - 2^{n-\ell}\mathrm{MSB}_{\ell,q}(x) \in \{0, \ldots, 2^{n-\ell} - 1\},$$

For instance, the most significant bit is 1 if $x \geq 2^{n-1}$, and 0 otherwise. However, this definition is not very well-suited to modular residues, since the most significant bit $\mathrm{MSB}_{1,q}(x)$ may in fact leak less than one bit of information : if q is very close to 2^{n-1}, then $\mathrm{MSB}_{1,q}(x)$ is most of the time equal to 0. Hence, Boneh and Venkatesan used in [48] another definition of most significant bits, which we will refer to as most significant modular bits. The ℓ most significant modular bits of an element $x \in \mathbb{F}_q$ are defined as the unique integer $\mathrm{MSMB}_{\ell,q}(x)$ such that

$$0 \leq x - \mathrm{MSMB}_{\ell,q}(x)q/2^{\ell} < q/2^{\ell}.$$

For example, the most significant modular bit is 0 if $x < q/2$, and 1 otherwise.

Now, recall that by definition of the DSA signature :

$$\alpha T(k,\mu) \equiv k - h(\mu)s(k,\mu)^{-1} \pmod{q},$$

where $T(k,\mu) = \left\lfloor r(k)s(k,\mu)^{-1} \right\rfloor_q$. It follows that for any integer ℓ :

$$\left|\alpha T(k,\mu) - h(\mu)s(k,\mu)^{-1} - 2^{n-\ell}\mathrm{MSB}_{\ell,q}(k) - 2^{n-\ell-1}\right|_q \leq 2^{n-\ell-1},$$

and

$$\left|\alpha T(k,\mu) - h(\mu)s(k,\mu)^{-1} - \mathrm{MSMB}_{\ell,q}(k)q/2^{\ell} - q/2^{\ell+1}\right|_q \leq q/2^{\ell+1}.$$

In other words, the ℓ most significant bits $\mathrm{MSB}_{\ell,q}(k)$ yield an approximation $\mathrm{APP}_{\ell-1,q}(\alpha T(k,\mu))$, while the ℓ most significant modular bits $\mathrm{MSMB}_{\ell,q}(k)$ yield an approximation $\mathrm{APP}_{\ell,q}(\alpha T(k,\mu))$. Hence, Theorems L.1 and L.2 also hold for most significant usual and modular bits, provided that we add one more bit in the case of most significant (usual) bits.

For oracles returning ℓ consecutive bits in the middle, one requires twice as many bits. The idea is to use a trick, which appeared in the work of Frieze *et al.* [101], see (2.13) in that work, on breaking truncated linear congruential generators, and which is based on the following statement for which we provide a proof somewhat simpler to that of [101]. The paper [101] invokes Lenstra's work [208] on integer programming with a fixed number of variables. Our arguments makes use of a simple technique based on continued fractions.

Lemma L.13 *There exists a polynomial-time algorithm which, given A and B in $[1, q]$ finds $\lambda \in \mathbb{Z}_q^*$ such that*

$$|\lambda|_q < B \qquad \textit{and} \qquad |\lambda A|_q \leq q/B.$$

Proof. Let P_i and Q_i denote respectively the numerator and denominator of the ith continued fraction convergents to the rational A/q, $i \geq 1$. There exists j such that $Q_j < B \leq Q_{j+1}$. Then we have

$$\left|\frac{A}{q} - \frac{P_j}{Q_j}\right| \leq \frac{1}{Q_j Q_{j+1}}$$

Therefore $|AQ_j - qP_j| \leq q/Q_{j+1}$. Selecting $\lambda = Q_j$, we obtain the desired statement. □

Now, assume that we are given the ℓ consecutive bits of a nonce $k \in \mathbb{F}_q^*$, starting at some known position j. More precisely, we are given an integer a such that $0 \leq a \leq 2^{\ell} - 1$ and $k = 2^j a + 2^{\ell+j} b + c$ for some integers $0 \leq b \leq q/2^{\ell+j}$ and $0 \leq c < 2^j$. We apply Lemma L.13 with $B = q2^{-j-\ell/2}$ and $A = 2^{j+\ell}$, to obtain $\lambda \in \mathbb{F}_q^*$ such that :

$$|\lambda|_q < q2^{-j-\ell/2} \qquad \text{and} \qquad |\lambda 2^{j+\ell}|_q \leq 2^{j+\ell/2}.$$

Multiplying by λ, the equation (L.1) can be rewritten as :

$$\alpha r(k)\lambda s(k,\mu)^{-1} \equiv \left(2^j a - s(k,\mu)^{-1}h(\mu)\right)\lambda + (c\lambda + 2^{\ell+j}b\lambda) \pmod q.$$

Notice that :

$$\begin{aligned} |c\lambda + 2^{\ell+j}b\lambda|_q &\leq c|\lambda|_q + b|2^{\ell+j}\lambda|_q \\ &< 2^j q 2^{-j-\ell/2} + q/2^{\ell+j}2^{j+\ell/2} = q/2^{\ell/2-1}. \end{aligned}$$

Thus, for arbitrary consecutive bits, one requires roughly twice as many bits. Note that [159] actually suggested that the bounds remained the same with arbitrary consecutive bits. More generally, by using high-dimensional lattice reduction, it is not difficult to show that when ℓ arbitrary bits at known positions are leaked, one requires roughly m as many bits, where m is the number of blocks of consecutive unknown bits.

L.5.2 Practical implications of our results

First of all we note that the constants in our theoretical results are effective and can be explicitly evaluated. One can also find a precise dependence of δ on ε in the above estimates. In particular, it would be interesting to obtain a non-asymptotic form of our theoretical results for the range of p and q corresponding to the real applications of DSA, that is, when q is a 160-bit prime and p is a 512-bit prime, see [231, 333].

The condition $W \leq |\mathcal{M}|^2 q^{-1+\delta}$ does not seem too restrictive, as one could expect $W \sim |\mathcal{M}|^2 q^{-1}$ for any "good" hash function.

It might be worth noting that Lemma L.10 implies that $r(k)$ takes exponentially many distinct values. Thus DSA indeed generates exponentially many distinct signatures. Certainly this fact has never been doubted in practice but our results provide its rigorous confirmation. On the other hand, the bound q^{δ} implied by Lemma L.10 on the number of distinct values of $r(k)$ falls far below the expected value of order about q. Obtaining such a lower bound is a very challenging question which probably requires some advanced number theoretic tools.

Finally, we observe that if for efficiency reasons, one chooses either a nonce k with fewer bits than q or a sparse nonce k (to speed up the exponentiation at each signature round), then our attack obviously applies, because one then either knows or guesses with high probability sufficiently many bits of the nonce k. We remark that it could be quite tempting to choose such "special" k. Indeed, the size of q is currently determined by the time required by the $q^{1/2}$-attacks on the signer's discrete log key (such as Pollard's rho algorithm, see the survey [341]). However, such attacks fail to recover the value of k from $r(k)$ because the double reduction (modulo p and then modulo q) seems to erase all useful properties of the exponential function. Thus, simple exhaustive search may *a priori* seem the only method to recover k from $r(k)$, and one may believe that taking k in the range $1 \leq k \leq q^{1/2}$ does not undermine the security of the scheme. Our results show that this choice is fatal for the whole scheme.

To establish the corresponding uniformity of distribution results one can use bounds of exponential sums when k runs over a part of the interval $[1, q-1]$. Namely, for any element $g \in \mathbb{F}_p$ of multiplicative order T the bound

$$\max_{1 \leq K \leq T} \max_{\gcd(c,p)=1} \left| \sum_{k=1}^{K} \mathbf{e}_p\left(cg^k\right) \right| = O(p^{1/2} \log p)$$

holds, see Lemma 2 of [192] or Theorem 8.2 of [271]. This bound is nontrivial only for $T \geq p^{1/2+\varepsilon}$. Accordingly, our method applies only to larger values of q than in Theorems L.1 and L.2, namely $q \geq p^{1/2+\varepsilon}$, but the attack itself still can be launched (even without rigorous proof of success). In fact one can obtain an analogue of Lemma L.8 for such short sums as well. For sparse exponents one can

use the approach of [99] to obtain the necessary bounds of the corresponding exponential sums. We remark that the results of [99] apply only to the case of a primitive root g but the technique can be expanded to g of arbitrary (but sufficiently large) multiplicative order modulo p.

Hence, our results show that it is really essential to the security of DSA that the nonce k be generated by a cryptographically secure pseudo-random number generator. We also remark that a very different heuristic attack on very small values of $k = O(q^{1/2})$ has recently been described in [196]. Even in this case, if the attacker is able to apply a *timing* or *power* attack and select signatures corresponding to small values of k then the whole signature scheme is insecure. Generally, any leakage of information on k could prove dramatic.

L.5.3 Related signature schemes

One might wonder to which extent our results also apply to other DSA-related signature schemes (see [231, Section 11.5]), such as Schnorr's [304] or El Gamal's [108]. We follow the notations of Section L.1.1.

Recall that in Schnorr's signature scheme, the signature of a message μ with a nonce $k \in \mathbb{F}_q^*$ is the pair $(e, s) \in \mathbb{F}_q^2$ defined as (the symbol $\|$ denoting as usual concatenation) :

$$\begin{aligned} e(k,\mu) &= h\left(\mu \,\|\, \left\lfloor g^k \right\rfloor_p\right) \\ s(k,\mu) &= \left\lfloor k + \alpha e(k,\mu) \right\rfloor_q \end{aligned}$$

Obviously, the same attack applies with different multipliers. For instance, suppose that the ℓ least significant bits of k are known for several signatures. Then, as in Section L.1.4, it can be seen that recovering the signer's secret key α is a HNP with multiplier

$$t(k,\mu) = \left\lfloor e(k,\mu) 2^{-\ell} \right\rfloor_q .$$

Under the random oracle model, $t(k,\mu)$ has (perfect) uniform distribution, making Lemmas L.4 and L.5 directly applicable. The same holds for any block of consecutive bits. Hence, our results are even simpler with Schnorr's signature scheme.

In El Gamal's signature scheme, there is only one large prime p, and g is a generator of $\mathbb{F}_p^*$. The signature of a message μ with a nonce $k \in [1, p-2]$ coprime with $p-1$ is the pair $(r, s) \in [1, p-1]^2$ defined as :

$$\begin{aligned} r(k) &= \left\lfloor g^k \right\rfloor_p \\ s(k,\mu) &= \left\lfloor k^{-1}(h(\mu) - \alpha r(k)) \right\rfloor_p \end{aligned}$$

This time, we cannot work with the least significant bits, as 2 is not invertible modulo $p-1$. So suppose instead that the ℓ most significant modular bits of k are known for several signatures. Then we obtain a HNP with multiplier :

$$t(k,\mu) = \left\lfloor r(k) s(k,\mu)^{-1} \right\rfloor_p .$$

The resemblance with the DSA case is of course natural. Statements similar to that of Section L.3 can be obtained on such multipliers. In fact, for this case, many things can be done in much stronger form. In particular, it has been shown in [324] that in El Gamal's signature scheme the pairs $(r(k), s(k,\mu))$ are uniformly distributed modulo p, rather than just their ratios $t(k,\mu)$. Thus we have no doubt that a similar attack applies to this scheme as well, however we are not able to give a rigorous proof of this statement because the above approach to the HNP fails to work modulo a composite. Probably some

further adjustments and modifications should be made to design an algorithm for the HNP modulo a composite, which could be an interesting problem by itself.

The modifications of DSA described in [231, Table 11.5]), see also [273], can be studied by our method as well, see [91]. Also, in [262] we have obtained similar results for the elliptic curve analogue of DSA.

Finally, the results and ideas of this paper have recently been used in [53] to design an attack on another DSA-based cryptosystem. It is shown in [53] that in the above cryptosystem there is a way to extract all necessary information from the protocol itself, thus no additional "leakage" is assumed. In fact, Lemma L.11 allows us to make the attack of [53] rigorously proved and also to extend it to other small subgroups of $\mathbb{F}_p^*$ (not only those with a power of 2 elements as in [53]).

Acknowledgements

We thank Daniel Bleichenbacher, Dan Boneh, Nick Howgrave–Graham and Ramarathnam Venkatesan for helpful discussions. Part of this work was done while the first author was visiting Stanford University, whose hospitality is gratefully acknowledged.

Annexe M

Experimenting with Faults, Lattices, and the DSA

PKC 2005
[246] avec David Naccache, Michael Tunstall (Gemplus) et Claire Whelan (Dublin City Univ.)

Abstract: *We present an attack on DSA smart-cards which combines physical fault injection and lattice reduction techniques. This seems to be the first (publicly reported) physical experiment allowing to concretely pull-out DSA keys out of smart-cards. We employ a particular type of fault attack known as a glitch attack, which will be used to actively modify the DSA nonce k used for generating the signature : k will be tampered with so that a number of its least significant bytes will flip to zero. Then we apply well-known lattice attacks on El Gamal-type signatures which can recover the private key, given sufficiently many signatures such that a few bits of each corresponding k are known. In practice, when one byte of each k is zeroed, 27 signatures are sufficient to disclose the private key. The more bytes of k we can reset, the fewer signatures will be required. This paper presents the theory, methodology and results of the attack as well as possible countermeasures.*

Keywords: *DSA, fault injection, glitch attacks, lattice reduction.*

M.1 Introduction

Over the past few years fault attacks on electronic chips have been investigated and developed. The theory developed was used to challenge public key cryptosystems [42] and symmetric ciphers in both block [29] and stream [145] modes.

The discovery of fault attacks (1970s) was accidental. It was noticed that elements naturally present in packaging material of semiconductors produced radioactive particles which in turn caused errors in chips [226]. These elements, while only present in extremely minute parts (two or three parts per million), were sufficient to affect the chips' behaviour, create a charge in sensitive silicon areas and, as a result, cause bits to flip. Since then various mechanisms for fault creation and propagation have been discovered and researched. Diverse research organisations such as the aerospace industry and the security community have endeavoured to develop different types of fault injection techniques and devise corresponding preventative methods. Some of the most popular fault injection techniques include variations in supply voltage, clock frequency, temperature or the use of white light, X-ray and ion beams.

The objectives of all these techniques is generally the same : corrupt the chip's behaviour. The outcomes have been categorised into two main groups based on the long term effect that the fault

produced. These are known as *permanent* and *transient* faults. Permanent faults, created by purposely inflicted defects to the chip's structure, have a permanent effect. Once inflicted, such destructions will affect the chip's behavior permanently. In a transient fault, silicon is locally ionized so as to induce a current that, when strong enough, is falsely interpreted by the circuit as an internal signal. As ionization ceases so does the induced current (and the resulting faulty signal) and the chip recovers its normal behavior.

Preventive measures come in the form of software and hardware protections (the most cost-effective solution being usually a combination of both). Current research is also looking into fault detection where, at stages through the execution of the algorithm, checks are performed to see whether a fault has been induced [169]. For a survey of the different types of fault injection techniques and the various software and hardware countermeasures that exist, we refer the reader to [21].

In this paper we will focus on a type of fault attack known as a glitch attack. Glitch attacks use transient faults where the attacker deliberately generates a voltage spike that causes one or more flip-flops to transition into a wrong state. Targets for insertion of such 'glitches' are generally machine instructions or data values transferred between registers and memory. Results can include the replacement of critical machine instructions by almost arbitrary ones or the corruption of data values.

The strategy presented in this paper is the following : we will use a glitch to reset some of the bytes of the nonce k, used during the generation of DSA signatures. As the attack ceases, the system will remain fully functional. Then, we will use classical lattice reduction techniques to extract the private signature key from the resulting glitched signatures (which can pass the usual verification process). Such lattice attacks (introduced by Howgrave-Graham and Smart [159], and improved by Nguyễn and Shparlinski [261]) assume that a few bits of k are known for sufficiently many signatures, without addressing how these bits could be obtained. In [261], it was reported that in practice, the lattice attack required as few as three bits of k, provided that about a hundred of such signatures were available. Surprisingly, to the authors' knowledge, no fault attack had previously exploited those powerful lattice attacks.

The paper is organised as follows : In section 2 we will give a brief description of DSA, we will also introduce the notations used throughout this paper. An overview of the attack's physical and mathematical parts will be given in section 3. In section 4 we will present the results of our attack while countermeasures will be given in section 5.

Related work :

In [20] an attack against DSA is presented by Bao *et al.*, this attack is radically different from the one presented in this paper and no physical implementation results are given. This attack was extended in [84] by Dottax. In [114], Knudsen and Giraud introduce another fault attack on the DSA. Their attack requires around 2300 signatures (*i.e.* 100 times more than the attack presented here). The merits of the present work are thus twofold : we present a new (*i.e.* unrelated to [114, 20, 84]) efficient attack and describe what is, to the authors' best knowledge, the first (publicly reported) physical experiment allowing to concretely pull-out DSA keys out of smart-cards. The present work shows that the hypotheses made in the lattice attacks [159, 261] can be realistic in certain environments.

M.2 Background

In this section we will give a brief description of the DSA.

M.2.1 DSA Signature and Verification

The system parameters for DSA [248] are $\{p, q, g\}$, where p is prime (at least 512 bits), q is a 160-bit prime dividing $p-1$ and $g \in \mathbb{Z}_p^*$ has order q. The private key is an integer $\alpha \in \mathbb{Z}_q^*$ and the public key is the group element $\beta = g^\alpha \pmod p$.

Signature :

To sign a message m, the signer picks a random $k < q$ and computes :

$$r \leftarrow \left(g^k \pmod p\right) \pmod q \text{ and } s \leftarrow \frac{\mathrm{SHA}(m) + \alpha r}{k} \pmod q$$

The signature of m is the pair : $\{r, s\}$.

Verification :

To check $\{r, s\}$ the verifier ascertains that :

$$r \stackrel{?}{=} \left(g^{wh}\beta^{wr} \pmod p\right) \pmod q \text{ where } w \leftarrow \frac{1}{s} \pmod q \text{ and } h \leftarrow \mathrm{SHA}(m)$$

M.3 Attack Overview

The attack on DSA proceeds as follows : we first generate several DSA signatures where the random value generated for k has been modified so that a few of k's least[15] significant bytes are reset[16]. This faulty k will then be used by the card to generate a (valid) DSA signature. Using lattice reduction, the secret key α can be recovered from a collection of such signatures (see [261, 159]). In this section we will detail each of these stages in turn, showing first how we tamper with k in a closed environment and then how we apply this technique to a complete implementation.

M.3.1 Experimental Conditions

DSA was implemented on a chip known to be vulnerable to V_{cc} glitches. For testing purposes (closed environment) we used a separate implementation for the generation of k.

A 160-bit nonce is generated and compared to q. If $k \geq q-1$ the nonce is discarded and a new k is generated. This is done in order to ascertain that k is drawn uniformly in $\mathbb{Z}_q^*$ (assuming that the source used for generating the nonce is perfect). We present the code fragment (modified for simplicity) that we used to generate k :

```
PutModulusInCopro(PrimeQ) ;
RandomGeneratorStart() ;

status = 0 ;
do {
  IOpeak() ;
  for (i=0 ; i<PrimeQ[0] ; i++) {
    acCoproMessage[i+1] = ReadRandomByte() ;
  }
```

[15] It is also possible to run a similar attack by changing the most significant bytes of k. This is determined by the implementation.

[16] It would have also been possible to run a similar attack if these bytes were set to FF.

```
    IOpeak() ;

    acCoproMessage[0] = PrimeQ[0] ;
    LoadDataToCopro(acCoproMessage) ;

    status = 1 ;
    for (j=0 ; j<(PrimeQ[0]+1) ; j++) {
      if (acCoproResult[j] != acCoproMessage[j]) {
        status = 0 ;
      }
    }
  }
  while (status == 0) ;
  RandomGeneratorStop() ;
```

Note that `IOpeaks`[17], featured in the above code was also included in the implementation of DSA. The purpose of this is to be able to easily identify the code sections in which a fault can be injected to produce the desired effect. This could have been done by monitoring power consumption but would have greatly increased the complexity of the task.

The tools used to create the glitches can be seen in figure M.1 and figure M.2. Figure M.1 is a modified CLIO reader which is a specialised high precision reader that allows one glitch to be introduced following any arbitrarily chosen number of clock cycles after the command sent to the card. Figure M.2 shows the experimental set up of the CLIO reader with the oscilloscope used during our experiments. A BNC connector is present on the CLIO reader which allows the I/O to be easily read ; another connector produces a signal when a glitch is applied (in this case used as a trigger). Current is measured using a differential probe situated on top of the CLIO reader.

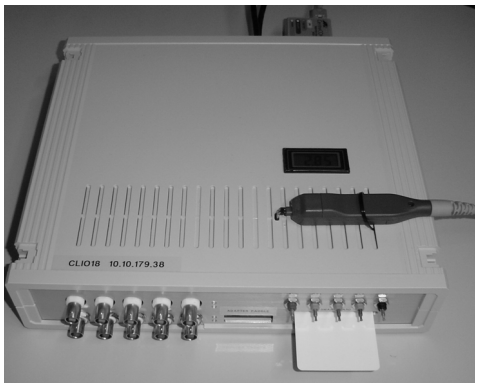


Fig. M.1 – A Modified CLIO Reader

M.3.2 Generating a Faulty k

The command that generated k was attacked in every position between the two `IOpeaks` in the code. It was found that the fault did not affect the assignment of k to the RAM *i.e.* the instruction `acCoproMessage[i+1] = ReadRandomByte() ;` which always executed correctly. However, it was possible to change the evaluation of `i` during the loop. This enabled us to select the number of least

[17]The I/O peak is a quick movement on the I/O from one to zero and back again. This is visible on an oscilloscope but is ignored by the card reader.

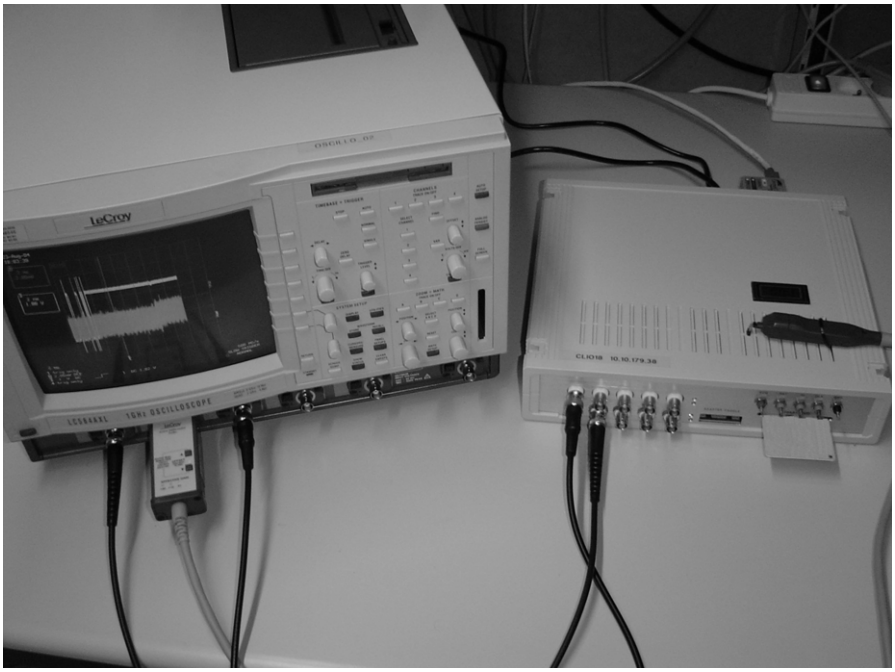


FIG. M.2 – Experimental Set Up

significant bytes to be reset. In theory, this would produce the desired fault in k with probability $q/2^{160}$, as if the modified k happens to be larger than q, it is discarded anyway. In practice this probability is likely to be lower as it is unusual for a fault to work correctly every time.

An evaluation of a position that resetted the last two bytes was performed. Out of 2000 attempts 857 were corrupted. This is significantly less than what one would expect, as the theoretical probability is $\simeq 0.77$. We expected the practical results to perform worse than theory due to a slight variation in the amount of time that the smart card takes to arrive at the position where the data corruption is performed. There are other positions in the same area that return k values with the same fault, but not as often.

M.3.3 The Attack : Glitching k During DSA Computations

The position found was equated to the generation of k in the command that generates the DSA signature. This was done by using the last I/O event at the end of the command sent as a reference point and gave a rough position of where the fault needs to be injected.

As changes in the value of k were not visible in the signature, results would only be usable with a certain probability. This made the attack more complex, as the subset signatures having faulty k values had to be guessed amongst those acquired by exhaustive search.

To be able to identify the correct signatures the I/O and the current consumption signals were monitored during the attacks. An example of such a monitoring is given in figure M.3. The object of these acquisitions was to measure the time T elapsed between the end of the command sent to the card and the beginning of the calculation of r. This can be seen in the current consumption, as the chip will require more energy when the crypto-coprocessor is ignited. If we denote by t the time that it takes to reach the start of the calculation of r knowing that the picked k was smaller that q (*i.e.* that it was not necessary to restart the picking process) then, if $T = t$ we know that the command has executed properly and that k was picked correctly the first time. If $T > t$ then any fault targeting k would be a miss (as k was regenerated given that the value of k originally produced was greater

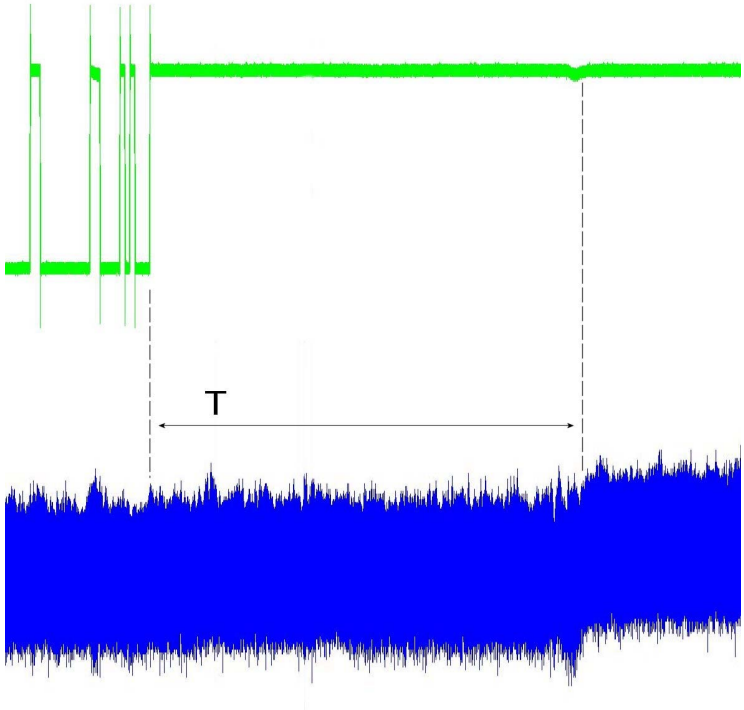


FIG. M.3 – I/O and Current Consumption (Beginning of the Trace of the Command Used to Generate Signatures).

than q). Signatures resulting from commands that feature such running times can be discarded as the value of k will not present any exploitable weaknesses. When $T < t$ we know that the execution of the code generating k has been cut short, so some of the least significant bytes will be equal to zero. This allows signatures generated from corrupted k values to be identified *a posteriori*.

As the position where the fault should be injected was only approximately identified, glitches were injected in twenty different positions until a position that produced signatures with the correct characteristics (as described above) was found. The I/O peaks left in the code were used to confirm these results. Once the correct position identified, more attacks were conducted at this position to acquire a handful of signatures. From a total of 200 acquisitions 38 signatures where $T < t$ were extracted.

This interpretation had to be done by a combination of the I/O and the current consumption, as after the initial calculation involving k the command no longer takes the same amount of time. This is because $0 < k \leq q$ and therefore k does not have a fixed size ; consequently any calculations k is involved in will not always take the same amount of time.

M.3.4 Use of Lattice Reduction to Retrieve α

We are now in a position to apply the well-known lattice attacks of [159, 261] on El Gamal-type signature schemes : given many DSA signatures for which a few bits of the corresponding k are known, such attacks recover the DSA signer's private key. In our case, these known bits are in fact 0 bits, but that does not matter for the lattice attack. We recall how the lattice attacks work, using the presentation of Nguyễn and Shparlinski [261]. Roughly speaking, lattice attacks focus on the linear part of DSA, that is, they exploit the congruence $s \leftarrow \frac{\mathrm{SHA}(m)+\alpha r}{k} \pmod{q}$ used in the

signature generation, not the other congruence $r \leftarrow \left(g^k \pmod p\right) \pmod q$ which is related to a discrete log problem. When no information on k is available, the congruence reveals nothing, but if partial information is available, each congruence discloses something about the private key α : by collecting sufficiently many signatures, there will be enough information to recover α. If ℓ bits of k are known for a certain number of signatures, we expect that about $160/\ell$ signatures will suffice to recover α. Here is a detailed description of the attack.

For a rational number z and $m \geq 1$ we denote by $\lfloor z \rfloor_m$ the unique integer a, $0 \leq a \leq m-1$ such that $a \equiv z \pmod m$ (provided that the denominator of z is relatively prime to m). The symbol $|.|_q$ is defined as $|z|_q = \min_{b \in Z} |z - bq|$ for any real z.

Assume that we know the ℓ least significant bits of a nonce $k \in \{0, \ldots, q-1\}$ which will be used to generate a DSA signature (for the case of other bits, like most significant bits or bits in the middle, see [261]).

That is, we are given an integer a such that $0 \leq a \leq 2^\ell - 1$ and $k - a = 2^\ell b$ for some integer $b \geq 0$. Given a message m (whose SHA hash is h) signed with the nonce k, the congruence

$$\alpha r \equiv sk - h \pmod q,$$

can be rewritten for $s \neq 0$ as :

$$\alpha r 2^{-\ell} s^{-1} \equiv (a - s^{-1}h)2^{-\ell} + b \pmod q. \tag{M.1}$$

Now define the following two elements

$$\begin{aligned} t &= \left\lfloor 2^{-\ell} r s^{-1} \right\rfloor_q, \\ u &= \left\lfloor 2^{-\ell}(a - s^{-1}h) \right\rfloor_q \end{aligned}$$

and remark that both t and u can easily be computed by the attacker from the publicly known information. Recalling that $0 \leq b \leq q/2^\ell$, we obtain

$$0 \leq \lfloor \alpha t - u \rfloor_q < q/2^\ell.$$

And therefore :

$$|\alpha t - u - q/2^{\ell+1}|_q \leq q/2^{\ell+1}. \tag{M.2}$$

Thus, the attacker knows an integer t and a rational number $v = u + q/2^{\ell+1}$ such that :

$$|\alpha t - v|_q \leq q/2^{\ell+1}.$$

In some sense, we know an approximation of αt modulo q. Now, suppose we can repeat this for many signatures, that is, we know d DSA signatures $\{r_i, s_i\}$ of hashes h_i (where $1 \leq i \leq d$) such that we know the ℓ least significant bits of the corresponding nonce k_i. From the previous reasoning, the attacker can compute integers t_i and rational numbers v_i such that :

$$|\alpha t_i - v_i|_q \leq q/2^{\ell+1}.$$

The goal of the attacker is to recover the DSA private key α. This problem is very similar to the so-called hidden number problem introduced by Boneh and Venkatesan in [48]. In [48, 261], the problem is solved by transforming it into a lattice closest vector problem (for background on lattice theory and its applications to cryptography, we refer the reader to the survey [267] ; a similar technique was recently used in [255]).

More precisely, consider the $(d+1)$-dimensional lattice L spanned by the rows of the following matrix :

$$\begin{pmatrix} q & 0 & \cdots & 0 & 0 \\ 0 & q & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \dots & 0 & q & 0 \\ t_1 & \dots & \dots & t_d & 1/2^{\ell+1} \end{pmatrix}. \tag{M.3}$$

The inequality $|v_i - \alpha t_i|_q \leq q/2^{\ell+1}$ implies the existence of an integer c_i such that :

$$|v_i - \alpha t_i - q c_i| \leq q/2^{\ell+1}. \tag{M.4}$$

Notice that the row vector $\mathbf{c} = (\alpha t_1 + q c_1, \dots, \alpha t_d + q c_d, \alpha/2^{\ell+1})$ belongs to L, since it can be obtained by multiplying the last row vector by α and then subtracting appropriate multiples of the first d row vectors. Since the last coordinate of this vector discloses the hidden number α, we call $\mathbf{c}$ the *hidden vector*. The hidden vector is very close to the (publicly known) row vector $\mathbf{v} = (v_1, \dots, v_d, 0)$. By trying to find the closest vector to $\mathbf{v}$ in the lattice L, one can thus hope to find the hidden vector $\mathbf{c}$ and therefore the private key α. The article [261] presents provable attacks of this kind, and explains how the attack can be extended to bits at other positions. Such attacks apply to DSA but also to any El Gamal-type signature scheme (see for instance [262] for the case of ECDSA).

In our case, we simply build the previously mentioned lattice and the target vector $\mathbf{v}$, and we try to solve the closest vector problem with respect to $\mathbf{v}$, using the so-called embedding technique that heuristically reduces the lattice closest vector problem to the shortest vector problem (see [261] for more details). >From each close vector candidate, we derive a candidate y for α from its last coordinate, and we check that the public key satisfies $\beta = g^y \pmod p$.

M.4 Results

As already mentioned in Section M.3.3, using a glitch attack, we were able to generate 38 DSA signatures such that the least significant byte of the corresponding k was expected to be zero. Next, we applied the lattice attack of Section M.3.4, using NTL's [319] implementation of Schnorr–Euchner's BKZ algorithm [307] with block size 20 as our lattice basis reduction algorithm. Out of the 38 signatures, we picked 30 at random to launch the lattice attack, and those turned out to be enough to disclose the DSA private key α after a few seconds on an Apple PowerBook G4. We only took 30 because we guessed from past experiments that 30 should be well sufficient.

To estimate more precisely the efficiency of the lattice attack, we computed success rates, by running the attack 100 times with different parameters. Results can be seen in Table 1. Because the number of signatures is small, the lattice dimension is relatively small, which makes the running time of the lattice attack negligible : for instance, on an Apple PowerBook G4, the lattice attack takes about 1 second for 25 signatures, and 20 seconds for 38 signatures. Table 1 shows how many signatures are required in practice to make the lattice attack work, depending on the number of least significant bytes reset in k. Naturally, there will be a tradeoff between the fault injection and the lattice reduction : when generating signatures with nonces with more reset bytes, the lattice phase of the attack will require less signatures. When only one signature is available, the lattice attack cannot work because there is not enough information in the single congruence used. However, if ever that signature is such that k has a large proportion of zero bytes, it might be possible to compute k by exhaustive search (using the congruence $\leftarrow \left(g^k \pmod p\right) \pmod q$), and then recover α. >From Table 1, we see that when two signatures are available, the lattice attack starts working when 11 bytes are reset in each k. When only one byte is reset in k, the lattice attack starts working (with non-negligible probability) with only 23 signatures.

TAB. M.1 – Experimental Attack Success Rates : n is the Number of Bytes Reset in k, and d is the Number of Signatures.

	Number d of Signatures															
$n \downarrow$	2	3	4	5	6	7	8	10	11	12	22	23	24	25	26	27
1											0%	10%	39%	63%	87%	100%
2								0%	69%	100%						
3					0%	69%	100%									
4				0%	100%											
5		0%	2%	100%												
6		0%	100%													
7	0%	96%	100%													
10	6%	100%														
11	100%															

It should be stressed that the lattice attack does not tolerate mistakes. For instance, 27 signatures with a single byte reset in k are enough to make the attack successful. But the attack will not work if for one of those 27 signatures, k has no reset bytes. It is therefore important that the signatures input to the lattice attack satisfy the assumption about the number of reset bytes. Hence, if ever one is able to obtain many signatures such that the corresponding k is expected (but not necessarily all the time) to have a certain number of reset bytes, then one should not input all the signatures to the lattice attack. Instead, one should pick at random a certain number of signatures from the whole set of available signatures, and launch the lattice attack on this smaller number of signatures : Table 1 can be used to select the minimal number of signatures that will make the lattice attack successful. This leads to a combination of exhaustive search and lattice reduction.

M.5 Countermeasures

The heart of this attack lies with the ability to induce faults that reset some of k's bits. Hence, any strategy allowing to avoid or detect such anomalies will help thwart the attacks described in this paper. Note that checking the validity of the signature after generation will not help, contrary to the case of fault attacks on RSA signatures [42] : the faulty DSA signatures used here are valid signatures which will pass the verification process. We recommend to use *simultaneously* the following tricks that cost very little in terms of code-size and speed :

- *Checksums* can be implemented in software. This is often complementary to hardware checksums, as software CRCs can be applied to buffers of data (sometimes fragmented over various physical addresses) rather than machine words.
- *Execution Randomization* : If the order in which operations in an algorithm are executed is randomized it becomes difficult to predict what the machine is doing at any given cycle. For most fault attacks this countermeasure will only slow down a determined adversary, as eventually a fault will hit the desired instruction. This will however thwart attacks that require faults in specific places or in a specific order.
  For instance, to copy 256 bytes from buffer a to buffer b, copy

$$b[f(i)] \leftarrow a[f(i)] \quad \text{for} \quad i = 0, \ldots, 255$$

  where $f(i) = (x \times (i \oplus w) + y \pmod{256}) \oplus z$ and $\{x, y, z, w\}$ are four random bytes (x odd) unknown to the attacker.
- *Ratification counters and baits* : baits are small (< 10 byte) code fragments that perform an operation and test its result. A typical bait writes, reads and compares data, performs xors,

additions, multiplications and other operations whose results can be easily checked. When a bait detects an error it increments an NVM counter and when this counter exceeds a tolerance limit (usually three) the card ceased to function.
– *Repeated refreshments* : refresh k by generating several nonces and exclusive-or them with each other, separating each nonce generation from the previous by a random delay. This forces the attacker to inject multiple faults at randomly shifting time windows in order to reset specific bits of k.

Finally, it may also be possible to have a real time testing of the random numbers being generated by the smart card, such as that proposed in the FIPS140-2. However, even if this is practical it may be of limited use as our attack requires very few signatures to be successful. Consequently, our attack may well be complete before it gets detected.

What is very important is that no information on k is leaked, and that k is cryptographically random.

M.6 Conclusion

We described a method for attacking a DSA smart card vulnerable to fault attacks. Similar attacks can be mounted on any other El Gamal-type signature scheme, such as ECDSA and Schnorr's signature. The attack consisted of two stages. The first stage dealt with fault injection. The second involved forming a lattice for the data gathered in the previous stage and solving a closest vector problem to reveal the secret key.

The attack was realised in the space of a couple of weeks and was made easier by the inclusion of peaks on the I/O. This information could have been derived by using power or electromagnetic analysis to locate the target area, but would have taken significantly longer. The only power analysis done during this attack was to note when the crypto-coprocessor started to calculate a modular exponentiation.

Annexe N

Can We Trust Cryptographic Software ? Cryptographic Flaws in GNU Privacy Guard v1.2.3

EUROCRYPT 2004
[255]

Abstract: *More and more software use cryptography. But how can one know if what is implemented is good cryptography ? For proprietary software, one cannot say much unless one proceeds to reverse-engineering, and history tends to show that bad cryptography is much more frequent than good cryptography there. Open source software thus sounds like a good solution, but the fact that a source code can be read does not imply that it is actually read, especially by cryptography experts. In this paper, we illustrate this point by examining the case of a basic Internet application of cryptography : secure email. We analyze parts of the source code of the latest version of GNU Privacy Guard (GnuPG or GPG), a free open source alternative to the famous PGP software, compliant with the OpenPGP standard, and included in most GNU/Linux distributions such as Debian, MandrakeSoft, Red Hat and SuSE. We observe several cryptographic flaws in GPG v1.2.3. The most serious flaw has been present in GPG for almost four years : we show that as soon as one (GPG-generated) ElGamal signature of an arbitrary message is released, one can recover the signer's private key in less than a second on a PC. As a consequence, ElGamal signatures and the so-called ElGamal sign+encrypt keys have recently been removed from GPG. Fortunately, ElGamal was not GPG's default option for signing keys.*

Keywords : *Public-key cryptography, GnuPG, GPG, OpenPGP, Cryptanalysis, RSA, ElGamal, Implementation.*

N.1 Introduction

With the advent of standardization in the cryptography world (RSA PKCS [197], IEEE P1363 [163], CRYPTREC [165], NESSIE [96], *etc.*), one may think that there is more and more good cryptography. But as cryptography becomes "global", how can one be sure that what is implemented in the real world is actually good cryptography ? Numerous examples (such as [33, 35, 218]) have shown that the frontier between good cryptography and bad cryptography is very thin. For proprietary software, it seems difficult to make any statement unless one proceeds to the tedious task of reverse-engineering. If a proprietary software claims to implement 2048-bit RSA and 128-bit AES, it does not say much

about the actual cryptographic security : which RSA is being used ? Could it be textbook RSA [47] (with zero-padding) encrypting a 128-bit AES key with public exponent 3 ? Are secret keys generated by a weak pseudo-random number generator like old versions of Netscape [115] ? Who knows if it is really RSA–OAEP which is implemented [218] ? With proprietary software, it is ultimately a matter of trust : unfortunately, history has shown that there is a lot of bad cryptography in proprietary software (see for instance [301, 133] for explanations). Open source software thus sounds like a good solution. However, the fact that a source code can be read does not necessarily imply that it is actually read, especially by cryptography experts.

The present paper illustrates this point by examining the case of "perhaps the most mature cryptographic technology in use on the Internet" (according to [27]) : secure email, which enables Internet users to authentify and/or encrypt emails. Secure email became popular in the early 90s with the appearance of the now famous Pretty Good Privacy (PGP) [283] software developed by Phil Zimmermann in the US. Not so long ago, because of strict export restrictions and other US laws, PGP was unsuitable for many areas outside the US. Although the source code of PGP has been published, it is unknown whether future versions of PGP will be shipped with access to the source code.

GNU Privacy Guard [128] (GnuPG, or GPG in short) was developed in the late 90s as an answer to those PGP issues. GPG is a full implementation of OpenPGP [280], the Internet standard that extends PGP. GPG has been released as free software under the GNU General Public License (GNU GPL) : As such, full access to the source code is provided at [128], and GPG can be viewed as a free replacement for PGP. The German Federal Ministry of Economics and Technology granted funds for the further development of GPG. GPG has a fairly significant user base : it is included in most GNU/Linux distributions, such as Debian, MandrakeSoft, Red Hat and SuSE. The first stable version of GPG was released on September 7th, 1999. Here, we review the main public-key aspects of the source code of v1.2.3, which was the current stable version (released on August 22nd, 2003) when this paper was submitted to Eurocrypt '04. Our comments seem to also apply to several previous versions of GPG. However, we stress that our analysis is not claimed to be complete, even for the public-key aspects of GPG.

We observe several cryptographic flaws in GPG v1.2.3. The most serious flaw (which turns out to have been present in GPG for almost four years) is related to ElGamal signatures : we present a lattice-based attack which recovers the signer's private key in less than a second on a PC, given any (GPG-generated) ElGamal signature of a (known) arbitrary message and the corresponding public key. This is because both a short private exponent and a short nonce are used for the generation of ElGamal signatures, when the GPG version in use is between 1.0.2 (January 2000) and 1.2.3 (August 2003). As a result, GPG–ElGamal signing keys have been considered compromised [186], especially the so-called primary ElGamal sign+encrypt keys : with such keys, one signature is always readily available, because such keys automatically come up with a signature to bind the user identity to the public key, thus leaking the private key used for both encryption and signature. Hence, ElGamal signatures and ElGamal sign+encrypt keys have recently been removed from GPG (see [128] for more information).

We notice that GPG encryption provides no chosen-ciphertext security, due to its compliance with OpenPGP [280], which uses the old PKCS #1 v1.5 standard [175] : Bleichenbacher's chosen-ciphertext attack [33] applies to OpenPGP, either when RSA or ElGamal is used. Although the relevance of chosen-ciphertext attacks to the context of email communications is debatable, we hope that OpenPGP will replace PKCS #1 v1.5 to achieve chosen-ciphertext security. The other flaws do not seem to lead to any reasonable attack, they only underline the lack of state-of-the-art cryptography in GPG and sometimes OpenPGP. It is worth noting that the OpenPGP standard is fairly loose : it gives non-negligible freedom over the implementation of cryptographic functions, especially regarding key generation. Perhaps stricter and less ambiguous guidelines should be given in order to decrease

security risks.

The only published research on the cryptographic strength of GPG we are aware of is [179, 167], which presented chosen-ciphertext attacks with respect to the symmetric encryption used in PGP and GPG. The rest of the paper is organized as follows. In Section N.2, we give an overview of the GPG software v1.2.3. In Section N.3, we review the GPG implementation of ElGamal and present the attack on ElGamal signatures. In Section N.4, we review the GPG implementation of RSA. There is no section devoted to the GPG implementation of DSA, since we have not found any noteworthy weakness in it. In Appendix N.5, we give a brief introduction to lattice theory, because the attack on ElGamal signatures uses lattices. In Appendix N.6, we provide a proof (in an idealized model) of the lattice-based attack on ElGamal signatures.

N.2 An Overview of GPG v1.2.3

GPG v1.2.3 [128] supports ElGamal (signature and encryption), DSA, RSA, AES, 3DES, Blowfish, Twofish, CAST5, MD5, SHA-1, RIPE-MD-160 and TIGER. GPG decrypts and verifies PGP 5, 6 and 7 messages : It is compliant with the OpenPGP standard [280], which is described in RFC 2440 [55].

GPG provides secrecy and/or authentication to emails : it enables users to encrypt/decrypt and/or sign/verify emails using public-key cryptography. The public-key infrastructure is the famous web of trust : users certify public key of other users.

GPG v.1.2.3 allows the user to generate several types of public/private keys, with the command `gpg -gen-key` :

- Choices available in the standard mode :
  - (1) DSA and ElGamal : this is the default option. The DSA keys are signing keys, while the ElGamal keys are encryption keys (type 16 in the OpenPGP terminology).
  - (2) DSA : only for signatures.
  - (5) RSA : only for signatures.
- Additional choices available in the expert mode :
  - (4) ElGamal for both signature and encryption. In the OpenPGP terminology, these are keys of type 20.
  - (7) RSA for both signature and encryption.

In particular, an ElGamal signing key is also an encryption key, but an ElGamal encryption key may be restricted to encryption. In GPG v1.2.3, ElGamal signing keys cannot be created unless one runs the expert mode : however, this was not always the case in previous versions. For instance, the standard mode of GPG v1.0.7 (which was released in April 2002) proposes the choices (1), (2), (4) and (5).

N.2.1 Encryption

GPG uses hybrid encryption to encrypt emails. A session key (of a symmetric encryption scheme) is encrypted by a public-key encryption scheme : either RSA or ElGamal (in a group $\mathbb{Z}_p^*$, where p is a prime number). The session key is formatted as specified by OpenPGP (see Figure N.1) : First, the session key is prefixed with a one-octet algorithm identifier that specifies the symmetric encryption algorithm to be used ; Then a two-octet checksum is appended which is equal to the sum of the preceding session key octets, not including the algorithm identifier, modulo 65536.

This value is then padded as described in PKCS#1 v1.5 block type 02 (see [175] and Figure N.2) : a zero byte is added to the left, as well as as many non-zero random bytes as necessary in such a way that the first two bytes of the final value are 00 02 followed by as many nonzero random bytes as necessary, and the rest. Note that this formatting is applied to both RSA and ElGamal encryption,

One-octet identifier of the symmetric encryption algorithm	Key of the symmetric encryption algorithm	Two-octet checksum over the key bytes

FIG. N.1 – Session key format in OpenPGP.

even though PKCS#1 v1.5 was only designed for RSA encryption. The randomness required to generate nonzero random bytes is obtained by a process following the principles of [132].

00	02	Non-zero random bytes	00	Message

FIG. N.2 – PKCS#1 v1.5 encryption padding, block type 02.

N.2.2 Signature

GPG supports the following signature schemes : RSA, DSA and ElGamal. The current GPG FAQ includes the following comment : *As for the key algorithms, you should stick with the default (i.e., DSA signature and ElGamal encryption). An ElGamal signing key has the following disadvantages : the signature is larger, it is hard to create such a key useful for signatures which can withstand some real world attacks, you don't get any extra security compared to DSA, and there might be compatibility problems with certain PGP versions. It has only been introduced because at the time it was not clear whether there was a patent on DSA.* The `README` file of GPG includes the following comment : *ElGamal for signing is available, but because of the larger size of such signatures it is strongly deprecated (Please note that the GnuPG implementation of ElGamal signatures is *not* insecure).* Thus, ElGamal signatures are not really recommended (mainly for efficiency reasons), but they are nevertheless supported by GPG, and they were not supposed to be insecure.

When RSA or ElGamal is used, the message is first hashed (using the hash function selected by the user), and the hash value is encoded as described in PKCS#1 v1.5 (see [175] and Figure N.3) : a certain constant (depending on the hash function) is added to the left, then a zero byte is added to the left, as well as as many FF bytes as necessary in such a way that the first two bytes of the final value are 00 01 followed by the FF bytes and the rest. With DSA, there is no need to apply a

00	01	FF bytes	00	Constant	Hashed message

FIG. N.3 – PKCS#1 v1.5 signature padding, block type 01.

signature padding, as the DSS standard completely specifies how a message is signed.

The randomness required by ElGamal and DSA is obtained by a process following the principles of [132].

N.3 The Implementation of ElGamal

GPG uses the same key generation for signature and encryption. It implements ElGamal in a multiplicative group $\mathbb{Z}_p^*$ (where p is a large prime) with generator g. The private key is denoted by x, and the corresponding public key is $y = g^x \pmod p$.

N.3.1 Key generation

The large prime number p is chosen in such a way that the factorization of $p-1$ is completely known and all the prime factors of $(p-1)/2$ have bit-length larger than a threshold q_{bit} depending on the requested bit-length of p. The correspondance between the size of p and the threshold is given by the so-called Wiener table (see Figure N.4) : notice that $4q_{bit}$ is always less than the bit-length of p.

Bit-length of p	512	768	1024	1280	1536	1792	2048	2304	2560	2816	3072	3328	3584	3840
q_{bit}	119	145	165	183	198	212	225	237	249	259	269	279	288	296

FIG. N.4 – The Wiener table used to generate ElGamal primes.

Once p is selected, a generator g of $\mathbb{Z}_p^*$ is found by testing successive potential generators (thanks to the known factorization of $p-1$), starting with the number 3 : If 3 turns out not to be a generator, then one tries with 4, and so on. The generation of the pair (p, g) is specified in the procedure `generate_elg_prime` of the file `cipher/primegen.c`. The generation of the pair of public and private keys is done in the procedure `generate` of the file `cipher/elgamal.c`. Although the generator g is likely to be small, we note that because all the factors of $(p-1)/2$ have at least $q_{bit} \geq 119$ bits, and $g > 2$, Bleichenbacher's forgery [31] of ElGamal signatures does not seem to apply here.

The private exponent x is not chosen as a pseudo-random number modulo $p-1$, although GPG makes the following comment : *select a random number which has these properties :* $0 < x < p-1$. *This must be a very good random number because this is the secret part.* Instead, x is chosen as a pseudo-random number of bit-length $3q_{bit}/2$, which is explained by the following comment (and which somehow contradicts the previous one) : *I don't see a reason to have a* x *of about the same size as the* p. *It should be sufficient to have one about the size of* q *or the later used* k *plus a large safety margin. Decryption will be much faster with such an* x. Thus, one chooses an x much smaller than p to speed-up the private operations. Unfortunately, we will see that this has implications on the security of GPG–ElGamal signatures.

N.3.2 Signature

Description.

The signature of a message already formatted as an integer m modulo p (as decribed in Section N.2.2), is the pair (a, b) where : $a = g^k \bmod p$ and $b = (m - ax)k^{-1} \pmod{p-1}$. The integer k is a "random" number coprime with $p-1$, which must be generated at each signature. GPG verifies a signature (a, b) by checking that $0 < a < p$ and $y^a a^b \equiv g^m \pmod{p}$. We note that such a signature verification does not prevent malleability (see [331] for a discussion on malleability) : if (a, b) is a valid signature of m, then $(a, b + u(p-1))$ is another valid signature of m for all integer u, because there is no range check over b. This is a minor problem, but there is worse.

Theoretically, k should be a cryptographically secure random number modulo $p-1$ such that k is coprime to $p-1$. Recent attacks on discrete-log signature schemes (see [261, 35, 159]) have shown that any leakage of information (or any peculiar property) on k may enable an attacker to recover the signer's private key in a much shorter time than what is usually required to solve discrete logarithms, provided that sufficiently many message/signature pairs are available. Intuitively, if partial information on k is available, each message/signature pair discloses information on the signer's private key, even though the signatures use different k's : when enough such pairs are gathered, it might become possible to extract the private key. Unfortunately, the GPG generation of k falls short of such recommendations.

The generation of k is described in the procedure `gen_k` of the file `cipher/elgamal.c`. It turns out that k is first chosen with $3q_{bit}/2$ pseudo-random bits (as in the generation of the private exponent

x, except that k may have less than $3q_{bit}/2$ bits). Next, as while as k is not coprime with $p-1$, k is incremented. Obviously, the final k is much smaller than p, and therefore far from being uniformly distributed modulo $p-1$: the bit-length of k should still be around $3q_{bit}/2$, while that of p is at least $4q_{bit}$. This is explained in the following comment : *IMO using a k much lesser than p is sufficient and it greatly improves the encryption performance. We use Wiener's table and add a large safety margin.* One should bear in mind that the same generation of k is used for both encryption and signature. However, the choice of a small k turns out to be dramatic for signature, rather than for encryption.

Attacking GPG–ElGamal signatures.

Independently of the choice of the private exponent x, because k is much smaller than $p-1$, one could apply the lattice-based attack of Nguyen-Shparlinski [261] with very slight modifications, provided that a few signatures are known. However, because x is so small, there is even a simpler attack, using only a single signature ! Indeed, we have the following congruence :

$$bk + ax \equiv m \pmod{p-1}. \tag{N.1}$$

In this congruence, only k and x are unknowns, and they are unusually small : From Wiener's table (Figure N.4), we know that k and x are much smaller than $\sqrt{p}$.

Linear congruences with small unknowns occur frequently in public-key cryptanalysis (see for instance the survey [267]), and they are typically solved with lattice techniques. We assume that the reader is familiar with lattice theory (see Appendix N.5 and the references of [267]). Following the classical strategy described in [267], we view the problem as a closest vector problem in a two-dimensional lattice, using lattices defined by a single linear congruence. The following lemma introduces the kind of two-dimensional lattices we need :

Lemma N.14 *Let $(\alpha, \beta) \in \mathbb{Z}^2$ and n be a positive integer. Let d be the greatest common divisor of α and n. Let e be the greatest common divisor of α, β and n. Let L be the set of all $(u, v) \in \mathbb{Z}^2$ such that $\alpha u + \beta v \equiv 0 \ (mod\, n)$. Then :*

1. *L is a two-dimensional lattice in $\mathbb{Z}^2$.*
2. *The determinant of L is equal to n/e.*
3. *There exists $u \in \mathbb{Z}$ such that $\alpha u + (\beta/e)d \equiv 0 \ (mod\, n)$.*
4. *The vectors $(n/d, 0)$ and $(u, d/e)$ form a basis of L.*

Proof. By definition, L is a subgroup of $\mathbb{Z}^2$, hence a lattice. Besides, L contains the two linearly independent vectors $(n, 0)$ and $(0, n)$, which proves statement 1. Let f be the function that maps $(u, v) \in \mathbb{Z}^2$ to $\alpha u + \beta v$ modulo n. f is a group morphism between $\mathbb{Z}^2$ and the additive group $\mathbb{Z}_n$. The image of f is the subgroup (of $\mathbb{Z}_n$) spanned by the greatest common divisor of α and β : it follows that the image of f has exactly n/e elements. Since L is the kernel of f, we deduce that the index of L in $\mathbb{Z}^2$ is equal to n/e, which proves statement 2. Statement 3 holds because the greatest common divisor of α and n is d, which divides the integer $(\beta/e)d$. By definition of u, the vector $(u, d/e)$ belongs to L. Obviously, the vector $(n/d, 0)$ belongs to L. But the determinant of those two vectors is n/e, that is, the determinant of L. This proves statement 4. □

□

We use the following lattice :

$$L = \{(u, v) \in \mathbb{Z}^2 : bu + av \equiv 0 \pmod{p-1}\}. \tag{N.2}$$

By Lemma N.14, a basis of L can easily be found. We then compute an arbitrary pair $(k', x') \in \mathbb{Z}^2$ such that $bk' + ax' \equiv m \pmod{p-1}$. To do so, we can apply the extended Euclidean algorithm to express the greatest common divisor of a, b and $p-1$ as a linear combination of a, b and $p-1$. This

gcd must divide m by (N.1), and therefore, a suitable multiplication of the coefficients of the linear combination gives an appropriate (k', x').

The vector $\mathbf{l} = (k' - k, x' - x)$ belongs to L and is quite close to the vector $\mathbf{t} = (k' - 2^{3q_{bit}/2-1}, x' - 2^{3q_{bit}/2-1})$. Indeed, k has about $3q_{bit}/2$ bits and x has exactly $3q_{bit}/2$ bits, therefore the distance between $\mathbf{l}$ and $\mathbf{t}$ is about $2^{(3q_{bit}-1)/2}$, which is much smaller than $\det(L)^{1/2}$, because from Lemma N.14 :

$$\det(L) = \frac{p-1}{\gcd(a, b, p-1)}.$$

From the structure of $p-1$ and the way a and b are defined, we thus expect $\det(L)$ to be around p. Hence, we can hope that $\mathbf{l}$ is the closest vector of $\mathbf{t}$ in the lattice L, due to the huge size difference between $2^{(3q_{bit}-1)/2}$ and $\sqrt{p}$, for all the values of q_{bit} given by Figure N.4 : this heuristic reasoning is frequent in lattice-based cryptanalysis, here it can however be proved if we assume that the distribution of a and b is uniform modulo $p-1$ (see Appendix N.6). If $\mathbf{l}$ is the closest vector of $\mathbf{t}$, $\mathbf{l}$ and therefore the private exponent x can be recovered from a two-dimensional closest vector computation (we know t and a basis of L). And such a closest vector computation can be done in quadratic time (see for instance [263]), using the classical Gaussian algorithm for two-dimensional lattice reduction. Figure N.5 sums up the attack, which clearly runs in polynomial time.

Input : The public parameters and a GPG–ElGamal signature (a, b) of m.
Expected output : The signer's private exponent x.

1. Compute a basis of the lattice L of (N.2), using statement 4 of Lemma N.14.
2. Compute $(k', x') \in \mathbb{Z}^2$ such that $bk' + ax' \equiv m \pmod{p-1}$, using the Euclidean algorithm.
3. Compute the target vector $\mathbf{t} = (k' - 2^{3q_{bit}/2-1}, x' - 2^{3q_{bit}/2-1})$.
4. Compute the lattice vector $\mathbf{l}$ closest to $\mathbf{t}$ in the two-dimensional lattice L.
5. Return x' minus the second coordinate of $\mathbf{l}$.

FIG. N.5 – An attack using a single GPG–ElGamal signature.

Alternatively, if one wants to program as less as possible, one can mount another lattice-based attack, by simply computing a shortest vector of the 4-dimensional lattice L' spanned by the following row vectors, where K is a large constant :

$$\begin{pmatrix} (p-1)K & 0 & 0 & 0 \\ -mK & 2^{3q_{bit}/2} & 0 & 0 \\ bK & 0 & 1 & 0 \\ aK & 0 & 0 & 1 \end{pmatrix}$$

This shortest vector computation can be done in quadratic time using the lattice reduction algorithm of [263]. The lattice L' contains the short vector $(0, 2^{3q_{bit}/2}, k, x)$ because of (N.1). This vector is expected to be a shortest lattice vector under roughly the same condition on q_{bit} and p as in the previous lattice attack (we omit the details). Thus, for all values of Wiener's table (see Figure N.4), one can hope to recover the private exponent x as the absolute value of the last coordinate of any shortest nonzero vector of L'.

We implemented the last attack with Shoup's NTL library [319], using the integer LLL algorithm to obtain short vectors. In our experiments, the attack worked for all the values of Wiener's table, and the total running time was negligible (less than a second).

Practical impact.

We have shown that GPG's implementation of the ElGamal signature is totally insecure : an attacker can recover the signer's private key from the public key and a single message/signature pair in less than a second. Thus, GPG–ElGamal signing keys should be be considered compromised, as announced by the GPG development team [186]. There are two types of ElGamal signing keys in GPG :

- The primary ElGamal sign+encrypt keys. When running the command `gpg -list-keys`, such keys can be spotted by a prefix of the form `pub 2048G/` where `2048` can be replaced by any possible keylength. The prefix `pub` specifies a primary key, while the capital letter `G` indicates an ElGamal sign+encrypt key.
- The ElGamal sign+encrypt subkeys. When running the command `gpg -list-keys`, such keys can be spotted by a prefix of the form `sub 2048G/` where `2048` can be replaced by any possible keylength. The prefix `sub` indicates a subkey.

The primary keys are definitely compromised because such keys automatically come up with a signature to bind the user identity to the public key, thus disclosing the private key immediately. The subkeys may not be compromised if no signature has ever been generated. In both cases, it is worth noting that the signing key is also an encryption key, so the damage is not limited to authentication : a compromised ElGamal signing key would also disclose all communications encrypted with the corresponding public key.

The mistake of using a small k and a small x dates back to GPG v1.0.2 (which was released in January 2000), when the generation of k and x was changed to improve performances : the flaw has therefore been present in GPG for almost four years. A signing key created prior to GPG v1.0.2 may still be compromised if a signature using that key has been generated with GPG v1.0.2 or later.

Nobody knows how many ElGamal sign+encrypt keys there are. What one knows is the number of ElGamal sign+encrypt keys that have been registered on keyservers. According to keyserver statistics (see [186]), there are 848 registered primary ElGamal sign+encrypt keys (which is a mere 0.04% percent of all primary keys on keyservers) and 324 registered ElGamal sign+encrypt subkeys : of course, GPG advised all the owners of such keys to revoke their keys. These (fortunately) small numbers can be explained by the fact that ElGamal signing keys were never GPG's default option for signing, and their use was not really advocated.

As a consequence, ElGamal signatures and ElGamal sign+encrypt keys have recently been removed from GPG, and the GNU/Linux distributions which include GPG have been updated accordingly.

N.3.3 Encryption

Let m be the message to be encrypted. The message m is formatted in the way described in Section N.2.1. The ciphertext is the pair (a, b) where : $a = g^k \bmod p$ and $b = my^k \bmod p$. The integer k is a "random" number coprime with $p - 1$. Theoretically, k should be a cryptographically secure random number modulo $p - 1$ such that k is coprime to $p - 1$. But the generation of k is performed using the same procedure `gen_k` called by the ElGamal signature generation process. Thus, k is first selected with $3q_{bit}/2$ pseudo-random bits. Next, as while as k is not coprime with $p - 1$, k is incremented. Hence, k is much smaller than $p - 1$.

The security assumption for the hardness of decryption is no longer the standard Diffie-Hellman problem : instead, this is the Diffie-Hellman problem with short exponents (see [279]). Because the key generation makes sure that all the factors of $(p - 1)/2$ have bit-length $\geq q_{bit}$, the best attack known to recover the plaintext requires at least $2^{q_{bit}/2}$ time, which is not a real threat.

However, the session key is formatted according to a specific padding, PKCS#1 v1.5 block type 02, which does not provide chosen-ciphertext security (see [33]). If we had access to a validity-checking oracle (which is weaker than a decryption oracle) that tells whether or not a given ciphertext is

the ElGamal encryption of a message formatted with PKCS#1 v1.5 block type 02, we could apply Bleichenbacher's attack [33] to decrypt any ciphertext. Indeed, even though Bleichenbacher's attack was originally described with RSA, it also applies to ElGamal due to its homomorphic property : if (a, b) and (a', b') are ElGamal ciphertexts of respectively m and m', then $(aa' \bmod p, bb' \bmod p)$ is an ElGamal ciphertext of $mm' \bmod p$. One could argue that a validity-checking oracle is feasible in the situation where a user has configured his software to automatically decrypt any encrypted emails he receives : if an encrypted email turns out not to be valid, the user would inform the sender. However, Bleichenbacher's attack require a large number of oracle calls, which makes the attack debatable in an email context. Nevertheless, it would be better if OpenPGP recommended a provably secure variant of ElGamal encryption such as ACE-KEM selected by NESSIE [96].

N.4 The Implementation of RSA

N.4.1 Key generation

To generate the parameters p, q, n, e, d, GPG implements the process described in Figure N.6. Although the process does not lead to any realistic attack, it is worth noting that the process leaks

Input : Bit-length k of the RSA modulus.
1. Repeat
2. Generate a pseudo-random prime p of $k/2$ bits.
3. Generate a pseudo-random prime q of $k/2$ bits.
4. If $p > q$, swap p and q.
5. $n \longleftarrow pq$.
6. Until the bit-length of n is equal to k.
7. If 41 is coprime with $\varphi(n)$, then $e \longleftarrow 41$
8. Else if 257 is coprime with $\varphi(n)$, then $e \longleftarrow 257$
9. Else
10. $e \longleftarrow 65537$
11. While e is not coprime with $\varphi(n)$, $e \longleftarrow e + 2$
12. Let d be the inverse of e modulo $\varphi(n)$.

Fig. N.6 – The RSA key generation in GnuPG.

information on the private key. Indeed, the value of the RSA public exponent e discloses additional information on $\varphi(n)$. For instance, if we see a GPG–RSA public key with $e \geq 65539$, we know that $\varphi(n)$ is divisible by the prime numbers 41, 257 and 65537 : we learn a 30-bit factor of $\varphi(n)$, namely $41 \times 257 \times 65537$. However, the probability of getting $e \geq 65539$ after the process is very small. To our knowledge, efficient attacks to factor n from partial knowledge of $\varphi(n)$ require a factor of $\varphi(n)$ larger than approximately $n^{1/4}$. Thus, this flaw does not lead to a serious attack, since the probability of getting a factor $\geq n^{1/4}$ after the process is way too small.

Nevertheless, any leakage on $\varphi(n)$ (apart from the fact that e is coprime with $\varphi(n)$) is not recommended : if one really wants a small public exponent, one should rather select e first, and then generate the primes p and q until both $p - 1$ and $q - 1$ are coprime with e.

N.4.2 Encryption

As already mentioned in Section N.2, GPG implements RSA encryption as defined by PKCS#1 v1.5. This is not state-of-the-art cryptography : like with ElGamal, Bleichenbacher's chosen-ciphertext attack [33] can decrypt any ciphertext. But, as mentioned in N.3.3, the relevance of such attacks to

the email world is debatable, in part because of the high number of oracle calls. We hope that future versions of the OpenPGP standard, will recommend better RSA encryption standards (see for instance PKCS#1 v2.1 [197] or NESSIE [96]).

N.4.3 Signature

GPG implements RSA signatures as defined by PKCS#1 v1.5. Again, this is not state-of-the-art cryptography (no security proof is known for this padding), but we are unaware of any realistic attack with the GPG setting, as opposed to some other paddings (see [74]). The RSA verification does not seem to check the range of the signature with respect to the modulus, which gives (marginal) malleability (see [331]) : given a signature s of m, one can forge another signature s' of m. As with encryption, we hope that future versions of the OpenPGP standard will recommend a better RSA signature standard (see for instance PKCS#1 v2.1 [197] or NESSIE [96]).

N.5 Annex : Lattices in a nutshell

We recall basic facts about lattices. To learn more about lattices, see [267] for a list of references. Informally speaking, a lattice is a regular arrangement of points in n-dimensional space. In this paper, by the term lattice, we actually mean an integral lattice.

An integral lattice is a subgroup of $(\mathbb{Z}^n, +)$, that is, a non-empty subset L of $\mathbb{Z}^n$ which is stable by subtraction : $\mathbf{x} - \mathbf{y} \in L$ whenever $(\mathbf{x}, \mathbf{y}) \in L^2$. The simplest lattice is $\mathbb{Z}^n$. It turns out that in any lattice L, not just $\mathbb{Z}^n$, there must exist linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_d \in L$ such that :

$$L = \left\{ \sum_{i=1}^{d} n_i \mathbf{b}_i \mid n_i \in \mathbb{Z} \right\}.$$

Any such d-uple of vectors $\mathbf{b}_1, \ldots, \mathbf{b}_d$ is called a basis of L : a lattice can be represented by a basis, that is, a matrix. Reciprocally, if one considers d integral vectors $\mathbf{b}_1, \ldots, \mathbf{b}_d \in \mathbb{Z}^n$, the previous set of all integral linear combinations of the $\mathbf{b}_i$'s is a subgroup of $\mathbb{Z}^n$, and therefore a lattice.

The *dimension* of a lattice L is the dimension d of the linear span of L : any basis of L has exactly d elements. It turns out that the d-dimensional volume of the parallelepiped spanned by an arbitrary basis of L only depends on L, not on the basis itself : this volume is called the *determinant* (or *volume*) of L. When the lattice is full-rank, that is, when the lattice dimension d equals the space dimension n, the determinant of L is simply the absolute value of the determinant of any basis. Thus, the volume of $\mathbb{Z}^n$ is 1.

Since our lattices are subsets of $\mathbb{Z}^n$, they must have a shortest nonzero vector : In any lattice $L \subseteq \mathbb{Z}^n$, there is at least one nonzero vector $\mathbf{v} \in L$ such that no other nonzero lattice vector has a Euclidean norm strictly smaller than that of $\mathbf{v}$. Finding such a vector $\mathbf{v}$ from a basis of L is called the shortest vector problem. When the lattice dimension is fixed, it is possible to solve the shortest vector problem in polynomial time (with respect to the size of the basis), using lattice reduction techniques. But the problem becomes much more difficult if the lattice dimension varies. In this article, we only deal with low-dimensional lattices, so the shortest vector problem is really not a problem.

The lattice determinant is often used to estimate the size of short lattice vectors. In a typical d-dimensional lattice L, if one knows a nonzero vector $\mathbf{v} \in L$ whose Euclidean norm is much smaller than $\det(L)^{1/d}$, then this vector is likely to be the shortest vector, in which case it can be found by solving the shortest vector problem, because any shortest vector would be expected to be equal to $\pm\mathbf{v}$. Although this can sometimes be proved, this is not a theorem : there are counter-examples, but it is often true with the lattices one is faced with in practice, which is what we mean by a typical lattice.

Another problem which causes no troubles when the lattice dimension is fixed is the closest vector problem : given a basis of $L \subseteq \mathbb{Z}^n$ and a point $\mathbf{t} \in \mathbb{Q}^n$, find a lattice vector $\mathbf{l} \in L$ minimizing the Euclidean norm of $\mathbf{l} - \mathbf{t}$. Again, in a typical d-dimensional lattice L, if one knows a vector $\mathbf{t}$ and a lattice vector $\mathbf{l}$ such that the norm of $\mathbf{t} - \mathbf{l}$ is much smaller than $\det(L)^{1/d}$, then $\mathbf{l}$ is likely to be the closest lattice vector of $\mathbf{t}$ in L, in which case $\mathbf{l}$ can be found by solving the closest vector problem. Indeed, if there was another lattice vector $\mathbf{l}'$ close to $\mathbf{t}$, then $\mathbf{l} - \mathbf{l}'$ would be a lattice vector of norm much smaller than $\det(L)^{1/d}$: it should be zero.

N.6 Annex : Proving the GPG–ElGamal attack

We use the same notation as in Section N.3.2. Let (a, b) be an GPG–ElGamal signature of m. If we make the simplifying assumption that both a and b are uniformly distributed modulo $p - 1$, then the attack of Figure N.5 can be proved, using the following lemma (which is not meant to be optimal, but is sufficient for our purpose) :

Lemma N.15 *Let $\varepsilon > 0$. Let p be a prime number such that all the prime factors of $(p-1)/2$ are $\geq 2^{q_{bit}}$. Let a and b be chosen uniformly at random over $\{0, \ldots, p-2\}$. Let L be the lattice defined by* (N.2)*. Then the probability (over the choice of a and b) that there exists a non-zero $(u, v) \in L$ such that both $|u|$ and $|v|$ are $< 2^{3q_{bit}/2+\varepsilon}$ is less than :*

$$\frac{2^{7q_{bit}/2+5+3\varepsilon} \log_2 p}{(p-1)q_{bit}}.$$

Proof. This probability P is less than the sum of all the probabilities $P_{u,v}$, where the sum is over all the $(u, v) \neq (0, 0)$ such that both $|u|$ and $|v|$ are $< 2^{3q_{bit}/2+\varepsilon}$, and $P_{u,v}$ denotes the probability (over the choice of a and b) that $(u, v) \in L$. Let $(u, v) \in \mathbb{Z}^2$ be fixed and nonzero. If $v = 0$, there are at most $2\gcd(u, (p-1)/2)$ values of b in the set $\{0, \ldots, p-2\}$ such that :

$$bu + av \equiv 0 \pmod{(p-1)/2} \tag{N.3}$$

It follows that :

$$P_{u,0} \leq \frac{2\gcd(u, (p-1)/2)}{p-1}.$$

If $v \neq 0$: for any b, there are at most $2\gcd(v, (p-1)/2)$ values of a in the set $\{0, \ldots, p-2\}$ which satisfy (N.3), therefore :

$$P_{u,v} \leq \frac{2\gcd(v, (p-1)/2)}{p-1}.$$

Hence :

$$P \leq S + 2^{3q_{bit}/2+1+\varepsilon} S \leq 2^{3q_{bit}/2+2+\varepsilon} S,$$

where

$$S = \sum_{0<|u|<2^{3q_{bit}/2+\varepsilon}} \frac{2\gcd(u, (p-1)/2)}{p-1} = \sum_{0<|v|<2^{3q_{bit}/2+\varepsilon}} \frac{2\gcd(v, (p-1)/2)}{p-1}.$$

To bound S, we split the sum in two parts, depending on whether or not $\gcd(u, (p-1)/2) > 1$. If $\gcd(u, (p-1)/2) > 1$, then $\gcd(u, (p-1)/2) \leq |u| < 2^{3q_{bit}/2+\varepsilon}$ and u must be divisible by a prime factor of $(p-1)/2$ which is necessarily $\geq 2^{q_{bit}}$: the number of such u's is less than $2^{q_{bit}/2+1+\varepsilon}(\log_2 p)/q_{bit}$ because the number of prime factors of $(p-1)/2$ is less than $(\log_2 p)/q_{bit}$. We obtain :

$$\begin{aligned} S &\leq 2^{3q_{bit}/2+1+\varepsilon} \times \frac{2}{p-1} + 2^{q_{bit}/2+1+\varepsilon}(\log_2 p)/q_{bit} \times \frac{2^{3q_{bit}/2+1+\varepsilon}}{p-1} \\ &\leq \frac{2^{2q_{bit}+3+2\varepsilon} \log_2 p}{(p-1)q_{bit}} \end{aligned}$$

This completes the proof since $P \leq 2^{3q_{bit}/2+2+\varepsilon}S$. □

□

Because p is always much larger than $2^{4q_{bit}}$, the lemma shows that if ε is not too big, then with overwhelming probability, there is no non-zero $(u, v) \in L$ such that both $|u|$ and $|v|$ are $< 2^{3q_{bit}/2+\varepsilon}$. If $\mathbf{l}$ was not the closest vector of $\mathbf{t}$ in L, there would be another lattice vector $\mathbf{l}' \in L$ closer to $\mathbf{t}$: the distance between $\mathbf{l}'$ and $\mathbf{t}$ would be less than $2^{(3q_{bit}-1)/2}$. But then, the lattice vector $(u, v) = \mathbf{l} - \mathbf{l}'$ would contradict the lemma, for some small ε. Hence, $\mathbf{l}$ is the closest vector of $\mathbf{t}$ in L with overwhelming probability, which proves the attack. However, the initial assumption that both a and b are uniformly distributed modulo $p - 1$ is an idealized model, compared to the actual way a and b are generated by GPG. In this sense, the lemma explains why the attack works, but it does not provide a complete proof.

Bibliographie

[1] L. M. Adleman. On breaking generalized knapsack publick key cryptosystems. In *Proc. of the 15th Symposium on the Theory of Computing*, pages 402–412. ACM, 1983.

[2] D. Aharonov and O. Regev. Lattice problems in NP ∩ coNP. *J. ACM*, 52(5) :749–765 (electronic), 2005.

[3] M. Ajtai. Generating random lattices according to the invariant distribution. Draft of March 2006.

[4] M. Ajtai. Generating hard instances of lattice problems. In *Proc. of the 28th Symposium on the Theory of Computing*, pages 99–108. ACM, 1996. Disponible chez [89] comme TR96-007.

[5] M. Ajtai. The shortest vector problem in L_2 is NP-hard for randomized reductions. In *Proc. of the 30th Symposium on the Theory of Computing*. ACM, 1998. Disponible chez [89] comme TR97-047.

[6] M. Ajtai. Random lattices and a conjectured 0 - 1 law about their polynomial time computable properties. In *Proc. 43rd Symposium on Foundations of Computer Science (FOCS 2002)*, pages 733–742, 2002.

[7] M. Ajtai. The worst-case behavior of Schnorr's algorithm approximating the shortest nonzero vector in a lattice. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, pages 396–406 (electronic), New York, 2003. ACM.

[8] M. Ajtai, R. Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, pages 601–610 (electronic), New York, 2001. ACM.

[9] A. Akhavi. Worst-case complexity of the optimal LLL algorithm. In *Proceedings of the 2000 Latin American Theoretical Informatics (LATIN 2000)*, volume 1776 of *Lecture Notes in Computer Science*, pages 355–366. Springer, 2000.

[10] A. Akhavi and C. Moreira dos Santos. Another view of the Gaussian algorithm. In *Proceedings of the 2004 Latin American Theoretical Informatics (LATIN 2004)*, volume 2976 of *Lecture Notes in Computer Science*, pages 474–487. Springer, 2004.

[11] M.-L. Akkar. *Attaques et Méthodes de Protections de Systèmes Cryptographiques Embarqués*. PhD thesis, Université de Versailles Saint-Quentin, 2004.

[12] N. Alon and J. H. Spencer. *The probabilistic method*. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley-Interscience [John Wiley & Sons], New York, second edition, 2000. With an appendix on the life and work of Paul Erdős.

[13] S. Ar, R. Lipton, R. Rubinfeld, and M. Sudan. Reconstructing algebraic functions from mixed data. *SIAM J. of Computing*, 28(2) :488–511, 1999.

[14] S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *Journal of Computer and System Sciences*, 54(2) :317–331, 1997.

[15] A. O. L. Atkin. The number of points on an elliptic curve modulo a prime. Email on the Number Theory mailing list, 1988.

[16] A. O. L. Atkin. The number of points on an elliptic curve modulo a prime. Email on the Number Theory mailing list, 1991.

[17] L. Babai. On Lovász lattice reduction and the nearest lattice point problem. *Combinatorica*, 6 :1–13, 1986.

[18] W. Backes and S. Wetzel. Heuristics on lattice reduction in practice. *ACM Journal of Experimental Algorithms*, 7 :1, 2002.

[19] F. Bahr, M. Boehm, J. Franke, and T. Kleinjung. Factorization of RSA-200. Public announcement on May 9th., 2005.

[20] F. Bao, R. H. Deng, Y. Han, A. B. Jeng, A. D. Narasimhalu, and T.-H. Ngair. Breaking public key cryptosystems on tamper resistant devices in the presence of transient faults. In *Security Protocols, 5th International Workshop, Paris, France, April 7-9, 1997, Proceedings*, volume 1361 of *Lecture Notes in Computer Science*, pages 115–124. Springer, 1998.

[21] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan. The sorcerers apprentice guide to fault attacks. *Proceedings of the IEEE*, 94(2), 2006.

[22] C. Batut, K. Belabas, D. Bernardi, H. Cohen, and M. Olivier. PARI/GP computer package version 2. Université de Bordeaux I.

[23] M. Bellare, S. Goldwasser, and D. Micciancio. "Pseudo-random" number generation within cryptographic algorithms : The DSS case. In *Proc. of Crypto '97*, volume 1294 of *Lecture Notes in Computer Science*. IACR, Springer, 1997.

[24] M. Bellare and P. Rogaway. Random oracles are practical : a paradigm for designing efficient protocols. In *Proc. of the 1st CCS*, pages 62–73. ACM Press, 1993.

[25] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *Proc. of Eurocrypt '94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. IACR, Springer, 1995.

[26] M. Bellare and P. Rogaway. The exact security of digital signatures - how to sign with RSA and Rabin. In *Proc. of Eurocrypt '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416. IACR, Springer, 1996.

[27] S. M. Bellovin. Cryptography and the internet. In *Proc. CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 46–55. Springer, 1998.

[28] A.-M. Bergé. Symplectic lattices. In *Quadratic forms and their applications (Dublin, 1999)*, volume 272 of *Contemp. Math.*, pages 9–22. Amer. Math. Soc., Providence, RI, 2000.

[29] E. Biham and A. Shamir. Differential fault analysis of secret key cryptosystems. In *Proc. CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer, 1997.

[30] I. F. Blake, G. Seroussi, and N. Smart. *Advances in Elliptic Curve Cryptography*, volume 317 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, 2004.

[31] D. Bleichenbacher. Generating ElGamal signatures without knowing the secret key. In *Proc. of Eurocrypt '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 10–18. IACR, Springer, 1996. Corrected version available from the author.

[32] D. Bleichenbacher. On the security of the KMOV public key cryptosystem. In *Proc. of Crypto '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 235–248. IACR, Springer, 1997.

[33] D. Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In *Proc. of Crypto '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 1–12. IACR, Springer, 1998.

[34] D. Bleichenbacher, 1999. Private communication.

[35] D. Bleichenbacher. On the generation of DSS one-time keys. Manuscript. The result was presented at the Monteverita workshop in March, 2001., February 2001.

[36] D. Bleichenbacher and P. Q. Nguyen. Noisy polynomial interpolation and noisy Chinese remaindering. In *Proc. of Eurocrypt '00*, volume 1807 of *Lecture Notes in Computer Science*. IACR, Springer-Verlag, 2000.

[37] H. F. Blichfeldt. The minimum values of positive quadratic forms in six, seven and eight variables. *Math. Z.*, 39(1) :1–15, 1935.

[38] J. Blömer and A. May. A tool kit for finding small roots of bivariate polynomials over the integers. In *Advances in Cryptology - Proc. of EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 251–267. Springer, 2005.

[39] M. I. Boguslavsky. Radon transforms and packings. *Discrete Appl. Math.*, 111(1-2) :3–22, 2001.

[40] D. Boneh. Twenty years of attacks on the RSA cryptosystem. *Notices of the AMS*, 46(2) :203–213, 1999.

[41] D. Boneh. Finding smooth integers in short intervals using CRT decoding. In *Proc. of 32nd STOC*. ACM, 2000.

[42] D. Boneh, R. A. DeMillo, and R. J. Lipton. On the importance of checking cryptographic protocols for faults. In *Proc. of Eurocrypt '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. IACR, Springer-Verlag, 1997.

[43] D. Boneh and G. Durfee. Cryptanalysis of RSA with private key d less than $n^{0.292}$. In *Proc. of Eurocrypt '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 1–11. Springer, 1999.

[44] D. Boneh, G. Durfee, and N. Howgrave-Graham. Factoring $n = p^r q$ for large r. In *Proc. of Crypto '99*, volume 1666 of *Lecture Notes in Computer Science*. Springer, 1999.

[45] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In *Advances in cryptology—CRYPTO 2001 (Santa Barbara, CA)*, volume 2139 of *Lecture Notes in Comput. Sci.*, pages 213–229. Springer, Berlin, 2001.

[46] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Comput.*, 32(3) :586–615 (electronic), 2003.

[47] D. Boneh, A. Joux, and P. Q. Nguyen. Why textbook ElGamal and RSA encryption are insecure. In *Proc. of Asiacrypt '00*, volume 1976 of *Lecture Notes in Computer Science*, pages 30–43. IACR, Springer, 2000.

[48] D. Boneh and R. Venkatesan. Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes. In *Proc. of Crypto '96*, Lecture Notes in Computer Science. IACR, Springer, 1996.

[49] D. Boneh and R. Venkatesan. Rounding in lattices and its cryptographic applications. In *Proc. of the 8th Symposium on Discrete Algorithms*, pages 675–681. ACM, 1997.

[50] D. Boneh and R. Venkatesan. Breaking RSA may not be equivalent to factoring. In *Proc. of Eurocrypt '98*, volume 1233 of *Lecture Notes in Computer Science*, pages 59–71. Springer, 1998.

[51] E. Brickell, D. M. Gordon, K. S. McCurley, and D. Wilson. Fast exponentiation with precomputation. In *Proc. of Eurocrypt '92*, volume 658 of *Lecture Notes in Computer Science*, pages 200–207. Springer, 1993.

[52] E. Brickell, D. Pointcheval, S. Vaudenay, and M. Yung. Design validations for discrete logarithm based signature schemes. In *Proc. of PKC '00*, volume 1751 of *Lecture Notes in Computer Science*, pages 276–292. Springer, 2000.

[53] D. R. L. Brown and A. J. Menezes. A small subgroup attack on a key agreement protocol of Arazi. Technical report, Dept. of Combinatorics and Optimization, Univ. of Waterloo, 2001. CORR 2001–50.

[54] P. Buser and P. Sarnak. On the period matrix of a Riemann surface of large genus. *Invent. Math.*, 117(1) :27–56, 1994. With an appendix by J. H. Conway and N. J. A. Sloane.

[55] J. Callas, L. Donnerhacke, H. Finney, and R. Thayer. OpenPGP message format : Request for Comments 2440. Available as `http ://www.ietf.org/rfc/rfc2440.txt`.

[56] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4) :557–594 (electronic), 2004.

[57] J. Cassels. *An Introduction to the Geometry of Numbers*. Springer, 1997. Réédition de 1971.

[58] J. W. S. Cassels. *Rational quadratic forms*, volume 13 of *London Mathematical Society Monographs*. Academic Press Inc. [Harcourt Brace Jovanovich Publishers], London, 1978.

[59] D. Catalano, P. Q. Nguyen, and J. Stern. The hardness of Hensel lifting : the case of RSA and discrete logarithm. In *Proc. of Asiacrypt '02*, volume 2501 of *Lecture Notes in Computer Science*, pages 299–310. IACR, Springer, 2002.

[60] S. Cavallar, B. Dodson, A. K. Lenstra, W. Lioen, P. L. Montgomery, B. Murphy, H. te Riele, K. Aardal, J. Gilchrist, G. Guillerm, P. Leyland, J. Marchand, F. Morain, A. Muffett, C. Putnam, C. Putnam, and P. Zimmermann. Factorization of 512-bit RSA key using the number field sieve. In *Proc. of Eurocrypt'2000*, Lecture Notes in Computer Science. IACR, Springer, 2000. Factorization announced in August, 1999.

[61] CEES. Efficient embedded security standards #1 : Implementation aspects of NTRU and NSS. Draft Version 3.0 available at `http ://www.ceesstandards.org`, July 2001.

[62] CEES. Efficient embedded security standards #1 : Implementation aspects of NTRU and NSS. Draft Version 1.0 available at `http ://www.ceesstandards.org`, November 2002.

[63] CEES. Efficient embedded security standards #1 : Implementation aspects of NTRUEncrypt and NTRUSign. Version 2.0 available at [164], June 2003.

[64] B. Chor and R. Rivest. A knapsack-type public key cryptosystem based on arithmetic in finite fields. *IEEE Trans. Inform. Theory*, 34, 1988.

[65] H. Cohen. *A Course in Computational Algebraic Number Theory*. Springer, 1995. Deuxième édition.

[66] H. Cohn and N. Elkies. New upper bounds on sphere packings. I. *Ann. of Math. (2)*, 157(2) :689–714, 2003.

[67] H. Cohn and A. Kumar. The densest lattice in twenty-four dimensions. *Electron. Res. Announc. Amer. Math. Soc.*, 10 :58–67 (electronic), 2004.

[68] J. Conway and N. Sloane. *Sphere Packings, Lattices and Groups*. Springer, 1998. Troisième édition.

[69] J. H. Conway and N. J. A. Sloane. A lattice without a basis of minimal vectors. *Mathematika*, 42 :175–177, 1995.

[70] D. Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *J. of Cryptology*, 10(4) :233–260, 1997.

[71] D. Coppersmith. Finding small solutions to small degree polynomials. In *Proc. of CALC '01*, volume 2146 of *Lecture Notes in Computer Science*. Springer, 2001.

[72] D. Coppersmith and A. Shamir. Lattice attacks on NTRU. In *Proc. of Eurocrypt '97*, volume 1233 of *Lecture Notes in Computer Science*. IACR, Springer, 1997.

[73] J.-S. Coron and A. May. Deterministic polynomial-time equivalence of computing the RSA secret key and factoring. *J. Cryptology*, 20(1) :39–50, 2007.

[74] J.-S. Coron, D. Naccache, and J. P. Stern. On the security of RSA padding. In *Proc. of Crypto '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 1–18. IACR, Springer, 1999.

[75] M. Coster, A. Joux, B. LaMacchia, A. Odlyzko, C.-P. Schnorr, and J. Stern. Improved low-density subset sum algorithms. *Comput. Complexity*, 2 :111–128, 1992.

[76] N. Courtois, A. Klimov, J. Patarin, and A. Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *Proc. of Eurocrypt'2000*, LNCS. Springer-Verlag, 2000.

[77] R. Crandall and C. Pomerance. *Prime Numbers – A Computational Perspective*. Springer, 2001.

[78] N. Cryptosystems. Technical reports. Available at `http ://www.ntru.com`, 2002.

[79] J. Daemen and V. Rijmen. The advanced encryption standard. `http ://csrc.nist.gov/encryption/aes/`.

[80] B. N. Delone and N. N. Sandakova. Theory of stereohedra. *Trudy Mathematics Institute Steklov*, 64 :28–51, 1961.

[81] R. Descombes. *Éléments de Théorie des Nombres*. Presses universitaires de France, 1986.

[82] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, IT-22 :644–654, Nov 1976.

[83] I. Dinur, G. Kindler, and S. Safra. Approximating CVP to within almost-polynomial factors is NP-hard. In *Proc. of the 39th Symposium on the Foundations of Computer Science*, pages 99–109. IEEE, 1998. Disponible chez [89] comme TR98-048.

[84] E. Dottax. Fault attacks on NESSIE signature and identification schemes. Technical report, NESSIE European Project, October 2002.

[85] M. Drmota and R. Tichy. *Sequences, discrepancies and applications*. Springer-Verlag, Berlin, 1997.

[86] V. Dubois, P.-A. Fouque, A. Shamir, and J. Stern. Practical cryptanalysis of SFLASH. In *Advances in Cryptology – Proceedings of CRYPTO '07*, Lecture Notes in Computer Science. Springer, 2007.

[87] G. Durfee and P. Q. Nguyen. Cryptanalysis of the RSA schemes with short secret exponent from Asiacrypt '99. In *Proc. of Asiacrypt '00*, volume 1976 of *Lecture Notes in Computer Science*. IACR, Springer, 2000.

[88] C. Dwork. *Lattices and Their Application to Cryptography*. Stanford University, 1998. Lecture Notes, Spring Quarter. Plusieurs chapitres sont traduits du cours de Claus Schnorr *Gittertheorie und algorithmische Geometrie, Reduktion von Gitterbasen un Polynomidealen* donné à l'université de Francfort en 1994.

[89] ECCC. `http ://www.eccc.uni-trier.de/eccc/`. The Electronic Colloquium on Computational Complexity.

[90] F. Eisenbrand and G. Rote. Fast reduction of ternary quadratic forms. In *Cryptography and lattices (Providence, RI, 2001)*, volume 2146 of *Lecture Notes in Comput. Sci.*, pages 32–44. Springer, Berlin, 2001.

[91] E. El Mahassni, P. Q. Nguyen, and I. E. Shparlinski. The insecurity of Nyberg–Rueppel and other DSA-like signature schemes with partially known nonce. In *Proc. Workshop on Cryptography and Lattices (CALC '01)*, volume 2146 of *Lecture Notes in Computer Science*, pages 97–109. Springer-Verlag, 2001.

[92] N. D. Elkies. Explicit isogenies. Draft, 1991.

[93] C. Ellison, C. Hall, R. Milbert, and B. Schneier. Protecting secret keys with personal entropy. *Future Generation Computer Systems*, 1999. To appear. Available at `http ://www.counterpane.com/`.

[94] P. Emde Boas. Another NP-complete problem and the complexity of computing short vectors in a lattice. Technical report, Mathematische Instituut, University of Amsterdam, 1981. Report 81-04.

[95] P. Erdös, P. M. Gruber, and J. Hammer. *Lattice points*, volume 39 of *Pitman Monographs and Surveys in Pure and Applied Mathematics*. Longman Scientific & Technical, 1989.

[96] European Union. European project IST-1999-12324 : New European Schemes for Signatures, Integrity, and Encryption (NESSIE). `http ://www.cryptonessie.org`.

[97] J. Faugère and A. Joux. Algebraic cryptanalysis of hidden field equation (HFE) cryptosystems using Gröbner bases. In *Proc. of Crypto '03*, volume 2729 of *Lecture Notes in Computer Science*, pages 44–60. Springer, 2003.

[98] U. Fincke and M. Pohst. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *Math. Comp.*, 44(170) :463–471, 1985.

[99] J. B. Friedlander and I. E. Shparlinski. On the distribution of diffie–hellman triples with sparse exponents. *SIAM J. Discr. Math.*, 14 :162–169, 2001.

[100] A. Frieze, M. Jerrum, and R. Kannan. Learning linear transformations. In *37th Annual Symposium on Foundations of Computer Science (Burlington, VT, 1996)*, pages 359–368. IEEE Comput. Soc. Press, Los Alamitos, CA, 1996.

[101] A. M. Frieze, J. Håstad, R. Kannan, J. C. Lagarias, and A. Shamir. Reconstructing truncated integer variables satisfying linear congruences. *SIAM J. Comput.*, 17(2) :262–280, 1988. Special issue on cryptography.

[102] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA–OAEP is secure under the RSA assumption. In *Proc. of Crypto '01*, Lecture Notes in Computer Science. IACR, Springer, 2001.

[103] N. Gama, N. Howgrave-Graham, H. Koy, and P. Q. Nguyen. Rankin's Constant and Blockwise Lattice Reduction. In *Advances in Cryptology – Proceedings of CRYPTO '06*, volume 4117 of *Lecture Notes in Computer Science*, pages 112–130. Springer, 2006.

[104] N. Gama, N. Howgrave-Graham, and P. Q. Nguyen. Symplectic Lattice Reduction and NTRU. In *Advances in Cryptology – Proceedings of EUROCRYPT '06*, volume 4004 of *Lecture Notes in Computer Science*, pages 233–253. Springer, 2006.

[105] N. Gama and P. Q. Nguyen. New chosen-ciphertext attacks on NTRU. In *Workshop on Practice and Theory in Public-Key Cryptography (PKC '07)*, volume 4450 of *Lecture Notes in Computer Science*, pages 89–106. Springer, 2007.

[106] N. Gama and P. Q. Nguyen. Finding short lattice vectors within Mordell's inequality. In *STOC '08 : Proc. 40th ACM Symposium on Theory of Computing*. ACM, 2008. A paraître.

[107] N. Gama and P. Q. Nguyen. Predicting lattice reduction. In *Advances in Cryptology – Proc. EUROCRYPT '08*. Springer, 2008. A paraître.

[108] T. E. Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inform. Theory*, 31 :469–472, 1985.

[109] C. Gauss. *Disquisitiones Arithmeticæ*. Leipzig, 1801.

[110] C. Gentry. Key recovery and message attacks on NTRU-composite. In *Proc. of Eurocrypt '01*, volume 2045 of *Lecture Notes in Computer Science*. IACR, Springer, 2001.

[111] C. Gentry, J. Jonsson, J. Stern, and M. Szydlo. Cryptanalysis of the NTRU signature scheme (NSS) from Eurocrypt 2001. In *Proc. of Asiacrypt '01*, volume 2248 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.

[112] C. Gentry and M. Szydlo. Cryptanalysis of the revised NTRU signature scheme. In *Proc. of Eurocrypt '02*, volume 2332 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002.

[113] N. Gilboa. Two party RSA key generation. In *Proc. of Crypto '99*, volume 1666 of *LNCS*, pages 116–129. Springer-Verlag, 1999.

[114] C. Giraud and E. W. Knudsen. Fault attacks on signature schemes. In *Proc. ACISP*, volume 3108 of *Lecture Notes in Computer Science*, pages 478–491. Springer, 2004.

[115] I. Goldberg and D. Wagner. Randomness and the Netscape browser. *Dr Dobb's*, January 1996.

[116] O. Goldreich and S. Goldwasser. On the limits of non-approximability of lattice problems. In *Proc. of the 30th Symposium on the Theory of Computing*. ACM, 1998. Disponible chez [89] comme TR97-031.

[117] O. Goldreich, S. Goldwasser, and S. Halevi. Challenges for the GGH cryptosystem. Série de cinq défis numériques disponibles à l'adresse `http ://theory.lcs.mit.edu/~shaih/challenge.html`.

[118] O. Goldreich, S. Goldwasser, and S. Halevi. Public-key cryptosystems from lattice reduction problems. In *Proc. of Crypto '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 112–131. IACR, Springer, 1997.

[119] O. Goldreich, D. Micciancio, S. Safra, and J.-P. Seifert. Approximating shortest lattice vectors is not harder than approximating closest lattice vectors. Disponible chez [89] comme TR99-002.

[120] O. Goldreich, D. Ron, and M. Sudan. Chinese remaindering with errors. In *Proc. of 31st STOC*. ACM, 1999. Also available at [89].

[121] D. Goldstein and A. Mayer. On the equidistribution of Hecke points. *Forum Math.*, 15(2) :165–189, 2003.

[122] G. Golub and C. Loan. *Matrix Computations*. Johns Hopkins Univ. Press, 1996.

[123] G. H. Golub and Charles F. Van Loan. *Matrix Computations*. The John Hopkins University Press, third edition, 1996.

[124] M. I. González Vasco and I. E. Shparlinski. On the security of Diffie-Hellman bits. In K.-Y. Lam, I. E. Shparlinski, H. Wang, and C. Xing, editors, *Proc. Workshop on Cryptography and Computational Number Theory (CCNT'99), Singapore*, pages 257–268. Birkhäuser, 2001.

[125] M. I. González Vasco and I. E. Shparlinski. Security of the most significant bits of the Shamir message passing scheme. *Math. Comp.*, 71(237) :333–342 (electronic), 2002.

[126] D. M. Gordon. A survey of fast exponentiation methods. *J. Algorithms*, 27(1) :129–146, 1998.

[127] L. Goubin. Théorie et pratique de la cryptologie sur carte à microprocesseur, 2003. Mémoire d'habilitation.

[128] GPG. The GNU privacy guard. `http ://www.gnupg.org`.

[129] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer, 1993.

[130] M. Gruber and C. G. Lekkerkerker. *Geometry of Numbers*. North-Holland, 1987.

[131] V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. *IEEE Trans. on Information Theory*, 45(6) :1757–1767, 1999. An extended abstract appeared in the Proc. of IEEE FOCS '98.

[132] P. Gutmann. Software generation of practically strong random numbers. In *Proc. of the 7th Usenix Security Symposium*, 1998.

[133] P. Gutmann. Lessons learned in implementing and deploying crypto software. In *Proc. of the 11th Usenix Security Symposium*, 2002.

[134] A. Haar. Der Massbegriff in der Theorie der kontinuierlichen Gruppen. *Ann. of Math. (2)*, 34(1) :147–169, 1933.

[135] T. C. Hales. A proof of the Kepler conjecture. *Ann. of Math. (2)*, 162(3) :1065–1185, 2005.

[136] T. C. Hales. Historical overview of the Kepler conjecture. *Discrete Comput. Geom.*, 36(1) :5–20, 2006.

[137] C. Hall, I. Goldberg, and B. Schneier. Reaction attacks against several public-key cryptosystems. In *Proc. of ICICS '99*, LNCS, pages 2–12. Springer-Verlag, 1999.

[138] G. Hanrot and D. Stehlé. Improved analysis of Kannan's shortest lattice vector algorithm. In *Advances in Cryptology – Proceedings of CRYPTO '07*, Lecture Notes in Computer Science. Springer, 2007.

[139] R. Harley, D. Doligez, D. de Rauglaudre, and X. Leroy. Ecc2k-108 challenge solved. Public announcement on April 4th., 2000.

[140] I. Haviv and O. Regev. Tensor-based hardness of the shortest vector problem to within almost polynomial factors. In *STOC'07 : Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, New York, 2007. ACM.

[141] B. Helfrich. Algorithms to construct Minkowski reduced and Hermite reduced bases. *Theoretical Computer Science*, 41 :125–139, 1985.

[142] C. Hermite. Extraits de lettres de M. Hermite à M. Jacobi sur différents objets de la théorie des nombres. *J. Reine Angew. Math.*, 40 :261–315, 1850. On trouve aussi ces lettres dans le premier tome (pp 100–135) de [143]. La première lettre est datée du 6 août 1845.

[143] C. Hermite. *Œuvres*. Gauthiers-Villars, 1905.

[144] N. J. Higham. The accuracy of floating point summation. *SIAM Journal on Scientific Computing*, 14 :783–799, 1993.

[145] J. J. Hoch and A. Shamir. Fault analysis of stream ciphers. In *Proc. CHES*, volume 3156 of *Lecture Notes in Computer Science*, pages 240–253. Springer, 2004.

[146] J. Hoffstein, N. A. H. Graham, J. Pipher, J. H. Silverman, and W. Whyte. NTRUSIGN : Digital signatures using the NTRU lattice. Full version of [148]. Draft of April 2, 2002, available on NTRU's website.

[147] J. Hoffstein, N. A. H. Graham, J. Pipher, J. H. Silverman, and W. Whyte. Performances improvements and a baseline parameter generation algorithm for NTRUsign. In *Proc. of Workshop on Mathematical Problems and Techniques in Cryptology*, pages 99–126. CRM, 2005.

[148] J. Hoffstein, N. A. Howgrave-Graham, J. Pipher, J. H. Silverman, and W. Whyte. NTRUSIGN : Digital signatures using the NTRU lattice. In *Proc. of CT-RSA*, volume 2612 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

[149] J. Hoffstein, J. Pipher, and J. Silverman. NTRU : A ring based public key cryptosystem. In *Algorithmic Number Theory – Proc. of ANTS-III*, volume 1423 of *Lecture Notes in Computer Science*. Springer, 1998.

[150] J. Hoffstein, J. Pipher, and J. H. Silverman. NSS : An NTRU lattice-based signature scheme. In *Proc. of Eurocrypt '01*, volume 2045 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.

[151] J. Hoffstein and J. H. Silverman. Optimizations for NTRU. In *Public-key Cryptography and Computational Number Theory*. DeGruyter, 2000. To appear, available at [78].

[152] J. Hoffstein and J. H. Silverman. Protecting NTRU against chosen ciphertext and reaction attacks. Technical report, NTRU Cryptosystems, June 2000. Report #016, version 1, available at [78].

[153] J. Hoffstein and J. H. Silverman. Random small Hamming weight products with applications to cryptography. *Discrete Appl. Math.*, 130(1) :37–49, 2003. The 2000 Com^{2}MaC Workshop on Cryptography (Pohang).

[154] J. Hong, J. W. Han, D. Kwon, and D. Han. Chosen-ciphertext attacks on optimized (ntru). Technical report, Cryptology ePrint Archive, 2002. Report 2002/188.

[155] N. Howgrave-Graham. Finding small roots of univariate modular equations revisited. In *Cryptography and Coding*, volume 1355 of *Lecture Notes in Computer Science*, pages 131–142. Springer, 1997.

[156] N. Howgrave-Graham, J. H. Silverman, A. Singer, and W. Whyte. NAEP : Provable security in the presence of decryption failures. Technical report, Cryptology ePrint Archive, 2003.

[157] N. A. Howgrave-Graham. Approximate integer common divisors. In *Proc. of CALC '01*, volume 2146 of *Lecture Notes in Computer Science*. Springer, 2001.

[158] N. A. Howgrave-Graham, P. Q. Nguyen, D. Pointcheval, J. Proos., J. H. Silverman, A. Singer, and W. Whyte. The impact of decryption failures on the security of NTRU encryption. In *Proc. of the 23rd Cryptology Conference (Crypto '03)*, volume 2729 of *Lecture Notes in Computer Science*, pages 226–246. IACR, Springer-Verlag, 2003.

[159] N. A. Howgrave-Graham and N. P. Smart. Lattice attacks on digital signature schemes. *Design, Codes and Cryptography*, 23 :283–290, 2001.

[160] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley & Sons, 2001.

[161] A. Hyvärinen and E. Oja. A fast fixed-point algorithm for independent component analysis. *Neural Computation*, 9(7) :1483–1492, 1997.

[162] IEEE. ANSI/IEEE standard 754-1985 for binary floating-point arithmetic. Reprinted in SIGPLAN Notices, 22(2) :9–25, 1987.

[163] IEEE. P1363 : Standard specifications for public-key cryptography. Available at `http ://grouper.ieee.org/groups/1363/`.

[164] IEEE. *P1363.1 Public-Key Cryptographic Techniques Based on Hard Problems over Lattices*, June 2003. IEEE., Available from `http ://grouper.ieee.org/groups/1363/lattPK/index.html`.

[165] IPA. Cryptrec : Evaluation of cryptographic techniques. Available at `http ://www.ipa.go.jp/security/enc/CRYPTREC/index-e.html`.

[166] K. Jacobs. *Measure and integral*. Academic Press [Harcourt Brace Jovanovich Publishers], New York, 1978. Probability and Mathematical Statistics, With an appendix by Jaroslav Kurzweil.

[167] K. Jallad, J. Katz, and B. Schneier. Implementation of chosen-ciphertext attacks against PGP and GnuPG. In *Proc. of ISC '02*, volume 2433 of *Lecture Notes in Computer Science*. Springer, 2002.

[168] E. Jaulmes and A. Joux. A chosen ciphertext attack on NTRU. In *Proc. of Crypto '00*, volume 1880 of *Lecture Notes in Computer Science*. IACR, Springer, 2000.

[169] N. Joshi, K. Wu, and R. Karri. Concurrent error detection schemes for involution ciphers. In *Proc. CHES*, volume 3156 of *Lecture Notes in Computer Science*, pages 400–412. Springer, 2004.

[170] A. Joux. *La Réduction des Réseaux en Cryptographie*. PhD thesis, École Polytechnique, 1993.

[171] A. Joux and J. Stern. Lattice reduction : A toolbox for the cryptanalyst. *J. of Cryptology*, 11 :161–185, 1998.

[172] M. Joye, A. K. Lenstra, and J.-J. Quisquater. Chinese remaindering based cryptosystems in the presence of faults. *J. Cryptology*, 12(4) :241–245, 1999.

[173] C. S. Jutla. On finding small solutions of modular multivariate polynomial equations. In *Proc. of Eurocrypt '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 158–170. IACR, Springer, 1998.

[174] M. Kaib and C. P. Schnorr. The generalized Gauss reduction algorithm. *J. Algorithms*, 21(3) :565–578, 1996.

[175] B. Kaliski. PKCS #1 : RSA encryption version 1.5 : Request for Comments 2313. Available as `http ://www.ietf.org/rfc/rfc2313.txt`.

[176] R. Kannan. Improved algorithms for integer programming and related lattice problems. In *Proc. of the 15th Symposium on the Theory of Computing*, pages 193–206. ACM, 1983.

[177] R. Kannan. Algorithmic geometry of numbers. *Annual review of computer science*, 2 :231–267, 1987.

[178] R. Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12(3) :415–440, 1987.

[179] J. Katz and B. Schneier. A chosen ciphertext attack against several E-Mail encryption protocols. In *Proc. of the 9th Usenix Security Symposium*, 2000.

[180] A. Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, IX, 1883. Numéros de janvier et février.

[181] A. Kipnis and A. Shamir. Cryptanalysis of the HFE public key cryptosystem by relinearization. In *Proc. of Crypto '99*, volume 1666 of *LNCS*, pages 19–30. Springer-Verlag, 1999.

[182] T. Kleinjung. Discrete logarithms in GF(p)— 160 digits. Public announcement on Feb 5th., 2007.

[183] N. Koblitz. Elliptic curve cryptosystems. *Math. Comp.*, 48 :203–209, 1987.

[184] N. Koblitz. *Algebraic aspects of cryptography*. Springer, 1998.

[185] N. Koblitz and A. Menezes. Another look at generic groups. *Adv. Math. Commun.*, 1(1) :13–28, 2007.

[186] W. Koch. GnuPG's ElGamal signing keys compromised. Internet public announcement on November 27th, 2003.

[187] P. C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Proc. of Crypto '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. IACR, Springer-Verlag, 1996.

[188] P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Proc. CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[189] S. V. Konyagin and I. E. Shparlinski. *Character sums with exponential functions and their applications*. Cambridge Univ. Press, Cambridge, 1999.

[190] A. Korkine and G. Zolotareff. Sur les formes quadratiques positives ternaires. *Math. Ann.*, 5 :581–583, 1872.

[191] A. Korkine and G. Zolotareff. Sur les formes quadratiques. *Math. Ann.*, 6 :336–389, 1873.

[192] N. M. Korobov. On the distribution of digits in periodic fractions. *Math. USSR – Sbornik*, 18 :659–676, 1972.

[193] H. Koy and C. P. Schnorr. Segment LLL-reduction of lattice bases. In *Proceedings of the 2001 Cryptography and Lattices Conference (CALC'01)*, volume 2146 of *Lecture Notes in Computer Science*, pages 67–80. Springer, 2001.

[194] H. Koy and C. P. Schnorr. Segment LLL-reduction of lattice bases with floating-point orthogonalization. In *Proceedings of the 2001 Cryptography and Lattices Conference (CALC'01)*, volume 2146 of *Lecture Notes in Computer Science*, pages 81–96. Springer, 2001.

[195] R. Kuipers and H. Niederreiter. *Uniform Distribution of Sequences*. Wiley-Interscience, NY, 1974.

[196] H. Kuwakado and H. Tanaka. On the security of the ElGamal-type signature scheme with small parameters. *IEICE Transactions on Fundamentals of Electronics, Commun., and Comp. Sci.*, E82-A :93–97, 1999.

[197] R. Laboratories. The public-key cryptography standards (PKCS). Available at `http ://www.rsasecurity.com/rsalabs`.

[198] J. C. Lagarias. Worst-case complexity bounds for algorithms in the theory of integral quadratic forms. *Journal of Algorithms*, 1 :142–186, 1980.

[199] J. C. Lagarias, H. W. Lenstra, Jr., and C. P. Schnorr. Korkin-Zolotarev bases and successive minima of a lattice and its reciprocal lattice. *Combinatorica*, 10 :333–348, 1990.

[200] J. C. Lagarias and A. M. Odlyzko. Solving low-density subset sum problems. *Journal of the Association for Computing Machinery*, January 1985.

[201] L. Lagrange. Recherches d'arithmétique. *Nouv. Mém. Acad.*, 1773. Pages 695–795 du Tome III des oeuvres de Lagrange.

[202] B. A. LaMacchia. Basis reduction algorithms and subset sum problems. Technical Report AITR-1283, MIT, 1991.

[203] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems*. SIAM Publications, 1995.

[204] A. K. Lenstra and H. W. Lenstra, Jr. *The Development of the Number Field Sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer, 1993.

[205] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Ann.*, 261 :513–534, 1982.

[206] A. K. Lenstra and E. R. Verheul. The XTR public key system. In *Proc. CRYPTO*, volume 1880 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2000.

[207] H. W. Lenstra, Jr. Integer programming with a fixed number of variables. Technical report 81-03, Mathematisch Instituut, Universiteit van Amsterdam, 1981.

[208] H. W. Lenstra, Jr. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8(4) :538–548, 1983.

[209] H. W. Lenstra, Jr. Factoring integers with elliptic curves. *Ann. of Math.*, 126 :649–673, 1987.

[210] H. W. Lenstra, Jr. On the Chor-Rivest knapsack cryptosystem. *J. Cryptology*, 3(3) :149–155, 1991.

[211] H. W. Lenstra, Jr. Flags and lattice basis reduction. In *Proceedings of the third European congress of mathematics, volume 1*. Birkhäuser, 2001.

[212] R. Lercier and F. Morain. Counting the number of points on elliptic curves over finite fields : strategies and performances. In *Proc. of Eurocrypt '95*, volume 921 of *LNCS*, pages 79–94. Springer-Verlag, 1995.

[213] Lidia. A library for computational number theory. Disponible à l'adresse `http ://www-jb.cs.uni-sb.de/LiDIA/linkhtml/lidia/lidia.html`.

[214] L. Lovász. *An Algorithmic Theory of Numbers, Graphs and Convexity*, volume 50. SIAM Publications, 1986. CBMS-NSF Regional Conference Series in Applied Mathematics.

[215] D. S. Mackey, N. Mackey, and F. Tisseur. Structured factorizations in scalar product spaces. *SIAM J. of Matrix Analysis and Appl.*, 2005. To appear.

[216] Magma. The Magma computational algebra system for algebra, number theory and geometry. Informations disponibles à l'adresse `http ://www.maths.usyd.edu.au :8000/u/magma/`.

[217] K. Mahler. A theorem on inhomogeneous diophantine inequalities. In *Nederl. Akad. Wetensch., Proc.*, volume 41, pages 634–637, 1938.

[218] J. Manger. A chosen ciphertext attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as standardized in PKCS #1 v2.0. In *Proc. of Crypto '01*, volume 2139 of *Lecture Notes in Computer Science*, pages 230–231. IACR, Springer, 2001.

[219] W. Maple. The Maple computational algebra system for algebra, number theory and geometry. Information available at `http ://www.maplesoft.com/products/Maple6/maple6info.html`.

[220] J. Martinet. *Les Réseaux Parfaits des Espaces Euclidiens*. Editions Masson, 1996. Traduction en cours chez Springer-Verlag.

[221] T. Matsumoto and H. Imai. Public quadratic polynominal-tuples for efficient signature-verification and message-encryption. In *Proc. of Eurocrypt '88*, volume 330 of *Lecture Notes in Computer Science*, pages 419–453. Springer, 1988.

[222] T. Matsumoto, K. Kato, and H. Imai. Speedings up secret computation with insecure auxiliary devices. In *Proc. of Crypto '88*, volume 403 of *Lecture Notes in Computer Science*, pages 497–506. Springer, 1989.

[223] A. May. Computing the RSA secret key is deterministic polynomial time equivalent to factoring. In *Proc. CRYPTO '04*, volume 3152 of *Lecture Notes in Computer Science*, pages 213–219. Springer, 2004.

[224] A. May. Secret exponent attacks on RSA-type schemes with moduli $n = p^r q$. In *Public Key Cryptography – Proc. of PKC 2004*, volume 2947 of *Lecture Notes in Computer Science*, pages 218–230. Springer, 2004.

[225] A. May and J. H. Silverman. Dimension reduction methods for convolution modular lattices. In *Proc. of CALC '01*, volume 2146 of *Lecture Notes in Computer Science*. Springer, 2001.

[226] T. May and M. Woods. A new physical mechanism for soft errors in dynamic memories. In *Proc. 16th International Reliability Physics Symposium*, pages 33–40, 1978.

[227] J. E. Mazo and A. M. Odlyzko. Lattice points in high-dimensional spheres. *Monatsh. Math.*, 110 :47–61, 1990.

[228] R. McEliece. A public-key cryptosystem based on algebraic number theory. Technical report, Jet Propulsion Laboratory, 1978. DSN Progress Report 42-44.

[229] W. Meier, June 2000. Private communication.

[230] A. Menezes, T. Okamoto, and S. A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Trans. Inform. Theory*, 39(5) :1639–1646, 1993.

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

[232] J. Merkle. Multi-round passive attacks on server-aided RSA protocols. In *ACM Conference on Computer and Communications Security*, pages 102–107, 2000.

[233] J. Merkle and R. Werchner. On the security of server-aided RSA protocols. In *Proc. PKC*, volume 1431 of *Lecture Notes in Computer Science*, pages 99–116. Springer, 1998.

[234] R. Merkle and M. Hellman. Hiding information and signatures in trapdoor knapsacks. *IEEE Trans. Inform. Theory*, IT-24 :525–530, Sept. 1978.

[235] T. Meskanen and A. Renvall. A wrap error attack against NTRUEncrypt. *Discrete Appl. Math.*, 154(2) :382–391, 2006.

[236] D. Micciancio. *On the hardness of the shortest vector problem*. PhD Thesis, MIT, 1998.

[237] D. Micciancio. The shortest vector problem is NP-hard to approximate within some constant. In *Proc. of the 39th Symposium on the Foundations of Computer Science*. IEEE, 1998. Disponible chez [89] comme TR98-016.

[238] D. Micciancio. The hardness of the closest vector problem with preprocessing. *IEEE Trans. Inform. Theory*, 47(3) :1212–1215, 2001.

[239] D. Micciancio. Improving lattice-based cryptosystems using the Hermite normal form. In *Proc. of CALC '01*, volume 2146 of *Lecture Notes in Computer Science*. Springer, 2001.

[240] D. Micciancio and S. Goldwasser. *Complexity of lattice problems*. The Kluwer International Series in Engineering and Computer Science, 671. Kluwer Academic Publishers, Boston, MA, 2002. A cryptographic perspective.

[241] V. Miller. Use of elliptic curves in cryptography. In *Proc. of Crypto '85*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426. IACR, Springer-Verlag, 1987.

[242] J. Milnor and D. Husemoller. *Symmetric Bilinear Forms*. Springer, 1973.

[243] H. Minkowski. *Geometrie der Zahlen*. Teubner-Verlag, Leipzig, 1896.

[244] F. Monrose, M. Reiter, and S. Wetzel. Password hardening based on keystroke dynamics. In *Proc. of 6th Conf. on Computer and Communications Security*. ACM, 1999.

[245] C. J. Moreno and O. Moreno. Exponential sums and Goppa codes, I. *Proc. Amer. Math. Soc.*, 111 :523–531, 1991.

[246] D. Naccache, P. Q. Nguyen, M. Tunstall, and C. Whelan. Experimenting with faults, lattices and the DSA. In *Proc. PKC 2005, 8th International Workshop on Theory and Practice in Public Key Cryptography*, volume 3386 of *Lecture Notes in Computer Science*, pages 16–28. Springer, 2005.

[247] M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation. In *Proc. of 31st STOC*, pages 245–254. ACM, 1999.

[248] National Institute of Standards and Technology (NIST). *FIPS Publication 186 : Digital Signature Standard*, Mai 1994.

[249] National Institute of Standards and Technology (NIST). *Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family*, Novembre 2007. Federal Register vol. 72, no. 212, Docket No. 070911510–7512–01.

[250] P. Nguyen. Cryptanalysis of the Goldreich-Goldwasser-Halevi cryptosystem from Crypto '97. In *Proc. of the 19th IACR Cryptology Conference (Crypto '99)*, volume 1666 of *Lecture Notes in Computer Science*, pages 288–304. Springer, 1999.

[251] P. Nguyen and J. Stern. Merkle-Hellman revisited : a cryptanalysis of the Qu-Vanstone cryptosystem based on group factorizations. In *Proc. of the 17th IACR Cryptology Conference (Crypto '97)*, volume 1294 of *Lecture Notes in Computer Science*, pages 198–212. Springer, 1997.

[252] P. Nguyen and J. Stern. Cryptanalysis of the Ajtai-Dwork cryptosystem. In *Proc. of the 18th IACR Cryptology Conference (Crypto '98)*, volume 1462 of *Lecture Notes in Computer Science*, pages 223–242. Springer, 1998.

[253] P. Nguyen and J. Stern. The Béguin-Quisquater server-aided RSA protocol from Crypto '95 is not secure. In *Proc. of Asiacrypt '98*, volume 1514 of *Lecture Notes in Computer Science*, pages 372–379. Springer, 1998.

[254] P. Q. Nguyen. The dark side of the hidden number problem : Lattice attacks on DSA. In K.-Y. Lam, I. E. Shparlinski, H. Wang, and C. Xing, editors, *Proc. Workshop on Cryptography and Computational Number Theory (CCNT'99), Singapore*, pages 321–330. Birkhäuser, 2001.

[255] P. Q. Nguyen. Can we trust cryptographic software ? Cryptographic flaws in GNU privacy guard v1.2.3. In *Advances in Cryptology – Proceedings of EUROCRYPT '04*, volume 3027 of *Lecture Notes in Computer Science*, pages 555–570. Springer, 2004.

[256] P. Q. Nguyen. A note on the security of NTRUSign. Technical report, Cryptology ePrint Archive, 2006. Report 2006/387.

[257] P. Q. Nguyen. *Public-Key Cryptanalysis*. AMS, 2008. To appear.

[258] P. Q. Nguyen and D. Pointcheval. Analysis and improvements of NTRU encryption paddings. In *Proc. of the 22nd Cryptology Conference (Crypto '02)*, volume 2442 of *Lecture Notes in Computer Science*, pages 210–225. IACR, Springer-Verlag, 2002.

[259] P. Q. Nguyen and O. Regev. Learning a Parallelepiped : Cryptanalysis of GGH and NTRU Signatures. In *Advances in Cryptology – Proceedings of EUROCRYPT '06*, volume 4004 of *Lecture Notes in Computer Science*, pages 215–233. Springer, 2006.

[260] P. Q. Nguyen and I. E. Shparlinski. On the insecurity of a server-aided RSA protocol. In *Proc. of Asiacrypt '01*, volume 2248 of *Lecture Notes in Computer Science*, pages 21–35. IACR, Springer, 2001.

[261] P. Q. Nguyen and I. E. Shparlinski. The insecurity of the Digital Signature Algorithm with partially known nonces. *J. Cryptology*, 15(3) :151–176, 2002.

[262] P. Q. Nguyen and I. E. Shparlinski. The insecurity of the elliptic curve Digital Signature Algorithm with partially known nonces. *Design, Codes and Cryptography*, 30(2) :201–217, 2003.

[263] P. Q. Nguyen and D. Stehlé. Low-dimensional lattice basis reduction revisited. In *Proceedings of the 6th International Algorithmic Number Theory Symposium, (ANTS-VI)*, volume 3076 of *Lecture Notes in Computer Science*, pages 338–357. Springer, 2004.

[264] P. Q. Nguyen and D. Stehlé. Floating-Point LLL Revisited. In *Advances in Cryptology – Proceedings of EUROCRYPT '05*, volume 3494 of *Lecture Notes in Computer Science*, pages 215–233. Springer, 2005.

[265] P. Q. Nguyen and D. Stehlé. LLL on the Average. In *Proceedings of the 7th International Algorithmic Number Theory Symposium, (ANTS-VII)*, volume 4076 of *Lecture Notes in Computer Science*, pages 238–256. Springer, 2006.

[266] P. Q. Nguyen and J. Stern. Lattice reduction in cryptology : An update. In *Algorithmic Number Theory – Proc. of ANTS-IV*, volume 1838 of *Lecture Notes in Computer Science*, pages 85–112. Springer-Verlag, 2000.

[267] P. Q. Nguyen and J. Stern. The two faces of lattices in cryptology. In *Proc. of CALC '01*, volume 2146 of *Lecture Notes in Computer Science*. Springer, 2001.

[268] P. Q. Nguyen and J. Stern. Adapting density attacks to low-weight knapsacks. In *Advances in Cryptology – Proceedings of ASIACRYPT '05*, volume 3788 of *Lecture Notes in Computer Science*, pages 41–58. Springer, 2005.

[269] P. Q. Nguyen and J. Stern. La cryptologie : enjeux et perspectives. In *Paradigmes et enjeux de l'informatique*, chapter 6. Lavoisier, 2005.

[270] P. Q. Nguyen and T. Vidick. Sieve algorithms for the shortest vector problem are practical. *J. of Mathematical Cryptology*, 2008. A paraître.

[271] H. Niederreiter. Quasi-Monte Carlo Methods and Pseudo-random Numbers. *Bull. Amer. Math. Soc.*, 84 :957–1041, 1978.

[272] H. Niederreiter. *Random Number Generation and Quasi–Monte Carlo Methods*, volume 63. SIAM, Philadelphia, 1992. CBMS-NSF Regional Conference Series in Applied Mathematics.

[273] K. Nyberg and R. A. Rueppel. Message recovery for signature schemes based on the discrete logarithm problem. *J. Cryptology*, 8 :27–37, 1995.

[274] A. M. Odlyzko. The rise and fall of knapsack cryptosystems. In *Cryptology and Computational Number Theory*, volume 42 of *Proc. of Symposia in Applied Mathematics*, pages 75–88. A.M.S., 1990.

[275] J. officiel. Loi numéro 2000-230 du 13 mars 2000 portant adaptation du droit de la preuve aux technologies de l'information et relative à la signature électronique. `http ://www.adminet.com/jo/20000314/`.

[276] T. Okamoto and D. Pointcheval. REACT : Rapid enhanced-security asymmetric cryptosystem transform. In *Proc. of CT-RSA '01*, volume 2020 of *Lecture Notes in Computer Science*. Springer, 2001.

[277] T. Okamoto, K. Tanaka, and S. Uchiyama. Quantum public-key cryptosystems. In *Advances in cryptology—CRYPTO 2000 (Santa Barbara, CA)*, volume 1880 of *Lecture Notes in Comput. Sci.*, pages 147–165. Springer, Berlin, 2000.

[278] K. Omura and K. Tanaka. Density attack to the knapsack cryptosystems with enumerative source encoding. *IEICE Trans. Fundamentals*, E84-A(1), 2001.

[279] P. Oorschot and M. J. Wiener. On Diffie-Hellman key agreement with short exponents. In *Proc. of Eurocrypt '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 332–343. IACR, Springer, 1996.

[280] OpenPGP. The OpenPGP standard. See `http ://www.openpgp.org`.

[281] R. Peralta and E. Okamoto. Faster factoring of integers of a special form. *IEICE Trans. Fund. of Electronics, Communications, and Computer Sciences*, 79(4), 1996.

[282] B. Pfitzmann and M. Waidner. Attacks on protocols for server-aided RSA computation. In *Proc. EUROCRYPT*, Lecture Notes in Computer Science, pages 153–162. Springer, 1992.

[283] PGP. Pretty good privacy. `http ://www.pgp.com`.

[284] B. Pinkas, 1999. Private communication.

[285] M. Pohst. On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications. *ACM SIGSAM Bulletin*, 15(1) :37–44, 1981.

[286] M. Pohst. A modification of the LLL reduction algorithm. *Journal of Symbolic Computation*, 4(1) :123–127, 1987.

[287] D. Pointcheval. *Provable Security for Public Key Schemes*, pages 133–185. Birkhäuser Verlag, 2005.

[288] D. Pointcheval and J. Stern. Security proofs for signature schemes. In *Proc. of Eurocrypt '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398. IACR, Springer, 1996.

[289] T. S. Project. MPFR, a LGPL-library for multiple-precision floating-point computations with exact rounding. Available at `http ://www.mpfr.org/`.

[290] J. Proof. Imperfect decryption and an attack on the NTRU encryption scheme. Technical report, Cryptology ePrint Archive, 2003. Report 2003/002.

[291] H.-G. Quebbemann. Modular lattices in Euclidean spaces. *J. Number Theory*, 54(2) :190–202, 1995.

[292] R. A. Rankin. On positive definite quadratic forms. *J. London Math. Soc.*, 28 :309–314, 1953.

[293] H. Ritter. Breaking knapsack cryptosystems by max-norm enumeration. In *Proc. of Pragocrypt '96*, pages 480–492. CTU Publishing House, 1996.

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

[295] S. S. Ryškov. On the reduction theory of positive quadratic forms. *Dokl. Akad. Nauk SSSR*, 198 :1028–1031, 1971.

[296] S. S. Ryskov. On Hermite, Minkowski and Venkov reduction of positive quadratic forms in n variables. *Soviet Mathematics Doklady*, 13 :1676–1679, 1972.

[297] S. S. Ryškov. The reduction of positive quadratic forms of n variables in the sense of Hermite, Minkowski and Venkov. *Dokl. Akad. Nauk SSSR*, 207 :1054–1056, 1972.

[298] R. Sakai, M. Morii, and M. Kasahara. New key generation algorithm for RSA cryptosystem. *IEICE Trans. Fundamentals*, E77-A(1) :89–97, 1994.

[299] W. Scharlau and H. Opolka. *From Fermat to Minkowski*. Undergraduate Texts in Mathematics. Springer-Verlag, New York, 1985. Lectures on the theory of numbers and its historical development, Translated from the German by Walter K. Bühler and Gary Cornell.

[300] W. M. Schmidt. The distribution of sublattices of $\mathbf{Z}^m$. *Monatsh. Math.*, 125, 1998.

[301] B. Schneier. Security in the real world : How to evaluate security technology. *Computer Security Journal*, XV(4), 1999.

[302] C. P. Schnorr. A hierarchy of polynomial lattice basis reduction algorithms. *Theoretical Computer Science*, 53 :201–224, 1987.

[303] C. P. Schnorr. A more efficient algorithm for lattice basis reduction. *J. of algorithms*, 9(1) :47–62, 1988.

[304] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4 :161–174, 1991.

[305] C. P. Schnorr. Block reduced lattice bases and successive minima. *Combinatorics, Probability and Computing*, 3 :507–522, 1994.

[306] C. P. Schnorr. Fast LLL-type lattice reduction. *Information and Computation*, 204 :1–25, 2006.

[307] C. P. Schnorr and M. Euchner. Lattice basis reduction : improved practical algorithms and solving subset sum problems. *Math. Programming*, 66 :181–199, 1994.

[308] C. P. Schnorr and H. H. Hörner. Attacking the Chor-Rivest cryptosystem by improved lattice reduction. In *Proc. of Eurocrypt '95*, volume 921 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 1995.

[309] A. Schönhage. Schnelle Berechnung von Kettenbruchentwicklungen. *Acta Informatica*, 1 :139–144, 1971.

[310] A. Schönhage. Factorization of univariate integer polynomials by Diophantine approximation and improved basis reduction algorithm. In *Proceedings of the 1984 International Colloquium on Automata, Languages and Programming (ICALP 1984)*, volume 172 of *Lecture Notes in Computer Science*, pages 436–447. Springer, 1984.

[311] A. Schönhage. Fast reduction and composition of binary quadratic forms. In *Proceedings of the 1991 International Symposium on Symbolic and Algebraic Computation (ISSAC'91)*, pages 128–133. ACM Press, 1991.

[312] A. Schönhage and V. Strassen. Schnelle Multiplikation grosser Zahlen. *Computing*, 7 :281–292, 1971.

[313] R. Schoof. Counting points on elliptic curves over finite fields. *J. Théor. Nombres Bordeaux*, 7 :219–254, 1995.

[314] I. A. Semaev. A 3-dimensional lattice reduction algorithm. In *Proc. of CALC '01*, volume 2146 of *Lecture Notes in Computer Science*. Springer, 2001.

[315] M. Seysen. Simultaneous reduction of a lattice basis and its reciprocal basis. *Combinatorica*, 13(3) :363–376, 1993.

[316] A. Shamir. A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem. In *Proc. of the 23rd Symposium on the Foundations of Computer Science*, pages 145–152. IEEE, 1982.

[317] A. Shamir. RSA for paranoids. *RSA Laboratories CryptoBytes*, 1(3) :1–4, 1995.

[318] P. W. Shor. Algorithms for quantum computation : discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science (Santa Fe, NM, 1994)*, pages 124–134. IEEE Comput. Soc. Press, Los Alamitos, CA, 1994.

[319] V. Shoup. Number Theory C++ Library (NTL). Disponible à l'adresse `http ://www.shoup.net/ntl/`.

[320] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in cryptology—EUROCRYPT '97 (Konstanz)*, volume 1233 of *Lecture Notes in Comput. Sci.*, pages 256–266. Springer, 1997.

[321] V. Shoup. OAEP reconsidered. In *Proc. of Crypto '01*, Lecture Notes in Computer Science. IACR, Springer, 2001.

[322] V. Shoup. Sequences of games : a tool for taming complexity in security proofs. Cryptology ePrint Archive : Report 2004/332, 2004.

[323] V. Shoup. *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, 2005. Also available on the Internet.

[324] I. E. Shparlinski. On the uniformity of distribution of the ElGamal signature. *Appl. Algebra Engrg. Comm. Comput.*, 13(1) :9–16, 2002.

[325] C. L. Siegel. A mean value theorem in geometry of numbers. *Ann. of Math. (2)*, 46 :340–347, 1945.

[326] C. L. Siegel. *Lectures on the Geometry of Numbers*. Springer, 1989.

[327] J. H. Silverman. Estimated breaking times for NTRU lattices. Technical report, NTRU Cryptosystems, March 1999. Report #012, available at [78].

[328] J. H. Silverman and W. Whyte. Estimating decryption failure probabilities for NTRUEncrypt. Technical report, NTRU Cryptosystems, May 2003. Report #018, available at [78].

[329] D. Simon. Solving quadratic equations using reduced unimodular quadratic forms. *Mathematics of Computation*, 74(251) :1531–1543, 2005.

[330] J. Stern. *La Science du Secret*. Éditions Odile Jacob, 1998.

[331] J. Stern, D. Pointcheval, J. Malone-Lee, and N. P. Smart. Flaws in applying proof methodologies to signature schemes. In *Proc. of Crypto '02*, volume 2442 of *Lecture Notes in Computer Science*, pages 93–110. IACR, Springer, 2002.

[332] M. Stevens, A. K. Lenstra, and B. de Weger. Chosen-prefix collisions for MD5 and colliding X.509 certificates for different identities. In *Proc. EUROCRYPT*, volume 4515 of *Lecture Notes in Computer Science*, pages 1–22. Springer, 2007.

[333] D. Stinson. *Cryptography : Theory and Practice*. CRC Press, 1995.

[334] D. R. Stinson. Some baby-step giant-step algorithms for the low Hamming weight discrete logarithm problem. *Math. Comp.*, 71(237) :379–391 (electronic), 2002.

[335] M. I. Stogrin. Regular Dirichlet-Voronoï partitions for the second triclinic group. *American Mathematical Society*, 1977. English translation of the proceedings of the Steklov Institute of Mathematics, Number 123 (1973).

[336] A. Storjohann. Faster algorithms for integer lattice basis reduction. Technical report, ETH Zürich, 1996.

[337] H.-M. Sun, W.-C. Yang, and C.-S. Laih. On the design of RSA with short secret exponent. In *Proc. ASIACRYPT*, volume 1716 of *Lecture Notes in Computer Science*, pages 150–164. Springer, 1999.

[338] M. Szydlo. Hypercubic lattice reduction and analysis of GGH and NTRU signatures. In *Proc. of Eurocrypt '03*, volume 2656 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

[339] P. P. Tammela. On the reduction theory of positive quadratic forms. *Soviet Mathematics Doklady*, 14 :651–655, 1973.

[340] A. Terras. *Harmonic analysis and symmetric spaces and applications. II.* Springer-Verlag, Berlin, 1988.

[341] E. Teske. Square-root algorithms for the discrete logarithm problem (A survey). In *Proc. of the Conference "Public Key Cryptography and Computational Number Theory", Warszawa, September 11-15, 2000*, pages 283–301, 2001.

[342] J. L. Thunder. Higher-dimensional analogs of Hermite's constant. *Michigan Math. J.*, 45(2) :301–314, 1998.

[343] B. Vallée. *Une Approche Géométrique de la Réduction de Réseaux en Petite Dimension*. PhD thesis, Université de Caen, 1986.

[344] B. Vallée. Gauss' algorithm revisited. *J. Algorithms*, 12(4) :556–572, 1991.

[345] S. Vaudenay. Hidden collisions on DSS. In *Proc. of Crypto '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 83–88. IACR, Springer, 1996.

[346] S. Vaudenay. Cryptanalysis of the Chor-Rivest cryptosystem. *J. Cryptology*, 14(2) :87–100, 2001.

[347] E. Verheul and H. van Tilborg. Cryptanalysis of less short RSA secret exponents. *Applicable Algebra in Engineering, Communication and Computing*, 8 :425–435, 1997.

[348] I. M. Vinogradov. *Elements of Number Theory*. Dover Publ., New York, 1954.

[349] G. Voronoï. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. *Journal für die reine und angewandte Mathematik*, 134 :198–287, 1908.

[350] B. Waerden. Die Reduktionstheorie der positiven quadratischen Formen. *Acta Math.*, 96 :265–309, 1956.

[351] X. Wang, Y. L. Yin, and H. Yu. Finding collisions in the full SHA-1. In *Advances in cryptology—CRYPTO 2005*, volume 3621 of *Lecture Notes in Comput. Sci.*, pages 17–36. Springer, Berlin, 2005.

[352] X. Wang and H. Yu. How to break MD5 and other hash functions. In *Proc. EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2005.

[353] B. M. M. d. Weger. Solving exponential Diophantine equations using lattice basis reduction algorithms. *J. Number Theory*, 26(3) :325–367, 1987.

[354] H. Weyl. *The classical groups*. Princeton Landmarks in Mathematics. Princeton University Press, 1997. Their invariants and representations, Fifteenth printing, Princeton Paperbacks.

[355] W. Whyte. Improved NTRUSign transcript analysis. Presentation at the rump session of Eurocrypt '06, on May 30, 2006.

[356] M. Wiener. Cryptanalysis of short RSA secret exponents. *IEEE Trans. Inform. Theory*, 36(3) :553–558, 1990.

[357] J. H. Wilkinson. *The Algebraic Eigenvalue Problem*. Oxford University Press, 1988.

[358] C. K. Yap. Fast unimodular reduction : planar integer lattices. In *Proceedings of the 1992 Symposium on the Foundations of Computer Science (FOCS 1992)*, pages 437–446. IEEE Computer Society Press, 1992.