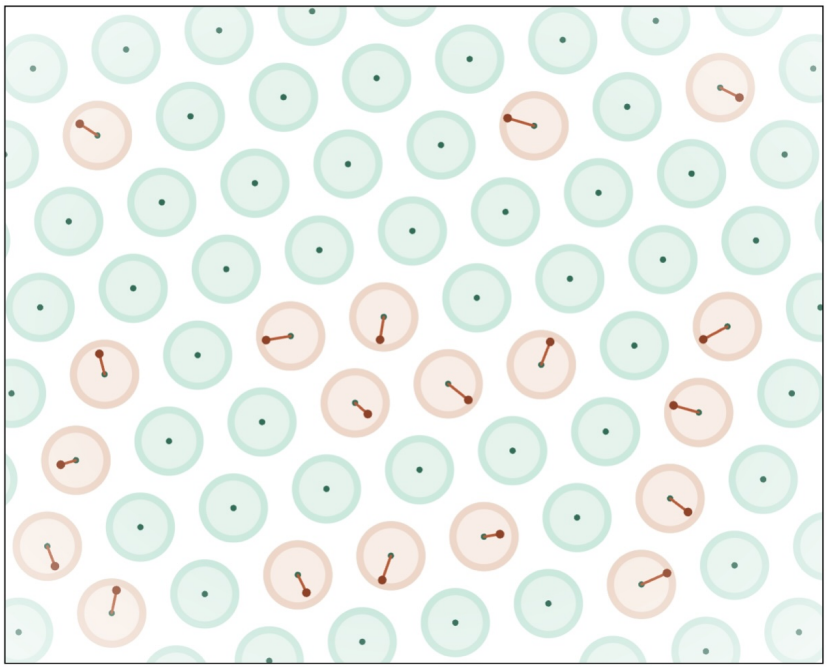
A Gentle Introduction to Lattice-Based Cryptography

Alfred Menezes
University of Waterloo
Email: ajmeneze@uwaterloo.ca
URL: https://cryptography101.ca

Version 3.4: October 28 2026

Abstract

We present four quantum-safe schemes: the Kyber (ML-KEM) and FrodoKEM key encapsulation mechanisms, and the Dilithium (ML-DSA) and Falcon (FN-DSA) signature schemes. We also develop the mathematical background on lattices needed to understand why Kyber, FrodoKEM, Dilithium and Falcon are regarded as lattice-based cryptosystems, and we provide insight into the computational hardness of the underlying lattice problems. The exposition is intended to be accessible to senior undergraduate students and beginning graduate students, and closely follows the accompanying video lecture series *PQC Explained*, *Kyber and Dilithium*, *The Mathematics of Lattice-Based Cryptography*, and *The LLL Algorithm*, which are available at the website cryptography101.ca.



Contents

List of Symbols . . . . . **vii**

List of Acronyms . . . . . **viii**

List of Algorithms . . . . . **ix**

List of Figures . . . . . **x**

List of Tables . . . . . **xii**

Revision history . . . . . **xiii**

1 Introduction **1**
1.1 Notes and further references 3

2 Lattices **5**
2.1 Basic definitions 5
2.2 Examples 6
2.3 Bases of a lattice 8
2.4 Successive minima 9
2.5 Fundamental lattice problems 13
2.5.1 Shortest Vector Problem (SVP) 13
2.5.2 Shortest Independent Vectors Problem (SIVP) 14
2.5.3 Closest Vector Problem (CVP) 14
2.5.4 Relationships 16
2.6 Notes and further references 16
2.7 Exercises 18

3 Short Integer Solution (SIS) problem **21**
3.1 Problem statement 21
3.2 The SIS lattice 23
3.3 Hardness of SIS 24
3.3.1 Dual attack on SIS_2 25
3.3.2 Average-case hardness of SIS 25
3.4 Notes and further references 26
3.5 Exercises 27

4 Learning With Errors (LWE) problem **28**
4.1 Problem statement 28
4.2 Lindner-Peikert public-key encryption 32
4.3 The LWE lattice 36
4.4 Hardness of LWE 37
4.4.1 Primal attack 38
4.4.2 Dual attack 39
4.4.3 Arora-Ge attack 40

4.4.4 Average-case hardness of LWE . . . 41
4.5 Notes and further references . . . 41
4.6 Exercises . . . 42

5 Module-SIS and Module-LWE . . . 45
5.1 The polynomial ring $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ . . . 45
5.2 Module-SIS (MSIS) . . . 48
5.3 Module-LWE (MLWE) . . . 50
5.4 Notes and further references . . . 52
5.5 Exercises . . . 52

6 Kyber (ML-KEM) . . . 54
6.1 Kyber public-key encryption . . . 54
6.2 Decryption error probability . . . 57
6.2.1 Centered binomial distributions . . . 57
6.2.2 Computing the decryption error probability . . . 59
6.2.3 ML-KEM-512 decryption error probability . . . 60
6.3 Ciphertext compression . . . 62
6.4 Kyber-KEM . . . 65
6.5 Notes and further references . . . 68
6.6 Exercises . . . 69

7 Dilithium (ML-DSA) . . . 71
7.1 Dilithium (toy version) . . . 71
7.1.1 Schnorr signature scheme . . . 71
7.1.2 Dilithium: first attempt . . . 71
7.1.3 Dilithium: second attempt . . . 72
7.1.4 Five limitations and their solutions . . . 73
7.2 Dilithium (without t compression) . . . 76
7.2.1 Supporting routines . . . 76
7.2.2 Signature scheme . . . 77
7.2.3 Security . . . 81
7.3 Dilithium (with t compression) . . . 81
7.3.1 Supporting routines . . . 82
7.3.2 Full signature scheme . . . 85
7.3.3 Number of iterations in signature generation . . . 87
7.3.4 Security . . . 88
7.4 Notes and further references . . . 88
7.5 Exercises . . . 90

8 Falcon (FN-DSA) . . . 91
8.1 Hash-and-sign . . . 91
8.1.1 RSA signature scheme . . . 91
8.1.2 SIS-based trapdoor one-way function . . . 92

8.1.3 SIS-based signature scheme . . . 93
8.1.4 A critical security weakness . . . 93
8.2 Discrete Gaussian distributions . . . 94
8.2.1 Gaussians . . . 94
8.2.2 Discrete Gaussians . . . 95
8.2.3 The smoothing parameter of a lattice . . . 98
8.3 GPV signature framework . . . 101
8.4 NTRU lattices . . . 104
8.4.1 Lattice definition . . . 105
8.4.2 A short basis . . . 106
8.5 NTRU key pair generation . . . 107
8.6 Falcon outline . . . 113
8.7 The Fast Fourier Transform . . . 117
8.8 Notes and further references . . . 118
8.9 Exercises . . . 120

9 FrodoKEM . . . 125
9.1 The error distribution . . . 125
9.2 Frodo public-key encryption . . . 128
9.2.1 FrodoPKE . . . 129
9.2.2 Decryption error probability . . . 133
9.2.3 Security . . . 135
9.3 FrodoKEM . . . 137
9.4 Notes and further references . . . 140
9.5 Exercises . . . 141

10 The LLL lattice basis reduction algorithm . . . 143
10.1 Gram-Schmidt orthogonalization . . . 143
10.2 Gauss's algorithm . . . 148
10.3 The LLL algorithm . . . 153
10.3.1 LLL-reduced bases and the Lovász condition . . . 153
10.3.2 Size reduction . . . 154
10.3.3 Algorithm description . . . 155
10.3.4 Termination . . . 157
10.3.5 Bit complexity . . . 160
10.3.6 A dimension-15 example . . . 165
10.4 LLL improvements . . . 167
10.4.1 Scheduling the Gram-Schmidt computations . . . 167
10.4.2 The parameter δ . . . 170
10.4.3 Using floating point arithmetic . . . 171
10.4.4 Deep insertion . . . 171
10.5 Notes and further references . . . 172
10.6 Exercises . . . 173

11 The BKZ lattice basis reduction algorithm . . . 177
11.1 BKZ-reduced bases . . . 177
11.2 Algorithm description . . . 180
11.3 Complexity analysis . . . 181
11.4 Enumeration . . . 183
11.4.1 Naive enumeration . . . 183
11.4.2 Basic enumeration . . . 184
11.4.3 Analysis . . . 185
11.5 Sieving . . . 185
11.5.1 Preliminaries . . . 186
11.5.2 The Nguyen-Vidick sieving algorithm . . . 189
11.5.3 Analysis . . . 189
11.6 Estimating the security level of Kyber . . . 192
11.6.1 Primal attack . . . 192
11.6.2 Geometric Series Assumption (GSA) . . . 194
11.6.3 Primal attack analysis . . . 195
11.7 Estimating the security level of Falcon . . . 196
11.7.1 Key recovery . . . 197
11.7.2 Signature forgery . . . 197
11.8 Notes and further references . . . 199
11.9 Exercises . . . 201

12 LLL applications . . . 203
12.1 Primal attack on short-secret LWE . . . 203
12.2 Dual attack on SIS_2 . . . 205
12.3 Finding integer relations . . . 206
12.4 Simultaneous Diophantine approximation . . . 209
12.5 Side-channel attack on ECDSA . . . 211
12.6 Finding small roots of a polynomial modulo N . . . 215
12.6.1 The Coppersmith/Howgrave-Graham method . . . 217
12.6.2 RSA encryption with fixed padding . . . 221
12.6.3 RSA encryption with random padding . . . 221
12.7 Notes and further references . . . 222
12.8 Exercises . . . 223

13 The Number-Theoretic Transform . . . 229
13.1 NTT definition . . . 229
13.2 NTT computation . . . 231
13.3 Dilithium NTT . . . 236
13.4 Kyber NTT . . . 240
13.5 Notes and further references . . . 244
13.6 Exercises . . . 244

A Security definitions . . . 247

A.1 Public-key encryption 247
A.2 Key encapsulation mechanisms 248
A.3 Signature schemes 249

B Answers to computational exercises 250

References 254

List of Symbols

$\mathbb{N}$	natural numbers
$\mathbb{Z}$	integers
$\mathbb{Q}$	rational numbers
$\mathbb{R}$	real numbers
$\mathbb{C}$	complex numbers
$\mathbb{Z}_n$	integers modulo n
$A \leq B$	problem A polynomial-time reduces to problem B
$\in_R S$	sample uniformly at random from S
$\lfloor x \rfloor$	largest integer $\leq x$
$\lfloor \cdot \rceil$	round to nearest integer, with ties broken by rounding towards 0
$\lceil \cdot \rfloor$	round to nearest integer, with ties broken by rounding away from 0
$\|x\|_\infty$	infinity norm of $x \in \mathbb{R}^n$ (page 5)
$\|x\|_2$	Euclidean norm of $x \in \mathbb{R}^n$ (page 5)
$L(B)$	lattice with basis B (Definition 2.1)
$P(B)$	fundamental parallelepiped of $L(B)$ (Definition 2.6)
$\mathrm{vol}(L)$	volume of lattice L (Definition 2.11)
$\lambda_i(L)$	i-th successive minimum of L (Definition 2.12)
γ_n	Hermite's constant (Definition 2.14)
L_A	LWE lattice (Definition 4.15)
$L_A^\perp$	SIS lattice (Definition 3.8)
mods	symmetric mod (Definitions 4.11 and 7.2)
R_q	polynomial ring $\mathbb{Z}_q[x]/(x^n + 1)$
$\|r\|_\infty$	infinity norm (size) of $r \in \mathbb{Z}_q$ (Definition 5.4)
$\|f\|_\infty$	infinity norm (size) of $f \in R_q$ (Definition 5.5)
S_η	small polynomials in R_q (Definition 5.6)
$\Phi_X(z)$	generating series of discrete random variable X (page 57, equation (22))
$\langle u, v \rangle$	inner product of $u, v \in \mathbb{R}^n$
V	vector subspace of $\mathbb{R}^n$
$V^\perp$	orthogonal complement of V
$\Pi_V(w)$	projection of $w \in \mathbb{R}^n$ onto V (Definition 10.1)
b_i^*	orthogonal basis vector (Theorem 10.3)
μ_{ij}	Gram-Schmidt coefficients (page 145, equation (62))
$\rho_{s,c}$	Gaussian function of width s and center c (Definition 8.1)
$D_{s,c}$	Gaussian distribution of width s and center c (Definition 8.2)
$D_{L,s,c}$	discrete Gaussian distribution of width s, center c, over lattice L (Definition 8.3)
$\eta_\varepsilon(L)$	smoothing parameter of lattice L (Definition 8.6)
Ψ_s	one-dimensional rounded Gaussian distribution of width s (Definition 9.1)
$\chi_{s,c}$	approximate rounded Gaussian distribution table of width s and support $\{-c, \ldots, c\}$ (Definition 9.3)

List of Acronyms

NIST	National Institute of Standards and Technology
FIPS	Federal Information Processing Standards
ISO	International Organization for Standardization
PKE	Public-Key Encryption
KEM	Key Encapsulation Mechanism
SVP	Shortest Vector Problem (page 13)
SIVP	Shortest Independent Vectors Problem (page 14)
CVP	Closest Vector Problem (page 14)
SIS	Short Integer Solution problem (page 21)
ISIS	Inhomogeneous Short Integer Solution problem (page 22)
nf-ISIS	normal-form ISIS problem (page 23)
SIS_2	Euclidean SIS problem (page 25)
LWE	Learning With Errors problem (page 28)
DLWE	Decisional Learning With Errors problem (page 30)
ss-LWE	short-secret LWE problem (page 31)
ss-DLWE	short-secret Decisional LWE problem (page 32)
BDD	Bounded Distance Decoding problem (page 38)
MSIS	Module Short Integer Solution problem (page 49)
MISIS	Module Inhomogeneous SIS problem (page 50)
nf-MISIS	normal-form MISIS problem (page 50)
MLWE	Module Learning With Errors problem (page 50)
D-MLWE	Decisional MLWE problem (page 51)
ss-MLWE	short-secret MLWE problem (page 51)
ss-D-MLWE	short-secret Decisional MLWE problem (page 52)
CBD	Centered Binomial Distribution (page 58)
GPV	Gentry-Peikert-Vaikuntanathan (page 91)
FFT	Fast Fourier Transform (page 113)
MatrixLWE	Matrix Learning With Errors problem (page 135)
ss-D-MatrixLWE	short-secret Decisional Matrix LWE problem (page 136)
LLL	Lenstra-Lenstra-Lovász (page 143)
BKZ	Block Korkine-Zolotarev (page 177)
ECDSA	Elliptic Curve Digital Signature Algorithm (page 211)
NTT	Number-Theoretic Transform (page 229)

List of Algorithms

4.13 Lindner-Peikert public-key encryption scheme 35

6.1 Kyber public-key enc. scheme (without ciphertext compression) 55
6.10 Kyber public-key enc. scheme (with ciphertext compression) 63
6.12 Kyber key encapsulation mechanism. 67

7.1 Dilithium signature scheme (toy version) 74
7.3 Dilithium signature scheme (without t compression) 78
7.9 Dilithium signature scheme (with t compression) 86

8.13 GPV signature scheme 103
8.23 NTRU key pair generation 111
8.26 Falcon signature scheme 115

9.4 Sampling from $\chi_{s,c}$ 128
9.7 Frodo public-key encryption scheme (FrodoPKE) 130
9.17 Frodo key encapsulation mechanism (FrodoKEM) 138

10.5 Gram-Schmidt orthogonalization process 146
10.10 Gauss's algorithm 151
10.18 Size reduction 155
10.20 LLL lattice basis reduction algorithm 156

11.6 BKZ lattice basis reduction algorithm 181
11.7 BKZ block operation 182
11.15 Nguyen-Vidick sieving method for solving SVP (simplified) 190

List of Figures

2.1 Three full-rank integer lattices in $\mathbb{R}^2$ . . . 7
2.2 Successive minima. . . . 10
2.3 A basis matrix for the Leech lattice $\Lambda_{24} \subseteq \mathbb{R}^{24}$ . . . 11
2.4 Finding a close lattice vector using rounding . . . 15
2.5 Known relationships between SVP, SIVP, CVP . . . 16

3.1 SIS problem. . . . 21
3.2 Proving the average-case hardness of SIS. . . . 26

4.1 LWE problem. . . . 28
4.2 Uniqueness of the LWE solution. . . . 29
4.3 The rounding function. . . . 33
4.4 Lindner-Peikert encryption. . . . 34
4.5 The BDD_α problem. . . . 38

6.1 Centered binomial distribution CBD_η with $\eta = 3$. . . . 59
6.2 Compression with $q = 19$ and $d = 2$. . . . 62

7.1 Rejection sampling in Dilithium. . . . 75
7.2 Example of $\mathrm{HighBits}_0$ and $\mathrm{LowBits}_0$ . . . 77
7.3 Example of Power2Round . . . 82
7.4 Original and adjusted definitions of HighBits and LowBits . . . 84

8.1 Signature points s in the parallelepiped $P_{1/2}(T)$ . . . 94
8.2 Examples of one-dimensional Gaussian distributions . . . 95
8.3 Examples of two-dimensional Gaussian distributions . . . 96
8.4 The one-dimensional discrete Gaussian distribution $D_{\mathbb{Z},5}$. . . . 97
8.5 The two-dimensional discrete Gaussian distribution $D_{\mathbb{Z}^2,3}$ . . . 98
8.6 The function $f(x) = \sum_{c \in L} D_{s,c}(x)$ for increasing widths s. . . . 99
8.7 Signature points in the GPV signature scheme . . . 103

9.1 The one-dimensional rounded Gaussian distribution Ψ_5. . . . 126
9.2 FrodoPKE decoding function $\mathrm{Decode} : \mathbb{Z}_{16} \to \mathbb{Z}_4$. . . . 129
9.3 FrodoPKE encryption. . . . 130

10.1 The projections of w onto V and $V^\perp$. . . . 144
10.2 Gram-Schmidt example in $\mathbb{R}^2$ . . . 149
10.3 Gram-Schmidt example in $\mathbb{R}^3$ . . . 149
10.4 An iteration of Gauss's algorithm. . . . 150
10.5 SageMath scripts . . . 157
10.6 A randomly-selected short basis of a 15-dimensional lattice . . . 167
10.7 The randomized, initial long basis of the 15-dimensional lattice . . . 168
10.8 The LLL-reduced basis . . . 169

11.1 The sieving method. 186

12.1 The LLL-reduced basis in Example 12.6 216
12.2 The 8×8 basis matrix B in Example 12.7. 219
12.3 The LLL-reduced basis $\tilde{B}$ in Example 12.7. 220

13.1 The NTT computation tree ($n = 8$). 232
13.2 The NTT tree ($n = 8$). 235
13.3 The "butterfly" formulas for computing NTT and NTT^{-1}. 236
13.4 NTT and NTT^{-1} pseudocode from FIPS 204 237
13.5 The NTT_2 tree ($n = 8$). 242

List of Tables

2.1 The known Hermite constants. 12

6.1 FIPS 203 parameter sets for Kyber-KEM (ML-KEM). 55
6.2 Probability distributions of A, B, C, D for ML-KEM-512 parameters 61
6.3 Probability distributions of C_u, C_v, F for ML-KEM-512 parameters 65

7.1 FIPS 204 parameter sets for Dilithium (ML-DSA). 73

8.1 One-dimensional discrete Gaussian distribution $D_{\mathbb{Z},5}$ 97
8.2 Parameter sets for Falcon (FN-DSA). 114
8.3 Size and performance comparison of Dilithium and Falcon (with floating-point arithmetic). 117

9.1 One-dimensional rounded Gaussian distribution $\Psi_5(x)$ 127
9.2 The error distributions χ used by FrodoKEM 127
9.3 Parameter sets for FrodoKem 130
9.4 Size and performance comparison of Kyber-KEM and FrodoKEM 140

10.1 Intermediate bases in an execution of the LLL algorithm 158
10.2 Swaps executed by the LLL algorithm 166
10.3 Lengths of basis vectors in the LLL algorithm 169
10.4 Matrix of Gram-Schmidt coefficients 170

11.1 Comparison of LLL, BKZ, enumeration, and sieving 183
11.2 Root Hermite factors for β-BKZ-reduced bases. 195
11.3 Core-SVP hardness for the standardized Kyber parameter sets. 196
11.4 Core-SVP hardness for the Falcon parameter sets. 198

Revision history

- Version 1.0: May 26, 2026.
- Version 1.1: June 9, 2026.
  - Added Example 2.13 (Leech lattice), Example 3.5 (ISIS instance), Example 4.8 (ss-LWE instance), Example 4.14 (Lindner-Peikert encryption).
  - Added §11.5 on sieving.
  - Added answers to computational exercises (Appendix B).
- Version 1.2: June 24, 2026.
  - Added the rounding method for solving approx-CVP (§2.5.3).
  - Described a relationship between SIS and LWE lattices (§4.3).
  - Notes and references were moved to individual sections.
  - Added some notes on BLISS (§7.4).
- Version 2.0: July 19, 2026.
  - Added §8 on Falcon.
  - Added §11.7 on estimating Falcon's security.
  - Added a description of general NTT in §13.2.
- Version 2.1: August 2, 2026 (made some cosmetic changes).
- Version 3.0: August 16, 2026.
  - Added §9 on FrodoKEM.
  - Added Appendix A on security definitions.
- Version 3.1: August 22, 2026 (made some minor changes).
- Version 3.2: September 1, 2026 (made some minor changes).
- Version 3.3: September 16, 2026 (made various minor changes).
- Version 3.4: October 28, 2026 (made some corrections and various minor changes).

1 Introduction

In August 2024, the U.S. government's National Institute of Standards and Technology (NIST) published a suite of standards for a key encapsulation mechanism and for digital signature schemes designed to be resistant to attacks by quantum computers. These quantum-safe schemes are intended to replace widely deployed RSA and elliptic curve cryptography (ECC), both of which are vulnerable to quantum computing attacks. Although the timeline for the development of cryptographically relevant quantum computers remains uncertain, efforts to transition from RSA and ECC to quantum-safe alternatives are rapidly gaining momentum.

Among the quantum-safe cryptographic schemes standardized by NIST, the key encapsulation mechanism Kyber and the digital signature scheme Dilithium are expected to see the broadest deployment in the near future. The security of Kyber relies on the presumed hardness of the Module Learning With Errors (MLWE) problem, whereas the security of Dilithium is based on both the MLWE problem and the Module Short Integer Solution (MSIS) problem. These problems are the module variants of the well-studied Learning With Errors (LWE) and Short Integer Solution (SIS) problems, respectively.

These notes give detailed descriptions and analyses of Kyber and Dilithium. We show how the SIS and LWE problems—together with their module variants, MSIS and MLWE—can be cast as computational problems on lattices, justifying the classification of Kyber and Dilithium as *lattice-based cryptosystems*. We also describe FrodoKEM, a key encapsulation mechanism built on the unstructured LWE problem and standardized by the International Organization for Standardization (ISO), and Falcon, an SIS-based signature scheme over NTRU lattices currently undergoing standardization by NIST.

Organization. The remainder of this document is organized as follows. In §2, we give an introduction to the theory of lattices and define fundamental computational lattice problems: the Shortest Vector Problem (SVP), the Shortest Independent Vectors Problem (SIVP), the Closest Vector Problem (CVP), and their approximate versions, approx-SVP, approx-SIVP, and approx-CVP. In §3 and §4, we show how SIS and LWE can be formulated as lattice problems, and explain their connections to SVP and SIVP. The module variants of SIS and LWE are introduced in §5. Detailed descriptions of Kyber and Dilithium are provided in §6 and §7, Falcon is introduced in §8, and FrodoKEM is described in §9. The lattice problems SIS, LWE, MSIS, and MLWE can be tackled using the LLL and BKZ lattice basis reduction algorithms described in §10 and §11. The performance of LLL and BKZ plays a central role in assessing the hardness of the lattice problems and in motivating the parameter choices used in Kyber, Dilithium, Falcon and FrodoKEM. Six applications of the LLL algorithm are presented in §12: solving short-secret LWE, solving SIS, finding integer relations, performing simultaneous Diophantine approximation, mounting a side-channel attack on the ECDSA signature scheme, and finding small roots of univariate polynomials modulo a composite integer with unknown factorization. In §13, we describe the Number-Theoretic Transform (NTT), a fast algorithm for polynomial multiplication that is essential for efficient implementation of

Kyber and Dilithium, and is also employed in Falcon.

Guide to using these notes. While the notes are intended to be introductory, several advanced topics may be skipped on a first reading:

- §7.3 Dilithium (with t compression): Understanding Dilithium signature compression requires digesting considerable technical details, yet yields only a modest decrease in signature size.
- §8 Falcon (FN-DSA): Although Falcon produces shorter signatures than Dilithium, understanding the scheme and implementing it efficiently and securely pose significant challenges. The initial Falcon proposal used FFT-based floating-point Gaussian sampling, which is difficult to implement safely in constant time. In September 2026, NIST announced plans to use only fixed-point arithmetic in the forthcoming draft standard and to disallow floating point. The present edition of these notes covers neither the FFT-based Gaussian sampling algorithm nor the fixed-point approach.
- §10.3.5 Bit complexity of the LLL algorithm.
- §10.4 LLL improvements.
- §11.5 Sieving: The enumeration algorithm (§11.4) for solving SVP is conceptually easier to understand and analyze than sieving.
- §12.3 Finding integer relations.
- §12.4 Simultaneous Diophantine approximation.
- §12.6 Finding small roots of a polynomial modulo N.
- §13 The Number-Theoretic Transform: This material is required for a high-performance implementation of Kyber and Dilithium, not for understanding the essential ideas behind the schemes.

Readers are encouraged to attempt some of the exercises at the end of each section (without relying on LLMs!). Computational exercises, marked with †, can be solved using a computer algebra system such as SageMath, Maple, or Mathematica; answers to these exercises are provided in Appendix B.

Advanced study. The research literature on lattices is vast and broad. While these notes offer a solid introduction to lattice-based cryptography, they lack coverage in certain areas, such as worst-case to average-case reductions, security proofs, and protocols with advanced functionalities. There are many possible research directions depending on whether your interests lie in the underlying mathematics, protocol design and analysis, implementation, theory, or applications. Once you have identified a topic that you find especially interesting, you could select a few foundational papers on that topic and read them carefully. For instance, if you are interested in lattice basis reduction, you could read the original LLL paper; if you are interested in KEMs, you could read the

paper that first described Kyber. To find topics and papers of interest, you can consult the "Notes and further references" section in each chapter, browse abstracts on the IACR ePrint Archive (https://eprint.iacr.org), or ask your advisor for suggestions. You may find many of the latest research papers difficult to read initially, but as your knowledge grows, grasping their main ideas will become much easier.

Acknowledgements. These notes originated from a mini-course on lattice-based cryptography that I delivered at the Vietnam Institute for Advanced Study in Mathematics (VIASM) in March 2026. They were subsequently expanded to incorporate feedback, questions, and discussions with attendees, including researchers from the Vietnam Institute of Cryptography Science and Technology. I am grateful to Sanjit Chatterjee, Paul Duncan, Steven Galbraith, Anna Henderson, Elizabeth van Oorschot, Paul van Oorschot, Bart Preneel, and Thang Cao Trinh for their helpful comments and feedback.

Feedback. If you spot any errors or have suggestions for new material to include in future revisions, please email me at ajmeneze@uwaterloo.ca.

1.1 Notes and further references

The video series *Post-Quantum Cryptography Explained* (https://cryptography101.ca/pqc) provides an elementary introduction to post-quantum cryptography and lattice-based cryptography.

Lattices are fundamental structures that play a central role in numerous areas of mathematics and computer science, including the geometry of numbers, coding theory, optimization, and cryptography. The *geometry of numbers* is a branch of number theory that applies geometric techniques, particularly those involving lattices, to problems concerning integers and algebraic numbers. The subject was founded by Hermann Minkowski in the late nineteenth century; a classical reference is the book by Cassels [49]. Notable applications of lattices include H. Lenstra's polynomial-time algorithm [145] for integer linear programming in fixed dimension n, Viazovska's solution [214] of the sphere packing problem in dimension 8, the work of Kannan and Lovász [127] on the covering minima of convex bodies, and the capacity-achieving lattice coding scheme for the additive white Gaussian noise (AWGN) channel developed by Erez and Zamir [83]. More recently, in 2023, Reis and Rothvoss [196] introduced a $(\log(2n))^{O(n)}$-time randomized algorithm for solving integer programs in n variables.

Kyber and Dilithium were standardized by NIST in August 2024 as ML-KEM (FIPS 203) [168] and ML-DSA (FIPS 204) [167], respectively, whereas Falcon is being standardized as FN-DSA in FIPS 206, which at the time of writing has not yet been finalized. FrodoKEM was standardized by ISO in June 2026 [124]. These notes focus on developing a thorough understanding of Kyber, Dilithium and FrodoKEM, and providing background information to appreciate Falcon. As part of its post-quantum cryptography standardization initiative, NIST has also finalized standards for the stateful hash-based signature schemes LMS and XMSS in SP 800-208 [166], as well as the stateless hash-based signature scheme

SPHINCS+ in FIPS 205 [169]. Menezes, van Oorschot and Preneel [158] give an overview of these schemes.

2 Lattices

Contents

2.1 Basic definitions 5
2.2 Examples 6
2.3 Bases of a lattice 8
2.4 Successive minima 9
2.5 Fundamental lattice problems 13
2.6 Notes and further references 16
2.7 Exercises 18

2.1 Basic definitions

The sets of natural numbers, integers, rational numbers, real numbers, and complex numbers are denoted by $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$, and $\mathbb{C}$, respectively. We write $\mathbb{R}^n$ for the n-dimensional real vector space consisting of all length-n vectors $x = (x_1, x_2, \ldots, x_n)$ with $x_i \in \mathbb{R}$. The *length* of such a vector x is $\|x\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$, which is also denoted $\|x\|_2$ and called the *Euclidean length*, *Euclidean norm*, or ℓ_2*-norm*. The *infinity norm* of x is $\|x\|_\infty = \max_i |x_i|$. The set $\mathbb{Z}^n$ denotes the subset of $\mathbb{R}^n$ consisting of all length-n vectors whose components are integers.

Informally, a lattice can be viewed as a regular, repeating arrangement of points in real space. Formally, lattices are defined as follows.

Definition 2.1 A *lattice* L in $\mathbb{R}^n$ is the set of all integer linear combinations of m linearly independent vectors $b_1, b_2, \ldots, b_m \in \mathbb{R}^n$, where $m \leq n$. The (ordered) set $B = [b_1, b_2, \ldots, b_m]$ is called a *basis* of L and we write $L = L(B)$. The *dimension* of L is n, and the *rank* of L is m.

The lattice L consists of all vectors of the form

$$v = c_1 b_1 + c_2 b_2 + \cdots + c_m b_m,$$

where the scalars c_i are integers. The basis B can be represented by an $n \times m$ *basis matrix*, also denoted by B, whose columns are the basis vectors $b_1, b_2, \ldots, b_m$:

$$B = \begin{bmatrix} | & | & \cdots & | \\ b_1 & b_2 & \cdots & b_m \\ | & | & \cdots & | \end{bmatrix}.$$

With this notation, the lattice can be described compactly as $L = \{Bx : x \in \mathbb{Z}^m\}$.

An equivalent characterization of lattices is given by the following definition.

Definition 2.2 A *lattice* L is a discrete additive subgroup of $\mathbb{R}^n$.

Here, L being an *additive subgroup* of $\mathbb{R}^n$ means that L is a non-empty subset of $\mathbb{R}^n$ that is closed under vector addition and subtraction; that is, $x + y, -x \in L$ for all $x, y \in L$. The term *discrete* means that there exists an $\varepsilon > 0$ such that, for every $x \in L$, there are no other lattice points within distance ε of x. We omit the proof that Definitions 2.1 and 2.2 are equivalent.

In cryptographic applications, one typically works with lattices whose basis vectors have integer coordinates, that is, $b_1, b_2, \ldots, b_m \in \mathbb{Z}^n$. In this case, $L \subseteq \mathbb{Z}^n$ and L is called an *integer lattice*. The lattices considered in the remainder of this document will primarily be integer lattices.

Definition 2.3 A *full-rank lattice* in $\mathbb{R}^n$ is a lattice of rank n.

If L is a full-rank lattice in $\mathbb{R}^n$ with basis $B = [b_1, b_2, \ldots, b_n]$, then B is also a basis of the vector space $\mathbb{R}^n$. The lattice L consists of all *integer* linear combinations of $b_1, b_2, \ldots, b_n$, whereas the vector space $\mathbb{R}^n$ consists of all *real* linear combinations of these same vectors.

Definition 2.4 Let L and L' be lattices in $\mathbb{R}^n$. Then L' is a *sublattice* of L if $L' \subseteq L$.

To show that L' is a sublattice of L, it suffices to verify that each vector b' in a basis B' of L' is an element of L, that is, can be expressed as an integer linear combination of the vectors in a basis B of L.

Henceforth, unless explicitly stated otherwise, all lattices and sublattices will be assumed to be full-rank integer lattices.

2.2 Examples

The most basic example of a full-rank integer lattice is $\mathbb{Z}^n$, whose basis matrix is the identity matrix $B = I_n$. We next give several examples of full-rank integer lattices in two dimensions.

Example 2.5 (*two-dimensional lattice*) Consider the basis $B_1 = [(1, 0), (0, 1)]$. The lattice generated by B_1 is

$$L_1 = L(B_1) = \{B_1 x \ : \ x \in \mathbb{Z}^2\},$$

where

$$B_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Thus, $L_1 = \mathbb{Z}^2$ is simply the integer grid in the plane, as illustrated in Figure 2.1(a).

Definition 2.6 Let $L = L(B)$ be a full-rank lattice in $\mathbb{R}^n$ with basis $B = [b_1, b_2, \ldots, b_n]$. The *fundamental parallelepiped* of L (with respect to the basis B) is defined as

$$P(B) = \{a_1 b_1 + a_2 b_2 + \cdots + a_n b_n \ : \ a_i \in [0, 1)\}.$$

Equivalently, $P(B) = \{Bx \ : \ x \in [0, 1)^n\}$.

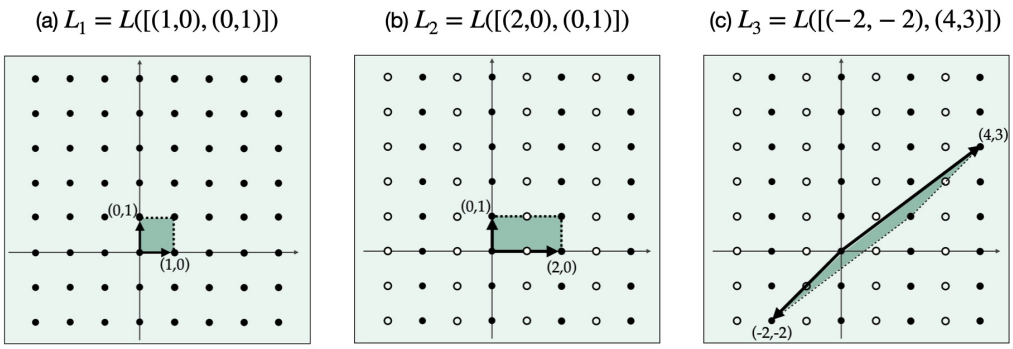


Figure 2.1: Three full-rank integer lattices in $\mathbb{R}^2$ and their fundamental parallelepipeds.

The translates $P(B) + x$, $x \in L(B)$, of the fundamental parallelepiped partition $\mathbb{R}^n$; that is, they form a *tiling* of $\mathbb{R}^n$ (Exercise 2.8). The "corners" of these parallelepipeds are precisely the lattice points of $L(B)$. Three examples of fundamental parallelepipeds are shown in Figure 2.1.

Example 2.7 (*two-dimensional lattice*) Consider the basis $B_2 = [(2,0),(0,1)]$. The lattice generated by B_2 is

$$L_2 = L(B_2) = \{B_2 x \ : \ x \in \mathbb{Z}^2\},$$

where

$$B_2 = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}.$$

The lattice L_2 and its fundamental parallelepiped $P(B_2)$ are shown in Figure 2.1(b).

Observe that L_2 is a sublattice of L_1. To see that $L_2 \neq L_1$, it suffices to exhibit one vector in L_1 that does not belong to L_2. For instance, $(1,0) \in L_1$ but

$$(1,0) = \frac{1}{2} \cdot (2,0) + 0 \cdot (0,1),$$

which is not an integer linear combination of the basis vectors of B_2. Hence, $(1,0) \notin L_2$.

Example 2.8 (*two-dimensional lattice*) Consider the basis $B_3 = [(-2,-2),(4,3)]$. The lattice generated by B_3 is

$$L_3 = L(B_3) = \{B_3 x \ : \ x \in \mathbb{Z}^2\},$$

where

$$B_3 = \begin{bmatrix} -2 & 4 \\ -2 & 3 \end{bmatrix}.$$

The lattice L_3 and its fundamental parallelepiped $P(B_3)$ are depicted in Figure 2.1(c).

Figures 2.1(b) and 2.1(c) suggest that L_2 and L_3 are in fact the same lattice. This can be confirmed by verifying that each lattice is a sublattice of the other. Indeed, $L_2 \subseteq L_3$ since

$$(2,0) = 3 \cdot (-2,-2) + 2 \cdot (4,3) \quad \text{and} \quad (0,1) = -2 \cdot (-2,-2) - 1 \cdot (4,3),$$

and $L_3 \subseteq L_2$ since

$$(-2,-2) = -1 \cdot (2,0) - 2 \cdot (0,1) \quad \text{and} \quad (4,3) = 2 \cdot (2,0) + 3 \cdot (0,1).$$

The basis $B_2 = [(2,0),(0,1)]$ for L_2 is "nicer" than the basis $B_3 = [(-2,-2),(4,3)]$, since the vectors in B_2 are shorter and mutually orthogonal. In general, the difficulty of computational problems on lattices depends heavily on the quality of the basis used to represent the lattice. For example, consider the Shortest Vector Problem (SVP), which asks for a shortest nonzero vector in a lattice, given a basis of the lattice. Examining the basis B_2 immediately reveals that $(0,1)$ is a shortest nonzero vector of L_2. In contrast, if L_2 were presented using the basis B_3, then identifying such a vector would be significantly less straightforward. For another illustration of the benefits of a "good" lattice basis, see Example 2.24.

A common approach to solving lattice problems is therefore to first find a good basis of the lattice in question, meaning one whose vectors are relatively short and nearly orthogonal. The most well-known algorithm for this task is the LLL algorithm, which is described in §10.

2.3 Bases of a lattice

Let $L = L(B)$ be an n-dimensional lattice in $\mathbb{R}^n$ with ordered basis $B = [b_1, b_2, \ldots, b_n]$. The following operations transform B into another basis of the same lattice:

(i) Exchange two basis vectors: $b_i \leftrightarrow b_j$.

(ii) Replace a basis vector by its negation: $b_i \leftarrow -b_i$.

(iii) Add an integer multiple of one basis vector to another: $b_i \leftarrow b_i + cb_j$ where $c \in \mathbb{Z}$.

Each of these corresponds to an elementary column operation on the basis matrix B. The first two operations change the sign of the determinant of B, while the third operation leaves the determinant unchanged.

By repeatedly applying these operations, one obtains infinitely many distinct bases of the same lattice (when $n \geq 2$). The following theorem characterizes all bases of a lattice.

Theorem 2.9 (*characterization of lattice bases*) Let $L = L(B_1)$ be an n-dimensional lattice. An $n \times n$ matrix B_2 is also a basis of L if and only if $B_2 = B_1 U$, where U is an $n \times n$ matrix with integer entries and determinant ± 1. (Such a matrix U is called *unimodular*.)

Proof: ($\Rightarrow$) Suppose that B_1 and B_2 are both bases of the lattice L. Since each is a basis of the vector space $\mathbb{R}^n$, there exists an invertible matrix $U \in \mathbb{R}^{n\times n}$ such that $B_2 = B_1U$. Because the columns of B_2 belong to L, the entries in U must be integers, and hence $U \in \mathbb{Z}^{n\times n}$. Similarly, there exists an invertible matrix $V \in \mathbb{Z}^{n\times n}$ such that $B_1 = B_2V$.

Substituting yields $B_1 = B_2V = (B_1U)V = B_1(UV)$. Since B_1 is invertible, it follows that $UV = I_n$. Taking determinants, we obtain $\det(UV) = \det(U)\det(V) = 1$, which implies that $\det(U) = \det(V) = 1$ or $\det(U) = \det(V) = -1$. Hence, U is unimodular and $B_2 = B_1U$.

($\Leftarrow$) Conversely, suppose that $B_2 = B_1U$ for some unimodular matrix U. Then $L(B_2) \subseteq L(B_1)$. Recall the inverse formula $U^{-1} = \operatorname{adj}(U)/\det(U)$. Since the adjugate $\operatorname{adj}(U)$ has integer entries and $\det(U) = \pm 1$, it follows that U^{-1} is also unimodular. Thus $B_1 = B_2U^{-1}$ implies that $L(B_1) \subseteq L(B_2)$. Therefore, $L(B_1) = L(B_2)$. □

Example 2.10 (*two bases of a lattice*) As we saw in Examples 2.7 and 2.8, the matrices

$$B_2 = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad B_3 = \begin{bmatrix} -2 & 4 \\ -2 & 3 \end{bmatrix}$$

are both bases of the same lattice in $\mathbb{R}^2$. Indeed, one verifies that $B_2 = B_3U$, where

$$U = \begin{bmatrix} 3 & -2 \\ 2 & -1 \end{bmatrix}$$

is unimodular.

The volume[1] of a lattice provides a quantitative measure of how densely the lattice points are distributed in space: larger volumes correspond to sparser lattices. If L is two-dimensional, then its volume coincides with the area of any of its parallelepipeds.

Definition 2.11 The *volume* of a full-rank lattice $L = L(B)$ is defined as $\operatorname{vol}(L) = |\det(B)|$.

If B' is any other basis of $L = L(B)$, then $B' = BU$ for some unimodular matrix U. Consequently, $\det(B') = \det(B)\det(U)$, and hence $\det(B') = \pm\det(B)$. It follows that $|\det(B')| = |\det(B)|$. This shows that the volume is a *lattice invariant*, meaning that it is independent of the particular basis chosen to represent the lattice.

2.4 Successive minima

The successive minima are important invariants of a lattice.

Definition 2.12 Let $L \subseteq \mathbb{R}^n$ be a full-rank lattice. For each $i \in [1, n]$, the *i-th successive minimum* $\lambda_i(L)$ is defined as the smallest real number r such that L contains i linearly independent vectors, each of length at most r.

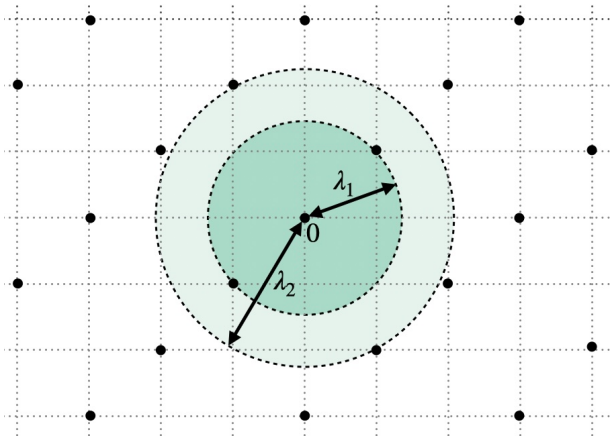


Figure 2.2: Successive minima.

Figure 2.2 shows the first two successive minima for a two-dimensional lattice. By definition, the successive minima form a non-decreasing sequence:

$$\lambda_1(L) \le \lambda_2(L) \le \cdots \le \lambda_n(L). \tag{1}$$

The first successive minimum, $\lambda_1(L)$, is simply the length of a shortest nonzero vector in L. Such a vector is not unique; for example, if $x \in L$ has length $\lambda_1(L)$, then so does $-x$.

Example 2.13 The *Leech lattice* Λ_{24} is a full-rank lattice in $\mathbb{R}^{24}$ with basis matrix shown in Figure 2.3. The length of a shortest nonzero vector in Λ_{24} is $\lambda_1(\Lambda_{24}) = 2$. In fact, there are 196,560 vectors of length 2 in Λ_{24}. This shows that a lattice can have a large number of shortest nonzero vectors.

Finding a shortest nonzero vector—or even determining its length—is a central problem in algorithmic number theory and lattice-based cryptography. In 1850, Hermite proved an exponential upper bound on the first successive minimum.

Definition 2.14 For each integer $n \ge 1$, the *Hermite constant*[2] γ_n is defined by

$$\gamma_n = \max_L \left(\frac{\lambda_1(L)}{\mathrm{vol}(L)^{1/n}} \right)^2, \tag{2}$$

where the maximum is taken over all full-rank lattices $L \subseteq \mathbb{R}^n$.

It follows immediately from (2) that every full-rank lattice $L \subseteq \mathbb{R}^n$ satisfies

$$\lambda_1(L) \le \sqrt{\gamma_n}\ \mathrm{vol}(L)^{1/n}. \tag{3}$$

[1]Formally, the volume of a measurable set $A \subseteq \mathbb{R}^n$ is defined as its Lebesgue measure, given by the integral $\int_A 1\,dx$. The volume of a lattice is defined to be the volume of its fundamental parallelepiped.

[2]γ_n is a constant in the sense that its value depends only on the dimension n, and not on any particular n-dimensional lattice.

$$B = \frac{1}{\sqrt{8}} \left[\begin{array}{cccc|cccc|cccc|cccc|cccc|cccc}
8 & 4 & 4 & 4 & 4 & 4 & 4 & 2 & 4 & 4 & 4 & 2 & 4 & 2 & 2 & 2 & 4 & 2 & 2 & 2 & 0 & 0 & 0 & -3 \\
0 & 4 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 2 & 2 & 0 & 0 & 1 \\
0 & 0 & 4 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 1 \\
0 & 0 & 0 & 4 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 2 & 0 & 0 & 1 \\
\hline
0 & 0 & 0 & 0 & 4 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 2 & 2 & 2 & 0 & 2 & 2 & 2 & 2 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 4 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 4 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 1 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 2 & 0 & 2 & 2 & 2 & 0 & 2 & 2 & 2 & 2 & 2 & 2 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 2 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 2 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 1 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 2 & 2 & 2 & 0 & 0 & 0 & 0 & 2 & 2 & 2 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 2 & 2 & 2 & 2 & 2 & 2 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 1 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 2 & 2 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}\right].$$

Figure 2.3: A basis matrix for the Leech lattice $\Lambda_{24} \subseteq \mathbb{R}^{24}$ (Example 2.13). The basis vectors are the columns of the matrix.

Hermite proved that

$$\gamma_n \leq \left(\frac{4}{3}\right)^{(n-1)/2}, \tag{4}$$

which yields an upper bound on $\lambda_1(L)$:

$$\lambda_1(L) \leq \left(\frac{4}{3}\right)^{(n-1)/4} \mathrm{vol}(L)^{1/n}. \tag{5}$$

In 1896, Minkowski gave a much improved upper bound on the first successive minimum.

Theorem 2.15 (*Minkowski's Theorem*) Let $L \subseteq \mathbb{R}^n$ be a full-rank lattice. Then

$$\lambda_1(L) \leq \sqrt{n}\ \mathrm{vol}(L)^{1/n}. \tag{6}$$

Minkowski's Theorem immediately gives a linear upper bound on the Hermite constants:

$$\gamma_n \leq n.$$

The exact values of the Hermite constants have been explicitly determined only for a few small dimensions, as shown in Table 2.1.

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
γ_n^n	4/3	2	4	8	64/3	64	256	2^{48}

Table 2.1: The known Hermite constants.

More generally, tight asymptotic bounds show that γ_n grows linearly in n:

$$\frac{n}{2\pi e} + \frac{\log(\pi n)}{2\pi e} + o(1) \leq \gamma_n \leq \frac{1.744n}{2\pi e}(1 + o(1)). \tag{7}$$


Herman Minkowski

The bound in Minkowski's Theorem is not tight in general (Exercise 2.15). Nevertheless, the *Gaussian heuristic* suggests that, for a suitably defined notion of a random full-rank lattice, this bound is close to optimal. More specifically, the expected length of a shortest nonzero vector in such a lattice is approximately

$$\lambda_1(L) \approx \sqrt{\frac{n}{2\pi e}}\ \mathrm{vol}(L)^{1/n}. \tag{8}$$

The quantity on the right-hand side of (8) is approximately equal to the radius of the n-dimensional sphere whose volume is $\mathrm{vol}(L)$.

Upper bounds for the second and higher successive minima that are similar to (3) and (6) are not possible; see Exercise 2.17. Nonetheless, the following theorem provides an upper bound on the geometric mean of the successive minima. In view of (1), this result is stronger than Theorem 2.15.

Theorem 2.16 (*Minkowski's Second Theorem*) Let $L \subseteq \mathbb{R}^n$ be a full-rank lattice. Then

$$\left(\prod_{i=1}^{n} \lambda_i(L)\right)^{1/n} \leq \sqrt{n}\ \mathrm{vol}(L)^{1/n}. \tag{9}$$

2.5 Fundamental lattice problems

We now introduce several fundamental computational problems on lattices whose presumed hardness underpins the security of lattice-based cryptosystems. Throughout, a lattice L is specified by a basis B, and all lattices are assumed to be full-rank integer lattices of dimension n.

2.5.1 Shortest Vector Problem (SVP)

Definition 2.17 The *Shortest Vector Problem* (SVP) is defined as follows: given a lattice $L = L(B)$, find a nonzero lattice vector of smallest length $\lambda_1(L)$.

SVP has been shown to be NP-hard, and it is therefore reasonable to assume that it is hard in the worst case. The fastest classical algorithm known for SVP runs in (heuristic) time $2^{0.292n+o(n)}$, whereas the best quantum algorithm known is slightly faster, with (heuristic) running time $2^{0.256n+o(n)}$. In both cases, the running time is fully exponential in the lattice dimension.

A natural relaxation of SVP is the approximate version, in which one seeks a nonzero lattice vector whose length is within a prescribed approximation factor of $\lambda_1(L)$.

Definition 2.18 The *Approximate Shortest Vector Problem* (approx-SVP or SVP_γ) is defined as follows: given a lattice $L = L(B)$, find a nonzero lattice vector of length at most $\gamma \cdot \lambda_1(L)$, where γ is the approximation factor.

Notice that SVP_1 coincides with SVP. As γ increases, SVP_γ becomes easier; more precisely, if $\gamma_1 \geq \gamma_2$ then $\mathrm{SVP}_{\gamma_1} \leq \mathrm{SVP}_{\gamma_2}$.[3] For constant approximation factors γ, SVP_γ remains NP-hard. However, SVP_γ is unlikely to be NP-hard when $\gamma > \sqrt{n}$. For approximation factor $\gamma = 2^k$, the fastest classical algorithm known for SVP_γ runs in time $2^{\tilde{\Theta}(n/k)}$, where the $\tilde{\Theta}$ notation suppresses a power of $\log n$. When the approximation factor is very large, more precisely when $\gamma > 2^{(n \log\log n)/\log n}$, SVP_γ can be solved in polynomial time

[3] Let A and B be two computational problems. We say that A *polynomial-time* reduces to B, denoted $A \leq B$, if there exists a polynomial-time algorithm for solving A that uses a polynomial-time algorithm for solving B as an oracle (a hypothetical subroutine). We say that A and B are *computationally equivalent* if $A \leq B$ and $B \leq A$.

by applying the BKZ lattice basis reduction algorithm (§11) to compute a short lattice basis and then selecting the shortest basis vector.

2.5.2 Shortest Independent Vectors Problem (SIVP)

Another important lattice problem is to find many short linearly independent lattice vectors.

Definition 2.19 The *Shortest Independent Vectors Problem* (SIVP) is defined as follows: given a lattice $L = L(B)$, find n linearly independent lattice vectors, each of length at most $\lambda_n(L)$.

Definition 2.20 The *Approximate Shortest Independent Vectors Problem* (approx-SIVP or SIVP_γ) is defined as follows: given a lattice $L = L(B)$, find n linearly independent lattice vectors, each of length at most $\gamma \cdot \lambda_n(L)$.

It is worth emphasizing that a solution to SIVP need not form a lattice basis; the vectors obtained might be linearly independent without spanning the full lattice (see Exercises 2.19 and 2.20).

The complexity status of SIVP and approx-SIVP closely parallels that of SVP and approx-SVP. In particular, SIVP is NP-hard, as is SIVP_γ for constant approximation factors γ, and the fastest algorithms for these problems have fully exponential running time. Moreover, there is a known reduction $\mathrm{SIVP}_{\gamma\sqrt{n}} \leq \mathrm{SVP}_\gamma$, further highlighting the close relationship between these fundamental lattice problems.

2.5.3 Closest Vector Problem (CVP)

The next fundamental lattice problem we consider is finding a lattice vector closest to a given target vector.

Definition 2.21 Let $L \subseteq \mathbb{R}^n$ be a lattice and let $t \in \mathbb{R}^n$. The *distance* from t to L is defined by

$$\mathrm{dist}(t, L) = \min_{x \in L} \|x - t\|_2.$$

Definition 2.22 The *Closest Vector Problem* (CVP) is defined as follows: given a lattice $L = L(B)$ and a target vector $t \in \mathbb{R}^n$, find a lattice vector $x \in L$ that is closest to t, that is, satisfying $\|x - t\|_2 = \mathrm{dist}(t, L)$.

Definition 2.23 The *Approximate Closest Vector Problem* (approx-CVP or CVP_γ) is defined as follows: given a lattice $L = L(B)$ and a target vector $t \in \mathbb{R}^n$, find a lattice vector $x \in L$ such that $\|x - t\|_2 \leq \gamma \cdot \mathrm{dist}(t, L)$.

Let $L \subseteq \mathbb{Z}^n$ be a lattice with basis $B = [b_1, \ldots, b_n]$, and let $t \in \mathbb{R}^n$ be a target vector. *Babai's rounding method* for finding a close lattice vector to t proceeds as follows. Express t as a linear combination of the basis vectors, $t = \sum_{i=1}^{n} t_i b_i$, then round each coefficient t_i

to the nearest integer (where ties are broken by rounding towards 0), yielding the lattice vector $x = \sum_{i=1}^{n} \lfloor t_i \rceil b_i$. Observe that the error vector $e = t - x$ lies in the parallelepiped

$$P_{1/2}(B) = \{a_1 b_1 + a_2 b_2 + \cdots + a_n b_n \ : \ -\tfrac{1}{2} \le a_i \le \tfrac{1}{2}\}.$$

It follows that the closeness of the approximation x to t depends on the quality of the basis B.

Example 2.24 (*finding close lattice vectors using rounding*) Consider the lattice $L \subseteq \mathbb{Z}^2$ depicted in Figure 2.4 with basis $B = [b_1, b_2]$ where $b_1 = (5, 3)$ and $b_2 = (2, 7)$. Another basis of L, which is a poorer-quality basis, is $B' = [b'_1, b'_2]$ where $b'_1 = (12, 13)$ and $b'_2 = (7, 10)$. Indeed,

$$\underbrace{\begin{bmatrix} 12 & 7 \\ 13 & 10 \end{bmatrix}}_{B'} = \underbrace{\begin{bmatrix} 5 & 2 \\ 3 & 7 \end{bmatrix}}_{B} \underbrace{\begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}}_{U}$$

where the matrix U is unimodular, confirming that B' is a basis of L.

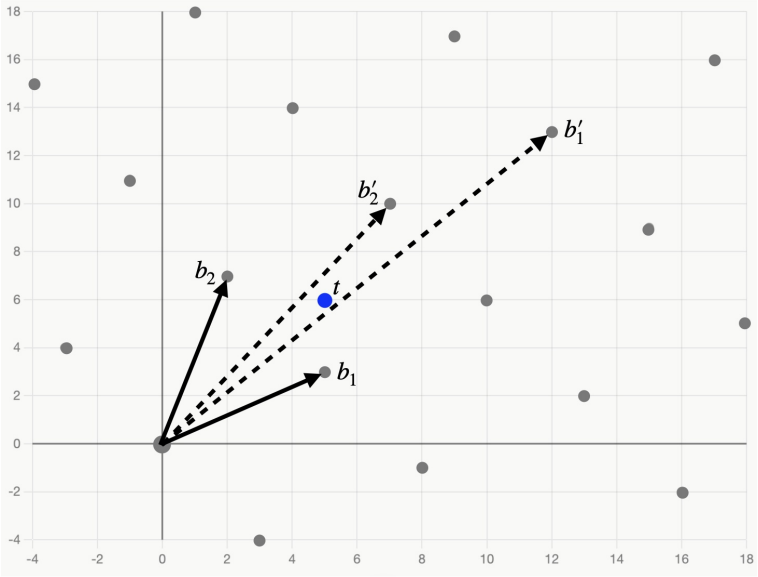


Figure 2.4: Finding a close lattice vector using rounding (cf. Example 2.24.

Consider the target vector $t = (5, 6)$. As can be seen from Figure 2.4, the closest lattice vector to t is $b_1 = (5, 3)$, and hence $\mathrm{dist}(t, L) = 3$. We have

$$t = \frac{23}{29} b_1 + \frac{15}{29} b_2 = \frac{8}{29} b'_1 + \frac{7}{29} b'_2.$$

Therefore, rounding with respect to basis B yields the lattice vector

$$x = \left\lfloor \frac{23}{29} \right\rceil b_1 + \left\lfloor \frac{15}{29} \right\rceil b_2 = 1 \cdot b_1 + 1 \cdot b_2 = (7, 10)$$

with $\|t - x\|_2 = \sqrt{20}$, whereas rounding with respect to B' yields

$$x' = \left\lfloor \frac{8}{29} \right\rceil b'_1 + \left\lfloor \frac{7}{29} \right\rceil b'_2 = 0 \cdot b'_1 + 0 \cdot b'_2 = (0, 0)$$

with $\|t - x'\|_2 = \sqrt{61}$. While rounding fails to find the closest lattice vector in either case, the choice of basis significantly affects the quality of the approximation.

To improve the quality of the approximation, one can first find a good basis $\tilde{B}$ of L using the LLL (§10) or BKZ (§11) algorithms, and then apply Babai's rounding method using $\tilde{B}$; see Exercise 12.5.

2.5.4 Relationships

The CVP and SIVP problems are closely related. It is known that $\text{SIVP}_\gamma \leq \text{CVP}_\gamma$ for every $\gamma \geq 1$, and $\text{CVP} \leq \text{SIVP}$. The CVP and SVP problems are also closely related: known relationships are $\text{SVP}_\gamma \leq \text{CVP}_\gamma$ for every $\gamma \geq 1$ (Exercise 10.25) and $\text{CVP}_{\gamma^2\sqrt{n}} \leq \text{SVP}_\gamma$. Figure 2.5 summarizes the known relationships between SVP, SIVP, and CVP, and their approximate versions.

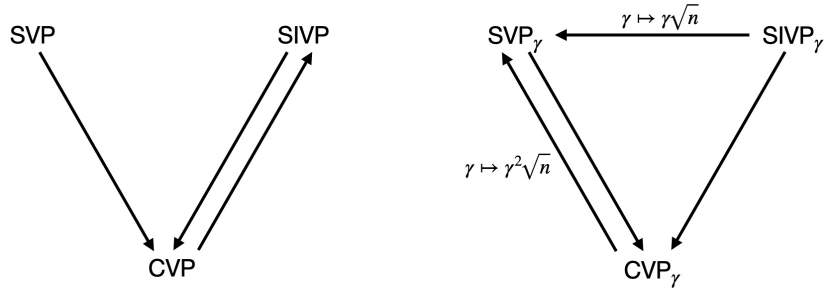


Figure 2.5: The known relationships between SVP, SIVP, CVP, and their approximate versions. An arrow $A \to B$ indicates that A polynomial-time reduces to B. Any loss in the approximation factor incurred by the reduction is shown next to the arrow.

2.6 Notes and further references

The first monograph devoted to algorithmic problems in lattices and their computational complexity is the 2002 book by Micciancio and Goldwasser [161]. Part IV of Galbraith's book [93] introduces lattices and explores their applications in cryptanalysis. Nguyen [172] explains the connections between the Hermite constants, lattice algorithms, and sphere packings. Peikert [186] provides a comprehensive survey of the major developments in lattice-based cryptography up to 2015, including fully homomorphic encryption (FHE) and attribute-based encryption (ABE). More recently, Lyubashevsky [153] offers a detailed exposition of the mathematical foundations and design principles underlying

SIS- and LWE-based cryptosystems, including Kyber and Dilithium. Kyber and Dilithium are fully specified in the NIST standards FIPS 203 [168] and FIPS 204 [167].

Conway and Sloane's book [60] is a rich source of information on the Leech lattice Λ_{24}. The basis matrix for Λ_{24} presented in Example 2.13 is taken from Figure 4.12 of that book. Bonnecaze and Solé [38] gave a construction of the Leech lattice from the extended quaternary Golay code of length 24 over $\mathbb{Z}_4$. The *sphere packing* problem is to find an arrangement of non-overlapping unit balls in $\mathbb{R}^n$ so that the proportion of space they cover is as large as possible. Cohn, Kumar, Miller, Radchenko and Viazovska [59] proved that the densest sphere packing in $\mathbb{R}^{24}$ is attained with unit balls centered at the points of the Leech lattice. This work builds on Viazovska's proof [214] that the densest sphere packing in $\mathbb{R}^8$ is attained by placing unit balls at the points of the scaled lattice $\sqrt{2}E_8$ (the E_8 lattice is defined in Exercise 10.11).

In 1998, Ajtai [4] proved that SVP is NP-hard under randomized reductions. Recently, in 2026, Wan [215] removed the randomness from Ajtai's reduction, finally establishing the NP-hardness of SVP. Khot [128] proved that the approximate version SVP_γ remains NP-hard under randomized reductions for constant approximation factors γ. The claim that SVP_γ is unlikely to be NP-hard for $\gamma > \sqrt{n}$ is supported by a result of Aharonov and Regev [2] that there exists $c > 0$ for which $\mathrm{GapSVP}_{c\sqrt{n}}$ lies in NP $\cap$ coNP (GapSVP_β is a decisional version of SVP). Blömer and Seifert [32] proved that SIVP_γ is NP-hard for constant approximation factors γ. Laarhoven, van de Pol and de Weger [136] and Stephens-Davidowitz [209] provide comprehensive overviews of the relationships between the main lattice problems. Stephens-Davidowitz [209] gave a proof that $\mathrm{SIVP}_{\gamma\sqrt{n}} \leq \mathrm{SVP}_\gamma$ for every $\gamma \geq 1$.

Micciancio [159] gave an elementary proof that CVP is NP-hard. In September 2026, Song [205] proved that $\mathrm{CVP}_{n^{1/2-\varepsilon}}$ is NP-hard for every fixed $0 < \varepsilon < 1/2$. Micciancio [160] gave polynomial-time reductions CVP $\leq$ SIVP and $\mathrm{SIVP}_\gamma \leq \mathrm{CVP}_\gamma$ for every $\gamma \geq 1$. Goldreich, Micciancio, Safra and Seifert [104] showed that for any $\gamma \geq 1$, SVP_γ polynomial-time reduces to CVP_γ. Conversely, Stephens-Davidowitz [209] proved that $\mathrm{CVP}_{\gamma^2\sqrt{n}}$ polynomial-time reduces to SVP_γ.

Babai [19] introduced two conceptually simple polynomial-time algorithms for CVP_γ. His *nearest plane algorithm* achieves an approximation factor of $\gamma = 2^{n/2}$, while his *rounding algorithm* (see Exercise 12.5) attains $\gamma = 1 + 2n(\frac{9}{2})^{n/2}$. The nearest plane algorithm has found applications in several areas beyond cryptography. In particular, Chen et al. [52] showed in 2025 that GPTQ, a widely used method for quantizing the weights of large language models (LLMs), can be viewed as an application of the nearest plane algorithm.

Gao [96] constructed a family of lattices for which the Bounded Distance Decoding (BDD) problem (Definition 4.19) is NP-hard but solvable in polynomial time on average, demonstrating that worst-case NP-hardness of a lattice problem does not imply its average-case hardness.

2.7 Exercises

Exercise 2.1 (†) Consider the lattice $L \subseteq \mathbb{Z}^3$ with basis matrix

$$B = \begin{bmatrix} 2 & 0 & -5 \\ 4 & 3 & 1 \\ -2 & -3 & 2 \end{bmatrix}.$$

(a) Show that $(7, -3, 2)$ is in L.
(b) Show that $(-4, 4, 0)$ is not in L.

Exercise 2.2 (†) Show that the following two matrices are basis matrices for the same lattice.

$$B_1 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 2 \end{bmatrix}, \qquad B_2 = \begin{bmatrix} 1 & 3 & 5 \\ 1 & 4 & 6 \\ 0 & 2 & 4 \end{bmatrix}.$$

Exercise 2.3 (†) Consider the lattices $L_1 = L(B_1)$ and $L_2 = L(B_2)$ where

$$B_1 = \begin{bmatrix} 2 & 1 & 1 \\ 0 & 3 & 2 \\ 1 & 2 & 4 \end{bmatrix} \quad \text{and} \quad B_2 = \begin{bmatrix} 3 & 4 & 2 \\ 2 & 6 & 5 \\ 5 & 5 & 6 \end{bmatrix}.$$

(a) Show that L_2 is a sublattice of L_1.
(b) Show that $L_1 \neq L_2$.

Exercise 2.4 Define $\mathbb{Z}[\sqrt{2}] = \{a + b\sqrt{2} : a, b \in \mathbb{Z}\} \subseteq \mathbb{R}$. Prove that $\mathbb{Z}[\sqrt{2}]$ is not a lattice. Hint: Let $\theta = \sqrt{2} - 1$, and show that $\theta^m \in \mathbb{Z}[\sqrt{2}]$ for all $m \geq 1$.

Exercise 2.5 Define $\sigma : \mathbb{Z}[\sqrt{2}] \longrightarrow \mathbb{R}^2$ by $\sigma(a + b\sqrt{2}) = (a + b\sqrt{2}, a - b\sqrt{2})$, and let $L = \sigma(\mathbb{Z}[\sqrt{2}]) \subseteq \mathbb{R}^2$.
(a) Prove that L is a subgroup of $\mathbb{R}^2$.
(b) Prove that σ is injective.
(c) Prove that L is a lattice.

Exercise 2.6 Let L_1 and L_2 be full-rank integer lattices with $L_1 \subseteq L_2$. Prove that $\mathrm{vol}(L_2)$ divides $\mathrm{vol}(L_1)$.

Exercise 2.7 Let $n \geq 1$. Prove that there is a unique full-rank integer lattice of volume 1 in $\mathbb{R}^n$.

Exercise 2.8 Let $L = L(B) \subseteq \mathbb{R}^n$ be a full-rank lattice. Recall that $P(B)$ is the fundamental parallelepiped of L with respect to B. Prove that every vector $x \in \mathbb{R}^n$ can be written uniquely as $x = u + v$ where $u \in L$ and $v \in P(B)$.

Exercise 2.9 Prove that $\lambda_1(L) = \min_{x,y \in L,\ x \neq y} \|x - y\|_2$.

Exercise 2.10 Prove that $\lambda_i(\mathbb{Z}^n) = 1$ for all $1 \le i \le n$.

Exercise 2.11 (†) Consider the two-dimensional integer lattice L with basis $b_1 = (2, 0)$, $b_2 = (0, 3)$. Determine $\lambda_1(L)$.

Exercise 2.12 (†) Consider the two-dimensional integer lattice L with basis $b_1 = (2, 1)$, $b_2 = (1, 2)$. Determine $\lambda_1(L)$.

Exercise 2.13 Consider the *hexagonal lattice* $L \subseteq \mathbb{R}^2$ with basis $b_1 = (1, 0)$, $b_2 = (\frac{1}{2}, \frac{\sqrt{3}}{2})$.
(a) (†) Find all lattice vectors of length 1.
(b) Prove that $\lambda_1(L) = 1$.
(c) Show that L satisfies Hermite's inequality (3) with equality.

Exercise 2.14 (†) Consider the lattice $L \subseteq \mathbb{Z}^2$ with basis vectors $b_1 = [4, 1]$, $b_2 = [1, 4]$. Show that $\lambda_1(L)^2 > \mathrm{vol}(L)$.
Hint: Gauss's algorithm (Algorithm 10.10) can be used to determine $\lambda_1(L)$.

Exercise 2.15 (†) Let d_i be positive real numbers with $d_1 \le d_2 \le \cdots \le d_n$. Prove that there exists a lattice $L \subseteq \mathbb{R}^n$ with $\lambda_i(L) = d_i$ for all $1 \le i \le n$.

Exercise 2.16 Minkowski's *Convex Body Theorem* is the following: Let $L \subseteq \mathbb{R}^n$ be a full-rank lattice, and let $C \subseteq \mathbb{R}^n$ be a subset that is symmetric about the origin (i.e., $C = -C$), convex (i.e., for any $c_1, c_2 \in C$ and $\lambda \in [0, 1]$, $\lambda c_1 + (1-\lambda)c_2 \in C$), and compact (i.e., closed and bounded). If $\mathrm{vol}(C) \ge 2^n \mathrm{vol}(L)$, then C contains at least one nonzero lattice point.
(a) Use Minkowski's convex body theorem to prove that every full-rank lattice $L \subseteq \mathbb{R}^n$ has a nonzero vector x satisfying $0 < \|x\|_\infty \le \mathrm{vol}(L)^{1/n}$.
Hint: Let $r = \mathrm{vol}(L)^{1/n}$ and consider the hypercube $C = [-r, r]^n$.
(b) Use Minkowski's convex body theorem to prove Minkowski's Theorem (Theorem 2.15).

Exercise 2.17 Let $0 < \varepsilon < 1$, and consider the lattice $L_\varepsilon \subseteq \mathbb{R}^2$ with basis matrix

$$B_\varepsilon = \begin{bmatrix} \varepsilon & 0 \\ 0 & 1/\varepsilon \end{bmatrix}.$$

Prove that $\mathrm{vol}(L_\varepsilon) = 1$, $\lambda_1(L_\varepsilon) = \varepsilon$, and $\lambda_2(L_\varepsilon) = 1/\varepsilon$. Conclude that, relative to the lattice volume, the first and second successive minima can be made arbitrarily small and arbitrarily large, respectively.

Exercise 2.18 An integer lattice L is said to be *q-ary* if $q\mathbb{Z}^n \subseteq L$. Prove that every integer lattice L is q-ary for $q = \mathrm{vol}(L)$.

Exercise 2.19 Let L be the set of all vectors $(x_1, x_2, x_3, x_4) \in \mathbb{Z}^4$ such that $\sum_{i=1}^4 x_i$ is even.
(a) Prove that L is a full-rank lattice.

(b) Prove that $\lambda_1(L) = \lambda_2(L) = \lambda_3(L) = \lambda_4(L) = \sqrt{2}$.
(c) Prove that the following is a basis matrix for L:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

(d) (†) Find a set of linearly independent lattice vectors $[b_1, b_2, b_3, b_4]$ that is not a lattice basis, but which is shortest possible, i.e., $\|b_i\|_2 = \sqrt{2}$.

Exercise 2.20 Let $n \geq 5$ and let e_i denote the i-th unit vector in $\mathbb{R}^n$. Consider the lattice $L = L(B) \subseteq \mathbb{Z}^n$ with basis $B = [2e_1, 2e_2, \ldots, 2e_{n-1}, e_1 + \cdots + e_n]$.
(a) Prove that $\lambda_1(L) = 2$.
(b) Prove that $I = [2e_1, 2e_2, \ldots, 2e_n]$ is a linearly independent set of lattice vectors.
(c) Prove that $\lambda_2(L) = \cdots = \lambda_n(L) = 2$.
(d) Prove that L does not have a basis $B = [b_1, \ldots, b_n]$ with $\max_i \|b_i\|_2 = \lambda_n(L)$.

Exercise 2.21 Let $i \geq 1$ and let $L \subseteq \mathbb{R}^i$ be a lattice whose shortest nonzero vectors have length $\sqrt{\gamma_i}\,\mathrm{vol}(L)^{1/i}$ (cf. Definition 2.14). Let B be a basis of L and define

$$B' = \begin{bmatrix} B & 0 \\ 0 & \sqrt{\gamma_i}\,\mathrm{vol}(L)^{1/i} \end{bmatrix}_{(i+1)\times(i+1)}.$$

(a) Prove that $\lambda_1(L') = \sqrt{\gamma_i}\,\mathrm{vol}(L)^{1/i}$ where $L' = L(B')$.
(b) Prove that $\gamma_i^i \leq \gamma_{i+1}^{i+1}$ for all $i \geq 1$.
(c) Prove that $\gamma_i^i \leq \gamma_j^j$ for all $1 \leq i \leq j$.

Exercise 2.22 Let B be a basis of a full-rank lattice $L \subseteq \mathbb{Z}^n$. Show that Babai's rounding method for finding a lattice vector x close to a target vector $t \in \mathbb{R}^n$ can be succinctly described as $x = B \cdot \lfloor B^{-1}t \rceil$.

3 Short Integer Solution (SIS) problem

Contents

3.1 Problem statement 21
3.2 The SIS lattice 23
3.3 Hardness of SIS 24
3.4 Notes and further references 26
3.5 Exercises 27

The security of the Dilithium signature scheme relies in part on the assumed hardness of the Short Integer Solution (SIS) problem. This section introduces SIS, presents a lattice formulation of SIS, and explores its relationship with SVP and SIVP.

3.1 Problem statement

Notation. q is a prime; $\mathbb{Z}_q = \{0, 1, 2, \ldots, q-1\}$ is the set of integers modulo q; $\mathbb{Z}_q^m$ is the set of length-m (column) vectors whose components are in $\mathbb{Z}_q$; $\mathbb{Z}_q^{n\times m}$ is the set of $n \times m$ matrices whose entries are in $\mathbb{Z}_q$; and $[-B, B]^m$ denotes the set of all length-m (column) vectors whose components are integers in the interval $[-B, B]$.

The SIS problem was introduced by Miklós Ajtai in 1996. It is parameterized by four values: positive integers n and m with $n \ll m$, a prime modulus q, and a bound B satisfying $B \ll q/2$.

Definition 3.1 The *Short Integer Solution* problem, denoted SIS(n, m, q, B), is defined as follows. Given a uniformly random matrix $A \in_R \mathbb{Z}_q^{n\times m}$, find a nonzero vector $z \in \mathbb{Z}^m$ such that $Az = 0 \pmod q$ and $z \in [-B, B]^m$; see Figure 3.1.

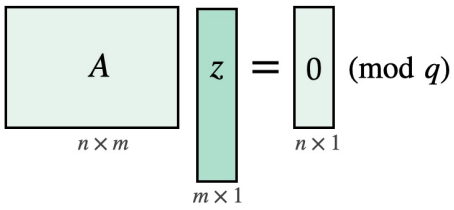


Figure 3.1: SIS problem.

SIS seeks to find a small nonzero solution in the null space of $A \pmod q$. While Gaussian elimination can efficiently determine a basis of this null space, the challenge is to find a nonzero element all of whose coordinates lie within $[-B, B]$.

When $n \geq m$, one expects that A has rank m, meaning that $z = 0$ is the only solution to $Az = 0 \pmod q$, and thus no SIS solution exists. This explains the requirement that $n \ll m$, which also ensures that A has full rank over $\mathbb{Z}_q$ with overwhelming probability. Throughout the remainder of this section, we therefore assume that A has rank n.

If $(B+1)^m > q^n$, the pigeonhole principle guarantees the existence of two distinct vectors $z_1, z_2 \in [0, B]^m$ such that $Az_1 = Az_2 \pmod q$. Consequently, their difference $z = z_1 - z_2$ is a solution to the SIS problem, establishing that a solution must exist. It is important to note, however, that an SIS solution is not unique. Indeed, if z is one SIS solution, then so is $-z$.

Example 3.2 (*SIS instance*) Let $n = 3$, $m = 5$, $q = 13$, and $B = 3$. Consider the SIS instance[4]

$$A = \begin{bmatrix} 10 & 6 & 3 & 6 & 0 \\ 9 & 11 & 0 & 10 & 3 \\ 7 & 11 & 5 & 7 & 8 \end{bmatrix} \in \mathbb{Z}_{13}^{3\times 5}.$$

We need to find nonzero $z = (z_1, z_2, z_3, z_4, z_5) \in [-3, 3]^5$ such that $Az = 0 \pmod{13}$.

Gaussian elimination (mod 13) on A yields the reduced matrix

$$A' = \begin{bmatrix} 1 & 0 & 0 & 8 & 12 \\ 0 & 1 & 0 & 5 & 7 \\ 0 & 0 & 1 & 0 & 11 \end{bmatrix}.$$

The rows of A' each provide one equation, and thus the complete solution to $Az = 0 \pmod{13}$ is $z_1 = 5z_4 + z_5$, $z_2 = 8z_4 + 6z_5$, $z_3 = 2z_5$, $z_4 \in \mathbb{Z}_{13}$, $z_5 \in \mathbb{Z}_{13}$. Among the $13^2 = 169$ solutions $z \in \mathbb{Z}_{13}^5$ are four SIS solutions (easily found by trial and error): $z = \pm(2, -2, 0, 3, 0)$ and $z = \pm(3, -3, 0, -2, 0)$.

We now give an example of a cryptographic primitive whose security relies on the hardness of SIS.

Example 3.3 (*cryptographic hash function*) Let $A \in_R \mathbb{Z}_q^{n\times m}$ with $m > n \log_2 q$. Define the hash function $H_A : \{0,1\}^m \longrightarrow \mathbb{Z}_q^n$ by

$$H_A(z) = Az \pmod q.$$

The function H_A is collision resistant under the assumption that $\mathrm{SIS}(n, m, q, 1)$ is hard. To justify this, suppose there exists an efficient algorithm that finds a collision for H_A, meaning two distinct $z_1, z_2 \in \{0,1\}^m$ such that $H_A(z_1) = H_A(z_2)$. Then $Az_1 = Az_2 \pmod q$, which implies that $Az = 0 \pmod q$ where $z = z_1 - z_2$. Since $z \neq 0$ and $z \in [-1, 1]^m$, the vector z is an $\mathrm{SIS}(n, m, q, 1)$ solution that has been found efficiently.

Henceforth, we will assume that $(B+1)^m \gg q^n$, ensuring that an SIS solution exists. We next present two alternative formulations of the SIS problem.

Definition 3.4 The *Inhomogeneous Short Integer Solution* problem $\mathrm{ISIS}(n, m, q, B)$ is the following. Given $A \in_R \mathbb{Z}_q^{n\times m}$ and $b \in_R \mathbb{Z}_q^n$, find $z \in \mathbb{Z}^m$ such that $Az = b \pmod q$ and $z \in [-B, B]^m$. Here, $B \ll q/2$.

[4] In this example, the condition $(B+1)^m > q^n$ is not satisfied.

Our assumption that $(B+1)^m \gg q^n$ implies that $(2B+1)^m \gg q^n$, and thus an ISIS solution is likely to exist.

Example 3.5 (*ISIS instance*) Let $n = 3$, $m = 5$, $q = 29$, and $B = 5$. Consider the ISIS instance

$$A = \begin{bmatrix} 4 & 19 & 5 & 4 & 22 \\ 8 & 10 & 19 & 1 & 7 \\ 13 & 10 & 20 & 7 & 12 \end{bmatrix} \in \mathbb{Z}_{29}^{3\times 5}, \quad b = \begin{bmatrix} 7 \\ 26 \\ 10 \end{bmatrix} \in \mathbb{Z}_{29}^{3}.$$

We need to find $z = (z_1, z_2, z_3, z_4, z_5) \in [-5, 5]^5$ such that $Az = b \pmod{29}$.

By trial and error, we find four ISIS solutions: $z = (-5, 3, -4, 3, -1)$, $(-4, -3, 3, -4, 5)$, $(0, 2, 2, -3, 0)$, $(4, -2, 3, 0, -2)$.

Theorem 3.6 SIS and ISIS are computationally equivalent.

Proof: (ISIS $\leq$ SIS) Let (A, b) be an ISIS(n, m, q, B) instance. Select $j \in_R [1, m+1]$ and $c \in_R [-B, B]$ with $c \neq 0$. Let A' be the $n \times (m+1)$ matrix obtained by inserting $-c^{-1}b \bmod q$ as a new jth column in A. Now, use an SIS solver to find an SIS solution $z' \in [-B, B]^{m+1}$ to $A'z' = 0 \pmod{q}$. If indeed the jth entry in z' is c, then $Az = b \pmod{q}$, where $z \in [-B, B]^m$ is obtained from z' by deleting its jth entry. Thus, z is an ISIS solution that we have efficiently found with non-negligible probability at least $1/(2B(m+1))$.

(SIS $\leq$ ISIS) Let A be an SIS(n, m, q, B) instance. Write $A = [A' \mid -b']$ where $A' \in \mathbb{Z}_q^{n\times(m-1)}$ and $b' \in \mathbb{Z}_q^n$. Use an ISIS solver to find an ISIS solution z' to the ISIS instance (A', b'). We then have $A'z' = b' \pmod{q}$ and $z' \in [-B, B]^{m-1}$. Then $z = (z', 1) \in \mathbb{Z}^m$ satisfies $Az = 0 \pmod{q}$, $z \neq 0$, and $z \in [-B, B]^m$. Thus, z is an SIS solution that we have efficiently found. □

Definition 3.7 The *normal-form ISIS* problem nf-ISIS(n, m, q, B) is the following. Given $A \in_R \mathbb{Z}_q^{n\times m}$ and $b \in_R \mathbb{Z}_q^n$, find $z \in \mathbb{Z}^{m+n}$ such that $[A \mid I_n]z = b \pmod{q}$ and $z \in [-B, B]^{m+n}$. Here, I_n is the $n \times n$ identity matrix and $B \ll q/2$.

Exercise 3.3 shows that nf-ISIS(n, m, q, B) and ISIS$(n, m+n, q, B)$ are computationally equivalent.

3.2 The SIS lattice

We next introduce the lattice associated with an SIS instance.

Definition 3.8 Given an instance A of SIS(n, m, q, B), the associated *SIS lattice* is defined by

$$L_A^{\perp} = \{z \in \mathbb{Z}^m : Az = 0 \pmod{q}\}.$$

The proof of Theorem 3.10 relies on the following standard result on free (additive) abelian groups.

Theorem 3.9 Let G be a free abelian group of rank m with $\mathbb{Z}$-basis $x_1, \ldots, x_m$, and let H be a rank-m subgroup of G with $\mathbb{Z}$-basis $y_1, \ldots, y_m$. For each $1 \le i \le m$, let $y_i = \sum_j a_{ij} x_j$ where $a_{ij} \in \mathbb{Z}$. Then the quotient group G/H is finite and $|G/H| = |\det(a_{ij})|$.

Theorem 3.10 The SIS lattice $L_A^\perp \subseteq \mathbb{R}^m$ is a full-rank integer lattice with volume q^n.

Proof: It is easy to check that $L_A^\perp$ is closed under addition and negation, and hence forms an additive subgroup of $\mathbb{Z}^m$. Since all of its elements have integral coordinates, the set is discrete, and thus $L_A^\perp$ is an integer lattice.

To see that $L_A^\perp$ has full rank, observe that the lattice $q\mathbb{Z}^m$, which consists of all integer vectors whose coordinates are multiples of q, is generated by the m linearly independent vectors $(q, 0, \ldots, 0), (0, q, \ldots, 0), \ldots, (0, 0, \ldots, q)$. Hence, $q\mathbb{Z}^m$ is a full-rank lattice. Moreover, $q\mathbb{Z}^m$ is a sublattice of $L_A^\perp$, which implies that $L_A^\perp$ itself is full rank.

Lastly, note that both $\mathbb{Z}^m$ and $L_A^\perp$ are free abelian groups of rank m. Since $L_A^\perp$ is a subgroup of $\mathbb{Z}^m$, Theorem 3.9 asserts that $\mathrm{vol}(L_A^\perp) = |\mathbb{Z}^m / L_A^\perp|$. To compute this quantity, we need to determine the number of cosets of $L_A^\perp$ in $\mathbb{Z}^m$. Now, let $x, y \in \mathbb{Z}^m$. Then x and y are in the same coset of $L_A^\perp$, meaning that $L_A^\perp + x = L_A^\perp + y$, if and only if $x - y \in L_A^\perp$, i.e., $A(x-y) = 0 \pmod q$. The latter condition is equivalent to $Ax = Ay \pmod q$. Thus, the number of cosets of $L_A^\perp$ in $\mathbb{Z}^m$ equals the size of the image of $A \pmod q$. Since A has rank n over $\mathbb{Z}_q$, the image has size q^n, and hence $\mathrm{vol}(L_A^\perp) = |\mathbb{Z}^m / L_A^\perp| = q^n$. □

The lattice $L_A^\perp$ is an example of a *q-ary lattice*: membership is determined entirely by reducing the vector modulo q. We now describe an explicit basis of this lattice.

Theorem 3.11 Suppose that the first n columns of A are linearly independent over $\mathbb{Z}_q$. Then A can be row-reduced over $\mathbb{Z}_q$ to the form $\widetilde{A} = [I_n \mid \overline{A}]$, where $\overline{A} \in \mathbb{Z}_q^{n \times (m-n)}$. In this case, the matrix

$$C = \begin{bmatrix} qI_n & -\overline{A} \\ 0 & I_{m-n} \end{bmatrix} \in \mathbb{Z}^{m \times m} \tag{10}$$

is a basis matrix for the SIS lattice $L_A^\perp$.

Proof: Since A and $\widetilde{A}$ are row equivalent over $\mathbb{Z}_q$, they have the same null space modulo q and hence define the same SIS lattice: $L_{\widetilde{A}}^\perp = L_A^\perp$. It therefore suffices to find a basis of $L_{\widetilde{A}}^\perp$.

Each column v of C satisfies $\widetilde{A}v = 0 \pmod q$, and thus lies in $L_{\widetilde{A}}^\perp$. The columns of C are linearly independent over $\mathbb{R}$, since $\det(C) = q^n \neq 0$. Consequently, C is a basis matrix for a full-rank sublattice L of $L_{\widetilde{A}}^\perp$. Finally, the determinant of C shows that $\mathrm{vol}(L) = q^n = \mathrm{vol}(L_{\widetilde{A}}^\perp)$. This implies that $L_{\widetilde{A}}^\perp = L$, and therefore C is a basis matrix for $L_A^\perp$. □

3.3 Hardness of SIS

We are now ready to reformulate the SIS problem purely as a lattice problem: namely, the task of finding a short nonzero vector in a specific lattice.

Definition 3.12 The *Short Integer Solution* problem $\mathrm{SIS}(n, m, q, B)$ is defined as follows. Given a uniformly random matrix $A \in_R \mathbb{Z}_q^{n\times m}$, find a vector $z \in \mathbb{Z}^m$ with $0 < \|z\|_\infty \le B$ that lies in the SIS lattice $L_A^\perp = L(C)$, where C is the basis matrix defined in (10).

SIS hardness is usually studied using the Euclidean norm rather than the infinity norm. The two formulations are closely related since for every $z \in \mathbb{R}^m$, we have $\|z\|_\infty \le \|z\|_2 \le \sqrt{m}\,\|z\|_\infty$. This motivates the following closely related variant of SIS.

Definition 3.13 The $\mathrm{SIS}_2(n, m, q, B)$ problem is defined as follows. Given $A \in_R \mathbb{Z}_q^{n\times m}$, find a vector $z \in \mathbb{Z}^m$ in $L_A^\perp$ such that $0 < \|z\|_2 \le B$.

3.3.1 Dual attack on SIS_2

Recall that the SIS lattice $L_A^\perp$ has rank m and volume q^n. By Minkowski's Theorem (Theorem 2.15), the length of a shortest nonzero vector in $L_A^\perp$ satisfies

$$\lambda_1(L_A^\perp) \le \sqrt{m}\, q^{n/m}.$$

We henceforth assume that $B \ge \sqrt{m}\, q^{n/m}$, which guarantees the existence of a solution to SIS_2. According to the Gaussian heuristic (equation (8)), one expects that

$$\lambda_1(L_A^\perp) \approx \sqrt{m/(2\pi e)}\, q^{n/m}.$$

Let γ denote the ratio between the SIS_2 solution bound B and the shortest length predicted by the Gaussian heuristic, namely

$$\gamma = B\sqrt{2\pi e}/(\sqrt{m}\, q^{n/m}). \tag{11}$$

With this notation, the SIS_2 problem can be viewed as an instance of the approximate shortest vector problem SVP_γ in the lattice $L_A^\perp$ with approximation factor γ. This strategy of reducing SIS_2 to an instance of SVP_γ is called a *dual attack*.[5] An example is presented in §12.2.

3.3.2 Average-case hardness of SIS

It is natural to conjecture that SIS is hard in the *worst case*. However, in cryptographic applications an SIS instance is generated by choosing a matrix A uniformly at random. Therefore, what is required is not merely worst-case hardness, but rather that every efficient algorithm succeeds only with negligible probability over the choice of A. In other words, we need SIS to be hard *on average*.

Another concern with SIS is that the corresponding approx-SVP instance has considerable structure. In particular, the SIS lattice admits a basis matrix of the special form given by (10), which is upper-triangular with q's in the upper-left diagonal block and

[5] The term "dual" mirrors the duality between the linear code over $\mathbb{Z}_q$ generated by the rows of A and its dual code, which has A as a parity-check matrix and whose integer lift is $L_A^\perp$; cf. Theorem 4.18.

1's in the lower-right diagonal block. One might therefore be concerned that, even if approx-SVP is hard in general, it could be significantly easier on the restricted family of lattices arising from SIS instances.

In his seminal 1996 work, Ajtai alleviated these concerns by proving that SIS is hard *on average* under standard complexity-theoretic assumptions. Specifically, he showed that if approx-SIVP is hard in the worst case, then SIS is hard on average. Reductions of this type are known as *worst-case to average-case reductions*, and they provide (conditional) assurances for the average-case hardness of SIS since approx-SIVP is a natural and well-studied lattice problem, at least for relatively small approximation factors.

Ajtai's reduction is technically involved and beyond the scope of this introductory treatment. At a very high level (see Figure 3.2), the reduction starts from an *arbitrary*

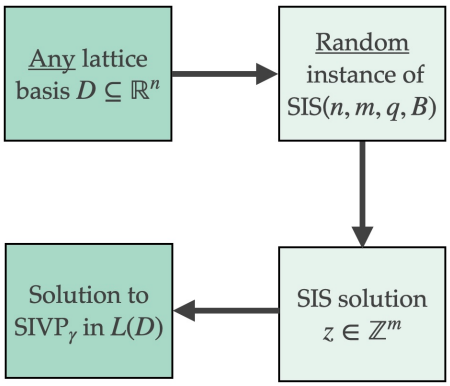


Figure 3.2: Proving the average-case hardness of SIS.

n-dimensional lattice with basis D and efficiently generates a *random* SIS instance. A solution to this SIS instance can then be efficiently transformed into a solution to approx-SIVP in the lattice generated by D. Consequently, an efficient algorithm for solving SIS on average would imply an efficient algorithm for solving approx-SIVP in the worst case.

Ajtai's worst-case to average-case reduction for SIS is one of the most remarkable results in lattice-based cryptography. It should be noted, however, that the guarantee it provides for the average-case hardness of SIS is asymptotic in nature. Furthermore, the reduction is highly non-tight, which means that the algorithm that solves approx-SIVP using an SIS solver is very inefficient. The asymptotic nature and non-tightness of the reduction prevents it from being used directly to derive concrete parameters for practical cryptographic schemes based on SIS.

3.4 Notes and further references

The SIS problem was introduced by Ajtai [3], who provided a worst-case to average-case reduction from approx-SIVP to SIS. The approximation factor in Ajtai's reduction was later improved by Micciancio and Regev [163]. Gentry, Peikert and Vaikuntanathan [98]

subsequently gave a simpler and slightly tighter reduction that works with a smaller modulus q.

Gentry, Peikert and Vaikuntanathan (GPV) [98, 99] introduced a framework for constructing hash-and-sign signature schemes. This framework was later employed in the design of the Falcon signature scheme by Fouque et al. [89]. Falcon (Fast-Fourier Lattice-Based Compact Signatures over NTRU) derives its security from the hardness of SIS on NTRU lattices. Falcon is currently undergoing standardization by NIST, with a draft standard, FIPS 206, anticipated in 2027. The GPV framework and Falcon are introduced in §8.

Cash, Hofheinz, Kiltz and Peikert [48] introduced the concept of lattice-based *bonsai trees*, using it to construct a stateless SIS-based signature scheme whose security proof does not depend on the random oracle model.

A proof of Theorem 3.9 can be found in §1.6 of Stewart and Tall's book [210].

3.5 Exercises

Exercise 3.1 (†) Write a program to find all solutions z to the following SIS$(4, 6, 23, 5)$ instance:

$$A = \begin{bmatrix} 16 & 12 & 19 & 0 & 7 & 3 \\ 12 & 6 & 5 & 1 & 20 & 11 \\ 14 & 9 & 14 & 15 & 1 & 10 \\ 1 & 0 & 16 & 19 & 17 & 2 \end{bmatrix}.$$

Exercise 3.2 (†) Write a program to find all solutions z to the following ISIS$(3, 5, 29, 5)$ instance:

$$A = \begin{bmatrix} 2 & 9 & 6 & 27 & 15 \\ 22 & 9 & 22 & 19 & 24 \\ 0 & 2 & 16 & 19 & 11 \end{bmatrix}, \quad b = \begin{bmatrix} 13 \\ 25 \\ 6 \end{bmatrix}.$$

Exercise 3.3 Prove that nf-ISIS(n, m, q, B) and ISIS$(n, m + n, q, B)$ are computationally equivalent.

4 Learning With Errors (LWE) problem

Contents

4.1 Problem statement 28
4.2 Lindner-Peikert public-key encryption 32
4.3 The LWE lattice 36
4.4 Hardness of LWE 37
4.5 Notes and further references 41
4.6 Exercises 42

The security of a wide range of cryptographic protocols, including the Kyber key encapsulation mechanism and the Dilithium signature scheme, relies on the hardness of the Learning With Errors (LWE) problem. This section introduces LWE, presents a lattice formulation of LWE, and explores its relationships with SVP and SIVP.

4.1 Problem statement

The LWE problem was introduced by Oded Regev in 2005. It is parameterized by four values: positive integers m and n with $m \gg n$, a prime modulus q, and a noise bound B satisfying $B \ll q/2$.

Definition 4.1 The *Learning With Errors* problem, denoted LWE(m, n, q, B), is defined as follows. Choose a matrix $A \in_R \mathbb{Z}_q^{m \times n}$, a secret vector $s \in_R \mathbb{Z}_q^n$, and an error vector $e \in_R [-B, B]^m$, and let $b = As + e \pmod q \in \mathbb{Z}_q^m$. Given the pair (A, b), the goal is to recover s; see Figure 4.1. The parameter m is the number of *LWE samples*.

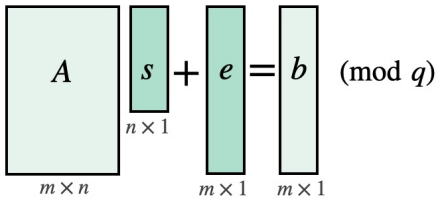


Figure 4.1: LWE problem.

Observe that if $B = 0$, so that $e = 0$, then the problem is to solve the system of linear equations $As = b \pmod q$ for s. In this case, the set of all solutions can be efficiently determined using Gaussian elimination over $\mathbb{Z}_q$. Hardness of LWE arises because of the introduction of the noise e. We henceforth assume that $B \neq 0$.

Remark 4.2 (*on the primality of q*) The requirement that q be prime in Definition 4.1 is a matter of convenience rather than necessity. Its main effect is that $\mathbb{Z}_q$ is a field, so that $\mathbb{Z}_q^n$ is a vector space and the usual tools of linear algebra are available. For example,

a uniformly random matrix $A \in \mathbb{Z}_q^{m \times n}$ with $m \gg n$ has rank n except with negligible probability. As another example, the proof that LWE $\leq$ DLWE (Theorem 4.6) relies on the primality of q. Similarly, a prime q is preferred in the MLWE setting of Kyber and Dilithium (see §5), where the known security reductions are established for prime moduli. The requirement is relaxed in §9, where the FrodoKEM scheme takes q to be a power of two (2^{15} or 2^{16}), which makes modular reduction particularly efficient.

Remark 4.3 (*distribution choices in LWE*) In Definition 4.1, the secret vector s is sampled uniformly over $\mathbb{Z}_q^n$ and the error vector e uniformly from the range $[-B, B]^m$. More generally, s and e can be sampled from any distribution. For instance, discrete Gaussian distributions are used in numerous cryptographic protocols including the Falcon signature scheme (§8) and FrodoKEM (§9), and are central to establishing worst-case to average-case security proofs. Alternatively, the centered binomial distribution is employed in Kyber; see §6.2. Furthermore, each coordinate of s and each coordinate of e can be drawn from the same distribution, as is done in the short-secret variant of LWE (Definition 4.7).

The statement of the LWE problem implicitly assumes that the LWE solution s is unique. In fact, the solution is expected to be unique with overwhelming probability provided that the number m of rows of A is sufficiently greater than the number n of columns of A.

To support this claim, observe that multiplying A by a vector $s \in \mathbb{Z}_q^n$ results in a vector $t = As \pmod q$ in $\mathbb{Z}_q^m$, a vector space with q^m elements. For each $s \in \mathbb{Z}_q^n$, the sphere in $\mathbb{Z}_q^m$ centered at $t = As \pmod q$ is defined as $T_s = \{t + e \pmod q \ : \ e \in [-B, B]^m\}$; note that the number of vectors in each sphere is $|T_s| = (2B + 1)^m$. Uniqueness of the LWE solution is guaranteed if and only if no two of the q^n spheres overlap (see Figure 4.2).

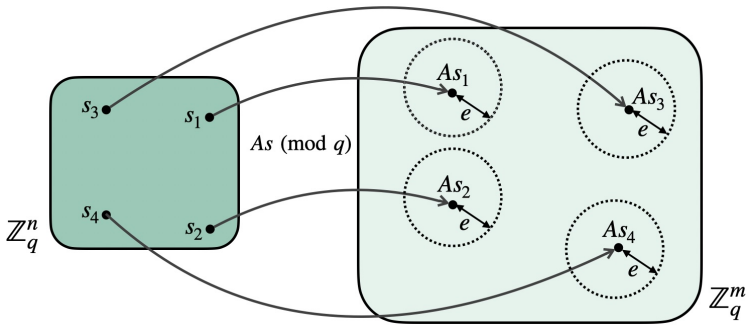


Figure 4.2: Uniqueness of the LWE solution.

Now[6], the two spheres centered at As_1 and As_2 overlap if and only if there exist $e_1, e_2 \in [-B, B]^m$ such that $As_1 + e_1 = As_2 + e_2 \pmod q$. Letting $\Delta s = s_1 - s_2$ and $\Delta e = e_2 - e_1$, we see that the LWE solution is not unique if and only if there exist

[6] Thang Cao Trinh showed me this argument.

$\Delta s \in \mathbb{Z}_q^n \setminus \{0\}$ and $\Delta e \in [-2B, 2B]^m$ such that[7] $A\Delta s = \Delta e$. We make the heuristic assumption that $A\Delta s$ is a uniformly random vector in $\mathbb{Z}_q^m$ for each nonzero Δs. Then, for each nonzero $\Delta s \in \mathbb{Z}_q^n$, the probability that $A\Delta s$ is in $[-2B, 2B]^m$ is

$$\left(\frac{4B+1}{q}\right)^m.$$

By the union bound, the probability that $A\Delta s$ is in $[-2B, 2B]^m$ for at least one nonzero $\Delta s \in \mathbb{Z}_q^n$ is at most

$$(q^n - 1)\left(\frac{4B+1}{q}\right)^m.$$

For $B \ll q/2$ and $n \ll m$, this probability is negligible, so the LWE solution is unique with overwhelming probability. In the remainder of these notes, we will assume that the LWE solution is unique.

Example 4.4 (*LWE instance*) Let $m = 5$, $n = 3$, $q = 47$, and $B = 2$. Consider the LWE(m, n, q, B) instance

$$A = \begin{bmatrix} 27 & 13 & 13 \\ 1 & 46 & 23 \\ 16 & 13 & 30 \\ 18 & 16 & 19 \\ 22 & 0 & 3 \end{bmatrix} \in \mathbb{Z}_{47}^{5\times 3}, \qquad b = \begin{bmatrix} 30 \\ 28 \\ 25 \\ 34 \\ 32 \end{bmatrix} \in \mathbb{Z}_{47}^5.$$

The LWE challenge is to find $s \in \mathbb{Z}_{47}^3$ and $e \in [-2, 2]^5$ such that $As + e = b \pmod{47}$. It turns out that this LWE instance has three solutions: $s = (2, 15, 12)$, $e = (1, 0, 2, 0, -1)$; $s = (19, 34, 12)$, $e = (0, 2, 0, 1, 1)$; and $s = (41, 33, 7)$, $e = (1, 0, -1, -2, 2)$.

Decisional LWE problem. In the decisional variant of LWE, the challenge is to distinguish an LWE instance (A, b) from a random instance (A, r) where $r \in \mathbb{Z}_q^m$ is randomly selected. In the latter case, one expects that $As + e = r \pmod{q}$ does not have an LWE solution. Essentially, the decisional problem is to decide whether a solution exists.

Definition 4.5 The *Decisional Learning With Errors* problem DLWE(m, n, q, B) is the following. Let $A \in_R \mathbb{Z}_q^{m\times n}$, $s \in_R \mathbb{Z}_q^n$, $e \in_R [-B, B]^m$, $r \in_R \mathbb{Z}_q^m$, and let $b = As + e \pmod{q}$. With equal probability $\frac{1}{2}$, set $c = b$ or $c = r$. Given a problem instance (A, c), decide (with success probability significantly greater than $\frac{1}{2}$) whether $c = b$ or $c = r$.

Theorem 4.6 LWE and DLWE are computationally equivalent.

Proof: We will first prove that LWE reduces to DLWE (denoted LWE $\leq$ DLWE) and then prove that DLWE reduces to LWE (denoted DLWE $\leq$ LWE).

[7] Henceforth, the $\pmod{q}$ operator will often be omitted when clear from context.

(LWE $\leq$ DLWE) Let (A, b) be an LWE instance (where $b = As + e$), and assume that we have a DLWE-solver.[8] We will use the solver to test guesses for the coordinates of s, one at a time, beginning with the first coordinate s_1. This procedure may need to be repeated up to qn times before s is determined.

Let $d \in \mathbb{Z}_q$. To test whether $s_1 = d$, select $\Delta \in_R \mathbb{Z}_q^m$. Let A' be the matrix obtained by adding Δ to the first column of A, and let $b' = b + d\Delta$.

Now, if $s_1 = d$, then $b' = A's + e$ so (A', b') is a valid LWE instance. On the other hand, if $s_1 \neq d$, then $b' = A's + e + (d - s_1)\Delta$. Since $d - s_1$ is nonzero, and Δ is uniformly random and independent of A', s and e, it follows that b' is uniformly random and independent of A'. Thus, (A', b') is a valid DLWE instance, whence the DLWE solver with input (A', b') will inform us whether or not $s_1 = d$. We eventually find s_1 and the other s_i's similarly.

(DLWE $\leq$ LWE) Let (A, c) be a DLWE instance, and assume that we have an LWE-solver. We will use the LWE-solver to efficiently solve the DLWE instance.

Now, if $c = b$, then (A, c) is an LWE instance and so one expects that $As + e = c \pmod q$ has a unique LWE solution (s, e). If $c = r$, as noted earlier, one expects that $As + e = c \pmod q$ has no LWE solution. So, the LWE solver is run with input (A, c). If a valid LWE solution is returned, then we conclude that $c = b$. If the LWE solver terminates without a valid LWE solution, or fails to terminate, then we conclude that $c = r$. □

Short-secret LWE problem. In the short-secret variant of the LWE problem, the coordinates of the secret vector s are sampled from the same distribution as those of the error vector e.

Definition 4.7 The *Short-secret Learning With Errors* problem, denoted ss-LWE(m, n, q, B), is defined as follows. Choose a matrix $A \in_R \mathbb{Z}_q^{m \times n}$, a secret vector $s \in_R [-B, B]^n$, and an error vector $e \in_R [-B, B]^m$, and define $b = As + e \pmod q \in \mathbb{Z}_q^m$. Given the pair (A, b), the goal is to recover the secret vector s.

Unlike the situation for LWE, the condition $n \ll m$ is not needed to ensure that ss-LWE has a unique solution with high probability (Exercise 4.3).

Example 4.8 (*ss-LWE instance*) Let $m = 5$, $n = 5$, $q = 97$, and $B = 4$. Consider the ss-LWE(m, n, q, B) instance

$$A = \begin{bmatrix} 15 & 30 & 51 & 23 & 77 \\ 6 & 46 & 54 & 84 & 23 \\ 32 & 12 & 80 & 78 & 34 \\ 32 & 4 & 88 & 18 & 4 \\ 52 & 49 & 67 & 31 & 26 \end{bmatrix} \in \mathbb{Z}_{97}^{5 \times 5}, \qquad b = \begin{bmatrix} 34 \\ 40 \\ 90 \\ 85 \\ 29 \end{bmatrix} \in \mathbb{Z}_{97}^5.$$

[8] We assume that the DLWE solver returns correct answers with overwhelming probability.

The ss-LWE challenge is to find s and e in $[-4,4]^5$ such that $As + e = b \pmod{97}$. It turns out that this ss-LWE instance has a unique solution $s = (1,-1,0,3,1)$, $e = (0,-1,-4,-1,4)$.

Theorem 4.9 The LWE and ss-LWE problems are computationally equivalent. More precisely, ss-LWE$(m,n,q,B) \leq$ LWE(m,n,q,B) and LWE$(m,n,q,B) \leq$ ss-LWE$(m-n,n,q,B)$.

Proof: First consider an instance (A,b) of ss-LWE(m,n,q,B), where $b = As+e \pmod{q}$ with $s \in_R [-B,B]^n$ and $e \in_R [-B,B]^m$. Select a random vector $d \in_R \mathbb{Z}_q^n$ and define

$$b' = b + Ad = (As+e) + Ad = A(s+d) + e \pmod{q}.$$

Then (A,b') forms an LWE(m,n,q,B) instance. If (s',e) is a solution to this LWE instance, then $(s'-d,e)$ is immediately a solution to the original ss-LWE instance. This establishes that ss-LWE$(m,n,q,B) \leq$ LWE(m,n,q,B).

Now suppose (A,b) is an LWE(m,n,q,B) instance, so that $b = As + e \pmod{q}$. Partition the matrix A and vectors b and e as

$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \quad e = \begin{bmatrix} e_1 \\ e_2 \end{bmatrix},$$

where $A_1 \in \mathbb{Z}_q^{n\times n}$, $A_2 \in \mathbb{Z}_q^{(m-n)\times n}$, $b_1 \in \mathbb{Z}_q^n$, $e_1 \in [-B,B]^n$, $b_2 \in \mathbb{Z}_q^{m-n}$, and $e_2 \in [-B,B]^{m-n}$. Assuming A_1 is invertible modulo q, define

$$A' = -A_2A_1^{-1} \in \mathbb{Z}_q^{(m-n)\times n} \text{ and } b' = A'b_1 + b_2 \in \mathbb{Z}_q^{m-n}.$$

Then

$$\begin{aligned} b' &= A'b_1 + b_2 = (-A_2A_1^{-1})(A_1s + e_1) + (A_2s + e_2) \\ &= -A_2s - A_2A_1^{-1}e_1 + A_2s + e_2 = A'e_1 + e_2. \end{aligned}$$

Therefore, (A',b') forms an ss-LWE$(m-n,n,q,B)$ instance. Solving this ss-LWE instance yields e_1, from which the original secret s can be immediately recovered, thereby solving the original LWE instance. Hence, LWE$(m,n,q,B) \leq$ ss-LWE$(m-n,n,q,B)$. □

4.2 Lindner-Peikert public-key encryption

We now give an example of a public-key encryption scheme whose security relies on the hardness of the decisional version of ss-LWE, which is computationally equivalent to the ss-LWE problem (Exercise 4.4).

Definition 4.10 The *short-secret Decisional Learning With Errors* problem, denoted ss-DLWE(m,n,q,B), is defined as follows. Choose a matrix $A \in_R \mathbb{Z}_q^{m\times n}$, a secret vector $s \in_R [-B,B]^n$, an error vector $e \in_R [-B,B]^m$, and a random vector $r \in_R \mathbb{Z}_q^m$. Define $b = As + e \pmod{q} \in \mathbb{Z}_q^m$. Let $c = b$ with probability $\frac{1}{2}$ and $c = r$ with probability $\frac{1}{2}$. Given the pair (A,c), the goal is to determine whether $c = b$ or $c = r$ with success probability significantly greater than $\frac{1}{2}$.

Decryption relies on a rounding procedure, which is introduced next.

Definition 4.11 Let $q \geq 3$ be an odd integer, and let $r \in \mathbb{Z}_q$. Define the *symmetric mod q* of r to be

$$r \bmod^s q = \begin{cases} r, & \text{if } r \leq (q-1)/2 \\ r - q, & \text{if } r > (q-1)/2. \end{cases}$$

The mods operation extends naturally to all integers: if $z \in \mathbb{Z}$ then

$$z \bmod^s q = (z \bmod q) \bmod^s q.$$

Note that $-(q-1)/2 \leq r \bmod^s q \leq (q-1)/2$. For example, if $q = 17$ and $r \in \mathbb{Z}_{17}$ then $-8 \leq r \bmod^s 17 \leq 8$. We have $6 \bmod^s 17 = 6$ and $13 \bmod^s 17 = -4$.

Definition 4.12 Define the rounding function $\text{Round}_q : \mathbb{Z}_q \to \{0, 1\}$ as follows:

$$\text{Round}_q(x) = \begin{cases} 0, & \text{if } -q/4 < x \bmod^s q < q/4, \\ 1, & \text{otherwise.} \end{cases}$$

The rounding function is depicted in Figure 4.3. For example, if $q = 1223$ then $\text{Round}_q(x) = 0$ if $0 \leq x \leq 305$ or $918 \leq x \leq 1222$, and $\text{Round}_q(x) = 1$ if $306 \leq x \leq 917$.

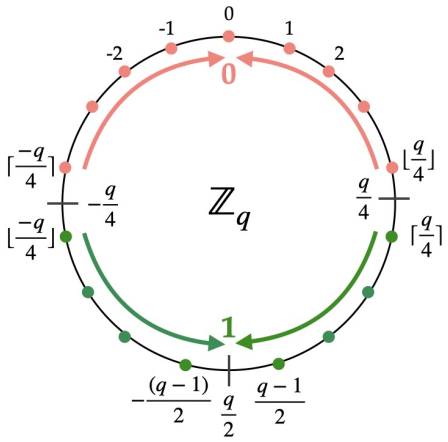


Figure 4.3: The rounding function.

In the Lindner-Peikert encryption scheme (Algorithm 4.13), we assume that the parameters q, n, B satisfy

$$B \leq \sqrt{\frac{q}{4(2n+1)}} \quad \text{and} \quad q > 6(2n+1). \tag{12}$$

These conditions guarantee the correctness of the decryption procedure. Also, $\lceil x \rfloor$ denotes the closest integer to x, with ties broken upwards.

Correctness of decryption. Observe that

$$\begin{aligned} v - s^T u &= (b^T r + e_2 + m\lceil q/2 \rfloor) - s^T (A^T r + e_1) \\ &= (s^T A^T + e^T) r + e_2 + m\lceil q/2 \rfloor - s^T (A^T r + e_1) \\ &= e^T r - s^T e_1 + e_2 + m\lceil q/2 \rfloor . \end{aligned}$$

Hence decryption succeeds, meaning $m = \mathrm{Round}_q(v - s^T u)$, if

$$|e^T r - s^T e_1 + e_2 \bmod q| < \lceil q/4 \rfloor .$$

Let $x = q/(4(2n+1))$. Using (12), we have

$$\begin{aligned} |e^T r - s^T e_1 + e_2| &\le nB^2 + nB^2 + B \\ &\le 2nx + \sqrt{x} \\ &= q/4 - \sqrt{x}(\sqrt{x} - 1). \end{aligned}$$

Now, $q > 6(2n+1)$ implies that $x > 3/2$. Since $x - \sqrt{x}$ is increasing for $x > 1/4$, we have $\sqrt{x}(\sqrt{x} - 1) > 1/4$ for $x > 3/2$. It follows that

$$|e^T r - s^T e_1 + e_2| < q/4 - 1/4 \le \lceil q/4 \rfloor$$

since q is odd, and thus the rounding step correctly recovers m.

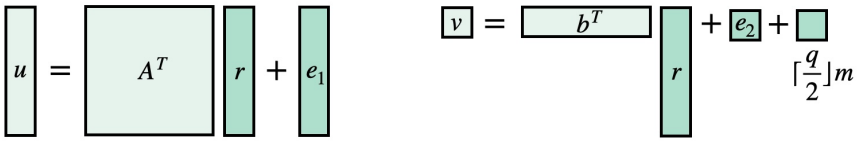


Figure 4.4: Lindner-Peikert encryption.

Example 4.14 (*Lindner-Peikert encryption*)
Domain parameters. $q = 997$, $n = 4$, and $B = 5$.
Key generation. Bob selects

$$A = \begin{bmatrix} 301 & 352 & 400 & 269 \\ 585 & 916 & 438 & 560 \\ 510 & 108 & 14 & 157 \\ 265 & 620 & 886 & 348 \end{bmatrix}, \quad s = \begin{bmatrix} 2 \\ 3 \\ -5 \\ 1 \end{bmatrix}, \quad e = \begin{bmatrix} 4 \\ 5 \\ 0 \\ 1 \end{bmatrix},$$

and computes

$$b = As + e \pmod q = \begin{bmatrix} 928 \\ 299 \\ 434 \\ 303 \end{bmatrix}.$$

Algorithm 4.13: Lindner-Peikert public-key encryption scheme

Domain parameters. q, n, B with $B \leq \sqrt{q/(4(2n+1))}$ and $q > 6(2n+1)$.

Key generation. Bob does the following:

1 Select $A \in_R \mathbb{Z}_q^{n \times n}$, $s \in_R [-B, B]^n$, and $e \in_R [-B, B]^n$.
2 Compute $b = As + e \pmod{q}$.
3 Bob's encryption (public) key is (A, b); his decryption (private) key is s.

Encryption. To encrypt a message $m \in \{0, 1\}$ for Bob, Alice does the following:

1 Obtain an authentic copy of Bob's encryption key (A, b).
2 Select $r \in_R [-B, B]^n$, $e_1 \in_R [-B, B]^n$, and $e_2 \in_R [-B, B]$.
3 Compute $u = A^T r + e_1 \pmod{q}$ and $v = b^T r + e_2 + \lceil \frac{q}{2} \rfloor m \pmod{q}$ (see Figure 4.4).
4 Output $c = (u, v)$.

Decryption. To decrypt $c = (u, v)$, Bob does:

1 Compute $m = \text{Round}_q(v - s^T u)$.

Bob's encryption key is (A, b), and his decryption key is s.
Encryption. To encrypt the plaintext message $m = 0$ for Bob, Alice selects

$$r = \begin{bmatrix} -3 \\ -3 \\ 4 \\ -5 \end{bmatrix}, \quad e_1 = \begin{bmatrix} -1 \\ 2 \\ -3 \\ -5 \end{bmatrix}, \quad e_2 = -3,$$

and computes

$$u = A^T r + e_1 = \begin{bmatrix} 50 \\ 509 \\ 88 \\ 384 \end{bmatrix} \quad \text{and} \quad v = b^T r + e_2 + \lceil 997/2 \rfloor m = 525.$$

The ciphertext is $c = (u, v)$.
Decryption. To decrypt c, Bob uses his decryption key s to compute

$$v - s^T u = 948$$

and rounds it to recover the plaintext $m = 0$.

Security. Lindner-Peikert encryption is indistinguishable under chosen-plaintext attack (Definition A.2) assuming that ss-DLWE is hard. To justify this claim, observe that the encryption operation can be written as

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} A^T \\ b^T \end{bmatrix} r + \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \lceil \frac{q}{2} \rfloor m \end{bmatrix}.$$

Under the ss-DLWE(n, n, q, B) hardness assumption, the matrix

$$\begin{bmatrix} A^T \\ b^T \end{bmatrix}$$

is indistinguishable from a random matrix. Consequently, by the ss-DLWE$(n+1, n, q, B)$ hardness assumption,

$$\begin{bmatrix} A^T \\ b^T \end{bmatrix} r + \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} = \begin{bmatrix} A^T r + e_1 \\ b^T r + e_2 \end{bmatrix}$$

is also indistinguishable from a random vector. Thus, from the adversary's perspective, the ciphertext component v looks like the sum of a random element $b^T r + e_2 \in \mathbb{Z}_q$ and the plaintext $\lceil \frac{q}{2} \rfloor m$. Therefore, the adversary can gain no information about the plaintext m.

Improvements. The Lindner-Peikert encryption scheme has two main shortcomings. First, it supports the encryption of only a single bit. Second, it achieves security against chosen-plaintext attacks, but not against chosen-ciphertext attacks (Exercise 4.5). The first issue can be addressed by using the module-LWE variant (introduced in §5), where elements in $\mathbb{Z}_q$ are replaced by polynomials in the ring $R_q = \mathbb{Z}_q[x]/(x^n + 1)$. This modification, which closely resembles the Kyber public-key encryption scheme (§6.1), enables the encryption of n bits simultaneously. The second issue can be resolved by applying the Fujisaki-Okamoto transform, a generic technique that upgrades a public-key encryption scheme from chosen-plaintext security to chosen-ciphertext security. This transformation effectively yields the Kyber key encapsulation mechanism (§6.4).

4.3 The LWE lattice

As with SIS, LWE admits a natural interpretation in terms of lattices.

Definition 4.15 Given an LWE(m, n, q, B) instance (A, b), the associated *LWE lattice* is defined by

$$L_A = \{y \in \mathbb{Z}^m : y = Az \pmod q \text{ for some } z \in \mathbb{Z}^n\}.$$

Let A_1 denote the $n \times n$ submatrix consisting of the first n rows of A. With overwhelming probability over the choice of A, the matrix A_1 is invertible modulo q. Throughout the remainder of this section, we assume that this condition holds.

Theorem 4.16 The LWE lattice $L_A \subseteq \mathbb{R}^m$ is a full-rank q-ary integer lattice of volume q^{m-n}.

Proof: As in the proof of Theorem 3.10, it is straightforward to show that L_A is a discrete additive subgroup of $\mathbb{Z}^m$, and hence an integer lattice. Moreover, $q\mathbb{Z}^m$ is a sublattice of L_A, which immediately implies that L_A has full rank.

To determine the volume of L_A, write

$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix},$$

where $A_1 \in \mathbb{Z}_q^{n\times n}$ and $A_2 \in \mathbb{Z}_q^{(m-n)\times n}$. Define the matrix $D_2 = A_2 A_1^{-1} \pmod q \in \mathbb{Z}_q^{(m-n)\times n}$, and consider the matrix

$$D = \begin{bmatrix} I_n & 0 \\ D_2 & qI_{m-n} \end{bmatrix} \in \mathbb{Z}^{m\times m}.$$

We claim that the columns of D form a basis of L_A, from which it follows that $\mathrm{vol}(L_A) = |\det(D)| = q^{m-n}$.

Since $\det(D) = q^{m-n} \neq 0$, the columns of D are linearly independent over $\mathbb{R}$. Let $y \in \mathbb{Z}^m$, and decompose it as

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

where $y_1 \in \mathbb{Z}^n$ and $y_2 \in \mathbb{Z}^{m-n}$. By definition, $y \in L_A$ if and only if there exists a vector $z \in \mathbb{Z}^n$ such that $y = Az \pmod q$, which in turn is true if and only if $y_1 = A_1 z \pmod q$ and $y_2 = A_2 z \pmod q$. Thus, $y \in L_A$ if and only if $y_2 = A_2 A_1^{-1} y_1 \pmod q$, or equivalently, $y_2 = D_2 y_1 + qc$ for some $c \in \mathbb{Z}^{m-n}$. In this case, we can write

$$y = D \begin{bmatrix} y_1 \\ c \end{bmatrix},$$

showing that every vector in L_A is an integer linear combination of the columns of D. It follows that D is indeed a basis matrix for L_A. □

Let $A \in \mathbb{Z}_q^{m\times n}$. Recall that the SIS lattice (Definition 3.8) associated with A^T is

$$L_{A^T}^{\perp} = \{z \in \mathbb{Z}^m \ : \ A^T z \equiv 0 \pmod q\}.$$

The LWE lattice L_A and the SIS lattice $L_{A^T}^{\perp}$ are closely related. More precisely, L_A and $L_{A^T}^{\perp}$ are duals of scalings of each other.

Definition 4.17 Let L be a full-rank lattice in $\mathbb{R}^n$. The *dual* of L is

$$L^* = \{x \in \mathbb{R}^n \ : \ \langle x, v\rangle \in \mathbb{Z} \text{ for all } v \in L\}.$$

Let B be a basis of L. It can be shown that L^* is a full-rank lattice with basis matrix $B^* = (B^{-1})^T$. Moreover, $(L^*)^* = L$ (Exercise 4.7).

Theorem 4.18 Let $A \in \mathbb{Z}_q^{m\times n}$. Then

$$L_A^* = \tfrac{1}{q} L_{A^T}^{\perp} \ \text{ and } \ (L_{A^T}^{\perp})^* = \tfrac{1}{q} L_A.$$

The proof of Theorem 4.18 is left as an exercise (Exercise 4.8).

4.4 Hardness of LWE

This section describes three attacks on LWE and briefly discusses the average-case hardness of LWE.

4.4.1 Primal attack

Consider an LWE instance (A, b) with $b = As + e \pmod q$, and define $y = b - e \in L_A$. If we are able to determine y from (A, b), then the secret vector s could be efficiently recovered by solving the overdetermined system of linear equations $As = y \pmod q$.

Since the error vector satisfies $e \in [-B, B]^m$, we have

$$\|b - y\|_2 = \|e\|_2 \le \sqrt{m}\, B.$$

In other words, the lattice vector y is close to the target vector b. This observation naturally leads to a connection between LWE and the Bounded Distance Decoding (BDD) problem, illustrated in Figure 4.5, which seeks a lattice vector that is close to a given target vector.

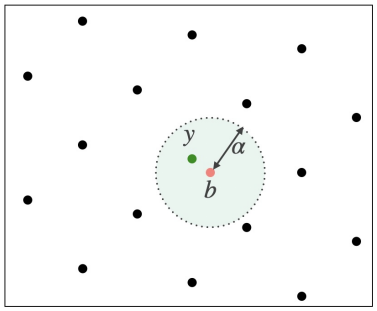


Figure 4.5: The BDD_α problem.

Definition 4.19 The *Bounded Distance Decoding Problem*, denoted BDD_α, is defined as follows: Given a lattice $L = L(D) \subseteq \mathbb{R}^m$ and a vector $b \in \mathbb{R}^m$ with the guarantee that there is a unique $y \in L$ within distance α of b, find y.

BDD_α can be seen as a promise version of CVP_γ (Definition 2.23). We next use *Kannan's embedding technique* (see also Exercise 12.6) to reduce BDD_α to unique SVP (uSVP) under the assumption that

$$\alpha < \lambda_1(L)/\sqrt{2}. \tag{13}$$

This strategy of reducing LWE to an instance of uSVP is known as the *primal attack* on LWE. An example of the primal attack on the short-secret variant of LWE is presented in §12.1.

Define

$$D' = \begin{bmatrix} D & -b \\ 0 & \alpha \end{bmatrix} \in \mathbb{R}^{(m+1)\times(m+1)} \text{ and } L' = L(D') = \left\{ \begin{bmatrix} v - cb \\ c\alpha \end{bmatrix} : v \in L(D) \text{ and } c \in \mathbb{Z} \right\}.$$

Notice that for $(v, c) = (y, 1)$ we obtain the lattice vector

$$\tilde{v} = \begin{bmatrix} y - b \\ \alpha \end{bmatrix} \in L' \text{ with } \|\tilde{v}\|_2 = \sqrt{\|y - b\|_2^2 + \alpha^2} \le \sqrt{2}\alpha.$$

Thus, $\lambda_1(L') \leq \sqrt{2}\alpha < \lambda_1(L)$ by (13).

Now, let

$$v' = \begin{bmatrix} v - cb \\ c\alpha \end{bmatrix} \in L'$$

be a shortest nonzero vector, i.e., $\|v'\|_2 = \lambda_1(L')$. We can rule out several possibilities for c. If $c = 0$, then

$$\|v'\|_2 = \|v\|_2 \geq \lambda_1(L) > \lambda_1(L'),$$

a contradiction. If $|c| \geq 2$, then

$$\|v'\|_2 \geq 2\alpha > \sqrt{2}\alpha \geq \lambda_1(L'),$$

also a contradiction. Hence, the only possibilities are $c = \pm 1$. Suppose that $c = 1$, so

$$v' = \begin{bmatrix} v - b \\ \alpha \end{bmatrix}$$

for some $v \in L$. (The case $c = -1$ is similar, except for signs.) If $v \neq y$, then $\|v - b\|_2 > \|y - b\|_2$ implying that

$$\|v'\|_2 = \sqrt{\|v - b\|_2^2 + \alpha^2} > \sqrt{\|y - b\|_2^2 + \alpha^2} = \|\tilde{v}\|_2;$$

this contradicts the assumption that v' is a nonzero vector of shortest length in L'. We conclude that $\pm\tilde{v}$ are the only vectors of length $\lambda_1(L')$ in L', and solving SVP in L' immediately yields the BDD solution y.

4.4.2 Dual attack

This attack reduces DLWE to SIS_2 (Definition 3.13), which is then solved using the dual attack described in §3.3.1. Let (A, c) be a DLWE(m, n, q, B) instance, where either $c = b$ or $c = r$ (see Definition 4.5). Select a positive integer β such that

$$\beta B\sqrt{m} \ll q/4.$$

Apply the SIS_2 dual attack to obtain a nonzero vector $x \in \mathbb{Z}^m$ such that $A^T x = 0 \pmod q$ and $\|x\|_2 \leq \beta$. If (A, c) is an LWE instance, so that $c = As + e \pmod q$ for some error vector $e \in [-B, B]^m$, then

$$c^T x = s^T A^T x + e^T x = e^T x \pmod q.$$

Since both $\|e\|_2 \leq B\sqrt{m}$ and $\|x\|_2 \leq \beta$, the Cauchy-Schwarz inequality[9] implies that

$$|e^T x| \leq \beta B\sqrt{m} \ll q/4.$$

[9]The Cauchy-Schwarz inequality states that $|u^T v| \leq \|u\| \cdot \|v\|$ for all $u, v \in \mathbb{R}^n$.

Hence,

$$|c^T x \bmod q| = |e^T x \bmod q| \ll q/4.$$

In contrast, if (A, c) is a random instance, so that c is uniformly random, then one expects that

$$|c^T x \bmod q| \approx q/4.$$

Consequently, the value $v = |c^T x \bmod q|$ can be used to distinguish LWE instances from random ones.

4.4.3 Arora-Ge attack

The Arora-Ge attack on LWE runs in polynomial time when the error bound B is fixed. However, it is inefficient because its running time function is a polynomial of very high degree in n. Furthermore, the attack is ineffective against LWE instances arising in Kyber and Dilithium since it requires a large number of LWE samples ($m \gg n^{2B+1}$), whereas in Kyber and Dilithium one has $m \approx n$.

We first treat the case $B = 1$. Suppose that $m \gg n^3$, and let (s, e) be the solution to an LWE$(m, n, q, 1)$ instance (A, b). For each $1 \le \ell \le m$, we have

$$a_\ell^T s - b_\ell = e_\ell,$$

where a_ℓ^T denotes the ℓ-th row of A, and e_ℓ and b_ℓ denote the ℓ-th components of e and b, respectively. Since $e_\ell \in \{-1, 0, 1\}$, it follows that

$$(a_\ell^T s - b_\ell - 1) \cdot (a_\ell^T s - b_\ell) \cdot (a_\ell^T s - b_\ell + 1) = 0. \tag{14}$$

Viewing the components $s_1, s_2, \ldots, s_n$ of s as variables over $\mathbb{Z}_q$, equation (14) can be expanded into the form

$$\sum_{1 \le i \le j \le k \le n} \alpha_{ijk} s_i s_j s_k + \sum_{1 \le i \le j \le n} \beta_{ij} s_i s_j + \sum_{1 \le i \le n} \gamma_i s_i = c_\ell, \tag{15}$$

where the coefficients α_{ijk}, β_{ij}, γ_i and c_ℓ are known elements of $\mathbb{Z}_q$. This nonlinear system of equations can be solved using the method of *linearization*.

For each $1 \le i \le j \le k \le n$, introduce new variables

$$x_{ijk} = s_i s_j s_k, \quad y_{ij} = s_i s_j, \quad z_i = s_i.$$

Equation (15) then becomes

$$\sum_{1 \le i \le j \le k \le n} \alpha_{ijk} x_{ijk} + \sum_{1 \le i \le j \le n} \beta_{ij} y_{ij} + \sum_{1 \le i \le n} \gamma_i z_i = c_\ell. \tag{16}$$

Notice that (16) is a linear equation in $N = \binom{n+2}{3} + \binom{n+1}{2} + n$ variables, and there is one such equation for each $1 \le \ell \le m$. Since $m \gg n^3 > N$ for all $n \ge 3$, we have an

overdetermined system of m linear equations over $\mathbb{Z}_q$ which can be solved efficiently. Since $z_i = s_i$, the solution immediately yields the secret s.

Because the linear algebra step takes cubic time, the running time of the Arora-Ge attack is

$$O(N^3) = O(n^9).$$

More generally, for arbitrary fixed B, the same linearization approach applies with monomials of degree up to $2B + 1$, yielding $O(n^{2B+1})$ variables. The running time is then

$$O((n^{2B+1})^3) = O(n^{6B+3}),$$

provided that $m \gg n^{2B+1}$.

4.4.4 Average-case hardness of LWE

As with SIS, it is reasonable to conjecture that LWE is hard in the *worst case*. However, LWE instances are generated by selecting A and s uniformly at random and e from the error distribution. For cryptographic applications, it is therefore crucial to establish *average-case hardness*, namely that every efficient algorithm succeeds only with negligible probability over the random choice of (A, s, e).

In his seminal 2005 work, Regev proved that LWE is hard *on average* against quantum attacks, assuming that approx-SIVP is quantum hard (i.e., intractable on quantum as well as classical computers) in the worst case. This worst-case to average-case reduction provides (conditional) evidence for the average-case hardness of LWE, since it is reasonable to assume that approx-SIVP is quantum hard in the worst case, at least for relatively small approximation factors. As with Ajtai's reduction for SIS (§3.3.2), Regev's result is asymptotic and non-tight, and thus does not directly yield concrete parameter choices for practical cryptosystems.

4.5 Notes and further references

The LWE problem was introduced by Regev [195], who gave a worst-case to average-case quantum reduction from approx-SIVP to LWE. For a concrete analysis of the tightness gap in Regev's reduction, see the work of Chatterjee, Koblitz, Menezes and Sarkar [51].

The Lindner-Peikert public-key encryption scheme [149] forms the basis for the Kyber public-key encryption scheme specified in FIPS 203, and also for the Frodo public-key encryption scheme presented in §9. LWE forms the basis for a wide variety of cryptographic protocols, including the identity-based encryption scheme of Gentry, Peikert and Vaikuntanathan [98, 99]; the hierarchical identity-based encryption scheme of Cash, Hofheinz, Kiltz and Peikert [48]; the attribute-based encryption scheme of Gorbunov, Vaikuntanathan and Wee [105]; and the fully-homomorphic encryption schemes of Gentry, Sahai and Waters [100], of Cheon, Kim, Kim and Song [56], and of Chillotti, Gama, Georgieva and Izabachène [57].

The embedding technique used in the primal attack on LWE described in §4.4.1 was originally introduced by Kannan [126] to reduce approx-CVP to SVP. Albrecht [8]

proposed a variant of the dual attack tailored to short-secret LWE. Howgrave-Graham [121] introduced *hybrid attacks*, which combine lattice basis reduction with meet-in-the-middle techniques.

The method of transforming an LWE instance into a system of high-degree multivariate polynomial equations that can then be solved via linearization was introduced by Arora and Ge [17]. Another known attack on LWE is the *BKW attack* introduced by Blum, Kalai and Wasserman [33]. This attack has been refined by many authors, including Kirchner and Fouque [130], Guo, Johansson, Mårtensson and Wagner [109], and Budroni, Guo, Johansson, Mårtensson and Wagner [45]. However, none of these attacks are effective against Kyber and Dilithium, as they require an impractically large number of LWE samples. Machine learning-based approaches have also been investigated, most recently by Alfarano, Saxena, Wenger, Charton and Lauter [11], although these methods have so far achieved only limited success.

The *Learning With Rounding (LWR) problem* is a variant of LWE in which the random error term is replaced by deterministic rounding. Let integers $q \geq p \geq 2$ be fixed. Define the *rounding function* $\lfloor \cdot \rfloor_p : \mathbb{Z}_q \to \mathbb{Z}_p$ by $\lfloor x \rfloor_p = \lfloor (p/q) \cdot x \rfloor$. This function naturally extends to vectors in $\mathbb{Z}_q^m$ by applying it componentwise. The LWR problem is then defined as follows: given a uniformly random matrix $A \in_R \mathbb{Z}_q^{m \times n}$ and a vector $b = \lfloor As \rfloor_p$, where $s \in_R \mathbb{Z}_q^n$ is chosen uniformly at random, determine s. The LWR problem was introduced by Banerjee, Peikert and Rosen [22], and a tighter reduction from LWE to LWR was later provided by Alwen, Krenn, Pietrzak and Wichs [14]. One advantage of LWR over LWE is that it requires fewer random bits, since the noise is deterministic (via rounding) rather than randomly sampled. Notable KEMs based on the module version of the LWR assumption include Saber, designed by D'Anvers, Karmakar, Roy and Vercauteren [67], and SMAUG-T [55], which is currently undergoing standardization in South Korea.

4.6 Exercises

Exercise 4.1 (†) Write a program to find all solutions (s, e) to the following LWE$(5, 3, 53, 2)$ instance:

$$A = \begin{bmatrix} 24 & 4 & 37 \\ 51 & 41 & 17 \\ 2 & 11 & 13 \\ 44 & 52 & 50 \\ 1 & 0 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} 5 \\ 3 \\ 12 \\ 10 \\ 6 \end{bmatrix}.$$

Exercise 4.2 (†) Find all solutions s to the following ss-LWE$(4, 4, 53, 3)$ instance:

$$A = \begin{bmatrix} 5 & 22 & 14 & 47 \\ 45 & 25 & 17 & 49 \\ 39 & 14 & 26 & 51 \\ 31 & 9 & 46 & 24 \end{bmatrix}, \quad b = \begin{bmatrix} 36 \\ 14 \\ 46 \\ 30 \end{bmatrix}.$$

Exercise 4.3 Suppose that $B \ll \sqrt{q}/4$. Give a heuristic justification for why the solution to ss-LWE(n, n, q, B) is expected to be unique.

Exercise 4.4 Prove that ss-LWE and ss-DLWE are computationally equivalent.

Exercise 4.5 Show that the Lindner-Peikert public-key encryption scheme is not indistinguishable under chosen-ciphertext attacks.

Exercise 4.6 Consider the Lindner-Peikert public-key encryption scheme with domain parameters $q = 997$, $n = 4$, $B = 5$ and keys

$$A = \begin{bmatrix} 956 & 579 & 385 & 946 \\ 325 & 648 & 268 & 820 \\ 329 & 837 & 955 & 923 \\ 224 & 287 & 608 & 837 \end{bmatrix}, \quad s = \begin{bmatrix} -2 \\ 5 \\ 5 \\ -2 \end{bmatrix}, \quad e = \begin{bmatrix} 4 \\ -2 \\ 1 \\ 0 \end{bmatrix}, \quad b = \begin{bmatrix} 23 \\ 294 \\ 475 \\ 359 \end{bmatrix}.$$

(a) Verify that the encryption of plaintext $m = 1$ using random quantities

$$r = \begin{bmatrix} 2 \\ -1 \\ 4 \\ -4 \end{bmatrix}, \quad e_1 = \begin{bmatrix} 5 \\ -4 \\ 4 \\ 0 \end{bmatrix}, \quad e_2 = -4,$$

yields the ciphertext $c = (u, v)$ where

$$u = \begin{bmatrix} 18 \\ 712 \\ 897 \\ 419 \end{bmatrix} \quad \text{and} \quad v = 711.$$

(b) Decrypt c.

Exercise 4.7 Let $L = L(B)$ be a full-rank lattice in $\mathbb{R}^n$. Recall that the *dual* of L is $L^* = \{x \in \mathbb{R}^n : \langle x, v \rangle \in \mathbb{Z} \text{ for all } v \in L\}$.
(a) Prove that $(c\mathbb{Z}^n)^* = \frac{1}{c}\mathbb{Z}^n$ for all $c \in \mathbb{R} \setminus \{0\}$.
(b) Prove that L^* is a lattice with basis matrix $(B^{-1})^T$.
(c) Prove that $(L^*)^* = L$.
(d) Prove that $\mathrm{vol}(L^*) = 1/\mathrm{vol}(L)$.
(e) Prove that $(qL)^* = \frac{1}{q}L^*$ for all $q \in \mathbb{R} \setminus \{0\}$.
(f) Prove that $L \subseteq L^*$ if and only if L is an *integral* lattice, i.e., $\langle x, y \rangle \in \mathbb{Z}$ for all $x, y \in L$.

Exercise 4.8 Let $A \in \mathbb{Z}_q^{m \times n}$. Prove that $L_A^* = \frac{1}{q}L_{A^T}^{\perp}$ and $(L_{A^T}^{\perp})^* = \frac{1}{q}L_A$.

Exercise 4.9 Let $L \subseteq \mathbb{R}^n$ be a full-rank lattice.
(a) Use Minkowski's Theorem (Theorem 2.15) to prove that $\lambda_1(L) \cdot \lambda_1(L^*) \leq n$.
(b) Prove that $\lambda_1(L) \cdot \lambda_n(L^*) \geq 1$.

Exercise 4.10 Let $L = L(B)$ be a lattice in $\mathbb{R}^n$, not necessarily of full rank. The *dual* of L is $L^* = \{x \in \mathrm{Span}(L) : \langle x, v\rangle \in \mathbb{Z} \text{ for all } v \in L\}$. Here, $\mathrm{Span}(L)$ is the usual linear span of L in $\mathbb{R}^n$.
(a) Prove that L^* is a lattice.
(b) Prove that $B(B^TB)^{-1}$ is a basis matrix for L^*.
(c) Prove that if $L' \subseteq L$ is a lattice of the same rank as L, then $L^* \subseteq (L')^*$.

5 Module-SIS and Module-LWE

Contents

5.1 The polynomial ring $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ 45
5.2 Module-SIS (MSIS) 48
5.3 Module-LWE (MLWE) 50
5.4 Notes and further references 52
5.5 Exercises 52

The security of Kyber relies on the assumed hardness of the module version of the LWE problem, whereas Dilithium's security relies on the module versions of both the SIS and LWE problems. These variants are derived from SIS and LWE by replacing integers modulo q with elements of the polynomial ring $R_q = \mathbb{Z}_q[x]/(x^n + 1)$.

5.1 The polynomial ring $R_q = \mathbb{Z}_q[x]/(x^n + 1)$

Let q be a prime and n a positive integer. The *polynomial ring* $R_q = \mathbb{Z}_q[x]/(x^n+1)$ is composed of the polynomials in $\mathbb{Z}_q[x]$ of degree less than n, with multiplication performed using the *reduction polynomial* $x^n + 1$.

To multiply two polynomials $f(x), g(x) \in R_q$, one (i) multiplies them using ordinary polynomial multiplication in $\mathbb{Z}_q[x]$ to obtain a polynomial $h(x) \in \mathbb{Z}_q[x]$ of degree at most $2n - 2$; and then (ii) divides $h(x)$ by $x^n + 1$ to get a remainder polynomial $r(x) \in \mathbb{Z}_q[x]$ of degree at most $n - 1$. The product of $f(x)$ and $g(x)$ in R_q is the remainder polynomial $r(x)$.

Example 5.1 (*polynomial ring*) Let $q = 43$ and $n = 4$. Then $R_q = \mathbb{Z}_{43}[x]/(x^4 + 1)$ is composed of the polynomials in $\mathbb{Z}_{43}[x]$ of degree at most 3.

Let $f(x) = 33 + 41x + 12x^3 \in R_q$ and $g(x) = 2 + 17x + 39x^2 + 14x^3 \in R_q$. Then their sum and difference in R_q are

$$f(x) + g(x) = 35 + 15x + 39x^2 + 26x^3$$

and

$$f(x) - g(x) = 31 + 24x + 4x^2 + 41x^3.$$

The product of f and g in $\mathbb{Z}_{43}[x]$ is the degree-6 polynomial

$$h(x) = 23 + 41x + 6x^2 + 21x^3 + 4x^4 + 38x^5 + 39x^6.$$

The division of $h(x)$ by $x^4 + 1$ can be accomplished by replacing x^4 by -1, x^5 by $-x$, and x^6 by $-x^2$, and then simplifying. We obtain

$$r(x) = 23 + 41x + 6x^2 + 21x^3 - 4 - 38x - 39x^2 = 19 + 3x + 10x^2 + 21x^3.$$

Thus, the product of f and g in R_q is

$$f(x) \times g(x) = r(x) = 19 + 3x + 10x^2 + 21x^3.$$

In §13, we will describe the Number-Theoretic Transform (NTT), a very fast method for performing polynomial multiplication in R_q.

Polynomials as vectors. Let $R_q = \mathbb{Z}_q[x]/(x^n + 1)$. A polynomial $f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1}x^{n-1} \in R_q$ can be represented by its length-n vector of coefficients $f = (a_0, a_1, a_2, \ldots, a_{n-1})$.

Example 5.2 (*representing polynomials as vectors*) Let $q = 31$ and $n = 5$, so $R_q = \mathbb{Z}_{31}[x]/(x^5 + 1)$. Let $f(x) = 12 + 3x + 4x^3 + 30x^4 \in R_q$ and $g(x) = 7 + 13x + 22x^2 + 27x^3 + 28x^4 \in R_q$. The vector representations of $f(x)$ and $g(x)$ are

$$f = (12, 3, 0, 4, 30) \text{ and } g = (7, 13, 22, 27, 28).$$

One can verify that the vector representations of the sum, difference, and product of $f(x)$ and $g(x)$ in R_q are

$$f + g = (19, 16, 22, 0, 27), \quad f - g = (5, 21, 9, 8, 2), \text{ and } f \times g = (18, 29, 1, 12, 28).$$

Polynomial multiplication via matrices. Polynomial multiplication in R_q can be expressed as a matrix-vector product. Let $a(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1}x^{n-1} \in R_q$ and $b(x) = b_0 + b_1 x + b_2 x^2 + \cdots + b_{n-1}x^{n-1} \in R_q$, and let $c(x) = a(x) \times b(x)$ in R_q. Then

$$\begin{aligned} c(x) &= a(x)(b_0 + b_1 x + b_2 x^2 + \cdots + b_{n-1}x^{n-1}) \\ &= b_0 a(x) + b_1 x \cdot a(x) + b_2 x^2 \cdot a(x) + \cdots + b_{n-1}x^{n-1} \cdot a(x). \end{aligned} \tag{17}$$

Now,

$$\begin{aligned} x \cdot a(x) &= a_0 x + a_1 x^2 + \cdots + a_{n-2}x^{n-1} + a_{n-1}x^n \\ &= -a_{n-1} + a_0 x + a_1 x^2 + \cdots + a_{n-2}x^{n-1} \pmod{x^n + 1} \\ &\leftrightarrow (-a_{n-1}, a_0, a_1, \ldots, a_{n-2}) \end{aligned}$$

since $x^n = -1 \pmod{x^n + 1}$. Thus, the vector representation of $x \cdot a(x)$ is the right cyclic shift of the vector representation of $a(x)$, with the cycled element negated. It follows from (17) that if $c(x) = a(x) \times b(x)$ in R_q, then

$$\underbrace{\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{bmatrix}}_{c} = \underbrace{\begin{bmatrix} a_0 & -a_{n-1} & -a_{n-2} & \cdots & -a_1 \\ a_1 & a_0 & -a_{n-1} & \cdots & -a_2 \\ a_2 & a_1 & a_0 & \cdots & -a_3 \\ \vdots & \vdots & \vdots & & \vdots \\ a_{n-1} & a_{n-2} & a_{n-3} & \cdots & a_0 \end{bmatrix}}_{\mathcal{A}(a)} \underbrace{\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{n-1} \end{bmatrix}}_{b}.$$

The $n \times n$ matrix above is an *anti-circulant matrix*, denoted $A = \mathcal{A}(a)$ or $A = \overline{\text{circ}}(a)$.

Example 5.3 (*matrix representation of R_q multiplication*) Let $q = 17$ and $n = 5$, so $R_q = \mathbb{Z}_{17}[x]/(x^5+1)$. Let $a(x) = 11 + 2x + 9x^3 + 13x^4$, $b(x) = 5 + 3x + 14x^2 + 7x^3 + 15x^4 \in R_q$. Then the vector representation of $c(x) = a(x)b(x)$ is

$$c = \underbrace{\begin{bmatrix} 11 & 4 & 8 & 0 & 15 \\ 2 & 11 & 4 & 8 & 0 \\ 0 & 2 & 11 & 4 & 8 \\ 9 & 0 & 2 & 11 & 4 \\ 13 & 9 & 0 & 2 & 11 \end{bmatrix}}_{\mathcal{A}(a)} \underbrace{\begin{bmatrix} 5 \\ 3 \\ 14 \\ 7 \\ 15 \end{bmatrix}}_{b} = \begin{bmatrix} 13 \\ 2 \\ 2 \\ 6 \\ 16 \end{bmatrix}.$$

Hence, $a(x)b(x) = 13 + 2x + 2x^2 + 6x^3 + 16x^4$ in R_q.

We next define a notion of "size" for integers in $\mathbb{Z}_q$ and polynomials in R_q.

Definition 5.4 Let $q \geq 3$ be an odd integer. The *size* of an integer $r \in \mathbb{Z}_q$, also called the *infinity norm* of r, is $\|r\|_\infty = |r \bmod q|$ where $|\cdot|$ denotes absolute value.

Note that $0 \leq \|r\|_\infty \leq (q-1)/2$. For example, if $q = 17$ then $0 \leq \|r\|_\infty \leq 8$. We have $\|6\|_\infty = 6$ and $\|13\|_\infty = 4$.

Definition 5.5 The *size* of a polynomial $f(x) = f_0 + f_1 x + f_2 x^2 + \cdots + f_{n-1}x^{n-1} \in R_q$ is $\|f\|_\infty = \max_i \|f_i\|_\infty$, also called the *infinity norm* of $f(x)$.

Definition 5.6 Let η be a positive integer that is small compared to $q/2$. The set of *small polynomials* in R_q is $S_\eta = \{f \in R_q \mid \|f\|_\infty \leq \eta\}$.

The product of small polynomials is also (relatively) small, as shown next.

Theorem 5.7 Fix $R_q = \mathbb{Z}_q[x]/(x^n+1)$. Let η_1 and η_2 be positive integers such that $n\eta_1\eta_2 < q/2$. Let $f \in S_{\eta_1}$ and $g \in S_{\eta_2}$. Then $fg \in S_{n\eta_1\eta_2}$.

Proof: Consider $f(x) = f_0 + f_1 x + \cdots + f_{n-1}x^{n-1} \in S_{\eta_1}$ and $g(x) = g_0 + g_1 x + \cdots + g_{n-1}x^{n-1} \in S_{\eta_2}$. Let $h(x) = h_0 + h_1 x + \cdots + h_{n-1}x^{n-1}$ be the product of $f(x)$ and $g(x)$ in R_q. By multiplying $f(x)$ and $g(x)$, then replacing x^{n+j} by $-x^j$ for $0 \leq j \leq n-2$, and then simplifying, one can observe that the coefficient h_i of x^i in $h(x)$ for each $i \in [0, n-1]$ is:

$$h_i = f_0 g_i + f_1 g_{i-1} + \cdots + f_i g_0 - f_{i+1}g_{n-1} - f_{i+2}g_{n-2} - \cdots - f_{n-2}g_{i+2} - f_{n-1}g_{i+1}. \quad (18)$$

Hence, $\|h_i\|_\infty \leq n\eta_1\eta_2$, whence $\|h\|_\infty \leq n\eta_1\eta_2$. □

Example 5.8 (*small polynomials*) Let $q = 379$ and $n = 5$ so $R_q = \mathbb{Z}_{379}[x]/(x^5 + 1)$. Let $f(x) = 378 + x^2 + 2x^3 + 377x^4 \in R_q$. The mods q representation of f is $f(x) = -1 + x^2 + 2x^3 - 2x^4$ so $f \in S_2$ is a small polynomial (with respect to $\eta_1 = 2$). Similarly, let $g(x) = 3 + 376x + 2x^2 + 378x^3 + 377x^4 \in S_3$ be a small polynomial (with respect to $\eta_2 = 3$). Then the product of f and g in R_q is $h(x) = 367 + 11x + 3x^2 + 371x^4$ which is in S_{12} since its mods q representation is $h(x) = -12 + 11x + 3x^2 - 8x^4$. Note that $12 \leq n\eta_1\eta_2 = 30 < q/2$.

5.2 Module-SIS (MSIS)

A *module* extends the notion of a vector space by replacing the vector space's field of scalars with a ring. The module of interest here is R_q^k, which consists of length-k vectors of polynomials in R_q. The *size* of module element $a = (a_1, a_2, \ldots, a_k) \in R_q^k$ is defined by

$$\|a\|_\infty = \max\{\|a_i\|_\infty \ : \ 1 \leq i \leq k\},$$

where the size $\|a_i\|_\infty$ of polynomial $a_i \in R_q$ is given in Definition 5.5. Addition and subtraction of elements in R_q^k are performed componentwise, ensuring that the result remains within R_q^k. The *inner product* of two vectors $a = (a_1, a_2, \ldots, a_k)$ and $b = (b_1, b_2, \ldots, b_k)$ in R_q^k is defined to be

$$a^T \cdot b = a_1b_1 + a_2b_2 + \cdots + a_kb_k,$$

which is a polynomial in R_q.

Example 5.9 (*module*) Let $q = 31$, $n = 5$ (so $R_q = \mathbb{Z}_{31}[x]/(x^5 + 1)$) and $k = 3$. Let $a = (a_1, a_2, a_3)$ and $b = (b_1, b_2, b_3)$ be:

$$a = \begin{bmatrix} 12 + 7x + x^3 + 29x^4 \\ 3 + 17x + 22x^2 + 9x^3 + 16x^4 \\ 14 + 8x + 29x^2 + 7x^3 + 4x^4 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 25 + 12x + 5x^2 + 23x^3 + 18x^4 \\ 3 + 29x + x^2 + 4x^3 + 7x^4 \\ 3 + 24x + 14x^2 + 4x^3 + 7x^4 \end{bmatrix} \in R_q^3.$$

Then

$$a + b = \begin{bmatrix} 6 + 19x + 5x^2 + 24x^3 + 16x^4 \\ 6 + 15x + 23x^2 + 13x^3 + 23x^4 \\ 17 + x + 12x^2 + 11x^3 + 11x^4 \end{bmatrix}, \quad a - b = \begin{bmatrix} 18 + 26x + 26x^2 + 9x^3 + 11x^4 \\ 19x + 21x^2 + 5x^3 + 9x^4 \\ 11 + 15x + 15x^2 + 3x^3 + 28x^4 \end{bmatrix},$$

and

$$a^T \cdot b = a_1b_1 + a_2b_2 + a_3b_3 = 4 + x + 25x^2 + 13x^3 + 18x^4.$$

MSIS is a variant of SIS in which elements of $\mathbb{Z}_q$ are replaced by polynomials in the ring $R_q = \mathbb{Z}_q[x]/(x^n + 1)$. Given a set of randomly selected elements in the module R_q^k, MSIS involves finding a non-trivial linear combination of the polynomial vectors that equals 0, with the scalar multipliers of the linear combination being small polynomials.

Definition 5.10 The *Module Short Integer Solution* problem MSIS(k, ℓ, n, q, η) is the following. Given $A \in_R R_q^{k\times\ell}$, find $z \in R_q^\ell$ such that $Az = 0$ in R_q^k, where $z \neq 0$ and $\|z\|_\infty \leq \eta$. Here, $k < \ell$, $\eta \ll q/2$, and $R_q = \mathbb{Z}_q[x]/(x^n + 1)$.

Note that an MSIS solution is not unique. Indeed, if $z = (z_1, z_2, \ldots, z_\ell)$ is one MSIS solution, then so is $(xz_1, xz_2, \ldots, xz_\ell)$. Thus, an MSIS solution yields as many as $2n - 1$ other distinct solutions.

Example 5.11 (*MSIS instance*) Let $k = 2$, $\ell = 3$, $n = 4$, $q = 71$ (so $R_q = \mathbb{Z}_{71}[x]/(x^4 + 1)$), and $\eta = 10$. Consider the MSIS instance

$$A = \begin{bmatrix} 16 + 58x^2 + 30x^3 & 22 + 40x + 4x^2 + 57x^3 & 53 + 8x + 21x^2 + 38x^3 \\ 61 + 32x + 3x^2 + 45x^3 & 23 + 5x + 58x^2 + 40x^3 & 49 + 40x + 29x^2 + 54x^3 \end{bmatrix} \in R_q^{2\times 3}.$$

Given A, the task is to find nonzero $z = (z_1, z_2, z_3) \in R_q^3$ such that $Az = 0$ and $\|z\|_\infty \leq 10$.

Recall that polynomial multiplication in R_q can be expressed as a matrix-vector product. Consequently, an equivalent formulation of this MSIS instance is to solve $\overline{A}\overline{z} = 0 \pmod{71}$ for nonzero $\overline{z} \in [-10, 10]^{12}$, where

$$\overline{A} = \left[\begin{array}{cccc|cccc|cccc} 16 & 41 & 13 & 0 & 22 & 14 & 67 & 31 & 53 & 33 & 50 & 63 \\ 0 & 16 & 41 & 13 & 40 & 22 & 14 & 67 & 8 & 53 & 33 & 50 \\ 58 & 0 & 16 & 41 & 4 & 40 & 22 & 14 & 21 & 8 & 53 & 33 \\ 30 & 58 & 0 & 16 & 57 & 4 & 40 & 22 & 38 & 21 & 8 & 53 \\ \hline 61 & 26 & 68 & 39 & 23 & 31 & 13 & 66 & 49 & 17 & 42 & 31 \\ 32 & 61 & 26 & 68 & 5 & 23 & 31 & 13 & 40 & 49 & 17 & 42 \\ 3 & 32 & 61 & 26 & 58 & 5 & 23 & 31 & 29 & 40 & 49 & 17 \\ 45 & 3 & 32 & 61 & 40 & 58 & 5 & 23 & 54 & 29 & 40 & 49 \end{array}\right]_{8\times 12}.$$

The 4×4 blocks in $\overline{A}$ are the anti-circulant matrices corresponding to the polynomials in A.

Gaussian elimination $\pmod{71}$ on $\overline{A}$ yields the following rank-8 matrix in reduced form:

$$\tilde{A} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 37 & 4 & 48 & 8 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 63 & 37 & 4 & 48 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 23 & 63 & 37 & 4 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 67 & 23 & 63 & 37 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 52 & 19 & 47 & 54 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 17 & 52 & 19 & 47 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 24 & 17 & 52 & 19 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 52 & 24 & 17 & 52 \end{bmatrix}.$$

Since the null space of $\overline{A}$ has dimension 4, the total number of solutions $\overline{z}$ to $\overline{A}\overline{z} = 0 \pmod{71}$ is $71^4 = 25,411,681$. Checking all these solutions $\overline{z}$ yields 16 nonzero solutions all of whose coordinates are in $[-10, 10]$. The two MSIS solutions, up to multiplication by $\pm 1, \pm x, \pm x^2, \pm x^3$, are $\overline{z} = (0, -5, 9, 1, -10, 4, -10, 10, -9, 5, 9, -6)$ and $\overline{z} =$

$(3, -10, -7, -4, 5, 0, 1, 5, 4, -9, 6, -6)$. The polynomial form of the first solution is $z = (-5x + 9x^2 + x^3,\ -10 + 4x - 10x^2 + 10x^3,\ -9 + 5x + 9x^2 - 6x^3)$.

We next define the inhomogeneous MSIS problem and its normal-form variant. Both of these problems are equivalent to MSIS (Exercises 5.3 and 5.4).

Definition 5.12 The *Module Inhomogeneous Short Integer Solution* problem MISIS(k, ℓ, n, q, η) is the following. Given $A \in_R R_q^{k\times\ell}$ and $b \in_R R_q^k$, find $z \in R_q^\ell$ such that $Az = b$, where $\|z\|_\infty \le \eta$. Here, $k < \ell$ and $\eta \ll q/2$.

Definition 5.13 The *normal-form MISIS* problem nf-MISIS(k, ℓ, n, q, η) is the following. Given $A \in_R R_q^{k\times\ell}$ and $b \in_R R_q^k$, find $z \in R_q^{k+\ell}$ such that $[A \mid I_k]z = b$ and $\|z\|_\infty \le \eta$.

5.3 Module-LWE (MLWE)

MLWE is a variant of LWE in which integers modulo q are replaced by polynomials in the ring $R_q = \mathbb{Z}_q[x]/(x^n + 1)$.

Definition 5.14 The *Module Learning With Errors* problem MLWE(k, ℓ, n, q, η) is the following. Let $A \in_R R_q^{k\times\ell}$, $s \in_R R_q^\ell$ and $e \in_R S_\eta^k$. Here, $k > \ell$, $\eta \ll q/2$, and $R_q = \mathbb{Z}_q[x]/(x^n + 1)$. Given A and $t = As + e \in R_q^k$, find s.

The columns $A_1, A_2, \ldots, A_\ell$ of A belong to the module R_q^k. MLWE asks to express the module element t, up to a small error e, as a linear combination of the A_i, where the coefficients of the linear combination are polynomials.

Example 5.15 (*MLWE instance*) Let $k = 3$, $\ell = 2$, $n = 4$, $q = 37$ (so $R_q = \mathbb{Z}_{37}[x]/(x^4+1)$), and $\eta = 1$. Randomly select

$$A = \begin{bmatrix} 33 + 34x^2 + 8x^3 & 21 + 10x + 19x^2 + 18x^3 \\ 3 + 13x + 19x^2 + 18x^3 & 34 + 19x + 15x^2 + 4x^3 \\ 16 + 6x + 5x^2 + 7x^3 & 24 + 6x + 35x^2 + 31x^3 \end{bmatrix} \in R_q^{3\times 2},$$

$$s = \begin{bmatrix} 32 + 15x + 16x^2 + 3x^3 \\ 10 + 28x + 6x^2 + 2x^3 \end{bmatrix} \in R_q^2, \quad \text{and } e = \begin{bmatrix} 1 - x \\ 1 - x^2 - x^3 \\ -1 + x - x^3 \end{bmatrix} \in S_1^3,$$

and define t to be

$$t = As + e = \begin{bmatrix} 2 + 29x + 6x^2 + 14x^3 \\ 28 + 20x + 9x^2 + 17x^3 \\ 13 + 5x + 12x^2 + 3x^3 \end{bmatrix} \in R_q^3.$$

Given (A, t), the MLWE challenge is to find $s \in R_q^2$ and $e \in S_1^3$ such that $As + e = t$.

Recall that polynomial multiplication in R_q can be expressed as a matrix-vector product. Consequently, an equivalent formulation of this MLWE instance is to solve $\overline{A}\overline{s}+\overline{e} = \overline{t} \pmod{37}$ for $\overline{s} \in \mathbb{Z}_{37}^8$ and $\overline{e} \in [-1,1]^{12}$, where

$$\overline{A} = \left[\begin{array}{cccc|cccc} 33 & 29 & 3 & 0 & 21 & 19 & 18 & 27 \\ 0 & 33 & 29 & 3 & 10 & 21 & 19 & 18 \\ 34 & 0 & 33 & 29 & 19 & 10 & 21 & 19 \\ 8 & 34 & 0 & 33 & 18 & 19 & 10 & 21 \\ \hline 3 & 19 & 18 & 24 & 34 & 33 & 22 & 18 \\ 13 & 3 & 19 & 18 & 19 & 34 & 33 & 22 \\ 19 & 13 & 3 & 19 & 15 & 19 & 34 & 33 \\ 18 & 19 & 13 & 3 & 4 & 15 & 19 & 34 \\ \hline 16 & 30 & 32 & 31 & 24 & 6 & 2 & 31 \\ 6 & 16 & 30 & 32 & 6 & 24 & 6 & 2 \\ 5 & 6 & 16 & 30 & 35 & 6 & 24 & 6 \\ 7 & 5 & 6 & 16 & 31 & 35 & 6 & 24 \end{array}\right]_{12\times 8} \quad \text{and} \quad \overline{t} = \left[\begin{array}{c} 2 \\ 29 \\ 6 \\ 14 \\ \hline 28 \\ 20 \\ 9 \\ 17 \\ \hline 13 \\ 5 \\ 12 \\ 3 \end{array}\right]_{12\times 1}.$$

The 4×4 blocks in $\overline{A}$ are the anti-circulant matrices corresponding to the polynomials in A, whereas the 4×1 blocks in $\overline{t}$ are the vector representations of the polynomials in t.

As it turns out, there are two MLWE solutions:

$$\overline{s} = (17, 4, 36, 30, 35, 31, 13, 12), \quad \overline{e} = (-1, 0, 1, 0, -1, -1, 1, 0, 0, -1, -1, -1);$$

and

$$\overline{s} = (32, 15, 16, 3, 10, 28, 6, 2), \quad \overline{e} = (1, -1, 0, 0, 1, 0, -1, -1, -1, 1, 0, -1).$$

The polynomial form of the first solution is $s = (17 + 4x + 36x^2 + 30x^3,\ 35 + 31x + 13x^2 + 12x^3)$, $e = (-1 + x^2,\ -1 - x + x^2,\ -x - x^2 - x^3)$. The second solution is the (s, e) pair that was used to construct this MLWE instance.

The decisional variant D-MLWE of MLWE is defined next. Analogously to Theorem 4.6, one can show that MLWE and D-MLWE are computationally equivalent (Exercise 5.7).

Definition 5.16 The *Decisional Module Learning With Errors* problem D-MLWE(k, ℓ, n, q, η) is the following. Let $A \in_R R_q^{k\times\ell}$, $s \in_R R_q^{\ell}$, $e \in_R S_\eta^k$, $r \in_R R_q^k$, and let $t = As + e \in R_q^k$. Let $z = t$ with probability $\frac{1}{2}$ and $z = r$ with probability $\frac{1}{2}$. Given A and z, decide (with success probability significantly greater than $\frac{1}{2}$) whether $z = t$ or $z = r$.

In the short-secret variants of MLWE and D-MLWE, the secret polynomial vector s is required to have small coefficients. Each of these variants is equivalent to MLWE (Exercises 5.8 and 5.9).

Definition 5.17 The *short-secret Module Learning With Errors* problem ss-MLWE$(k, \ell, n, q, \eta_1, \eta_2)$ is the following. Let $A \in_R R_q^{k\times\ell}$, $s \in_R S_{\eta_1}^{\ell}$ and $e \in_R S_{\eta_2}^k$. Here, $k \geq \ell$ and $\eta_1, \eta_2 \ll q/2$. Given A and $t = As + e \in R_q^k$, find s.

Definition 5.18 The *short-secret Decisional Module Learning With Errors* problem ss-D-MLWE$(k, \ell, n, q, \eta_1, \eta_2)$ is the following. Let $A \in_R R_q^{k\times\ell}$, $s \in_R S_{\eta_1}^{\ell}$, $e \in_R S_{\eta_2}^{k}$, $r \in_R R_q^k$, and let $t = As + e \in R_q^k$. Let $z = t$ with probability $\frac{1}{2}$ and $z = r$ with probability $\frac{1}{2}$. Here, $k \geq \ell$ and $\eta_1, \eta_2 \ll q/2$. Given A and z, decide (with success probability significantly greater than $\frac{1}{2}$) whether $z = t$ or $z = r$.

5.4 Notes and further references

Ring-SIS, an analogue of SIS that employs certain polynomial rings, was formulated by Lyubashevsky and Micciancio [154]. The module-based analogue, Module-SIS (MSIS), which employs vectors of polynomials, was developed by Langlois and Stehlé [139].

Ring-LWE, a variant of LWE that employs certain polynomial rings, was formulated by Lyubashevsky, Peikert and Regev [155], who also provided a worst-case to average-case quantum reduction from approx-SIVP in ideal lattices to Ring-LWE. The module variant, Module-LWE (MLWE), was first introduced by Brakerski, Gentry and Vaikuntanathan [43], and a worst-case to average-case quantum reduction was developed by Langlois and Stehlé [139]. A concrete analysis of the tightness gaps in the Ring-LWE and MLWE reductions was carried out by Koblitz, Samajder, Sarkar and Singha [132], who argued that the approximation factor in the SIVP problem is too large to yield meaningful security guarantees in practice. Wenger et al. [216] provide concrete benchmarks for several lattice-based attacks on LWE with parameter choices from Kyber and a Ring-LWE-based homomorphic encryption scheme.

5.5 Exercises

Exercise 5.1 Let $R_q = \mathbb{Z}_{677}[x]/(x^4 + 1)$, and let $a(x) = 625 + 436x + 84x^3 \in R_q$ and $b(x) = 166 + 285x + 376x^2 + 282x^3 \in R_q$. Verify the following computations in R_q:
(a) $a(x) + b(x) = 114 + 44x + 376x^2 + 366x^3$.
(b) $a(x) - b(x) = 459 + 151x + 301x^2 + 479x^3$.
(c) $a(x) \cdot b(x) = 186 + 246x + 457x^2 + 59x^3$.

Exercise 5.2 (†) Write a program to find all solutions z to the following MSIS$(2, 3, 4, 83, 10)$ instance:

$$A = \begin{bmatrix} 68 + 68x^2 + 54x^3 & 13 + 30x + 75x^2 + 18x^3 & 24 + 16x + 68x^2 + 79x^3 \\ 36 + 19x + 70x^2 + 36x^3 & 3 + 63x + 42x^2 + 53x^3 & 81x + 13x^2 + 45x^3 \end{bmatrix}.$$

Exercise 5.3 Prove that MSIS and MISIS are computationally equivalent.

Exercise 5.4 Prove that nf-MISIS$(k, \ell, n, q, \eta) \leq$ MISIS$(k, k + \ell, n, q, \eta)$ and MISIS$(k, k + \ell, n, q, \eta) \leq$ nf-MISIS(k, ℓ, n, q, η).

Exercise 5.5 (†) Write a program to find all solutions (s, e) to the following MLWE$(3, 2, 4, 37, 1)$ instance:

$$A = \begin{bmatrix} 32x^2 + 11x^3 & 14 + 13x + 22x^2 + 4x^3 \\ 5 + 11x + 14x^2 + 17x^3 & 7 + 6x + 5x^2 + 20x^3 \\ 21 + 10x + 5x^2 + 7x^3 & 6x + 5x^2 + 14x^3 \end{bmatrix}, \quad t = \begin{bmatrix} 2 + 31x + 18x^2 + 27x^3 \\ 3 + 18x + 14x^2 + 8x^3 \\ 5 + 27x^2 + 31x^3 \end{bmatrix}.$$

Exercise 5.6 Give a heuristic justification for why the solution to MLWE(k, ℓ, n, q, η) where $k > \ell$ and $\eta \ll q/2$ is expected to be unique.

Exercise 5.7 Prove that MLWE and D-MLWE are computationally equivalent.

Exercise 5.8 Prove that MLWE and ss-MLWE are computationally equivalent.

Exercise 5.9 Prove that ss-MLWE and ss-D-MLWE are computationally equivalent.

6 Kyber (ML-KEM)

Contents

6.1 Kyber public-key encryption 54
6.2 Decryption error probability 57
6.3 Ciphertext compression 62
6.4 Kyber-KEM 65
6.5 Notes and further references 68
6.6 Exercises 69

Kyber-KEM is a quantum-safe key encapsulation mechanism (KEM) standardized by NIST in FIPS 203, where its official name is Module-Lattice-based KEM (ML-KEM). It is designed to offer strong security guarantees against attacks from both classical and quantum computers. Kyber-KEM is based on the Kyber public-key encryption scheme (Kyber-PKE), described next.

6.1 Kyber public-key encryption

The Kyber public-key encryption scheme can be viewed as a natural extension of the Lindner-Peikert scheme described in §4.2, with elements in $\mathbb{Z}_q$ replaced by polynomials in the ring $R_q = \mathbb{Z}_q[x]/(x^n + 1)$.

Notation. Recall that $S_\eta = \{f \in R_q \mid \|f\|_\infty \leq \eta\}$, and that $\lceil x \rfloor$ denotes the nearest integer to x with ties broken upward. The plaintext space is $\{0,1\}^n$. A plaintext $m = (m_0, m_1, \ldots, m_{n-1}) \in \{0,1\}^n$ is associated with the polynomial $m(x) = m_0 + m_1 x + \cdots + m_{n-1}x^{n-1} \in R_q$. The rounding function Round_q (Definition 4.12) extends naturally to polynomials in R_q by applying it coefficientwise. For instance, if $q = 3329$ then $\mathrm{Round}_q(100 + 1000x + 2000x^2 + 3000x^3) = x + x^2$.

The Kyber public-key encryption scheme without ciphertext compression is described in Algorithm 6.1. The standardized version, which incorporates ciphertext compression, is presented in §6.3. The three Kyber parameter sets specified in FIPS 203 are listed in Table 6.1. Their intended security levels against classical attacks are approximately 128, 192 and 256 bits.

Example 6.2 (*Kyber-PKE*)
Domain parameters. $q = 991$, $n = 3$, $k = 3$, $\eta_1 = 3$, and $\eta_2 = 3$.
Key generation. Bob selects

$$A = \begin{bmatrix} 329 + 26x + 34x^2 & 295 + 745x + 292x^2 & 732 + 248x + 641x^2 \\ 833 + 849x + 647x^2 & 396 + 363x + 924x^2 & 312 + 803x + 31x^2 \\ 176 + 501x + 802x^2 & 738 + 941x + 521x^2 & 19 + 582x + 224x^2 \end{bmatrix},$$

$$s = \begin{bmatrix} 1 + 3x^2 \\ 1 + x \\ -2 + 2x^2 \end{bmatrix}, \text{ and } e = \begin{bmatrix} -2 + 3x - x^2 \\ 1 - x + 2x^2 \\ 3 + x - x^2 \end{bmatrix},$$

Parameter	Description	ML-KEM -512	ML-KEM -768	ML-KEM -1024
q	prime modulus	3329	3329	3329
n	$R_q = \mathbb{Z}_q[x]/(x^n + 1)$	256	256	256
k	dimensions of A	2	3	4
η_1	coefficient bound for s, e, r	3	2	2
η_2	coefficient bound for e_1, e_2	2	2	2
(d_u, d_v)	compression parameters	(10,4)	(10,4)	(11,5)
λ	security level	≈ 128	≈ 192	≈ 256

Table 6.1: FIPS 203 parameter sets for Kyber-KEM (ML-KEM).

Algorithm 6.1: Kyber public-key enc. scheme (without ciphertext compression)

Domain parameters. q, n, k, η_1, η_2.

Key generation. Bob does the following:

1 Select $A \in_R R_q^{k \times k}$, $s \in_R S_{\eta_1}^k$, and $e \in_R S_{\eta_1}^k$.
2 Compute $t = As + e$.
3 Bob's encryption (public) key is (A, t); his decryption (private) key is s.

Encryption. To encrypt a message $m \in \{0,1\}^n$ for Bob, Alice does the following:

1 Obtain an authentic copy of Bob's encryption key (A, t).
2 Select $r \in_R S_{\eta_1}^k$, $e_1 \in_R S_{\eta_2}^k$, and $e_2 \in_R S_{\eta_2}$.
3 Compute $u = A^T r + e_1$ and $v = t^T r + e_2 + \lceil \frac{q}{2} \rfloor m$ // see Figure 4.4
4 **return** $(c = (u, v))$.

Decryption. To decrypt $c = (u, v)$, Bob does:

1 Compute $m = \text{Round}_q(v - s^T u)$.
2 **return** (m).

and computes

$$t = As + e = \begin{bmatrix} 274 + 180x + 257x^2 \\ 484 + 971x + 42x^2 \\ 664 + 145x + 399x^2 \end{bmatrix}.$$

Bob's encryption key is (A, t), and his decryption key is s.
Encryption. To encrypt the plaintext message $m = 110 \leftrightarrow 1 + x$ for Bob, Alice selects

$$r = \begin{bmatrix} -2 - 2x + x^2 \\ 3x^2 \\ -1 - x + 3x^2 \end{bmatrix}, \quad e_1 = \begin{bmatrix} 3 - 2x + 2x^2 \\ 2 - 3x - x^2 \\ -3 + x - 2x^2 \end{bmatrix}, \quad \text{and } e_2 = 3 - 3x + 2x^2$$

and computes

$$u = A^T r + e_1 = \begin{bmatrix} 918 + 176x + 944x^2 \\ 77 + 530x + 160x^2 \\ 572 + 989x + 130x^2 \end{bmatrix}$$

and

$$v = t^T r + e_2 + \lceil 991/2 \rfloor m = 636 + 169x + 320x^2.$$

The ciphertext is $c = (u, v)$.
Decryption. To decrypt c, Bob uses his decryption key s to compute

$$v - s^T u = 478 + 492x + 3x^2$$

and rounds its coefficients to obtain $1 + x$, thereby recovering the plaintext $m = 110$.

Correctness of decryption. Decryption succeeds precisely when $\text{Round}_q(v - s^T u) = m$. Now, substituting $u = A^T r + e_1$ and $v = t^T r + e_2 + \lceil \frac{q}{2} \rfloor m$ into $v - s^T u$ yields

$$\begin{aligned} v - s^T u &= (t^T r + e_2 + \lceil \frac{q}{2} \rfloor m) - s^T (A^T r + e_1) \\ &= (s^T A^T + e^T) r + e_2 + \lceil \frac{q}{2} \rfloor m - s^T (A^T r + e_1) \qquad \text{(after replacing } t^T \text{ by } s^T A^T + e^T) \\ &= e^T r - s^T e_1 + e_2 + \lceil \frac{q}{2} \rfloor m. \end{aligned}$$

Thus, $\text{Round}_q(v - s^T u) = m$ if each coefficient E_i of the *decryption error polynomial*

$$E(x) = e^T r - s^T e_1 + e_2 \tag{19}$$

satisfies $|E_i \bmod q| < \lceil q/4 \rfloor$, i.e., $\|E\|_\infty < \lceil q/4 \rfloor$. Notice that e, e_1, e_2, r, and s all have small coefficients, so one can hope that $E(x)$ also has relatively small coefficients (cf. Theorem 5.7). Indeed, since $\|e\|_\infty \le \eta_1$ and $\|r\|_\infty \le \eta_1$, we have $\|e^T r\|_\infty \le kn\eta_1^2$ if $kn\eta_1^2 < q/2$. Similarly, $\|s^T e_1\|_\infty \le kn\eta_1\eta_2$ if $kn\eta_1\eta_2 < q/2$. Thus, if

$$kn\eta_1^2 + kn\eta_1\eta_2 + \eta_2 < \lceil q/4 \rfloor, \tag{20}$$

then $\|E_i\|_\infty \le kn\eta_1^2 + kn\eta_1\eta_2 + \eta_2 < \lceil q/4 \rfloor$ and decryption succeeds.

Example 6.3 (*decryption failure probability*) The ML-KEM-512 parameters (Table 6.1) are $q = 3329$, $n = 256$, $k = 2$, $\eta_1 = 3$, $\eta_2 = 2$. For these parameters, $kn\eta_1^2 + kn\eta_1\eta_2 + \eta_2 = 7682 \not< \lceil \frac{q}{4} \rfloor$, and hence decryption is not guaranteed to succeed. However, significant cancellation is expected when computing $E(x) = e^T r - s^T e_1 + e_2$ so its coefficients are expected to be substantially smaller than the pessimistic bound above. Indeed, we will show in §6.2 that the probability that $\|E\|_\infty \geq \lceil \frac{q}{4} \rfloor$ is less than 2^{-233}. Hence, the probability of decryption failure is negligibly small.

Security. The security of Kyber-PKE is captured in the following theorem.

Theorem 6.4 The Kyber public-key encryption scheme is indistinguishable under chosen-plaintext attack (Definition A.2) assuming that ss-D-MLWE is intractable.

Proof: The encryption operation can be written as

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} A^T \\ t^T \end{bmatrix} r + \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \lceil \frac{q}{2} \rfloor m \end{bmatrix}.$$

By the ss-D-MLWE$(k, k, n, q, \eta_1, \eta_1)$ intractability assumption,

$$\begin{bmatrix} A^T \\ t^T \end{bmatrix}$$

is indistinguishable from random. Next, by the ss-D-MLWE$(k+1, k, n, q, \eta_1, \eta_2)$ intractability assumption,

$$\begin{bmatrix} A^T \\ t^T \end{bmatrix} r + \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} = \begin{bmatrix} A^T r + e_1 \\ t^T r + e_2 \end{bmatrix} \tag{21}$$

is indistinguishable from random. Thus, from the adversary's perspective, v is the sum of the random polynomial $t^T r + e_2$ and the scaled message polynomial $\lceil \frac{q}{2} \rfloor m$. Hence, the adversary learns nothing about m. □

6.2 Decryption error probability

In this section, we analyze the probability of decryption failure in Kyber (without ciphertext compression).

6.2.1 Centered binomial distributions

One detail not mentioned in the description of Kyber in §6.1 is that all coefficients of polynomials in the vectors s, e, r, e_1 and in the polynomial e_2 are sampled from centered binomial distributions instead of uniform distributions.

Let $S \subseteq \mathbb{Z}$ be a finite set, and let X be a discrete random variable taking values in S. For each $j \in S$, let p_j denote the probability that $X = j$. The *generating series* of X is

$$\Phi_X(z) = \sum_{j \in S} p_j z^j. \tag{22}$$

The coefficient of z^j in $\Phi_X(z)$ is precisely p_j, which we denote using the coefficient operator $[\cdot]$ by

$$p_j = [z^j]\,\Phi_X(z).$$

Now, let Y be another random variable taking values in S, and suppose that X and Y are independent. Then the generating series of the sum $X + Y$ is

$$\Phi_{X+Y}(z) = \Phi_X(z) \times \Phi_Y(z). \tag{23}$$

Definition 6.5 For $\eta \geq 1$, define the random variable

$$C_\eta = \sum_{i=1}^{\eta} (a_i - b_i), \tag{24}$$

where the a_i and b_i are independent random variables, each uniformly distributed over $\{0, 1\}$. Then C_η is said to follow the *centered binomial distribution with parameter* η, written

$$C_\eta \sim \mathrm{CBD}_\eta.$$

The support of C_η is $\{-\eta, \ldots, \eta\}$, and the distribution is symmetric, i.e.,

$$\Pr(C_\eta = j) = \Pr(C_\eta = -j) \;\text{ for }\; -\eta \leq j \leq \eta.$$

Theorem 6.6 Let $C_\eta \sim \mathrm{CBD}_\eta$. Then $E(C_\eta) = 0$ and $E(C_\eta^2) = \eta/2$. Moreover,

$$\Pr(C_\eta = j) = \binom{2\eta}{\eta + j} / 2^{2\eta} \;\text{ for all }\; -\eta \leq j \leq \eta. \tag{25}$$

Proof: Since the distribution of C_η is symmetric about 0, we have $E(C_\eta) = 0$.

Let A_η and B_η be independent binomial random variables, each corresponding to the sum of η independent Bernoulli trials with success probability $p = \frac{1}{2}$. Recall that

$$E(A_\eta) = E(B_\eta) = \eta p = \frac{\eta}{2} \;\text{ and }\; \mathrm{Var}(A_\eta) = \mathrm{Var}(B_\eta) = \eta p(1-p) = \frac{\eta}{4}.$$

Since $C_\eta = A_\eta - B_\eta$, we obtain

$$\begin{aligned} E(C_\eta^2) &= E((A_\eta - B_\eta)^2) = E(A_\eta^2) - 2E(A_\eta)E(B_\eta) + E(B_\eta^2) \\ &= 2E(A_\eta^2) - 2E(A_\eta)^2 = 2\,\mathrm{Var}(A_\eta) = \frac{\eta}{2}. \end{aligned}$$

Now, for each $1 \leq i \leq \eta$, the generating series of a_i and $-b_i$ are

$$\Phi_{a_i}(z) = \frac{1}{2} + \frac{z}{2} \quad \text{and} \quad \Phi_{-b_i}(z) = \frac{1}{2} + \frac{1}{2z},$$

respectively. Hence by (23), the generating series of $C_\eta = A_\eta - B_\eta$ is

$$\Phi_{C_\eta}(z) = \left(\left(\frac{1}{2} + \frac{z}{2}\right)\left(\frac{1}{2} + \frac{1}{2z}\right)\right)^\eta = \frac{z^{-\eta}(z+1)^{2\eta}}{2^{2\eta}}.$$

Thus, for $0 \leq j \leq \eta$, we have

$$\Pr(C_\eta = j) = [z^j]\,\Phi_{C_\eta}(z) = [z^j]\frac{z^{-\eta}(z+1)^{2\eta}}{2^{2\eta}} = [z^{\eta+j}]\frac{(z+1)^{2\eta}}{2^{2\eta}} = \binom{2\eta}{\eta+j}/2^{2\eta}.$$

The result for negative j follows by symmetry. □

Figure 6.1 depicts the probability distribution CBD_3.

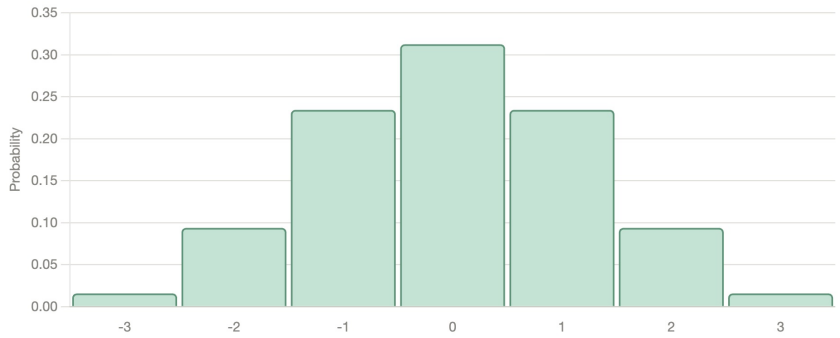


Figure 6.1: Centered binomial distribution CBD_η with $\eta = 3$.

In the standardized version of Kyber, the coefficients of the polynomials in the vectors s, e and r are sampled from the centered binomial distribution with parameter η_1, rather than uniformly from $[-\eta_1, \eta_1]$; we denote this by

$$s, e, r \in_{CBD} S^k_{\eta_1}. \tag{26}$$

Similarly, the coefficients of the polynomials in e_1 and the polynomial e_2 are sampled from the centered binomial distribution with parameter η_2, so that

$$e_1 \in_{CBD} S^k_{\eta_2} \text{ and } e_2 \in_{CBD} S_{\eta_2}. \tag{27}$$

Kyber uses the centered binomial distribution as its noise distribution for three main reasons. First, it is efficient to sample in practice: each sample can be generated using only a small number of uniform random bits (the a_i and b_i) and simple integer operations (additions and subtractions), avoiding costly floating-point arithmetic (required for Gaussian sampling) and rejection sampling (needed for uniform sampling in intervals $[-\eta, \eta]$). This makes it well-suited for constant-time implementations. Second, the distribution places higher probability on values close to 0, which helps reduce the probability of decryption errors. Third, it provides a good approximation to a discrete Gaussian, the standard choice in theoretical security analyses in lattice-based cryptography.

6.2.2 Computing the decryption error probability

Recall that the decryption error polynomial is $E(x) = e^T r - s^T e_1 + e_2$, and that decryption is correct when $\|E\|_\infty < \lceil q/4 \rfloor$. For each $0 \leq i \leq n-1$, let δ_i be the probability that the

ith coefficient of the plaintext is decrypted incorrectly. Note that the δ_i's are all equal, so we will denote them by δ. Since an incorrect decryption of the ith coefficient implies $|E_i| \geq \lceil q/4 \rfloor$, we have $\delta \leq \Pr(|E_i| \geq \lceil q/4 \rfloor)$. We will compute this probability exactly, and denote it by δ'.

Let $A \sim \mathrm{CBD}_{\eta_1}$, $A' \sim \mathrm{CBD}_{\eta_1}$, and $B \sim \mathrm{CBD}_{\eta_2}$ be random variables. Define the random variables $C = AA'$ and $D = AB$ with supports contained in $[-\eta_1^2, \eta_1^2]$ and $[-\eta_1\eta_2, \eta_1\eta_2]$, respectively. Let $f \in_{CBD} S_{\eta_1}$ and $g \in_{CBD} S_{\eta_1}$, and consider their product in $\mathbb{Z}[x]/(x^n+1)$, i.e., without applying the mods q operation to the coefficients. Then each coefficient of the product is a random variable $\widehat{C} = \sum_{j=1}^{n} C_j$, where each C_j is an independent and identically distributed (i.i.d.) copy of C. Similarly, each coefficient of the product of polynomials $f \in_{CBD} S_{\eta_1}$ and $g \in_{CBD} S_{\eta_2}$ in $\mathbb{Z}[x]/(x^n + 1)$ is a random variable $\widehat{D} = \sum_{j=1}^{n} D_j$, where each D_j is an i.i.d. copy of D. It follows that when $E(x)$ is considered a polynomial in $\mathbb{Z}[x]/(x^n + 1)$, each coefficient is a random variable

$$E_i = \sum_{\ell=1}^{k} \widehat{C}_\ell + \sum_{\ell=1}^{k} \widehat{D}_\ell + B, \tag{28}$$

where the $\widehat{C}_\ell$ and $\widehat{D}_\ell$ are i.i.d. copies of $\widehat{C}$ and $\widehat{D}$. The three terms on the right-hand side of (28) correspond to the coefficients of x^i in $e^T r$, $-s^T e_1$ and e_2, respectively. Notice that the support of E_i is $[-kn\eta_1^2 - kn\eta_1\eta_2 - \eta_2,\ kn\eta_1^2 + kn\eta_1\eta_2 + \eta_2]$. Thus, the generating series of E_i is

$$\Phi_{E_i}(z) = \Phi_{\widehat{C}}(z)^k \cdot \Phi_{\widehat{D}}(z)^k \cdot \Phi_B(z) = \Phi_C(z)^{kn} \cdot \Phi_D(z)^{kn} \cdot \Phi_B(z). \tag{29}$$

Therefore,

$$\delta \leq \delta' = 1 - \sum_{j=-\lceil q/4 \rfloor+1}^{\lceil q/4 \rfloor-1} [z^j]\, \Phi_{E_i}(z). \tag{30}$$

By the union bound, an upper bound on the decryption error probability p is

$$p \leq n\delta \leq n\delta'.$$

6.2.3 ML-KEM-512 decryption error probability

The ML-KEM-512 parameters (see Table 6.1) are

$$q = 3329, \quad n = 256, \quad k = 2, \quad \eta_1 = 3, \quad \eta_2 = 2.$$

Let $A \sim \mathrm{CBD}_3$, $A' \sim \mathrm{CBD}_3$, and $B \sim \mathrm{CBD}_2$ be random variables, and let $C = AA'$ and $D = AB$. The probability distributions of A, B, C and D are shown in Table 6.2. Hence,

A	-3	-2	-1	0	1	2	3
$64 \cdot \Pr(A)$	1	6	15	20	15	6	1

B	-2	-1	0	1	2
$16 \cdot \Pr(B)$	1	4	6	4	1

C	-9	-6	-4	-3	-2	-1	0	1	2	3	4	6	9
$2048 \cdot \Pr(C)$	1	12	36	30	180	225	1080	225	180	30	36	12	1

D	-6	-4	-3	-2	-1	0	1	2	3	4	6
$1024 \cdot \Pr(D)$	2	12	8	78	120	584	120	78	8	12	2

Table 6.2: Probability distributions of A, B, C and D for the ML-KEM-512 parameters.

the generating series of B, C and D are:

$$\Phi_B(z) = \frac{1}{16z^2} + \frac{1}{4z} + \frac{3}{8} + \frac{z}{4} + \frac{z^2}{16},$$

$$\Phi_C(z) = \frac{1}{2048z^9} + \frac{3}{512z^6} + \frac{9}{512z^4} + \frac{15}{1024z^3} + \frac{45}{512z^2} + \frac{225}{2048z} + \frac{135}{256} + \frac{225z}{2048} + \frac{45z^2}{512} + \frac{15z^3}{1024} + \frac{9z^4}{512} + \frac{3z^6}{512} + \frac{z^9}{2048},$$

$$\Phi_D(z) = \frac{1}{512z^6} + \frac{3}{256z^4} + \frac{1}{128z^3} + \frac{39}{512z^2} + \frac{15}{128z} + \frac{73}{128} + \frac{15z}{128} + \frac{39z^2}{512} + \frac{z^3}{128} + \frac{3z^4}{256} + \frac{z^6}{512}.$$

Therefore, the generating series of each E_i is

$$\Phi_{E_i}(z) = \Phi_C(z)^{512} \cdot \Phi_D(z)^{512} \cdot \Phi_B(z)$$

and an upper bound on δ is

$$\delta \leq \delta' = 1 - \sum_{j=-831}^{831} [z^j]\, \Phi_{E_i}(z).$$

Finally, an upper bound on p is

$$p \leq 256\delta'.$$

The Maple computer algebra system was used to extract coefficients from the generating series $\Phi_{E_i}(z)$, yielding the upper bound

$$p \leq 2^{-233}$$

on the decryption error probability for Kyber (without ciphertext compression) when used with the ML-KEM-512 parameters.

Similarly, upper bounds on the decryption error probability for Kyber (without ciphertext compression) when used with the ML-KEM-768 and ML-KEM-1024 parameter sets are $p \leq 2^{-300}$ and $p \leq 2^{-230}$, respectively.

6.3 Ciphertext compression

The FIPS 203 standard specifies a method for compressing and decompressing Kyber-PKE ciphertexts. The approach involves discarding the "low-order" bits of the coefficients of all polynomials in the ciphertext $c = (u, v)$. This compression is lossy, but the compression parameters were carefully selected to ensure that the errors introduced during decompression do not significantly affect the decryption error probability.

For a *compression parameter* $1 \leq d < \log_2 q$ and integer $X \in [0, q-1]$, define

$$\text{Compress}_q(X, d) = \lceil (2^d/q) \cdot X \rfloor \bmod 2^d. \tag{31}$$

For $Y \in [0, 2^d - 1]$, define

$$\text{Decompress}_q(Y, d) = \lceil (q/2^d) \cdot Y \rfloor \bmod q. \tag{32}$$

The decompression error is quantified next.

Theorem 6.7 Consider compression parameter $1 \leq d < \log_2 q$. Let $X \in [0, q-1]$, $Y = \text{Compress}_q(X, d)$, and $X' = \text{Decompress}_q(Y, d)$. Then $\|X' - X\|_\infty \leq \lceil q/2^{d+1} \rfloor$.

Example 6.8 Let $q = 19$ and $d = 2$. For each $X \in [0, 18]$, Figure 6.2 shows the $Y = \text{Compress}_{19}(X, 2)$ values (in blue) and the $X' = \text{Decompress}_{19}(Y, 2)$ values (in red).

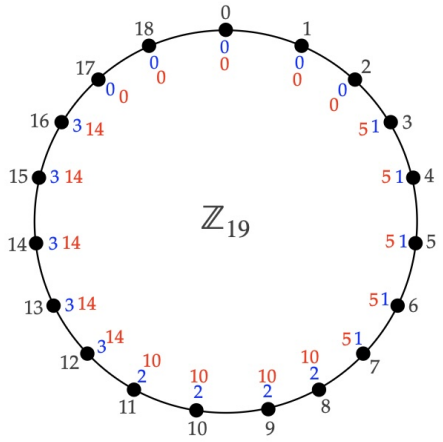


Figure 6.2: Compression with $q = 19$ and $d = 2$.

The functions Compress and Decompress extend to polynomials in R_q and polynomial vectors in R_q^k by applying them to each coefficient of a polynomial.

Example 6.9 Let $q = 3329$. If $X \in [0, q-1]$, $Y = \text{Compress}_q(X, 10)$, and $X' = \text{Decompress}_q(Y, 10)$, then $\|X' - X\|_\infty \leq 2$. Also, if $Y = \text{Compress}_q(X, 4)$, and $X' = \text{Decompress}_q(Y, 4)$, then $\|X' - X\|_\infty \leq 104$.

For example, if

$$U = 2426 + 1198x + 2055x^2 + 2409x^3 \in \mathbb{Z}_q[x]/(x^4 + 1),$$

then

$$V = \text{Compress}_q(U, 10) = 746 + 369x + 632x^2 + 741x^3$$

and

$$U' = \text{Decompress}_q(V, 10) = 2425 + 1200x + 2055x^2 + 2409x^3.$$

We have $U' - U = -1 + 2x$.

Also,

$$W = \text{Compress}_q(U, 4) = 12 + 6x + 10x^2 + 12x^3,$$

$$U'' = \text{Decompress}_q(W, 4) = 2497 + 1248x + 2081x^2 + 2497x^3,$$

and $U'' - U = 71 + 50x + 26x^2 + 88x^3$.

The Kyber public-key encryption scheme with ciphertext compression is described in Algorithm 6.10. The *compression parameters* d_u and d_v specified in FIPS 203 are listed in Table 6.1. Ciphertext compression reduces the ciphertext size from 1152, 1536, and 1920 bytes for the ML-KEM-512, ML-KEM-768, and ML-KEM-1024 parameter sets, respectively, to 768, 1088, and 1568 bytes.

Algorithm 6.10: Kyber public-key enc. scheme (with ciphertext compression)

Domain parameters. q, n, k, η_1, η_2, d_u, d_v.

Key generation. Bob does the following:

1 Select $A \in_R R_q^{k \times k}$, $s \in_{CBD} S_{\eta_1}^k$, and $e \in_{CBD} S_{\eta_1}^k$.
2 Compute $t = As + e$.
3 Bob's encryption (public) key is (A, t); his decryption (private) key is s.

Encryption. To encrypt a message $m \in \{0,1\}^n$ for Bob, Alice does the following:

1 Obtain an authentic copy of Bob's encryption key (A, t).
2 Select $r \in_{CBD} S_{\eta_1}^k$, $e_1 \in_{CBD} S_{\eta_2}^k$, and $e_2 \in_{CBD} S_{\eta_2}$.
3 Compute $u = A^T r + e_1$ and $v = t^T r + e_2 + \lceil \frac{q}{2} \rfloor m$. // see Figure 4.4
4 Compute $c_1 = \text{Compress}_q(u, d_u)$ and $c_2 = \text{Compress}_q(v, d_v)$.
5 **return** $(c = (c_1, c_2))$.

Decryption. To decrypt $c = (c_1, c_2)$, Bob does:

1 Compute $u' = \text{Decompress}_q(c_1, d_u)$ and $v' = \text{Decompress}_q(c_2, d_v)$.
2 Compute $m = \text{Round}_q(v' - s^T u')$.
3 **return** (m).

Correctness of decryption. Ciphertext compression introduces additional noise into the decryption procedure. Decryption succeeds precisely when $m = \mathrm{Round}_q(v' - s^T u')$. Let $e_u = u' - u \in R_q^k$ and $e_v = v' - v \in R_q$ denote the compression errors. Then

$$v' - s^T u' = (v + e_v) - s^T(u + e_u) = e^T r - s^T e_1 + e_2 - s^T e_u + e_v + \lceil \frac{q}{2} \rfloor m.$$

Hence, decryption is correct if every coefficient E_i of the *decryption error polynomial*

$$E(x) = e^T r - s^T e_1 + e_2 - s^T e_u + e_v \tag{33}$$

satisfies $|E_i \bmod q| < \lceil q/4 \rfloor$, i.e., $\|E\|_\infty < \lceil q/4 \rfloor$. The probability that this condition holds can be estimated in the same manner as in §6.2.

Assume that the quantities u and v are uniformly distributed over R_q^k and R_q, respectively. Let C_u and C_v denote the random variables capturing the compression error of a uniformly random element of $\mathbb{Z}_q$ under parameters d_u and d_v. The two terms $-s^T e_1$ and $-s^T e_u$ share the coefficients of s and thus cannot be treated as independent. Let $F = A(B + C_u)$, where $A \sim \mathrm{CBD}_{\eta_1}$, $B \sim \mathrm{CBD}_{\eta_2}$, and C_u are independent. Then, analogously to (29), the generating series of the random variable E_i is

$$\Phi_{E_i}(z) = \Phi_C(z)^{kn} \cdot \Phi_F(z)^{kn} \cdot \Phi_B(z) \cdot \Phi_{C_v}(z), \tag{34}$$

where the factors $\Phi_F(z)^{kn}$ and $\Phi_{C_v}(z)$ account for $-s^T(e_1 + e_u)$ and e_v in (33). Consequently, an upper bound on the decryption error probability is given by $p \le n\delta'$ where

$$\delta' = 1 - \sum_{j=-\lceil q/4 \rfloor + 1}^{\lceil q/4 \rfloor - 1} [z^j]\, \Phi_{E_i}(z).$$

ML-KEM-512 decryption error probability. The ML-KEM-512 parameters (see Table 6.1) are

$$q = 3329, \quad n = 256, \quad k = 2, \quad \eta_1 = 3, \quad \eta_2 = 2, \quad d_u = 10, \quad d_v = 4.$$

The probability distributions of C_u, C_v and F are given in Table 6.3. This yields the following upper bound on the decryption error probability for Kyber with ciphertext compression:

$$p \le 2^{-138}.$$

Using the same approach, upper bounds on the decryption error probability for Kyber with ciphertext compression when used with the ML-KEM-768 and ML-KEM-1024 parameter sets are $p \le 2^{-164}$ and $p \le 2^{-174}$, respectively.

Remark 6.11 (*selecting Kyber parameters*) The ML-KEM-512, ML-KEM-768 and ML-KEM-1024 parameter sets for Kyber specified in FIPS 203 (see Table 6.1) were carefully chosen to balance security, public key size, ciphertext size, decryption error probability, and implementation efficiency. The following notes illustrate the delicate balance required when selecting the parameters n, q, k, η_1, η_2, d_u and d_v.

C_u	-2	-1	0	1	2
$3329 \cdot \Pr(C_u)$	128	1024	1024	1024	129

C_v	-104	-103	-102	$\cdots$	102	103	104
$3329 \cdot \Pr(C_v)$	8	16	16	$\cdots$	16	16	9

F	± 12	± 9	± 8	± 6	± 4
$16 \cdot 64 \cdot 3329 \cdot \Pr(F)$	257	3076	1542	30238	74547

F	± 3	± 2	± 1	0
$16 \cdot 64 \cdot 3329 \cdot \Pr(F)$	69696	318066	353340	1707372

Table 6.3: Probability distributions of C_u, C_v and F for the ML-KEM-512 parameters.

1. To facilitate fast polynomial multiplication in R_q via the Number-Theoretic Transform (see §13.4), the degree n was chosen to be a power of two, and the prime q was chosen to satisfy the condition $q \equiv 1 \pmod{n}$.

2. A larger modulus q reduces the decryption error probability, but at the expense of larger public keys and ciphertexts.

3. The parameter k can be increased to achieve higher levels of security against MLWE attacks without altering the underlying polynomial ring R_q. However, increasing k also increases the sizes of both public keys and ciphertexts.

4. As demonstrated by the decryption error polynomial in (33), smaller secret and error bounds η_1 and η_2 reduce the decryption error probability. Conversely, decreasing η_1 and η_2 also makes the underlying MLWE problem easier.

5. Smaller compression parameters d_u and d_v yield smaller ciphertexts, but their use increases the decryption error probability.

The impact of the parameters n, q, k, η_1 and η_2 on the concrete security of Kyber is analyzed further in §11.6.

6.4 Kyber-KEM

The official name of Kyber-KEM in FIPS 203 is Module-Lattice-based KEM (ML-KEM). As shown in Algorithm 6.12, Kyber-KEM is derived by applying a variant of the Fujisaki-Okamoto (FO) transform to the Kyber public-key encryption scheme. The FO transform is a generic method for converting a public-key encryption scheme that is indistinguishable under chosen-plaintext attack to one that is indistinguishable under chosen-ciphertext attack (Definition A.3).

The transform utilizes three hash functions, G, H, and J, which are derived from SHA3 and SHAKE256. Kyber-PKE is employed to encrypt a randomly chosen bit string m of length 256. During encapsulation, *derandomization* is applied, whereby m and the

encapsulation key ek are hashed to produce a random seed R and the secret key K. The random polynomial vectors r, e_1 and the random polynomial e_2 required for Kyber-PKE are derived from R. This process transforms the randomized Kyber public-key encryption scheme into a deterministic one, meaning the ciphertext c depends only on the message m and encapsulation key ek, without relying on any additional random bits generated during encryption.

In decapsulation, the recipient decrypts the Kyber-PKE ciphertext c to recover m', then hashes m' and ek to obtain R' and K'. He then re-encrypts m' using R' and compares the resulting ciphertext c' with the received c. If the encryption was performed correctly, then m' will equal m with overwhelming probability and consequently $R' = R$ and $K' = K$. This ensures that the re-encryption of m results in the ciphertext c, which is why a deterministic encryption scheme is necessary. However, if decryption fails and $m' \neq m$, the re-encrypted ciphertext c' will differ from c with overwhelming probability.

Therefore, if c' equals c, Bob accepts K as the shared key. Otherwise, he generates a random key $\overline{K}$ obtained by hashing c and the secret z; in this case the decapsulation is deemed to have failed.

Kyber-KEM exhibits *plaintext awareness*, meaning that decapsulation will produce K (and not $\overline{K}$) only if the entity who performed the encapsulation knew K. This property is designed to ensure resistance to chosen-ciphertext attacks because the adversary cannot gain any useful information by submitting a ciphertext to Bob for decapsulation. Either the adversary already knew K (in which case nothing new is learned when Bob returns the same K), or the adversary does not know K, in which case Bob will almost certainly return a key $\overline{K}$ that is randomly selected and independent of K, preventing the adversary from learning anything useful.

Decapsulation fails when c does not equal c', in which case the key $\overline{K}$ outputted is different from the encapsulated key K. This failure can occur even if Alice and Bob behave honestly, as there is a very small chance that a failure in the underlying Kyber-PKE will cause m' to differ from m. However, for the domain parameters specified in FIPS 203, the probability of Kyber-PKE decryption failure is negligible, and consequently the probability of a Kyber-KEM failure is also extremely low.

Decapsulation error probability. Kyber-KEM decapsulation fails if $\overline{K}$ is returned. The probability p of this event coincides with the decryption error probability for Kyber encryption, namely $p \leq 2^{-138}$, 2^{-164}, and 2^{-174} for the ML-KEM-512, ML-KEM-768, and ML-KEM-1024 parameter sets, respectively. In comparison, the targeted security levels for these parameter sets are approximately 128, 192 and 256 bits. Ideally, the decryption error probabilities for ML-KEM-768 and ML-KEM-1024 would be closer to 2^{-192} and 2^{-256} in view of the following potential chosen-ciphertext attack.

Decapsulation fails when at least one coefficient E_i of the error polynomial $E(x) = e^T r - s^T e_1 + e_2 - s^T e_u + e_v$ satisfies $\|E_i\|_\infty \geq \lceil q/4 \rfloor$. An adversary selects a random m, computes the corresponding ciphertext c, and submits c to a decapsulation oracle. This procedure is repeated until a decapsulation error occurs. Upon such a failure, the

Algorithm 6.12: Kyber key encapsulation mechanism.

Domain parameters. Kyber-PKE parameters q, $n = 256$, k, η_1, η_2, and hash functions $G : \{0,1\}^* \to \{0,1\}^{2n}$, $H : \{0,1\}^* \to \{0,1\}^n$, $J : \{0,1\}^* \to \{0,1\}^n$.

Key generation. Bob does the following:

1 Use the Kyber-PKE key generation algorithm to select a Kyber-PKE encryption key (A, t) and decryption key s.
2 Select $z \in_R \{0,1\}^{256}$.
3 Bob's encapsulation (public) key is $ek = (A, t)$; his decapsulation (private) key is $dk = (s, ek, H(ek), z)$.

Encapsulation. To establish an n-bit shared secret key with Bob, Alice does:

1 Obtain an authentic copy of Bob's encapsulation key ek.
2 Select $m \in_R \{0,1\}^{256}$.
3 Compute $h = H(ek)$ and $(K, R) = G(m, h)$ where $K, R \in \{0,1\}^{256}$.
4 Use the Kyber-PKE encryption algorithm to encrypt m with ek, and using R to derive the random quantities needed for encryption; the resulting ciphertext is c.
5 **return** (K, c). // K is the secret key; c is the ciphertext.

Decapsulation. To recover the secret key K from c using $dk = (s, ek, H(ek), z)$, Bob does the following:

1 Use the Kyber-PKE decryption algorithm to decrypt c using decryption key s; the resulting plaintext is m'.
2 Compute $(K', R') = G(m', H(ek))$.
3 Compute $\overline{K} = J(z, c)$.
4 Use the Kyber-PKE enc. algorithm to encrypt m' with ek, and using R' to derive the random quantities needed for encryption; the resulting ciphertext is c'.
5 If $c \neq c'$ then return($\overline{K}$).
6 **return** (K').

adversary learns *some* information about the secret (s, e); specifically, the adversary has learned values r, e_1, e_2, e_u and e_v such that $\|E(x)\|_\infty \geq \lceil q/4 \rfloor$.

For the ML-KEM-768 parameter set, an adversary can make $t = 2^{64}$ queries to the decapsulation oracle and expects to observe at least one failure with probability $\varepsilon \approx 2^{-101}$. In this setting, we have $t/\varepsilon \approx 2^{165}$ which is below the target security level of 2^{192}. However, it is not known how to effectively exploit the information leaked by a single decapsulation failure to break the KEM, i.e., to distinguish between (K, C) and $(\overline{K}, C)$ where (K, C) is a valid key-ciphertext pair and $\overline{K}$ is an independently generated random key.

Some strategies have been proposed to increase the probability of decapsulation failure. For example, an adversary might repeatedly sample random m until the corresponding values r and e_1 have unusually large norms, thereby increasing the likelihood of failure. However, no such strategies are currently known to be effective in breaking Kyber-KEM, especially when the number of decapsulation queries is limited to 2^{64}, which is a realistic upper bound in practice.

Security. Kyber-KEM has been proven to be indistinguishable under chosen-ciphertext attack (Definition A.5) assuming that ss-D-MLWE is intractable, and that G, H, and J are modelled as random oracles. Kyber-KEM has also been proven indistinguishable under chosen-ciphertext attack by a quantum adversary, who is permitted to make queries to G, H and J in quantum superposition.

6.5 Notes and further references

Kyber-PKE was developed by Bos et al. [42]. The Fujisaki-Okamoto transform was introduced in 1999 [92]. A variant of it, due to Hofheinz, Hövelmanns and Kiltz [118], was used by Bos et al. [42] in the design of Kyber-KEM. For a comprehensive description of Kyber-KEM, including the specifications of the hash functions G, H, and J, the eXtendable Output Function (XOF) used to generate A, and the pseudorandom function (PRF) used to generate s, e, r, e_1 and e_2, see FIPS 203 [168]. Examples of Kyber-KEM for the three standardized parameter sets can be found at the CrypTool-Online website [65].

D'Anvers et al. [66] showed how information leaked through Kyber-KEM decapsulation failure can be exploited to formulate an MLWE instance with a secret that is smaller than the original pair (s, e), thereby making the problem easier to solve. They then applied standard cost estimates for solving MLWE to approximate the time required to recover the secret key. Their analysis indicates that the attack is ineffective when the number of decapsulation queries is limited to 2^{64}. In subsequent work, D'Anvers, Rossi and Virdia [68] proposed a technique called *directional failure boosting*, in which a previously identified failing ciphertext is leveraged to dramatically accelerate the discovery of additional ciphertexts that also fail decapsulation.

Side-channel attacks against Kyber implementations continue to be an active area of research. A comprehensive survey on side-channel attacks on Kyber implementations was presented in 2024 by Ravi, Chattopadhyay, D'Anvers and Baksi [194]. Recent notable

contributions include the works of Hermelink, Ning, Petri and Strieder [114], Bernstein et al. [30], and Fouque, Marion, Nguyen and Wallet [91].

The security of Kyber and Dilithium rests on the hardness of problems in *structured* lattices: as explained in §5, the lattices underlying the Module-LWE and Module-SIS problems possess considerable structure in that their basis matrices consist of anti-circulant blocks. FrodoKEM, the key encapsulation mechanism described in §9, takes a different approach, basing its security on the hardness of LWE in *unstructured* lattices. The absence of algebraic structure in FrodoKEM's lattices affords a more conservative security guarantee than that of Kyber, but comes at the considerable cost of larger public keys and ciphertexts, slower encapsulation, and slower decapsulation.

Most PQC deployments adopt a *hybrid* approach. In this model, traditional elliptic-curve key agreement such as elliptic-curve Diffie-Hellman (ECDH) is combined with a quantum-resistant key establishment mechanism such as Kyber-KEM. The final shared secret key is derived by combining the outputs of both schemes via a key derivation function. Hybrid constructions mitigate *harvest-now, decrypt-later* attacks by ensuring that recorded traffic remains secure even if large-scale quantum computers become available in the future. At the same time, the inclusion of a classical and well-studied primitive such as ECDH provides a hedge against the possibility that unforeseen weaknesses are later discovered in the quantum-resistant component. This dual protection, however, is lost if cryptographically relevant quantum computers become available *and* major weaknesses are discovered in the quantum-resistant scheme.

Giacon, Heuer and Poettering [101] studied key combiners for hybrid key agreement that remain secure provided that at least one of the constituent KEMs is secure. NIST's SP 800-227 [170] recommends the following generic key combiner. For $i \in \{1, 2\}$, let ek_i, c_i and K_i denote the encapsulation key, ciphertext and shared secret key for two KEMs. The combined key is then computed as $K = H(K_1, K_2, c_1, c_2, ek_1, ek_2, \text{domain_sep})$, where "domain_sep" is a string that uniquely identifies the composite KEM scheme in use. A hybrid KEM tailored to combine X25519 elliptic curve key agreement with ML-KEM-768 was proposed by Barbosa et al. [24]; this construction achieves improved efficiency compared to generic combiners. The IETF has standardized several methods for hybrid key agreement, including RFC 9954 [206], RFC 10024 [135], and RFC 10042 [125].

6.6 Exercises

Exercise 6.1 Consider an instance of Kyber-PKE with domain parameters $q = 937$, $n = 3$, $k = 3$, $\eta_1 = 3$ and $\eta_2 = 3$. Bob's encryption key is (A, t) where

$$A = \begin{bmatrix} 849 + 691x + 562x^2 & 523 + 589x + 133x^2 & 326 + 726x + 311x^2 \\ 553 + 157x + 10x^2 & 301 + 135x + 449x^2 & 777 + 905x + 49x^2 \\ 806 + 516x + 661x^2 & 540 + 37x + 438x^2 & 533 + 675x + 676x^2 \end{bmatrix}$$

and

$$t = \begin{bmatrix} 889 + 194x + 301x^2 \\ 597 + 204x + 448x^2 \\ 707 + 348x + 50x^2 \end{bmatrix},$$

and his decryption key is

$$s = \begin{bmatrix} -2 - 3x - 2x^2 \\ 2 + x \\ -3 + 3x - 2x^2 \end{bmatrix}.$$

To encrypt the plaintext $m = 011$ for Bob, Alice selects

$$r = \begin{bmatrix} 1 + 2x - x^2 \\ 1 + 3x - 2x^2 \\ -3 - 2x \end{bmatrix}, \quad e_1 = \begin{bmatrix} 2 - x + 2x^2 \\ 1 \\ -2 + 3x^2 \end{bmatrix} \quad \text{and} \quad e_2 = 1 - 3x + 3x^2.$$

(a) Verify that the ciphertext that Alice computes is $c = (u, v)$ where

$$u = \begin{bmatrix} 159 + 689x + 268x^2 \\ 264 + 639x + 589x^2 \\ 747 + 58x + 209x^2 \end{bmatrix} \quad \text{and} \quad v = 933 + 361x + 229x^2.$$

(b) Verify that decryption of c by Bob will recover m.

Exercise 6.2 Show that the Kyber public-key encryption scheme is not indistinguishable under chosen-ciphertext attack.

Exercise 6.3 Prove Theorem 6.7.

Exercise 6.4 Let c be a Kyber-PKE ciphertext generated using public key (A, t) and randomness (r, e_1, e_2). Show that the plaintext m can be efficiently determined by an adversary who knows (A, t), c, r, e_1 and e_2.

Exercise 6.5 Show that a Kyber-KEM ciphertext (with compression) has size 768, 1088, and 1568 bytes, when used with the ML-KEM-512, ML-KEM-768, and ML-KEM-1024 parameter sets, respectively.

7 Dilithium (ML-DSA)

Contents

7.1 Dilithium (toy version) 71
7.2 Dilithium (without t compression) 76
7.3 Dilithium (with t compression) 81
7.4 Notes and further references 88
7.5 Exercises 90

Dilithium is a quantum-safe signature scheme standardized by NIST in FIPS 204, where it is officially called the Module-Lattice-based Digital Signature Algorithm (ML-DSA). It is designed to provide strong security guarantees against attacks from both classical and quantum computers. In §7.1, we introduce a "toy" version of Dilithium to illustrate the main ideas underlying the scheme. We then examine several weaknesses of this simplified scheme and address them in §7.2. Finally, we incorporate public key compression in §7.3 to obtain the full version of Dilithium.

7.1 Dilithium (toy version)

The design of Dilithium is inspired by the Schnorr signature scheme, which we briefly recall next.

7.1.1 Schnorr signature scheme

Let g be a generator of a cyclic group G of prime order n, and let H be a hash function. Alice's secret signing key is an integer $a \in_R [1, n-1]$, and her corresponding public verification key is the group element $A = g^a$. Observe that recovering a from A is precisely an instance of the discrete logarithm problem in G.

To sign a message $M \in \{0,1\}^*$, Alice first selects a secret integer $y \in_R [1, n-1]$ and then computes $w = g^y$, $c = H(M\|w)$, and $z = y + ca \bmod n$. The value w is called the *commitment*, c is the *challenge*, and z is the *response*. Alice's signature on M is the pair $\sigma = (c, z)$. To verify a purported signature, Bob uses Alice's public key A to compute $w' = g^z A^{-c}$ and accepts the signature if and only if $c = H(M\|w')$.

Correctness follows by observing that

$$z \equiv y + ca \pmod{n} \;\Leftrightarrow\; g^z = g^{y+ca} \;\Leftrightarrow\; g^y = g^z (g^a)^{-c} \;\Leftrightarrow\; w = g^z A^{-c}.$$

Therefore $w' = w$, so Bob and Alice compute the same challenge value.

7.1.2 Dilithium: first attempt

In our initial description of Dilithium, we work with parameters q, n, k, ℓ, η, γ_1 and τ. Alice randomly selects a matrix $A \in_R R_q^{k\times\ell}$ whose entries are polynomials in $R_q = \mathbb{Z}_q[x]/(x^n + 1)$, together with two small polynomial vectors $s_1 \in_R S_\eta^\ell$ and $s_2 \in_R S_\eta^k$.

(Recall that S_η denotes the set of polynomials $f \in R_q$ satisfying $\|f\|_\infty \leq \eta$.) She then computes $t = As_1 + s_2 \in R_q^k$. Alice's public verification key is (A, t), and her secret signing key is (s_1, s_2). Observe that recovering the secret key from the public key corresponds to an instance of ss-MLWE, the short-secret variant of MLWE (Definition 5.17).

To sign a message $M \in \{0,1\}^*$, Alice first selects a secret vector $y \in_R S_{\gamma_1}^\ell$ and then computes the *commitment* $w = Ay \in R_q^k$. Here, γ_1 is chosen to be significantly smaller than $q/2$, so that recovering y from w corresponds to an instance of the inhomogeneous variant of MSIS (Definition 5.12). Alice then computes the *challenge* $c = H(M\|w)$, where $H : \{0,1\}^* \longrightarrow B_\tau$ is a hash function whose output is interpreted as an element of B_τ. The set B_τ comprises all polynomials in S_1 having exactly τ coefficients (mods q) equal to ± 1; note that $|B_\tau| = 2^\tau \binom{n}{\tau}$. Finally, Alice computes the *response* $z = y + cs_1 \in R_q^\ell$. Alice's signature on M is $\sigma = (c, z)$.

The polynomial vector cs_1 is designed to mask the secret y, even though both c and z are revealed. Let $\beta = \tau\eta$. The parameters γ_1, η and τ are chosen so that $\gamma_1 + \beta < q/2$. Since $c \in B_\tau$ and $s_1 \in S_\eta^\ell$, it follows that $\|cs_1\|_\infty \leq \beta$. Furthermore, $\|z\|_\infty = \|y + cs_1\|_\infty \leq \gamma_1 + \beta < q/2$.

To verify Alice's signature $\sigma = (c, z)$ on a message M, Bob, who possesses Alice's public key (A, t), must reconstruct the commitment w and then check whether $c = H(M\|w)$. Multiplying both sides of $z = y + cs_1$ by A, we obtain

$$Az = Ay + c(As_1) = w + c(t - s_2),$$

which can be rearranged as

$$Az - ct = w - cs_2. \tag{35}$$

Thus, Bob can compute $w - cs_2$, although not w itself. However, note that the coefficients of the polynomials in cs_2 are small, specifically $\|cs_2\|_\infty \leq \beta$. To overcome the difficulty of Bob being unable to compute w, we will introduce the notions of "HighBits" and "LowBits" of an integer. These operations, which will be explained in §7.2, are analogous to the "most significant bits" and "least significant bits" in the standard binary representation of an integer. Using this approach, one then defines HighBits(g) and LowBits(g) for a polynomial vector g by applying the HighBits and LowBits operations to all coefficients of all polynomials in g.

7.1.3 Dilithium: second attempt

The signing procedure is then adjusted as follows to yield our toy version of Dilithium. Alice repeatedly selects $y \in_R S_{\gamma_1}^\ell$ and computes the commitment $w_1 = \text{HighBits}(w)$ and challenge $c = H(M\|w_1)$ until the LowBits of all coefficients of the polynomials in $w - cs_2$ are "sufficiently small" (this notion will be made precise later). This ensures that adding cs_2 to $w - cs_2$ does not affect any HighBits, so that $\text{HighBits}(w - cs_2) = \text{HighBits}(w)$. She then computes the response $z = y + cs_1$. As before, her signature on M is $\sigma = (c, z)$. The verifier, Bob, is now able to recover the commitment w_1 since

$$w_1 = \text{HighBits}(w) = \text{HighBits}(w - cs_2) = \text{HighBits}(Az - ct),$$

and can therefore check that the challenge was correctly derived from M and w_1. The resulting signature scheme is presented in Algorithm 7.1. Table 7.1 lists the standardized FIPS 204 parameter sets. Their intended security levels against classical attacks are approximately 128, 192 and 256 bits.

Parameter	Description	ML-DSA-44	ML-DSA-65	ML-DSA-87
q	prime modulus	$2^{23} - 2^{13} + 1$	$2^{23} - 2^{13} + 1$	$2^{23} - 2^{13} + 1$
n	$R_q = \mathbb{Z}_q[x]/(x^n + 1)$	256	256	256
(k, ℓ)	dimensions of A	(4,4)	(6,5)	(8,7)
η	coefficient bound for s_1, s_2	2	4	2
γ_1	coefficient bound for y	2^{17}	2^{19}	2^{19}
τ	number of ± 1's in c	39	49	60
β	$\beta = \tau\eta$	78	196	120
γ_2	HighBits/LowBits param.	$(q-1)/88$	$(q-1)/32$	$(q-1)/32$
α	$\alpha = 2\gamma_2$	$(q-1)/44$	$(q-1)/16$	$(q-1)/16$
λ	collision strength of $\tilde{c}$	128	192	256
d	# of dropped bits from t	13	13	13
ω	max # of 1's in hint vector	80	55	75
Λ	security level	≈ 128	≈ 192	≈ 256

Table 7.1: FIPS 204 parameter sets for Dilithium (ML-DSA).

In Algorithm 7.1, the vector y is drawn from $\tilde{S}^{\ell}_{\gamma_1}$, the set of polynomial vectors in R^{ℓ}_q all of whose coefficients lie in the interval $(-\gamma_1, \gamma_1]$. Notice that the endpoint $-\gamma_1$ is excluded from this interval. Therefore, each coefficient admits exactly $2 \cdot \gamma_1$ possible values, which is a power of two when $\gamma_1 = 2^{17}$ or $\gamma_1 = 2^{19}$, as specified in Table 7.1. As a result, the coefficients of candidate polynomial vectors y can be generated efficiently by sampling uniformly random binary strings of length 18 or 20, respectively.

7.1.4 Five limitations and their solutions

Next, we examine five issues with the toy version of Dilithium and modifications that address them.

Issue #1: Large public keys. For the ML-DSA-87 parameters, the matrix $A \in R_q^{k \times \ell}$ has size 41,216 bytes, and the vector $t \in R_q^k$ has size 5,888 bytes, so the public key (A, t) has size 47,104 bytes. To reduce this, A is generated from a randomly selected 256-bit seed ρ, and its entries are derived from ρ using a publicly known pseudorandom bit generator called ExpandA. The updated public key, (ρ, t), has size 5,920 bytes.

FIPS 204 also specifies a method for compressing t, reducing its size from 5,888 bytes to 2,560 bytes, further reducing the public key size to 2,592 bytes. This compression method will be presented in §7.3.

Algorithm 7.1: Dilithium signature scheme (toy version)

Domain parameters. q, n, k, ℓ, η, γ_1, τ (see Table 7.1).

Key generation. Alice does the following:

1 Select $A \in_R R_q^{k \times \ell}$, $s_1 \in_R S_\eta^\ell$, and $s_2 \in_R S_\eta^k$.
2 Compute $t = As_1 + s_2$.
3 Alice's verification (public) key is (A, t); her signing (private) key is (s_1, s_2).

Signature generation. To sign a message $M \in \{0,1\}^*$, Alice does the following:

1 Found $\leftarrow$ false.
2 **while** Found = false **do**
3 Select $y \in_R \tilde{S}_{\gamma_1}^\ell$. // y is the per-message secret
4 Compute $w = Ay$ and $w_1 = \text{HighBits}(w)$. // w_1 is the commitment
5 Compute $c = H(M \| w_1)$. // $c \in B_\tau$ is the challenge
6 **if** LowBits$(w - cs_2)$ are "sufficiently small" **then**
7 Compute $z = y + cs_1$. // z is the response
8 Found $\leftarrow$ true.
9 **return** $(\sigma \leftarrow (c, z))$.

Signature verification. To verify Alice's signature $\sigma = (c, z)$ on M, Bob does:

1 Obtain an authentic copy of Alice's verification key (A, t).
2 Compute $w_1' = \text{HighBits}(Az - ct)$.
3 Accept if and only if $c = H(M \| w_1')$.

Issue #2: Large private keys. For the ML-DSA-87 parameters, the secret key $(s_1, s_2) \in S_\eta^\ell \times S_\eta^k$ has a size of 1,440 bytes. To reduce this, s_1 and s_2 are generated from a randomly selected 512-bit secret seed ρ' using a pseudorandom bit generator called ExpandS.

Issue #3: Repeated hashing of M. If the message M is very large, then the hash operation $H(M\|w_1)$ in step 5 will be the most time-consuming operation in signature generation. Moreover, this step might be executed several times. To reduce the time cost, a message representative μ is first computed before the while loop by hashing M, and then μ is used in place of M in step 5.

Issue #4: Random bits required for signing. A large number of random bits are needed to generate $y \in \tilde{S}_{\gamma_1}^\ell$ in each iteration of the signature generation process. To eliminate the need for random bits, y is generated using a secret seed ρ'' and a counter κ via a pseudorandom bit generator called ExpandMask. The seed ρ'' is derived by hashing a secret key K and the message hash μ. This approach makes the signature process *deterministic*, meaning a signature is solely dependent on the message and the signing key. Additionally, FIPS 204 allows the signer to supplement the input to ExpandMask with a 256-bit randomly generated string. If this option is chosen, then the signing process becomes *randomized*.

Issue #5: A signature (c, z) may reveal information about s_1. Since $y \in \tilde{S}_{\gamma_1}^\ell$, we have $-\gamma_1 < \text{coeff}(y) \le \gamma_1$ where $\text{coeff}(y)$ denotes the mods q representation of an arbitrary coefficient of an arbitrary polynomial in y. Also, since $c \in B_\tau$ and $s_1 \in S_\eta^\ell$, we have $-\beta \le \text{coeff}(cs_1) \le \beta$ where $\beta = \tau\eta$. Thus, since $z = y + cs_1$ and $\gamma_1 + \beta < q/2$, we have $-\gamma_1 - \beta < \text{coeff}(z) \le \gamma_1 + \beta$.

Now, if a coefficient of a polynomial in z happens to equal $\gamma_1 + \beta$, then the corresponding coefficient of cs_1 must be β. This leaks *some* information about the coefficients of the private key s_1. Similarly, if a coefficient of a polynomial in z is $\gamma_1 + \beta - 1$, the corresponding coefficient of cs_1 must be either $\beta - 1$ or β, again leaking *some* information about s_1.

More generally (as shown in Figure 7.1), if a coefficient of a polynomial in z is $\gamma_1 + a$ for some $-\beta + 1 \le a \le \beta$ (or the coefficient is $-\gamma_1 + a$ for some $-\beta < a \le \beta$), the corresponding coefficient of cs_1 must fall within the interval $[a, \beta]$ (resp. in $[-\beta, a-1]$).

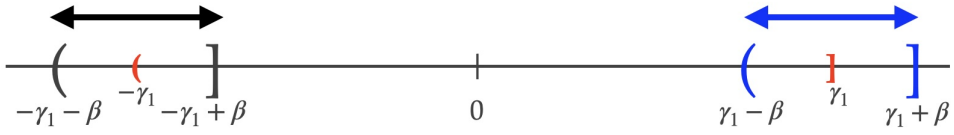


Figure 7.1: Rejection sampling in Dilithium.

To prevent this leakage of information about s_1, *rejection sampling* is employed. The signer repeatedly selects $y \in_R \tilde{S}_{\gamma_1}^\ell$ and computes $z = y + cs_1$ until $-\gamma_1 + \beta < \text{coeff}(z) \le$

$\gamma_1 - \beta$; FIPS 204 imposes the slightly stricter requirement that $\|z\|_\infty < \gamma_1 - \beta$. Rejection sampling aims to ensure that z does not reveal any information about cs_1, and thus does not leak information about s_1.

7.2 Dilithium (without t compression)

This section presents a slightly simplified variant of the standardized Dilithium signature scheme, drawing on the discussion from the previous section. We introduce the concepts of HighBits_0 and LowBits_0 for integers modulo q, which will later be refined in §7.3 to support compression of the public key component t.

7.2.1 Supporting routines

We begin by extending the symmetric mod q operation (Definition 4.11) to even moduli.

Definition 7.2 Let $\alpha \geq 2$ be an even integer, and let $x \in \mathbb{Z}_\alpha$. Define the *symmetric mod* α of x to be

$$x \bmod^s \alpha = \begin{cases} x, & \text{if } x \leq \alpha/2 \\ x - \alpha, & \text{if } x > \alpha/2. \end{cases}$$

The mods operation extends naturally to all integers: if $z \in \mathbb{Z}$ then

$$z \bmod^s \alpha = (z \bmod \alpha) \bmod^s \alpha.$$

Note that $-\alpha/2 < z \bmod^s \alpha \leq \alpha/2$.

HighBits and LowBits (preliminary). Let α be an even divisor of $q - 1$, and set $m = (q-1)/\alpha$. For integers $r \in [0, q-1]$, define

$$r_0 = r \bmod^s \alpha \quad \text{and} \quad r_1 = (r - r_0)/\alpha.$$

Then r can be written uniquely as

$$r = r_1\alpha + r_0, \text{ where } 0 \leq r_1 \leq m \text{ and } -\alpha/2 < r_0 \leq \alpha/2.$$

We define

$$\text{HighBits}_0(r, \alpha) = r_1 \text{ and } \text{LowBits}_0(r, \alpha) = r_0. \tag{36}$$

Figure 7.2 illustrates this decomposition for all $r \in [0, q-1]$ in the case $q = 21$ and $\alpha = 4$. For a polynomial vector $w \in R_q^k$, the functions $\text{HighBits}_0(w, \alpha)$ and $\text{LowBits}_0(w, \alpha)$ are obtained by applying the corresponding operations coefficientwise to every polynomial in w.

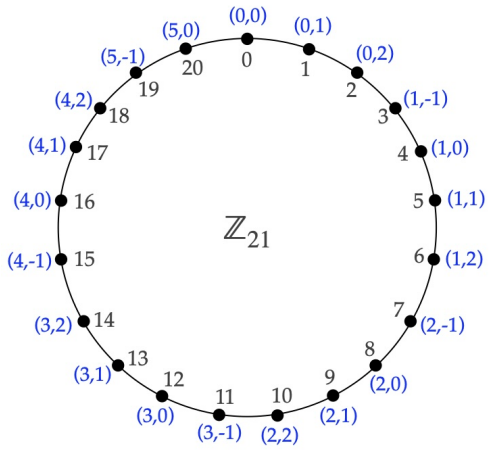


Figure 7.2: $(\text{HighBits}_0(r, 4), \text{LowBits}_0(r, 4))$ for $q = 21$ and $0 \le r \le 20$.

7.2.2 Signature scheme

Algorithm 7.3 presents a streamlined version of Dilithium that incorporates the resolutions to the five issues identified in the toy construction in §7.1. Additional implementation details, omitted from Algorithm 7.3, can be found in FIPS 204. Specifically, the pseudorandom bit generator ExpandA is instantiated using SHAKE128, while the pseudorandom bit generators ExpandS and ExpandMask are implemented using SHAKE256. The variable-output length hash function H, which maps input $m \in \{0,1\}^*$ and a length parameter $d \ge 1$ to an output $H(m, d) \in \{0,1\}^d$, is also instantiated using either SHAKE128 or SHAKE256. Finally, the function SampleInBall maps bitstrings in $\{0,1\}^{2\lambda}$ to polynomials in B_τ.

Signature verification works. Recall (equation 35) that $Az - ct = w - cs_2$. Since $\|\text{LowBits}_0(w - cs_2,\ 2\gamma_2)\|_\infty < \gamma_2 - \beta$ and $\|cs_2\|_\infty \le \beta$, we have

$$w_1' = \text{HighBits}_0(Az - ct,\ 2\gamma_2) = \text{HighBits}_0(w - cs_2,\ 2\gamma_2) = \text{HighBits}_0(w,\ 2\gamma_2) = w_1.$$

Thus, $H(\mu \| w_1',\ 2\lambda) = H(\mu \| w_1,\ 2\lambda)$ and the signature will be accepted.

Example 7.4 (*toy example of Dilithium signature scheme per Algorithm 7.3*)
Domain parameters. $q = 98993$ and $n = 4$ (so $R_q = \mathbb{Z}_{98993}[x]/(x^4 + 1)$), $k = 5$, $\ell = 4$, $\eta = 15$, $\gamma_1 = 2^{12} = 4096$, $\gamma_2 = (q-1)/16 = 6187$, $\tau = 3$, $\beta = 45$.

Algorithm 7.3: Dilithium signature scheme (without t compression)

Domain parameters. q, n, k, ℓ, η, γ_1, γ_2, τ, β, λ (see Table 7.1).

Key generation. Alice does the following:

Select $\xi \in_R \{0,1\}^{256}$.
Compute $(\rho, \rho', K) = H(\xi, 1024)$, where $\rho \in \{0,1\}^{256}$, $\rho' \in \{0,1\}^{512}$, $K \in \{0,1\}^{256}$.
Compute $A = \text{ExpandA}(\rho) \in R_q^{k \times \ell}$. // ExpandA is a PRBG
Compute $(s_1, s_2) = \text{ExpandS}(\rho') \in S_\eta^\ell \times S_\eta^k$. // ExpandS is a PRBG
Compute $t = As_1 + s_2$.
Compute $tr = H(\rho \| t,\ 512)$.
Alice's verification (public) key is (ρ, t);
her signing (private) key is (ρ, K, tr, s_1, s_2).

Signature generation. To sign a message $M \in \{0,1\}^*$, Alice does the following:

Compute $A = \text{ExpandA}(\rho)$.
Compute $\mu = H(tr \| M,\ 512)$. // μ is the message representative
Compute $\rho'' = H(K \| \text{rnd} \| \mu,\ 512)$ where either $\text{rnd} = 0^{256}$ or $\text{rnd} \in_R \{0,1\}^{256}$.
Set $\kappa = 0$ and Found = false.
while Found = false **do**
    Compute $y = \text{ExpandMask}(\rho'', \kappa) \in \tilde{S}_{\gamma_1}^\ell$. // y is the per-message secret
    Compute $w = Ay$ and $w_1 = \text{HighBits}_0(w, 2\gamma_2)$.
    // w_1 is the commitment
    Compute $\tilde{c} = H(\mu \| w_1, 2\lambda)$ and $c = \text{SampleInBall}(\tilde{c})$. // c is the challenge
    Compute $r_0 = \text{LowBits}_0(w - cs_2, 2\gamma_2)$.
    if $\|r_0\|_\infty < \gamma_2 - \beta$ **then**
        Compute $z = y + cs_1$. // z is the response
        if $\|z\|_\infty < \gamma_1 - \beta$ **then**
            Found $\leftarrow$ true.
    $\kappa \leftarrow \kappa + \ell$ // increment the counter κ
return $(\sigma \leftarrow (\tilde{c}, z))$.

Signature verification. To verify Alice's signature $\sigma = (\tilde{c}, z)$ on M, Bob does:

Obtain an authentic copy of Alice's verification key (ρ, t).
if $\|z\|_\infty \geq \gamma_1 - \beta$ **then**
    Reject the signature.
Compute $A = \text{ExpandA}(\rho)$.
Compute $tr = H(\rho \| t, 512)$ and $\mu = H(tr \| M, 512)$.
Compute $c = \text{SampleInBall}(\tilde{c})$.
Compute $w_1' = \text{HighBits}_0(Az - ct, 2\gamma_2)$.
Check that $\tilde{c} = H(\mu \| w_1', 2\lambda)$; if not then reject the signature.
Accept the signature.

Key generation. Alice generates the following matrix[10] $A \in R_q^{5\times 4}$:

$$\left[\begin{array}{cc}
83831 + 86182x + 34460x^2 + 53827x^3 & 52753 + 4611x + 33768x^2 + 68049x^3 \\
26611 + 31871x + 51791x^2 + 53135x^3 & 77983 + 73743x + 73630x^2 + 76856x^3 \\
\hline
84109 + 26539x + 57487x^2 + 49250x^3 & 78952 + 68437x + 21583x^2 + 35337x^3 \\
7915 + 41093x + 54045x^2 + 86724x^3 & 95963 + 98131x + 25363x^2 + 57139x^3 \\
\hline
95705 + 84707x + 74757x^2 + 85481x^3 & 8110 + 6018x + 86989x^2 + 84426x^3 \\
38849 + 66776x + 12868x^2 + 40821x^3 & 42661 + 14351x + 34674x^2 + 64475x^3 \\
\hline
52599 + 20391x + 59361x^2 + 18240x^3 & 30833 + 51209x + 68814x^2 + 96318x^3 \\
51956 + 43227x + 14013x^2 + 88049x^3 & 19027 + 33104x + 17796x^2 + 44727x^3 \\
\hline
34258 + 93745x + 15427x^2 + 3945x^3 & 567 + 74669x + 28186x^2 + 94530x^3 \\
42638 + 83911x + 12415x^2 + 44545x^3 & 40249 + 60948x + 57689x^2 + 70091x^3
\end{array}\right].$$

Alice then selects the signing key components

$$s_1 = \begin{bmatrix} 3 + 13x - 6x^2 - 2x^3 \\ -10 + 7x \\ -2 + 5x - 13x^2 + 9x^3 \\ -7 + x^2 \end{bmatrix} \in S_{15}^4, \quad s_2 = \begin{bmatrix} 14 - 13x + 12x^2 + 3x^3 \\ 5 - 2x - 14x^2 - 14x^3 \\ 10 + 13x - 3x^2 - 11x^3 \\ -14 - 13x + 10x^2 + 11x^3 \\ 14 + 14x + 9x^2 + 10x^3 \end{bmatrix} \in S_{15}^5,$$

and computes the verification key component

$$t = As_1 + s_2 = \begin{bmatrix} 57924 + 81390x + 10162x^2 + 83131x^3 \\ 40264 + 19231x + 1880x^2 + 93259x^3 \\ 52802 + 92079x + 23039x^2 + 13512x^3 \\ 16927 + 8353x + 32968x^2 + 33005x^3 \\ 6826 + 54397x + 87113x^2 + 45439x^3 \end{bmatrix} \in R_q^5.$$

Here, the coefficients of polynomials in s_1 and s_2 are expressed in their mods q representation. Alice's verification key is (A, t); her signing key is (s_1, s_2).

Signature generation. To sign a message M, Alice generates

$$y \bmod q = \begin{bmatrix} 915 + 3143x - 1424x^2 + 893x^3 \\ 2 - 819x + 2904x^2 - 805x^3 \\ -797 - 3770x + 1184x^2 - 1217x^3 \\ 3051 + 1878x + 1447x^2 + 1773x^3 \end{bmatrix} \in \tilde{S}_{\gamma_1}^4$$

and computes

$$w = Ay = \begin{bmatrix} 17575 + 82813x + 17567x^2 + 46483x^3 \\ 75506 + 5076x + 88490x^2 + 21861x^3 \\ 23112 + 54839x + 57476x^2 + 21701x^3 \\ 84380 + 57159x + 73769x^2 + 8929x^3 \\ 27202 + 41260x + 96026x^2 + 74081x^3 \end{bmatrix}$$

[10] Due to space constraints, each row of A spans two lines.

and the commitment

$$w_1 = \text{HighBits}_0(w, 2\gamma_2) = \begin{bmatrix} 1 + 7x + x^2 + 4x^3 \\ 6 + 7x^2 + 2x^3 \\ 2 + 4x + 5x^2 + 2x^3 \\ 7 + 5x + 6x^2 + x^3 \\ 2 + 3x + 8x^2 + 6x^3 \end{bmatrix}.$$

Alice computes $\tilde{c} = H(\mu \| w_1)$ and $c = \text{SampleInBall}(\tilde{c})$ (details omitted), obtaining

$$c \bmod q = -x + x^2 - x^3 \in B_3.$$

She computes

$$w - cs_2 = \begin{bmatrix} 17597 + 82818x + 17537x^2 + 46522x^3 \\ 75508 + 5081x + 88497x^2 + 21854x^3 \\ 23107 + 54841x + 57490x^2 + 21695x^3 \\ 84392 + 57146x + 73759x^2 + 8938x^3 \\ 27187 + 41275x + 96016x^2 + 74090x^3 \end{bmatrix}$$

and

$$r_0 = \text{LowBits}_0(w - cs_2, 2\gamma_2) = \begin{bmatrix} 5223 - 3800x + 5163x^2 - 2974x^3 \\ 1264 + 5081x + 1879x^2 - 2894x^3 \\ -1641 + 5345x - 4380x^2 - 3053x^3 \\ -2226 - 4724x - 485x^2 - 3436x^3 \\ 2439 + 4153x - 2976x^2 - 154x^3 \end{bmatrix}.$$

Note that $\|r_0\|_\infty < \gamma_2 - \beta = 6142$.

Alice then computes the response

$$z = y + cs_1 = \begin{bmatrix} 932 + 3136x + 97557x^2 + 909x^3 \\ 9 + 98184x + 2887x^2 + 98205x^3 \\ 98223 + 95203x + 1186x^2 + 97796x^3 \\ 3050 + 1886x + 1440x^2 + 1779x^3 \end{bmatrix}.$$

Using mods q representation for coefficients, we have

$$z \bmod q = \begin{bmatrix} 932 + 3136x - 1436x^2 + 909x^3 \\ 9 - 809x + 2887x^2 - 788x^3 \\ -770 - 3790x + 1186x^2 - 1197x^3 \\ 3050 + 1886x + 1440x^2 + 1779x^3 \end{bmatrix}.$$

Note that $\|z\|_\infty < \gamma_1 - \beta = 4051$. Alice's signature on M is $\sigma = (\tilde{c}, z)$.

Signature verification. To verify Alice's signature $\sigma = (\tilde{c}, z)$ on M, Bob computes $c = \text{SampleInBall}(\tilde{c})$ obtaining $c \bmod q = -x + x^2 - x^3 \in B_3$,

$$Az - ct = \begin{bmatrix} 17597 + 82818x + 17537x^2 + 46522x^3 \\ 75508 + 5081x + 88497x^2 + 21854x^3 \\ 23107 + 54841x + 57490x^2 + 21695x^3 \\ 84392 + 57146x + 73759x^2 + 8938x^3 \\ 27187 + 41275x + 96016x^2 + 74090x^3 \end{bmatrix},$$

and

$$w_1' = \text{HighBits}_0(Az - ct, 2\gamma_2) = \begin{bmatrix} 1 + 7x + x^2 + 4x^3 \\ 6 + 7x^2 + 2x^3 \\ 2 + 4x + 5x^2 + 2x^3 \\ 7 + 5x + 6x^2 + x^3 \\ 2 + 3x + 8x^2 + 6x^3 \end{bmatrix}.$$

Since $w_1' = w_1$, we have $H(\mu \| w_1', 2\lambda) = H(\mu \| w_1, 2\lambda)$ so Bob accepts the signature.

7.2.3 Security

If the short-secret D-MLWE problem (Definition 5.18) is intractable, an adversary cannot derive any information about the secret key component s_1 (or s_2) from the public key (ρ, t). Evaluating the difficulty of signature forgery is more complex. To simplify the analysis, we will make a few simplifications to the signature scheme and provide an informal argument that a forger with access to the public key (ρ, t) cannot produce a valid signed message.

The forger's task is as follows: given the verification key (ρ, t), find a message M and signature $(\tilde{c}, z)$ such that $\|z\|_\infty < \gamma_1 - \beta$ and $\tilde{c} = H(\mu \| w_1)$, where $\mu = H(M)$, $w_1 = \text{HighBits}_0(Az - ct,\ 2\gamma_2)$, and $c = \text{SampleInBall}(\tilde{c})$. It is important to note that c depends on $\tilde{c}$, $\tilde{c}$ depends on w_1, and w_1 depends on c, creating a circular dependency. Assuming that H has no weaknesses, it would appear that the forger's best strategy is to randomly select M and $w_1 \in R_q^k$, where each coefficient in w_1 is in the range $[0, (q-1)/(2\gamma_2)]$, and then compute $\mu = H(M)$, $\tilde{c} = H(\mu \| w_1)$, and $c = \text{SampleInBall}(\tilde{c})$. The remaining task is to find a z that satisfies the signature conditions.

Now, since $w_1 = \text{HighBits}_0(Az - ct,\ 2\gamma_2)$, we can write

$$Az - ct = 2\gamma_2 w_1 + w_0$$

where $w_0 \in R_q^k$ with $-\gamma_2 < \text{coeff}(w_0) \leq \gamma_2$. Rearranging terms gives

$$Az - w_0 = ct + 2\gamma_2 w_1.$$

Thus, the forger's task is to find a solution (z, w_0) to the linear system of equations

$$\begin{bmatrix} A \mid I_k \end{bmatrix} \begin{bmatrix} z \\ -w_0 \end{bmatrix} = ct + 2\gamma_2 w_1, \tag{37}$$

where $z \in R_q^\ell$ and $w_0 \in R_q^k$ are relatively small (more precisely, $\|z\|_\infty < \gamma_1 - \beta$ and $-\gamma_2 < \text{coeff}(w_0) \leq \gamma_2$). This is an instance of the normal-form version of Inhomogeneous MSIS (Definition 5.13).

7.3 Dilithium (with t compression)

The FIPS 204 standardized Dilithium signature scheme incorporates a compression technique for the public key component t. This requires adjusting the definitions of HighBits_0

and LowBits_0 and introducing the use of "hint bits", which in turn results in a modest increase in signature size. For the ML-DSA-44, ML-DSA-65 and ML-DSA-87 parameter sets, Dilithium public keys have sizes of 1312, 1952 and 2592 bytes, respectively, while the corresponding signature sizes are 2420, 3309 and 4627 bytes.

7.3.1 Supporting routines

We now describe five auxiliary routines used in the full Dilithium signature scheme: Power2Round, HighBits, LowBits, MakeHint, and UseHint.

Power2Round. Let d be an integer satisfying $1 \le d \le \log_2 q$. For integers $r \in [0, q-1]$, define

$$r_0 = r \bmod 2^d \quad \text{and} \quad r_1 = (r - r_0)/2^d.$$

Then r can be expressed as

$$r = r_1 2^d + r_0, \text{ where } -2^{d-1} < r_0 \le 2^{d-1} \text{ and } 0 \le r_1 \le \lceil (q-1)/2^d \rceil.$$

The function Power2Round[11] decomposes r into a high-order component r_1 and a low-order component r_0. Figure 7.3 illustrates this decomposition for the case $q = 23$ and $d = 3$. The Power2Round function extends naturally to polynomial vectors $t \in R_q^k$

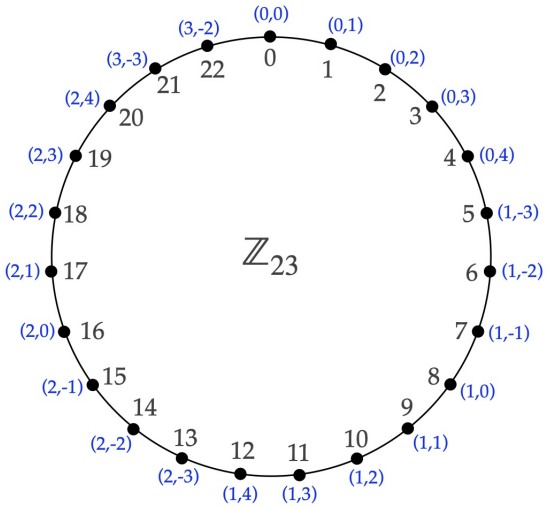


Figure 7.3: $\text{Power2Round}(r, d) = (r_1, r_0)$ for $q = 23$, $d = 3$, and $0 \le r \le 22$.

by applying the decomposition coefficientwise to each polynomial in t. For example, if $t = 5 + 2x + 19x^2 + 8x^3 \in R_q = \mathbb{Z}_{23}[x]/(x^4 + 1)$, then $\text{Power2Round}(t, 3) = (t_1, t_0)$ where $t_1 = 1 + 2x^2 + x^3$ and $t_0 = -3 + 2x + 3x^2$.

[11] Power2Round is similar to $\text{HighBits}_0/\text{LowBits}_0$, except that the modulus is required to be a power of two and does not necessarily divide $q - 1$.

t compression. Recall that the Dilithium verification key includes a polynomial vector $t \in R_q^k$. The size of this key can be reduced by discarding the d low-order bits of each coefficient of t. Concretely, Power2Round is used to write $t = t_1 2^d + t_0$. The compressed component t_1 is included in the verification key, while t_0 is retained as part of the signing key.

For the ML-DSA-87 parameter set, we have $k = 8$ and $d = 13$. Since each coefficient in R_q requires 23 bits, the full vector t has size $8 \times 256 \times 23$ bits, or 5,888 bytes. By contrast, the coefficients of t_1 have size 10 bits, giving a total size of $8 \times 256 \times 10$ bits, or 2,560 bytes. Consequently, the public key (ρ, t_1) has total size 2,592 bytes.

However, omitting t_0 from the verification key introduces an additional complication in verification. Recall that the verifier needs to compute $Az - ct$ in order to recover $\mathrm{HighBits}_0(w - cs_2, 2\gamma_2)$. Since only t_1 is available, the verifier instead computes

$$Az - ct_1 2^d = Az - c(t - t_0) = Az - ct + ct_0 = w - cs_2 + ct_0. \tag{38}$$

The coefficients of ct_0 are expected to be relatively small. In particular, we expect that $\|ct_0\|_\infty \leq \gamma_2$ with very high probability. When this condition holds, adding $-ct_0$ to $w - cs_2 + ct_0$ changes the corresponding $\mathrm{HighBits}_0$ by at most ± 1. Consequently, although the verifier cannot directly compute $\mathrm{HighBits}_0(w - cs_2, 2\gamma_2)$, they can still recover it from $w - cs_2 + ct_0$ provided that suitable auxiliary information is supplied.

Hint bits. To enable recovery of $\mathrm{HighBits}_0(w - cs_2, 2\gamma_2)$, the signer appends a small amount of auxiliary information to the signature in the form of "hint bits". Intuitively, these bits encode the "carry information" induced when the correction term $-ct_0$ is added to $w - cs_2 + ct_0$.

Let $r \in [0, q-1]$ with $r_1 = \mathrm{HighBits}_0(r, \alpha)$ and $r_0 = \mathrm{LowBits}_0(r, \alpha)$, and let $z \in [-\gamma_2, \gamma_2]$. A hint bit $h \in \{0, 1\}$ is intended to allow the computation of $\mathrm{HighBits}_0(r + z, \alpha)$ from r and h, without knowledge of z. In this setting, z models a coefficient of $-ct_0$, while r corresponds to a coefficient of $w - cs_2 + ct_0$.

A difficulty in defining such a hint bit arises from the fact that only two values, 0 and 1, are available, whereas there are three possible carry digits, namely -1, 0 and 1. To illustrate this issue, consider the case where $q = 21$, $\alpha = 4$, $\gamma_2 = 2$ and $m = 5$. For all $r \in [0, 18]$ except $r = 1$, one observes that $\mathrm{HighBits}_0(r, 4)$ changes when either $z = 2$ or $z = -2$ is added to r, but not both; see Figure 7.4. For instance, when $z = 2$ is added to $r = 6$ then $\mathrm{HighBits}_0(r, 4)$ is increased from 1 to 2, whereas if $z = -2$ is added to $r = 6$ then $\mathrm{HighBits}_0(r, 4)$ is unchanged. However, this behaviour does not hold for $r = 1$, $r = 19$ and $r = 20$. For example, if $z = 2$ is added to $r = 20$, then $\mathrm{HighBits}_0(r, 4)$ changes from 5 to 0, and adding $z = -2$ changes it from 5 to 4.

The underlying issue is that the multiples 20 and 0 of $\alpha = 4$ are side-by-side. More generally, the multiples $q - 1 = m \cdot \alpha$ and $0 = 0 \cdot \alpha$ of α lie next to each other, and the exceptional values of r are precisely those lying close to 0 or $q - 1$. To resolve this, the definitions of $\mathrm{HighBits}_0$ and $\mathrm{LowBits}_0$ are slightly adjusted.

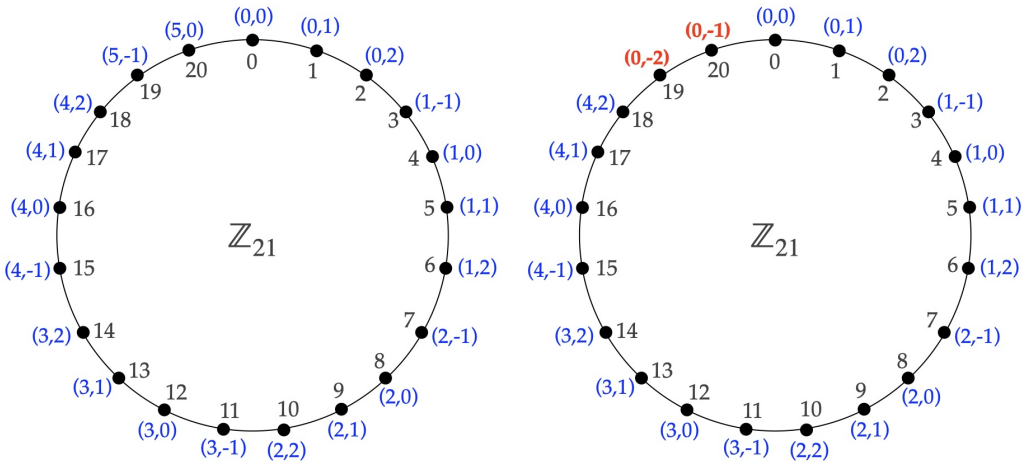


Figure 7.4: The original (left) and adjusted (right) definitions of HighBits(r, α) and LowBits(r, α) for $(q, \alpha) = (21, 4)$ and $0 \le r \le 20$.

HighBits and LowBits (adjusted). Returning to the example with $q = 21$ and $\alpha = 4$, we redefine (HighBits$(20, 4)$, LowBits$(20, 4)$) from $(5, 0)$ to $(0, -1)$, noting that $20 \equiv 0 \cdot \alpha - 1 \pmod{21}$. Similarly, we redefine (HighBits$(19, 4)$, LowBits$(19, 4)$) from $(5, -1)$ to $(0, -2)$, since $19 \equiv 0 \cdot \alpha - 2 \pmod{21}$. See Figure 7.4. With this adjustment, every $r \in [0, 20]$ satisfies the desired property: at most one of the two additions $z = 2$ and $z = -2$ changes the value of HighBits$(r, 4)$.

We now present the modified definitions of HighBits and LowBits.

Definition 7.5 Let α be an even divisor of $q - 1$, and let $r \in [0, q - 1]$. Let $s = r \bmod \alpha$. Define

$$(r_1, r_0) = \begin{cases} (0, s - 1), & \text{if } r - s = q - 1, \\ ((r - s)/\alpha, s), & \text{otherwise.} \end{cases}$$

Then HighBits$(r, \alpha) = r_1$ and LowBits$(r, \alpha) = r_0$.

MakeHint and UseHint. With this convention in place, a single hint bit h suffices to indicate unambiguously whether HighBits remained unchanged ($h = 0$) or whether it increases or decreases by one modulo m ($h = 1$).

Definition 7.6 Let α be an even divisor of $q - 1$. For $r \in [0, q - 1]$ and $z \in [-\alpha/2, \alpha/2]$, define the *hint bit*

$$\text{MakeHint}(z, r, \alpha) = \begin{cases} 1, & \text{if HighBits}(r + z, \alpha) \neq \text{HighBits}(r, \alpha), \\ 0, & \text{otherwise.} \end{cases}$$

The function UseHint is used to recover HighBits$(r + z, \alpha)$ from r and h.

Definition 7.7 Let α be an even divisor of $q-1$, and let $m = (q-1)/\alpha$. Let $r \in [0, q-1]$ and $h \in \{0,1\}$, and let $r_1 = \text{HighBits}(r, \alpha)$ and $r_0 = \text{LowBits}(r, \alpha)$. Define

$$\text{UseHint}(h, r, \alpha) = \begin{cases} r_1, & \text{if } h = 0, \\ r_1 + 1 \bmod m, & \text{if } h = 1 \text{ and } r_0 > 0, \\ r_1 - 1 \bmod m, & \text{if } h = 1 \text{ and } r_0 \le 0. \end{cases}$$

The following result shows that the functions MakeHint and UseHint operate as intended.

Lemma 7.8 Let α be an even divisor of $q-1$, and let $r \in [0, q-1]$ and $z \in [-\alpha/2, \alpha/2]$. Then

$$\text{UseHint}(\text{MakeHint}(z, r, \alpha), r, \alpha) = \text{HighBits}(r + z, \alpha).$$

The operations MakeHint and UseHint extend naturally to polynomial vectors in R_q^k.

7.3.2 Full signature scheme

Algorithm 7.9 presents the full version of the Dilithium signature scheme with t compression. The signer first checks[12] that $\|ct_0\|_\infty < \gamma_2$ and then constructs a *hint vector*

$$h = \text{MakeHint}(-ct_0,\ w - cs_2 + ct_0,\ 2\gamma_2),$$

which is appended to the signature. Since $-ct_0 \in R_q^k$, we have $h = (h_1, h_2, \ldots, h_k)$ where each $h_i \in \{0,1\}^{256}$. To further control the signature size, an additional constraint is imposed requiring that the total number of one bits in h does not exceed ω. The verifier applies the hint vector h to compute

$$w_1' = \text{UseHint}(h,\ Az - ct_1 2^d,\ 2\gamma_2).$$

Signature verification works. Recall (equation 38) that $Az - ct_1 2^d = w - cs_2 + ct_0$. Since $\|-ct_0\|_\infty < \gamma_2$, we have

$$\begin{aligned} w_1' &= \text{UseHint}(h, Az - ct_1 2^d,\ 2\gamma_2) = \text{HighBits}(Az - ct_1 2^d - ct_0,\ 2\gamma_2) \\ &= \text{HighBits}(Az - ct,\ 2\gamma_2) = \text{HighBits}(w - cs_2,\ 2\gamma_2). \end{aligned}$$

And, since $\|\text{LowBits}(w - cs_2,\ 2\gamma_2)\|_\infty < \gamma_2 - \beta$ and $\|cs_2\|_\infty \le \beta$, we have

$$w_1' = \text{HighBits}(w - cs_2,\ 2\gamma_2) = \text{HighBits}(w,\ 2\gamma_2) = w_1.$$

Thus, $H(\mu \| w_1',\ 2\lambda) = H(\mu \| w_1,\ 2\lambda)$ and the signature will be accepted.

[12] FIPS 204 specifies the condition $\|ct_0\|_\infty < \gamma_2$; alternatively, the check could have been defined as $\|ct_0\|_\infty \le \gamma_2$.

Algorithm 7.9: Dilithium signature scheme (with t compression)

Domain parameters. q, n, k, ℓ, η, γ_1, γ_2, τ, β, λ, d, ω (see Table 7.1).

Key generation. Alice does the following:

1 Select $\xi \in_R \{0,1\}^{256}$.
2 Compute $(\rho, \rho', K) = H(\xi, 1024)$, where $\rho \in \{0,1\}^{256}$, $\rho' \in \{0,1\}^{512}$, $K \in \{0,1\}^{256}$.
3 Compute $A = \text{ExpandA}(\rho) \in R_q^{k \times \ell}$.
4 Compute $(s_1, s_2) = \text{ExpandS}(\rho') \in S_\eta^\ell \times S_\eta^k$.
5 Compute $t = As_1 + s_2$ and $(t_1, t_0) = \text{Power2Round}(t, d)$.
6 Compute $tr = H(\rho \| t_1,\ 512)$.
7 Alice's verification (public) key is (ρ, t_1);
her signing (private) key is $(\rho, K, tr, s_1, s_2, t_0)$.

Signature generation. To sign a message $M \in \{0,1\}^*$, Alice does the following:

1 Compute $A = \text{ExpandA}(\rho)$.
2 Compute $\mu = H(tr \| M,\ 512)$.
3 Compute $\rho'' = H(K \| \text{rnd} \| \mu,\ 512)$ where either $\text{rnd} = 0^{256}$ or $\text{rnd} \in_R \{0,1\}^{256}$.
4 Set $\kappa = 0$ and Found = false.
5 **while** Found = false **do**
6 Compute $y = \text{ExpandMask}(\rho'', \kappa) \in \tilde{S}_{\gamma_1}^\ell$.
7 Compute $w = Ay$ and $w_1 = \text{HighBits}(w, 2\gamma_2)$.
8 Compute $\tilde{c} = H(\mu \| w_1, 2\lambda)$ and $c = \text{SampleInBall}(\tilde{c})$.
9 Compute $r_0 = \text{LowBits}(w - cs_2, 2\gamma_2)$ and $s_0 = ct_0$.
10 **if** $\|r_0\|_\infty < \gamma_2 - \beta$ and $\|s_0\|_\infty < \gamma_2$ **then**
11 Compute $h = \text{MakeHint}(-s_0,\ w - cs_2 + s_0,\ 2\gamma_2)$.
12 **if** the number of 1's in h is $\le \omega$ **then**
13 Compute $z = y + cs_1$.
14 **if** $\|z\|_\infty < \gamma_1 - \beta$ **then**
15 Found $\leftarrow$ true.
16 $\kappa \leftarrow \kappa + \ell$ (increment the counter κ).
17 **return** $(\sigma \leftarrow (\tilde{c}, z, h))$.

Signature verification. To verify Alice's signature $\sigma = (\tilde{c}, z, h)$ on M, Bob does:

1 Obtain an authentic copy of Alice's verification key (ρ, t_1).
2 **if** $\|z\|_\infty \ge \gamma_1 - \beta$ or the number of 1's in h is $> \omega$ **then**
3 Reject the signature.
4 Compute $A = \text{ExpandA}(\rho)$.
5 Compute $tr = H(\rho \| t_1, 512)$ and $\mu = H(tr \| M, 512)$.
6 Compute $c = \text{SampleInBall}(\tilde{c})$.
7 Compute $w_1' = \text{UseHint}(h, Az - ct_1 2^d, 2\gamma_2)$.
8 Check that $\tilde{c} = H(\mu \| w_1', 2\lambda)$; if not then reject the signature.
9 Accept the signature.

7.3.3 Number of iterations in signature generation

The main loop in signature generation terminates when the following four conditions are simultaneously satisfied:

1. $\|z\|_\infty < \gamma_1 - \beta$.
2. $\|r_0\|_\infty < \gamma_2 - \beta$.
3. $\|ct_0\|_\infty < \gamma_2$.
4. The total number of 1's in the hint vector h is at most ω.

We next estimate the probability of each event.

1. Let
$$p_1 = \Pr(\|z\|_\infty < \gamma_1 - \beta). \tag{39}$$
Recall that $z = y + cs_1 \in R_q^\ell$ where $y \in_R \tilde{S}_{\gamma_1}^\ell$, $c \in B_\tau$, and $s_1 \in_R S_\eta^\ell$. Suppose that a coefficient of cs_1 equals $u \in [-\beta, \beta]$. Then the corresponding coefficient of z lies in the interval $(-\gamma_1 + \beta,\ \gamma_1 - \beta)$ provided that the corresponding coefficient of y lies in $(-\gamma_1 + \beta - u,\ \gamma_1 - \beta - u)$. This interval contains $2(\gamma_1 - \beta) - 1$ valid integer values, independent of the choice of u. Hence,
$$p_1 = \left(\frac{2(\gamma_1 - \beta) - 1}{2\gamma_1}\right)^{256\ell} = \left(1 - \frac{\beta + 1/2}{\gamma_1}\right)^{256\ell} \approx e^{-256\ell\beta/\gamma_1}. \tag{40}$$

2. Let
$$p_2 = \Pr(\|r_0\|_\infty < \gamma_2 - \beta). \tag{41}$$
Assuming that $\text{LowBits}(w - cs_2,\ 2\gamma_2)$ is uniformly distributed in $[-\gamma_2, \gamma_2]^{256k}$, we obtain
$$p_2 = \left(\frac{2(\gamma_2 - \beta) - 1}{2\gamma_2 + 1}\right)^{256k} \approx \left(1 - \frac{\beta + 1/2}{\gamma_2}\right)^{256k} \approx e^{-256k\beta/\gamma_2}. \tag{42}$$

3. Let
$$p_3 = \Pr(\|ct_0\|_\infty < \gamma_2 \text{ and the total number of 1's in } h \text{ is at most } \omega). \tag{43}$$
Experimental evidence indicates that $p_3 > 0.98$ for the standardized parameter sets.

Assuming that these events are independent, the probability that all four requirements are satisfied in a single iteration of the main loop is $p_1 p_2 p_3$. Thus, the expected number of iterations required for signature generation is
$$\frac{1}{p_1 p_2 p_3} \approx \frac{1}{p_1 p_2} \approx e^{256\beta(\ell/\gamma_1 + k/\gamma_2)}.$$
For the standardized parameter sets ML-DSA-44, ML-DSA-65, and ML-DSA-87, the expected number of iterations is approximately 4.25, 5.1 and 3.85, respectively.

7.3.4 Security

Compressing the public key component t has no impact on the security of Dilithium. In particular, the full scheme has been proven existentially unforgeable under chosen-message attacks (Definition A.7), assuming the hardness of the ss-D-MLWE and MSIS problems, and modelling H as a random oracle.

Remark 7.10 (*selecting Dilithium parameters*) The ML-DSA-44, ML-DSA-65 and ML-DSA-87 parameter sets specified in FIPS 204 (see Table 7.1) were carefully engineered to optimize the trade-offs between security, public key size, signature size, implementation efficiency, and the expected number of iterations during signature generation. The following notes illustrate the delicate balance required when selecting the parameters q, n, k, ℓ, η, γ_1, γ_2, τ, β, d and ω.

1. To facilitate fast polynomial multiplication in R_q via the Number-Theoretic Transform (see §13.3), the degree n was chosen to be a power of two, and the prime q was chosen to satisfy the condition $q \equiv 1 \pmod{2n}$.

2. The dimensions k and ℓ can be increased to achieve higher security levels against MLWE and MSIS attacks without altering the underlying polynomial ring R_q. However, increasing k makes the public key larger (since $t_1 \in R_q^k$), while increasing ℓ makes the signature larger (since $z \in R_q^\ell$).

3. Minimizing γ_1 and γ_2 is desirable to maximize resistance to MSIS attacks; cf. equation (37). Conversely, larger values for γ_1 and γ_2 are preferred because they increase the acceptance probabilities p_1, p_2 and p_3 (see equations (40), (42) and (43), thereby reducing the expected number of iterations during signature generation.

4. A larger secret bound η enhances resistance against MLWE attacks, but it consequently increases β (since $\beta = \tau\eta$), which lowers the acceptance probabilities p_1 and p_2.

5. While a smaller value of τ is desirable because it reduces β, τ must remain sufficiently large to ensure that the size $2^\tau \binom{256}{\tau}$ of the challenge space B_τ provides adequate security.

6. Increasing the compression parameter d reduces the size of the public key component t_1, and decreasing ω reduces the size of the hint vector h. However, choosing a large d or a small ω reduces the acceptance probability p_3 (see equation (43)), which increases the expected number of iterations.

7.4 Notes and further references

Dilithium's design follows the Fiat-Shamir methodology [85], which is a generic method for converting a three-move interactive identification scheme into a non-interactive signature scheme by hashing the message and the prover's commitment together to generate the challenge. Dilithium was designed by Ducas, Kiltz, Lepoint, Lyubashevsky,

Schwabe and Seiler [75] and draws on ideas from several earlier works, most notably the papers of Lyubashevsky [151, 152] and Bai and Galbraith [20]. It was heavily optimized to reduce public-key and signature sizes, while ensuring fast and constant-time implementations. For a full description of Dilithium, including specifications of ExpandA, ExpandS, ExpandMask, H, and SampleInBall, see FIPS 204 [167]. Formal security proofs for Dilithium follow a reductionist approach, building on the "Fiat-Shamir with aborts paradigm" originally proposed by Lyubashevsky [151]. These proofs are notably complex; detailed analyses were presented by Barbosa et al. [23] and Devevey et al. [72].

Azevedo-Oliveira, Calle Viera, Cogliati and Goubin [18] developed a practical linear-programming method that recovers the private key component t_0 from a Dilithium public key (ρ, t_1) together with a collection of signed messages. Approximately 200,000 signatures were needed for the ML-DSA-44 parameter set, while about 500,000 signatures were required for the ML-DSA-65 and ML-DSA-87 parameter sets. These results support the statement in the FIPS 204 document that "The low-order bits of t can be reconstructed from a small number of signatures and, therefore, need not be regarded as secret."

The vulnerability of Dilithium implementations to side-channel attacks remains an active area of research. A comprehensive survey of such attacks was presented in 2024 by Ravi, Chattopadhyay, D'Anvers and Baksi [194]. Recent notable works include attacks exploiting potential information leakage arising from rejection sampling by Zhou, Wang, Sun and Yu [222] and by Liu, Wang, Wei, Ding, Zhang, Liu and Zhu [150], as well as fault attacks studied by Ulitzsch, Marzougui, Bagia, Tibouchi and Seifert [212], Krahmer, Pessl, Land and Güneysu [134], and Kosuge and Xagawa [133].

Threshold signature schemes distribute a secret signing key among multiple parties so that a valid signature can only be produced by a coalition of parties exceeding a given size threshold. In 2026, multi-party computation techniques were used by Bienstock, de Castro, Escudero, Polychroniadou and Takahashi [31], and by Celi, del Pino, Espitau, Niot and Prest [50], to construct threshold signing protocols for Dilithium.

Another lattice-based signature scheme that follows the Fiat-Shamir methodology is BLISS (Bimodal Lattice Signature Scheme), designed by Ducas, Durmus, Lepoint and Lyubashevsky [73]. A primary difference between BLISS and Dilithium is their choices of distributions for the per-message secret y and the signature component z. BLISS selects y from a discrete Gaussian distribution and uses rejection sampling to ensure that z follows a bimodal discrete Gaussian distribution that is independent of the signing key. In contrast, Dilithium selects y from a uniform distribution and uses rejection sampling to ensure that z follows a distribution that is independent of the signing key component s_1 and has coefficients bounded in the infinity norm. Consequently, BLISS has smaller public keys and signatures than Dilithium at the same security level. However, BLISS's Gaussian sampling and rejection sampling have been shown to be highly vulnerable to side-channel attacks. Specifically, Groot Bruinderink, Hülsing, Lange and Yarom [107] presented a key-recovery cache timing attack that targets the Gaussian sampler, and Espitau, Fouque, Gérard and Tibouchi [84] presented a key-recovery electromagnetic radiation attack that targets the rejection sampling. Marzougui, Wisiol, Gersch, Krämer and Seifert [156] devised power analysis attacks on GALACTICS, a constant-time mod-

ification of BLISS by Barthe, Belaïd, Espitau, Fouque, Rossi and Tibouchi [25].

HAWK is a lattice-based post-quantum signature scheme introduced by Ducas, Postlethwaite, Pulles and van Woerden [79], which reached the third round of NIST's standardization process for additional signature schemes. In 2026, Straznickas and Weis [211], working with Claude Mythos Preview, discovered an improved key-recovery attack on HAWK. The attack exploits a nontrivial automorphism of HAWK's key lattice and reduces key recovery to SVP calls in roughly half the dimension. For HAWK-512, the parameter set targeting NIST Level 1, the estimated gate-count cost of key recovery drops from 2^{150} to 2^{108}. Since restoring the intended security level would require roughly doubling key sizes, the HAWK team withdrew the scheme from NIST's standardization process.

7.5 Exercises

Exercise 7.1 Let $q = 9769$, $n = 5$, $\alpha = 296$, and

$$a(x) = \begin{bmatrix} 4024 + 2133x + 9066x^2 + 1163x^3 + 8416x^4 \\ 5179 + 5598x + 7747x^2 + 5777x^3 + 6635x^4 \\ 7720 + 8139x + 519x^2 + 7008x^3 + 9176x^4 \end{bmatrix} \in R_q^3.$$

Verify that

$$\mathrm{HighBits}(a(x), 296) = \begin{bmatrix} 14 + 7x + 31x^2 + 4x^3 + 28x^4 \\ 17 + 19x + 26x^2 + 20x^3 + 22x^4 \\ 26 + 27x + 2x^2 + 24x^3 + 31x^4 \end{bmatrix}$$

and

$$\mathrm{LowBits}(a(x), 296) = \begin{bmatrix} -120 + 61x - 110x^2 - 21x^3 + 128x^4 \\ 147 - 26x + 51x^2 - 143x^3 + 123x^4 \\ 24 + 147x - 73x^2 - 96x^3 \end{bmatrix}.$$

Exercise 7.2 Prove Lemma 7.8.

Exercise 7.3 Describe and analyze the FIPS 204 SampleInBall procedure for generating pseudorandom polynomials in B_τ. This procedure is based on the "Fisher-Yates shuffle".

Exercise 7.4 FIPS 204 specifies the following representation for the hint vector h. Let $h = (h_1, \ldots, h_k)$ where $h_i \in \{0,1\}^{256}$, and where the total number u of one bits in h is at most ω.
Let y be a byte array of length $\omega + k$, all of whose entries are initialized to 0. For each $1 \le i \le k$, let $y[\omega + i]$ store the total number of one bits in $h_1, \ldots, h_i$. Lastly, let $y[1], \ldots, y[u]$ store the positions of the u one bits in $h_1, \ldots, h_k$.
Describe a method for reconstructing $h_1, \ldots, h_k$ from the array y.

Exercise 7.5 Verify that Dilithium with the ML-DSA-44, ML-DSA-65 and ML-DSA-87 parameter sets (Table 7.1) has public keys of sizes 1312, 1952 and 2592 bytes, respectively, and signatures of sizes 2420, 3309 and 4627 bytes.

8 Falcon (FN-DSA)

Contents

8.1 Hash-and-sign 91
8.2 Discrete Gaussian distributions 94
8.3 GPV signature framework 101
8.4 NTRU lattices 104
8.5 NTRU key pair generation 107
8.6 Falcon outline 113
8.7 The Fast Fourier Transform 117
8.8 Notes and further references 118
8.9 Exercises 120

Falcon is a lattice-based quantum-safe signature scheme that was selected by NIST for standardization in July 2022. Falcon is expected to be published as FIPS 206, with a draft to be released in 2027. NIST's official name for Falcon is Fast-Fourier Transform over NTRU-Lattice-Based Digital Signature Algorithm (FN-DSA). Falcon's principal advantage over Dilithium lies in its significantly smaller public keys and signatures. However, it introduces unique implementation challenges. Key generation and signature generation rely heavily on intricate floating-point arithmetic, and achieving constant-time execution to mitigate side-channel attacks is notoriously difficult.

In September 2026, NIST announced plans to use only fixed-point arithmetic in the forthcoming draft standard and to disallow floating point. The present edition of these notes covers neither the FFT-based Gaussian sampling algorithm nor the fixed-point approach. The notes will be updated with a description of the fixed-point approach after a draft of FIPS 206 is released.

Falcon follows the framework for lattice-based hash-and-sign signature schemes (§8.1) introduced by Gentry, Peikert and Vaikuntanathan (GPV). A prominent feature of the GPV framework is the need to sample from discrete Gaussian distributions over lattices; these distributions are detailed in §8.2. After reviewing the GPV framework in §8.3, we introduce NTRU lattices in §8.4 and describe NTRU key pair generation in §8.5, before outlining the Falcon scheme itself in §8.6.

8.1 Hash-and-sign

8.1.1 RSA signature scheme

The RSA full-domain hash (FDH) signature scheme follows the hash-and-sign paradigm. Alice's public key is (N, e) and her private signing key is $d = e^{-1} \bmod \phi(N)$. To sign a message $M \in \{0,1\}^*$, she computes $h = H(M)$, where $H : \{0,1\}^* \to \mathbb{Z}_N$ is a hash function, and outputs the signature $s = h^d \bmod N$. To verify a signed message (M, s), Bob computes $h = H(M)$ and checks that $h = s^e \bmod N$.

The hash value h serves as a random challenge in $\mathbb{Z}_N$ whose unique preimage under

the RSA permutation $s \mapsto s^e \bmod N$ is $s = h^d \bmod N$. Computing this preimage is believed to be infeasible without knowledge of d. Thus, d serves as a *trapdoor* that allows Alice to efficiently invert the one-way function $f : \mathbb{Z}_N \to \mathbb{Z}_N$ defined by $f(x) = x^e \bmod N$. Such a function is called a *trapdoor one-way function.*

8.1.2 SIS-based trapdoor one-way function

Let q be a prime and $A \in_R \mathbb{Z}_q^{n\times m}$ where $m \gg n$, and let $u \in_R \mathbb{Z}_q^n$. Recall the inhomogeneous short integer solution problem (Definition 3.4) of finding a short solution $s \in \mathbb{Z}^m$ to the equation

$$As = u \pmod q.$$

Recall also that the SIS lattice (Definition 3.8) is the m-dimensional q-ary integer lattice

$$L_A^\perp = \{z \in \mathbb{Z}^m : Az = 0 \pmod q\}$$

with basis

$$B = \begin{bmatrix} qI_n & -\overline{A} \\ 0 & I_{m-n} \end{bmatrix} \in \mathbb{Z}^{m\times m} \tag{44}$$

where A is row-reduced over $\mathbb{Z}_q$ to the form $[I_n \mid \overline{A}]$. The basis vectors (the columns of B) are relatively long, with the first n vectors having length q and the remaining $m - n$ vectors having expected length approximately $q\sqrt{n/3} \approx 0.58q\sqrt{n}$.

Let $\beta \ll q\sqrt{m}$ be a solution bound, and define the set

$$D_n = \{s \in \mathbb{Z}^m \ : \ \|s\|_2 \le \beta\}.$$

Consider the *SIS function*

$$f_A : D_n \longrightarrow \mathbb{Z}_q^n$$

defined by

$$f_A(s) = As \bmod q.$$

Since $m \gg n$, f_A is expected to be surjective. Moreover, each $u \in \mathbb{Z}_q^n$ is expected to have many preimages. Given $u \in_R \mathbb{Z}_q^n$, finding any preimage of u is precisely an instance of (inhomogeneous) SIS, and thus is computationally infeasible.

Suppose now that $T = [t_1, \ldots, t_m]$ is a secret basis of $L_A^\perp$ whose vectors are unusually short, e.g., of length $O(\sqrt{q})$. This short basis T enables the efficient computation of a preimage of $u \in_R \mathbb{Z}_q^n$ via Babai's rounding method (§2.5.3). First, use linear algebra over $\mathbb{Z}_q$ to find an arbitrary integer solution $t \in \mathbb{Z}^m$ to the equation $At = u \pmod q$. Next, apply Babai's rounding method to find a lattice vector z close to t. Specifically, express the target t as a linear combination of the basis vectors, $t = \sum_{i=1}^m z_i t_i$ where $z_i \in \mathbb{Q}$, and let the nearby lattice point be $z = \sum_{i=1}^m \lfloor z_i \rceil t_i \in L_A^\perp$. Because T is a short lattice basis, one expects z to be close to t (cf. Example 2.24).

The parameter $\beta \ll q\sqrt{m}$ can be chosen to ensure that $\|t - z\|_2 \leq \beta$ holds with overwhelming high probability. Finally, let $s = t - z \in \mathbb{Z}^m$. Notice that

$$s = t - z = \sum_{i=1}^{m} (z_i - \lfloor z_i \rceil) t_i,$$

so s lies in the parallelepiped

$$P_{1/2}(T) = \{a_1 t_1 + a_2 t_2 + \cdots + a_m t_m \ : \ -\tfrac{1}{2} \leq a_i \leq \tfrac{1}{2}\}.$$

We have

$$As = A(t - z) = At - Az = u - 0 = u \pmod{q}.$$

Because $\|s\|_2 = \|t - z\|_2 \leq \beta$, $s \in D_n$ is a valid solution to the ISIS problem $As = u \pmod{q}$. Knowledge of the basis T enabled us to efficiently find a preimage of u, and thus f_A is a trapdoor one-way function with T serving as a trapdoor.

8.1.3 SIS-based signature scheme

SIS trapdoor one-way functions naturally yield a hash-and-sign signature scheme analogous to the RSA full-domain hash signature scheme.

Alice's public verification key is the matrix $A \in_R \mathbb{Z}_q^{n \times m}$, and her private signing key is a short basis $T \in \mathbb{Z}^{m \times m}$ of the SIS lattice $L_A^\perp$. To sign a message $M \in \{0,1\}^*$, Alice computes $u = H(M)$ where $H : \{0,1\}^* \to \mathbb{Z}_q^n$ is a hash function. She then uses Babai's rounding method with the trapdoor T to find a short vector $s \in D_n$ such that $As = u \pmod{q}$. Her signature on M is s. Bob verifies the signed message (M, s) by confirming that $As = H(M) \pmod{q}$ and $s \in D_n$, i.e., $\|s\|_2 \leq \beta$.

A complete specification of such a signature scheme requires a method for generating a key pair (A, T). This cannot be done by selecting A first and then computing T, as finding short vectors in arbitrary lattices is hard.

At first glance, forging a signature appears to require solving an instance of the approximate closest vector problem (§2.5.3). Indeed, to forge Alice's signature on a message M, an adversary can compute $u = H(M)$ and a target vector $t \in \mathbb{Z}^m$ satisfying $At = u \pmod{q}$. The success of the forgery attempt then depends on the adversary's ability to find a lattice vector $z \in L_A^\perp$ sufficiently close to t.

8.1.4 A critical security weakness

NTRUSign, the first practical lattice-based signature scheme, employed an SIS-based trapdoor function over a highly structured lattice called the NTRU lattice (§8.4). This algebraic structure allowed for compact public keys and signatures. The signing procedure used Babai's rounding method as described above.

However, NTRUSign had a critical vulnerability: every signed message (M, s) leaked geometric information about the secret basis T, specifically revealing that s is confined to the parallelepiped $P_{1/2}(T)$. Indeed, Nguyen and Regev showed that the secret basis T

could be efficiently reconstructed given a sufficient number of signatures; see Figure 8.1. For instance, the NTRUSign-251 parameter set, which was designed for 80-bit security, was completely broken using about 400 signed messages.

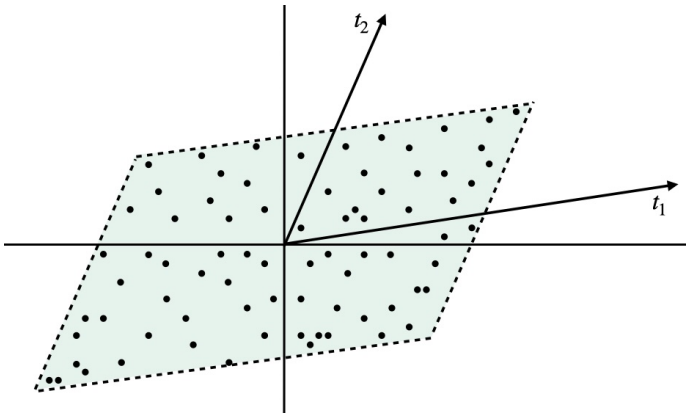


Figure 8.1: Signature points s in the parallelepiped $P_{1/2}(T) = \{a_1 t_1 + a_2 t_2 : -\frac{1}{2} \le a_i \le \frac{1}{2}\}$.

To prevent this leakage, Gentry, Peikert and Vaikuntanathan proposed randomizing the selection of a preimage s of $u = H(M)$. The solution bound β is chosen to ensure that each target $u \in \mathbb{Z}_q^n$ is expected to have a large number of preimages, but not render the ISIS problem easy to solve. A signature s is then sampled from a discrete Gaussian distribution over this preimage space, chosen so that the distribution of s can be proven independent of the secret basis T. One can then conclude that signatures do not leak any useful information about T.

8.2 Discrete Gaussian distributions

A discrete Gaussian distribution is obtained by restricting a continuous Gaussian distribution to a lattice and then renormalizing.

8.2.1 Gaussians

For the remainder of this section, we denote the Euclidean length $\|\cdot\|_2$ simply as $\|\cdot\|$.

Definition 8.1 The *Gaussian function* of width $s > 0$ and centered at $c \in \mathbb{R}^n$ is the function $\rho_{s,c} : \mathbb{R}^n \to \mathbb{R}$ defined by

$$\rho_{s,c}(x) = \exp\left(-\frac{\pi \|x - c\|^2}{s^2}\right). \tag{45}$$

The total mass of $\rho_{s,c}$ is

$$\int_{x \in \mathbb{R}^n} \rho_{s,c}(x)\, dx = s^n,$$

and hence $\rho_{s,c}(x)/s^n$ is a probability density function.

Definition 8.2 The (continuous) *Gaussian distribution* $D_{s,c}$ of width $s > 0$ and centered at $c \in \mathbb{R}^n$ is a probability distribution with density function

$$D_{s,c}(x) = \frac{\rho_{s,c}(x)}{s^n}, \quad x \in \mathbb{R}^n. \tag{46}$$

Gaussian distributions are also called *normal distributions*. With $\mu = c$ and

$$\sigma = s/\sqrt{2\pi}, \tag{47}$$

this becomes the familiar n-dimensional isotropic Gaussian $N(\mu, \sigma^2)$ with mean μ, standard deviation σ, and covariance matrix $\sigma^2 I_n$:

$$D_{\sigma,\mu}(x) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{\|x-\mu\|^2}{2\sigma^2}\right).$$

The parameter σ (equivalently s) controls the spread of the distribution. Figure 8.2 shows the 1-dimensional Gaussian distributions

$$D_{\sigma,0}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

with $\sigma = 1, 2, 3$. Figure 8.3 shows the 2-dimensional Gaussian distributions $D_{\sigma,0}$ with $\sigma = 0.5$ and $\sigma = 1.0$.

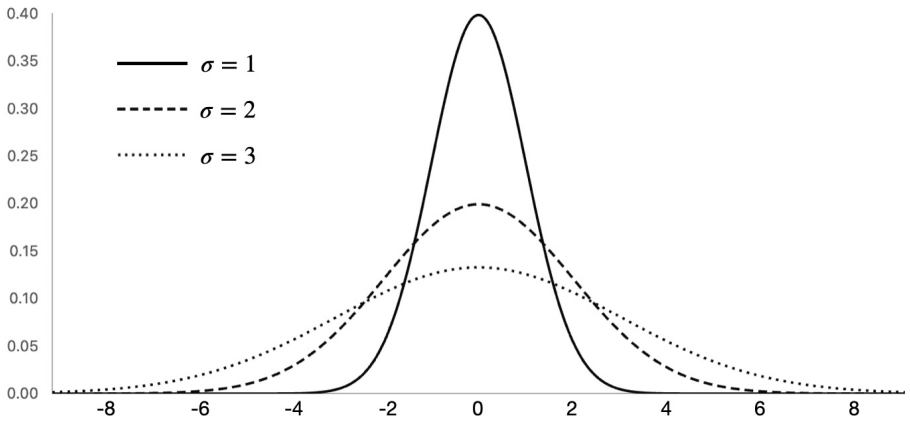


Figure 8.2: Gaussian distributions with centers $c = 0$ and standard deviations $\sigma = 1, 2, 3$.

To simplify notation, we often omit the center when it is 0, writing ρ_s and D_s instead of $\rho_{s,0}$ and $D_{s,0}$.

8.2.2 Discrete Gaussians

Let $L \subseteq \mathbb{R}^n$ be a lattice. Define the quantities

$$\rho_{s,c}(L) = \sum_{x \in L} \rho_{s,c}(x) \quad \text{and} \quad D_{s,c}(L) = \sum_{x \in L} D_{s,c}(x) = \frac{\rho_{s,c}(L)}{s^n},$$

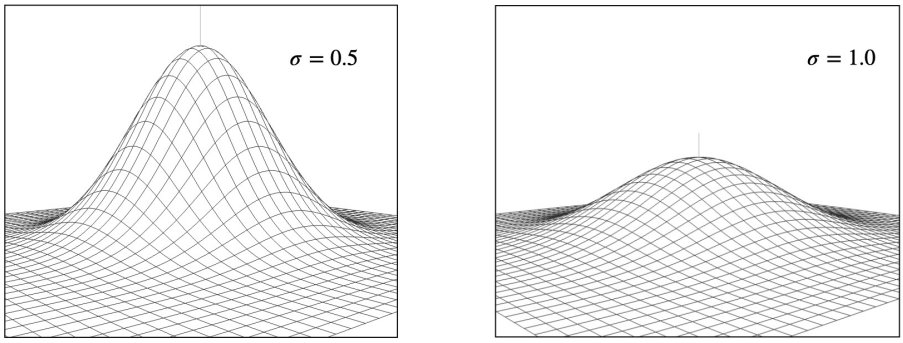


Figure 8.3: Two-dimensional Gaussian distributions with standard deviations $\sigma = 0.5$ and 1.0.

both of which converge for any $s > 0$.

Definition 8.3 The *discrete Gaussian distribution* $D_{L,s,c}$ of width $s > 0$ and center $c \in \mathbb{R}^n$ over a lattice $L \subseteq \mathbb{R}^n$ is a probability distribution with density function

$$D_{L,s,c}(x) = \frac{D_{s,c}(x)}{D_{s,c}(L)} = \frac{\rho_{s,c}(x)}{\rho_{s,c}(L)}, \quad x \in L. \tag{48}$$

To simplify notation, we often omit the center when it is 0, writing $D_{L,s}$ for $D_{L,s,0}$. The discrete Gaussian distribution $D_{L,s}$ is symmetric about the origin in the sense that $D_{L,s}(x) = D_{L,s}(-x)$ for all $x \in L$, since $-x \in L$ whenever $x \in L$, and because $\rho_s(x) = \rho_s(-x)$ (as ρ_s depends only on $\|x\|$).

Example 8.4 (*one-dimensional discrete Gaussian distribution*) Consider the integer lattice $L = \mathbb{Z}$. We have

$$S = \rho_5(\mathbb{Z}) = \sum_{x=-\infty}^{\infty} e^{-\pi x^2/25} \approx 5.$$

The one-dimensional discrete Gaussian distribution

$$D_{\mathbb{Z},5}(x) = e^{-\pi x^2/25}/S, \quad x \in \mathbb{Z},$$

is shown in Figure 8.4. Notice that $D_{\mathbb{Z},5}$ is symmetric about the origin. The values of $D_{\mathbb{Z},5}$ for $0 \le x \le 15$ are shown in Table 8.1. This Gaussian distribution has width $s = 5$ and standard deviation $\sigma = s/\sqrt{2\pi} \approx 1.9947$.

Example 8.5 (*two-dimensional discrete Gaussian distribution*) Consider the integer lattice $L = \mathbb{Z}^2$. We have

$$S = \rho_3(\mathbb{Z}^2) = \sum_{x_1=-\infty}^{\infty} \sum_{x_2=-\infty}^{\infty} e^{-\pi(x_1^2+x_2^2)/9} = \left(\sum_{x_1=-\infty}^{\infty} e^{-\pi x_1^2/9} \right) \left(\sum_{x_2=-\infty}^{\infty} e^{-\pi x_2^2/9} \right) \approx 9.$$

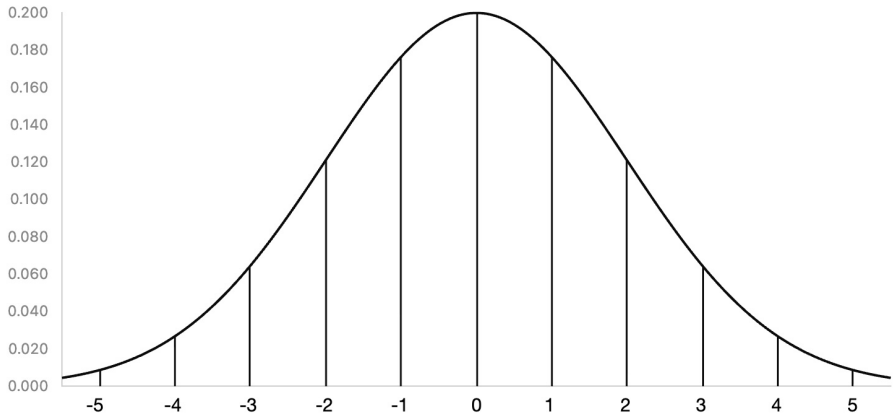


Figure 8.4: The one-dimensional discrete Gaussian distribution $D_{\mathbb{Z},5}$.

x	$D_{\mathbb{Z},5}(x)$	x	$D_{\mathbb{Z},5}(x)$
0	0.2000000000	8	0.0000643023
1	0.1763822757	9	$0.7593650578 \times 10^{-5}$
2	0.1209845126	10	$0.6974684712 \times 10^{-6}$
3	0.0645437967	11	$0.4982513029 \times 10^{-7}$
4	0.0267811443	12	$0.2768357654 \times 10^{-8}$
5	0.0086427837	13	$0.1196315814 \times 10^{-9}$
6	0.0021693421	14	$0.4020863978 \times 10^{-11}$
7	0.0004234989	15	$0.1051097035 \times 10^{-12}$

Table 8.1: Some values of the one-dimensional discrete Gaussian distribution $D_{\mathbb{Z},5}$.

The two-dimensional discrete Gaussian distribution

$$D_{\mathbb{Z}^2,3}(x) = e^{-\pi(x_1^2+x_2^2)/9}/S, \quad x = (x_1, x_2) \in \mathbb{Z}^2,$$

is shown in Figure 8.5. This Gaussian distribution has width $s = 3$ and standard deviation $\sigma = s/\sqrt{2\pi} \approx 1.1968$.

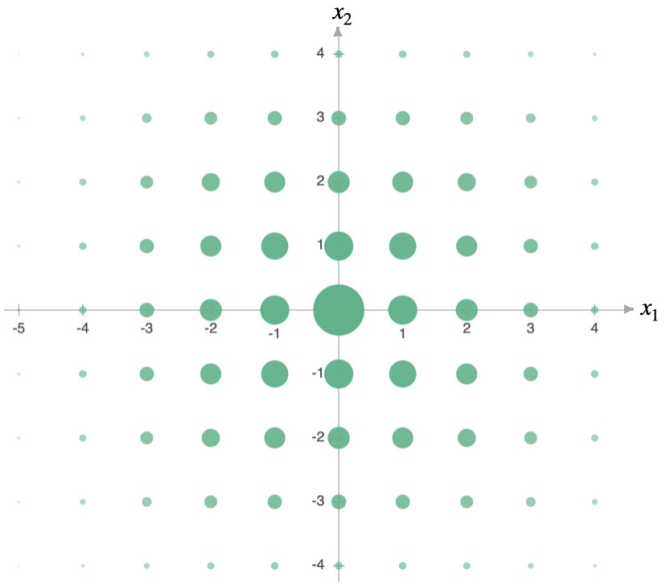


Figure 8.5: The two-dimensional discrete Gaussian distribution $D_{\mathbb{Z}^2,3}$. Each circle is centered at a lattice point $x = (x_1, x_2) \in \mathbb{Z}^2$, with radius proportional to $\rho_3(x)$.

8.2.3 The smoothing parameter of a lattice

The smoothing parameter is central to lattice-based cryptography. Although its formal definition is admittedly opaque, its cryptographic significance becomes clear through its applications.

Definition 8.6 Let $L \subseteq \mathbb{R}^n$ be a full-rank lattice, and let $\varepsilon > 0$. The *smoothing parameter* of L, denoted $\eta_\varepsilon(L)$, is the smallest real number $s > 0$ such that

$$\rho_{1/s}(L^* \setminus \{0\}) \leq \varepsilon \tag{49}$$

where L^* is the dual of L (Definition 4.17).

Because $\rho_{1/s}(0) = 1$, the inequality (49) is equivalent to $\rho_{1/s}(L^*) \leq 1 + \varepsilon$. Hence, for small ε, almost all of the mass $\rho_{1/s}(L^*)$ is concentrated at the origin. Larger ε relaxes the bound, permitting smaller s. Thus, $\eta_\varepsilon(L)$ strictly decreases as ε increases.

Let $s > 0$, and consider the function $f : \mathbb{R}^n \to \mathbb{R}$ defined by

$$f(x) = \sum_{c \in L} D_{s,c}(x).$$

The sum converges absolutely, so f is well-defined and smooth. It is easy to see that f is L-periodic, meaning $f(x + c_0) = f(x)$ for all $c_0 \in L$. Thus, f induces a well-defined function on the torus $\mathbb{R}^n/L$, also denoted f. Let $X \sim D_s$ and $Y = X \bmod L$. One can show that

$$\int_{\mathbb{R}^n/L} f(x)\,dx = \int_{\mathbb{R}^n} D_s\,dx = 1,$$

and hence f is the density function of Y on $\mathbb{R}^n/L$. The smoothing parameter $\eta_\varepsilon(L)$ is the threshold width above which Y is close to uniform. More precisely, we have the following result.

Theorem 8.7 (Micciancio and Regev) Let $L \subseteq \mathbb{R}^n$ be a full-rank lattice. Let $\varepsilon \in (0, 1)$ and let $s \geq \eta_\varepsilon(L)$. Then the density function f of $D_s \bmod L$ satisfies

$$\frac{1-\varepsilon}{\mathrm{vol}(L)} \leq f(x) \leq \frac{1+\varepsilon}{\mathrm{vol}(L)}, \quad \forall x \in \mathbb{R}^n/L.$$

The same bound holds for any center $c \in \mathbb{R}^n$, since the density of $D_{s,c} \bmod L$ is just a translate of f on the torus (Exercise 8.12).

Remark 8.8 (*illustration of the smoothing parameter*) The function $f(x) = \sum_{c \in L} D_{s,c}(x)$ can be visualized as placing a continuous Gaussian $D_{s,c}$ at each lattice point $c \in L$ and summing. As illustrated in Figure 8.6(a), when the width s is small compared to the spacing of the lattice points, the Gaussians appear as steep hills with very little overlap. As s increases, the hills broaden and overlap significantly (Figure 8.6(b)). When $s \geq \eta_\varepsilon(L)$, their sum f becomes almost flat (Figure 8.6(c)).

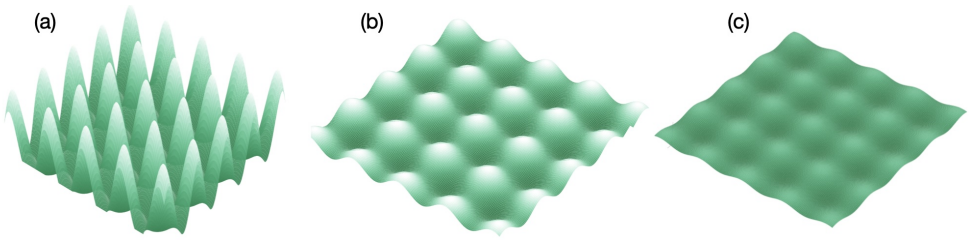


Figure 8.6: The function $f(x) = \sum_{c \in L} D_{s,c}(x)$ for increasing widths s.

We next state a weaker form of Theorem 8.7, expressed in terms of statistical distance, which we first define.

Definition 8.9 The *statistical distance* between discrete random variables X and Y over a countable set D is

$$\Delta(X, Y) = \frac{1}{2} \sum_{d \in D} |\Pr(X = d) - \Pr(Y = d)|.$$

The *statistical distance* between continuous random variables X and Y over $\mathbb{R}^n$ with probability density functions g_1 and g_2, respectively, is

$$\Delta(X, Y) = \frac{1}{2} \int_{\mathbb{R}^n} |g_1(x) - g_2(x)| \, dx.$$

Two distributions are *statistically indistinguishable* (or *statistically close*) if their statistical distance is negligible.

Note that $0 \leq \Delta(X, Y) \leq 1$, with $\Delta(X, Y) = 0$ if and only if X and Y are identically distributed, and $\Delta(X, Y) = 1$ if and only if they have disjoint support; see Exercise 8.9.

Recall that $P(B)$ is the fundamental parallelepiped of L with respect to basis $B = [b_1, \ldots, b_n]$. For any $x \in \mathbb{R}^n$, $x \bmod P(B)$ denotes the unique vector $y \in P(B)$ such that $x - y \in L$. To compute it, write $x = \sum_{i=1}^n c_i b_i$; then $y = \sum_{i=1}^n \tilde{c}_i b_i$ where $\tilde{c}_i = c_i - \lfloor c_i \rfloor$. The uniform distribution on a set S is denoted by $U(S)$.

Theorem 8.10 (Micciancio and Regev) Let $L \subseteq \mathbb{R}^n$ be a full-rank lattice. For any basis B of L, center $c \in \mathbb{R}^n$, parameter $\varepsilon > 0$, and width $s \geq \eta_\varepsilon(L)$, we have

$$\Delta\left(D_{s,c} \bmod P(B),\, U(P(B))\right) \leq \varepsilon/2.$$

Thus, if s exceeds the smoothing parameter, then for any center c and lattice basis B, folding $D_{s,c}$ onto the corresponding fundamental parallelepiped yields a distribution within statistical distance $\varepsilon/2$ of uniform regardless of the center. Micciancio and Regev proved this using the Poisson summation formula from Fourier analysis, and used it in their proof of the average-case hardness of SIS.

Near-uniformity modulo L is one application of the smoothing parameter. Another application concerns discrete Gaussian sampling. If $s \geq \eta_\varepsilon(L)$, then any sufficiently short basis yields an efficient sampler whose output is statistically close to the discrete Gaussian $D_{L,s,c}$ which is independent of the basis. Thus, a short secret basis T suffices for sampling, but the samples reveal essentially nothing about T. This basis-hiding property is central to the GPV signature scheme.

The next result bounds the probability that a discrete Gaussian sample deviates far from its center.

Theorem 8.11 (Micciancio and Regev) Let $L \subseteq \mathbb{R}^n$ be a full-rank lattice. For any center $c \in \mathbb{R}^n$, parameter $0 < \varepsilon < 1$, and width $s \geq \eta_\varepsilon(L)$, we have

$$\Pr_{x \sim D_{L,s,c}} \left\{ \|x - c\| > s\sqrt{n} \right\} \leq \frac{1 + \varepsilon}{1 - \varepsilon} \cdot \frac{1}{2^n}.$$

To conclude this subsection, we bound the smoothing parameter from above in terms of the Gram-Schmidt orthogonalization (see §10.1) of any basis of the lattice.

Theorem 8.12 (Gentry, Peikert and Vaikuntanathan) Let $L \subseteq \mathbb{R}^n$ be a full-rank lattice, and let $\varepsilon > 0$. For any basis B of L, we have

$$\eta_\varepsilon(L) \leq \|\tilde{B}\| \cdot \sqrt{\frac{\ln(2n(1+1/\varepsilon))}{\pi}},$$

where $\tilde{B}$ is the Gram-Schmidt orthogonalization of B, and $\|\tilde{B}\|$ denotes the length of the longest vector in $\tilde{B}$. Moreover, for any $\omega(\sqrt{\log n})$ function, there is a negligible $\varepsilon(n)$ for which $\eta_\varepsilon(L) \leq \|\tilde{B}\| \cdot \omega(\sqrt{\log n})$.

8.3 GPV signature framework

Gentry, Peikert and Vaikuntanathan (GPV) introduced a framework for secure hash-and-sign signature schemes. The key ingredient is a *preimage-sampleable trapdoor one-way function*. GPV gave a lattice-based construction whose security is based on SIS hardness. This section describes the lattice-based GPV signature framework. Falcon, a concrete instantiation of this framework, is presented in §8.6.

Preliminaries. The size of a basis $T = [t_1, \ldots, t_m]$ is $\|T\| = \max_i \|t_i\|$. The Gram-Schmidt orthogonalization $\tilde{T}$ of T (see §10.1) is denoted $\tilde{T} = [\tilde{t}_1, \ldots, \tilde{t}_m]$.

The GPV signature scheme has several parameters:

- n: the number of rows in the public-key matrix A.
- $q = \text{poly}(n)$: a prime modulus.[13]
- $m \geq 5n \log_2 q$: the number of columns in A.
- ℓ: bitlength of the message randomizer.
- $L_{GS} = m^{2.5}$: an upper bound on $\|\tilde{T}\|$.
- $\overline{s} \geq L_{GS} \cdot \omega(\sqrt{\log m})$: the Gaussian width.[14]
- $H : \{0,1\}^* \to \mathbb{Z}_q^n$: a hash function.

Recall that $L_A^\perp$ is the m-dimensional SIS lattice associated with the matrix $A \in \mathbb{Z}_q^{n \times m}$. Ajtai described an algorithm for generating a matrix $A \in \mathbb{Z}_q^{n \times m}$ and a short basis T for $L_A^\perp$ such that A is statistically close to uniform and $\|\tilde{T}\| \leq L_{GS}$. Furthermore, we can assume that $\|T\| \leq m^3$ (Exercise 10.23).

Now, let

$$D_n = \{s \in \mathbb{Z}^m : \|s\| \leq \overline{s}\sqrt{m}\}, \tag{50}$$

[13] The parameters q and $\overline{s}$ are described in asymptotic terms. A practical realization of the GPV signature scheme should specify concrete parameters.

[14] In §8.3 we are using $\overline{s}$ to denote the Gaussian width to avoid clashing with the signature component s.

and define the function $f_A : D_n \to \mathbb{Z}_q^n$ by

$$f_A(s) = As \bmod q. \tag{51}$$

The function f_A is one-way assuming inhomogeneous SIS is hard. Note that f_A is expected to be surjective but not injective. The secret basis T serves as a trapdoor for finding preimages under f_A as follows.

Given $u \in \mathbb{Z}_q^n$, find an arbitrary $t \in \mathbb{Z}^m$ such that $At = u \pmod q$. Next, use Klein's algorithm (not described here) and the bases T and $\tilde{T}$ to efficiently sample z from a distribution that is statistically close to the discrete Gaussian distribution $D_{L_A^\perp, \bar{s}, t}$. Finally, set $s = t - z$.

We have

$$As = A(t - z) = At - Az = u - 0 = u \pmod q.$$

Since $\bar{s} \geq L_{GS} \cdot \omega(\sqrt{\log m})$, Theorem 8.12 gives $\bar{s} \geq \eta_\varepsilon(L_A^\perp)$ for some negligible ε. By Theorem 8.11,

$$\Pr(\|t - z\| > \bar{s}\sqrt{m}) \leq \frac{1+\varepsilon}{1-\varepsilon} \cdot \frac{1}{2^m}.$$

Hence, $\|s\| \leq \bar{s}\sqrt{m}$ and $s \in D_n$ with overwhelming probability.

Notice that while z is relatively close to t, it is not as close to t as the vector produced using Babai's rounding method in §8.1. Indeed, since z is drawn from a distribution statistically indistinguishable from $D_{L_A^\perp, \bar{s}, t}$, which depends only on A, $\bar{s}$ and t (and not on T), z is (statistically) independent of T. Moreover, since t is derived using ordinary linear algebra from A and u, it is also independent of T. Thus neither z nor $s = t - z$ reveal any information about T. More precisely, s is statistically close to the conditional distribution of $D_{\mathbb{Z}^m, \bar{s}}$ given $As = u \pmod q$; see Remark 8.14 and Figure 8.7.

The GPV signature scheme. The GPV signature scheme employs the trapdoor one-way function $f_A : D_n \to \mathbb{Z}_q^n$ and is presented in Algorithm 8.13. The key innovation over the hash-and-sign SIS-based signature scheme described in §8.1.3 is the randomized selection of a vector z relatively close to the target t by sampling from a Gaussian distribution $D_{L_A^\perp, \bar{s}, t}$ centered at t, instead of using a deterministic method such as Babai's rounding.

Remark 8.14 (*sampleable trapdoor one-way functions*) The trapdoor function f_A satisfies two key properties.

- *Domain sampling with uniform outputs.* Consider a sample $s \sim D_{\mathbb{Z}^m, \bar{s}}$ and set $u = f_A(s) = As \bmod q$. Then the resulting distribution of u is statistically indistinguishable from uniform over $\mathbb{Z}_q^n$.
- *Preimage sampling with trapdoor.* Conversely, suppose that $u \in \mathbb{Z}_q^n$ is fixed, $t \in \mathbb{Z}^m$ is an arbitrary solution to $At = u \pmod q$, and z is sampled from $D_{L_A^\perp, \bar{s}, t}$. Then the distribution of $s = t - z$ is exactly the conditional distribution of $D_{\mathbb{Z}^m, \bar{s}}$ given $As = u \pmod q$.

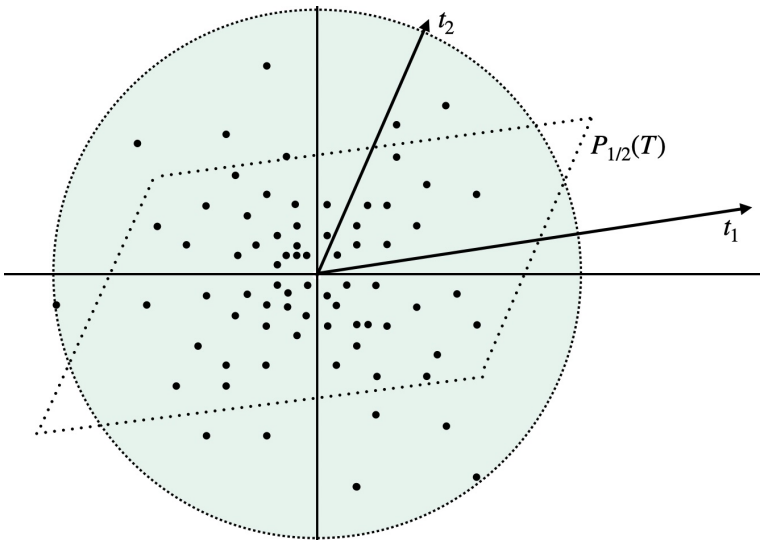


Figure 8.7: Signature points s in the GPV signature scheme; cf. Figure 8.1.

Algorithm 8.13: GPV signature scheme

Domain parameters. $n, m, q, \ell, L_{GS}, \bar{s}, H : \{0,1\}^* \to \mathbb{Z}_q^n$.

Key generation. Alice does the following:

1 Use Ajtai's algorithm to select $A \in \mathbb{Z}_q^{n \times m}$ and a short basis T for the SIS lattice $L_A^\perp$ such that (i) A is statistically indistinguishable from uniform, and (ii) $\|\tilde{T}\| \leq L_{GS}$.
2 Alice's verification (public) key is A; her signing (private) key is T.

Signature generation. To sign a message $M \in \{0,1\}^*$, Alice does the following:

1 Select $r \in_R \{0,1\}^\ell$ and compute $u = H(r, M)$.
2 Compute $t \in \mathbb{Z}^m$ such that $At = u \pmod q$.
3 Sample $z \sim D_{L_A^\perp, \bar{s}, t}$ using Klein's algorithm.
4 Compute $s = t - z$.
5 **return** $(\text{sig} = (r, s))$.

Signature verification. To verify Alice's signature $\text{sig} = (r, s)$ on M, Bob does:

1 Obtain an authentic copy of Alice's verification key A.
2 Verify that $\|s\| \leq \bar{s}\sqrt{m}$.
3 Compute $u = H(r, M)$.
4 Accept if and only if $As = u \pmod q$.

These properties are fundamental to proving that the GPV signature scheme is secure (Definition A.7) in the random oracle model under SIS hardness. The proof simulates a signer without T by programming H using the first property: on each new hash query (M, r), the simulator samples s directly from $D_{\mathbb{Z}^m,\bar{s}}$ and returns $u = f_A(s)$, so that the outputs of H remain statistically indistinguishable from uniform. The second property ensures that this s is distributed as in a real signature, so the simulator answers signing queries simply by revealing the previously sampled s without ever needing T.

Remark 8.15 (*message randomization*) The message M is randomized prior to signing by appending a randomly chosen *salt* $r \in \{0,1\}^\ell$. To see why randomization is necessary, suppose that Alice signs the same message M twice without a salt. Assuming the same t is computed both times (as would be the case with any deterministic method for solving $At = u \pmod q$), the signatures are $s_1 = t - z_1$ and $s_2 = t - z_2$ with $z_1, z_2 \sim D_{L_A^\perp,\bar{s},t}$ and $s_1 \neq s_2$ except with negligible probability. An adversary learning s_1 and s_2 obtains two distinct preimages of $u = H(M)$. Exposing multiple independent samples for the same target u breaks the GPV security proof, since the reduction can consistently program the random oracle only if at most one preimage of each target is revealed. Moreover, $s_1 - s_2 = z_2 - z_1$ is a short nonzero vector in $L_A^\perp$, i.e., a solution to the homogeneous SIS problem for A, giving the adversary information that is presumed infeasible to compute without knowledge of the secret basis T.

Making the GPV signature scheme practical requires several optimizations. A key optimization is the use of structured lattices, yielding compact public keys and signatures. Falcon achieves this by utilizing NTRU lattices, which are introduced next.

8.4 NTRU lattices

NTRU lattices are structured q-ary lattices with compact descriptions and short bases. This makes them attractive for lattice-based cryptography, including Falcon.

Preliminaries. Let n be a power of 2, and let q be a prime such that $2n$ divides $q-1$. Let $R = \mathbb{Z}[x]/(x^n+1)$ and $R_q = \mathbb{Z}_q[x]/(x^n+1)$. The restrictions on n and q ensure that the Number-Theoretic Transform (§13) yields efficient polynomial multiplication in R_q. The *size* of a polynomial $a = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1} \in R$ is the Euclidean norm of its vector representation $(a_0, a_1, \ldots, a_{n-1})$, i.e.,

$$\|a\|_2 = \sqrt{a_0^2 + a_1^2 + \cdots + a_{n-1}^2}.$$

Recall from §5.1 that polynomial multiplication in R_q can be expressed as a matrix-vector product. More precisely, if $a = \sum_{i=0}^{n-1} a_i x^i \in R_q$, $b = \sum_{i=0}^{n-1} b_i x^i \in R_q$, and $c =$

$a \times b = \sum_{i=0}^{n-1} c_i x^i \in R_q$, then

$$\underbrace{\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{bmatrix}}_{c} = \underbrace{\begin{bmatrix} a_0 & -a_{n-1} & -a_{n-2} & \cdots & -a_1 \\ a_1 & a_0 & -a_{n-1} & \cdots & -a_2 \\ a_2 & a_1 & a_0 & \cdots & -a_3 \\ \vdots & \vdots & \vdots & & \vdots \\ a_{n-1} & a_{n-2} & a_{n-3} & \cdots & a_0 \end{bmatrix}}_{\mathcal{A}(a)} \underbrace{\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{n-1} \end{bmatrix}}_{b},$$

where $\mathcal{A}(a)$ is an $n \times n$ anti-circulant matrix.

8.4.1 Lattice definition

Let $f, g \in R$ be polynomials of small norm (significantly smaller than q). Suppose also that $\gcd(f, x^n + 1) = 1$ in $\mathbb{Z}_q[x]$, so f is invertible in R_q and f^{-1} can be efficiently computed using the extended Euclidean algorithm (EEA) for polynomials. Let $h = g/f \in R_q$ and $\overline{A} = \mathcal{A}(h) \in \mathbb{Z}^{n \times n}$, and define the matrix

$$A = [I_n \mid \overline{A}] \in \mathbb{Z}^{n \times 2n}.$$

The *NTRU lattice* is the SIS lattice

$$L_A^\perp = \{z \in \mathbb{Z}^{2n} \ : \ Az = 0 \pmod q\}.$$

Remark 8.16 (*polynomial interpretation of* $L_A^\perp$) Let $z = (z_1, z_2) \in \mathbb{Z}^{2n}$ where $z_1, z_2 \in \mathbb{Z}^n$. Then $Az = z_1 + \mathcal{A}(h) z_2$. By identifying the integer vectors z_1 and z_2 with their corresponding polynomials in R, this relation simplifies to $Az = z_1 + z_2 h$ in R_q. Consequently, we can naturally describe the NTRU lattice using polynomial arithmetic as:

$$L_A^\perp = \{(z_1, z_2) \in R \times R \ : \ z_1 + z_2 h = 0 \text{ in } R_q\}.$$

Recall from §3.2 that $L_A^\perp$ is a full-rank q-ary lattice of volume q^n with basis matrix

$$B = \begin{bmatrix} qI_n & -\overline{A} \\ 0 & I_n \end{bmatrix} \in \mathbb{Z}^{2n \times 2n}.$$

The basis vectors are the columns of B. The first n columns have length q; the remaining n have length up to $\sqrt{n}q$. Hence $\|B\| \lesssim \sqrt{n}q$. In §8.4.2, we will find a much shorter basis of $L_A^\perp$.

Example 8.17 (*NTRU lattice*) Let $n = 4$ and $q = 9929$, so $R = \mathbb{Z}[x]/(x^4 + 1)$ and $R_q = \mathbb{Z}_{9929}[x]/(x^4 + 1)$. Select

$$f = 2 + 39x + 23x^2 - 55x^3 \in R \quad \text{and} \quad g = -20 + 60x + 34x^3 \in R.$$

We have $\gcd(f, x^4+1) = 1$ in $\mathbb{Z}_q[x]$,

$$f^{-1} = 4442 + 6644x + 4914x^2 + 9500x^3 \text{ in } R_q,$$

and

$$h = g \cdot f^{-1} = 8874 + 6280x + 7147x^2 + 7643x^3 \text{ in } R_q.$$

The NTRU lattice is $L_A^\perp$ where $A = [I_4 \mid \overline{A}]$ and $\overline{A} = \mathcal{A}(h)$. A basis of $L_A^\perp$ is

$$B = \begin{bmatrix} qI_4 & -\overline{A} \\ 0 & I_4 \end{bmatrix} = \left[\begin{array}{cccc|cccc} 9929 & 0 & 0 & 0 & -8874 & 7643 & 7147 & 6280 \\ 0 & 9929 & 0 & 0 & -6280 & -8874 & 7643 & 7147 \\ 0 & 0 & 9929 & 0 & -7147 & -6280 & -8874 & 7643 \\ 0 & 0 & 0 & 9929 & -7643 & -7147 & -6280 & -8874 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right].$$

8.4.2 A short basis

Suppose that $F, G \in R = \mathbb{Z}[x]/(x^n+1)$ are polynomials of small norm that satisfy the *NTRU equation*

$$fG - gF = q \text{ in } R. \tag{52}$$

Define the matrix

$$T = \begin{bmatrix} \mathcal{A}(g) & \mathcal{A}(G) \\ -\mathcal{A}(f) & -\mathcal{A}(F) \end{bmatrix} \in \mathbb{Z}^{2n \times 2n}.$$

To ease the notation, we sometimes identify polynomials with their anti-circulant matrices and write B and T as follows:

$$B = \begin{bmatrix} qI_n & -h \\ 0 & I_n \end{bmatrix} \quad \text{and} \quad T = \begin{bmatrix} g & G \\ -f & -F \end{bmatrix}.$$

In the proof of Theorem 8.18, we manipulate f, g, h, F, G as ring elements under matrix multiplication; this is justified since $\mathcal{A} : R \longrightarrow \mathbb{Z}^{n \times n}$ is a ring homomorphism (Exercise 8.13), so these matrices obey the same commutative algebra as polynomials f, g, h, F, G in R.

Theorem 8.18 T is a basis of the NTRU lattice $L_A^\perp$.

Proof: We first claim that the columns of T lie in $L_A^\perp$, from which it follows that the lattice L' spanned by the columns of T is a sublattice of $L_A^\perp$. Firstly,

$$A \cdot \begin{bmatrix} g \\ -f \end{bmatrix} = \begin{bmatrix} I_n & h \end{bmatrix} \cdot \begin{bmatrix} g \\ -f \end{bmatrix} = \begin{bmatrix} g - hf \end{bmatrix} = [0] \pmod q.$$

Secondly,

$$A \cdot \begin{bmatrix} G \\ -F \end{bmatrix} = \begin{bmatrix} I_n & h \end{bmatrix} \cdot \begin{bmatrix} G \\ -F \end{bmatrix} = \begin{bmatrix} G - hF \end{bmatrix}.$$

Now,

$$G - hF = G - (g/f)F = f^{-1}(fG - gF) = f^{-1}q = 0 \pmod{q},$$

which proves the claim.

We next prove that $\mathrm{vol}(L') = q^n$. Since the anti-circulant matrices corresponding to f, g, F and G commute, we have

$$\det(T) = \det(-gF + fG) = \det(qI_n) = q^n.$$

Hence, $\mathrm{vol}(L') = |\det(T)| = q^n$.

Finally, since $L' \subseteq L_A^\perp$ and $\mathrm{vol}(L') = \mathrm{vol}(L_A^\perp)$, we conclude that $L' = L_A^\perp$ and T is a basis of $L_A^\perp$. □

Example 8.19 (*short basis of an NTRU lattice*) Continuing Example 8.17, the polynomials

$$F = -37 + 24x + 21x^2 + 29x^3 \in R \quad \text{and} \quad G = -12 + 81x - 77x^2 - 49x^3 \in R$$

satisfy the NTRU equation $fG - gF = 9929$. Thus, a second basis of the NTRU lattice $L_A^\perp$ is

$$T = \begin{bmatrix} \mathcal{A}(g) & \mathcal{A}(G) \\ -\mathcal{A}(f) & -\mathcal{A}(F) \end{bmatrix} = \left[\begin{array}{rrrr|rrrr} -20 & -34 & 0 & -60 & -12 & 49 & 77 & -81 \\ 60 & -20 & -34 & 0 & 81 & -12 & 49 & 77 \\ 0 & 60 & -20 & -34 & -77 & 81 & -12 & 49 \\ 34 & 0 & 60 & -20 & -49 & -77 & 81 & -12 \\ \hline -2 & -55 & 23 & 39 & 37 & 29 & 21 & 24 \\ -39 & -2 & -55 & 23 & -24 & 37 & 29 & 21 \\ -23 & -39 & -2 & -55 & -21 & -24 & 37 & 29 \\ 55 & -23 & -39 & -2 & -29 & -21 & -24 & 37 \end{array}\right].$$

Notice that basis T is significantly shorter than basis B.

8.5 NTRU key pair generation

This section describes a method for generating an NTRU public key h and the corresponding private key (f, g, F, G), where $h = g/f$ in $\mathbb{Z}_q[x]/(x^n + 1)$ (which requires f and $x^n + 1$ to be coprime in $\mathbb{Z}_q[x]$) and $f, g, F, G \in \mathbb{Z}[x]/(x^n + 1)$ are short polynomials satisfying the NTRU equation $fG - gF = q$ modulo $x^n + 1$.

One might try to select short f and g, verify that $\gcd(f, x^n+1) = 1$ in $\mathbb{Z}_q[x]$, and then find $F', G' \in R$ such that $fG' - gF' = 1$; multiplying F' and G' by q would then solve the NTRU equation. But R is not a polynomial ring over a field, so the extended Euclidean algorithm for polynomials cannot be applied directly to f and g. Instead, Algorithm **??** descends to $\mathbb{Z}$, where the extended Euclidean algorithm is available, by passing from f and g to the integers R_f and R_g defined below.

Preliminaries. Let $R = \mathbb{Z}[x]/(x^n + 1)$, $R_q = \mathbb{Z}_q[x]/(x^n + 1)$, and $Q = \mathbb{Q}[x]/(x^n + 1)$. Since n is a power of two, $x^n + 1$ is irreducible over $\mathbb{Q}$. Thus, Q is a field, called a *cyclotomic field*. For $f = f_0 + f_1 x + \cdots + f_{n-1}x^{n-1} \in Q$, let $\|f\| = (f_0^2 + f_1^2 + \cdots + f_{n-1}^2)^{1/2}$ denote the Euclidean norm of its coefficient vector, and for a pair $(f, g) \in Q^2$, let $\|(f, g)\| = (\|f\|^2 + \|g\|^2)^{1/2}$. The *adjoint* of $f \in Q$ is

$$f^* = f(1/x) \bmod (x^n + 1) = f_0 - f_{n-1}x - f_{n-2}x^2 - \cdots - f_1 x^{n-1} \in Q,$$

the unique element of Q satisfying $f^*(\zeta) = \overline{f(\zeta)}$ for every complex root ζ of $x^n + 1$. The adjoint is additive and multiplicative, and $(ff^*)(\zeta) = |f(\zeta)|^2 \geq 0$. In particular, $d = ff^* + gg^*$ satisfies $d(\zeta) = |f(\zeta)|^2 + |g(\zeta)|^2 > 0$ for $(f, g) \neq (0, 0)$ (a nonzero element of Q cannot vanish at ζ, since $x^n + 1$ is irreducible over $\mathbb{Q}$), and hence d is invertible in Q. See Exercises 8.17 and 8.18 for proofs of these statements.

NTRU key generation. Algorithm 8.23 produces an NTRU private key consisting of small polynomials $f, g, F, G \in R$ such that $fG - gF = q$ in R, and a corresponding public key $h = g/f \in R_q$. The algorithm repeatedly selects small nonzero $f, g \in R$ until the following four conditions are satisfied:

(N1) $\|(f, g)\| \leq c\sqrt{q}$;

(N2) $\gcd(f, x^n + 1) = 1$ in $\mathbb{Z}_q[x]$;

(N3) $\|(qf^*/d, qg^*/d)\| \leq c\sqrt{q}$, where $d = ff^* + gg^* \in R$; and

(N4) $\gcd(R_f, R_g) = 1$, where R_f and R_g are the smallest positive integers such that $s = R_f f^{-1} \in R$ and $t = R_g g^{-1} \in R$. (Here, f^{-1} and g^{-1} denote the inverses of f and g in Q.)

By construction, $sf = R_f$ and $tg = R_g$. The extended Euclidean algorithm finds integers u and v with $uR_f + vR_g = 1$, and then $F_0 = -qvt \in R$ and $G_0 = qus \in R$ are computed. Combining these identities shows that (F_0, G_0) satisfies the NTRU equation.

Lemma 8.20 $fG_0 - gF_0 = q$ in R.

Proof: We have

$$fG_0 - gF_0 = q(usf + vtg) = q(uR_f + vR_g) = q \pmod{x^n + 1}. \qquad \square$$

While (F_0, G_0) satisfies the NTRU equation, its coefficients are generally very large, comparable in size to R_f and R_g; see Example 8.24. However, $(F_0 - kf, G_0 - kg)$ is also a solution for any $k \in R$ since

$$f(G_0 - kg) - g(F_0 - kf) = fG_0 - gF_0 = q \pmod{x^n + 1}.$$

Lemma 8.21 shows that the exact quotient $\kappa = (f^*F_0 + g^*G_0)/d$ is precisely the element of Q minimizing $\|(F_0 - \lambda f, G_0 - \lambda g)\|$. Step 22 of Algorithm 8.23 sets $k = \lfloor \kappa \rceil \in R$, the coefficient-wise rounding of κ. The resulting pair (F, G) is another solution, now with small coefficients.

Lemma 8.21 Let $f, g \in Q$ with $(f, g) \neq (0, 0)$. For given $F, G \in Q$, the function

$$\Phi(\lambda) = \|F - \lambda f\|^2 + \|G - \lambda g\|^2$$

attains its minimum at the unique point

$$\kappa = \frac{f^*F + g^*G}{ff^* + gg^*}.$$

Proof: Since $d = ff^* + gg^*$ is invertible, $\kappa = (f^*F + g^*G)/d \in Q$ is well-defined. Set $r_1 = F - \kappa f$ and $r_2 = G - \kappa g$, so that $\Phi(\kappa) = \|r_1\|^2 + \|r_2\|^2$. By definition of κ,

$$f^*r_1 + g^*r_2 = (f^*F + g^*G) - (ff^* + gg^*)\kappa = (f^*F + g^*G) - d\kappa = 0.$$

For any $\lambda \in Q$, let $h = \lambda - \kappa$, so $F - \lambda f = r_1 - fh$ and $G - \lambda g = r_2 - gh$. Then

$$\begin{aligned}\Phi(\lambda) &= \|r_1 - fh\|^2 + \|r_2 - gh\|^2 \\ &= \|r_1\|^2 + \|r_2\|^2 - 2(\langle r_1, fh\rangle + \langle r_2, gh\rangle) + \|fh\|^2 + \|gh\|^2.\end{aligned}$$

For the cross term, apply symmetry of $\langle\cdot,\cdot\rangle$ and the adjoint identity (Exercise 8.19(b)) to get

$$\langle r_1, fh\rangle + \langle r_2, gh\rangle = \langle fh, r_1\rangle + \langle gh, r_2\rangle = \langle h, f^*r_1\rangle + \langle h, g^*r_2\rangle = \langle h, f^*r_1 + g^*r_2\rangle = 0.$$

Therefore

$$\Phi(\lambda) = \Phi(\kappa) + \|f(\lambda - \kappa)\|^2 + \|g(\lambda - \kappa)\|^2. \tag{53}$$

The two added terms are ≥ 0, so $\Phi(\lambda) \geq \Phi(\kappa)$ for every $\lambda \in Q$; thus κ is a minimizer.

If $\Phi(\lambda) = \Phi(\kappa)$, then $f(\lambda - \kappa) = g(\lambda - \kappa) = 0$, so with $h = \lambda - \kappa$ and commutativity,

$$dh = (ff^* + gg^*)h = f^*(fh) + g^*(gh) = 0,$$

and invertibility of d implies that $h = 0$, so $\lambda = \kappa$. This proves that κ is the unique minimizer. □

The following lemma quantifies the size of (F, G).

Lemma 8.22 Suppose that $f, g \in R$ satisfy (N1) and (N3), that is $\|(f, g)\| \leq c\sqrt{q}$ and $\|(qf^*/d,\ qg^*/d)\| \leq c\sqrt{q}$ where $d = ff^* + gg^*$. Then the polynomials F, G returned by Algorithm 8.23 satisfy

$$\|(F, G)\| \leq \left(\frac{n}{2} + 1\right) c\,\sqrt{q}.$$

Proof: Let $\varepsilon = \kappa - k$, where $\kappa = (f^*F_0 + g^*G_0)/d$ and $k = \lfloor \kappa \rceil$ as in steps 21 and 22. Since k rounds each coefficient of κ to the nearest integer, every coefficient of ε lies in $[-\frac{1}{2}, \frac{1}{2}]$.

We first derive an explicit expression for (F, G). From $fG_0 - gF_0 = q$, we obtain

$$F_0 - \kappa f \;=\; \frac{F_0(ff^* + gg^*) - (f^*F_0 + g^*G_0)f}{d} \;=\; \frac{g^*(gF_0 - fG_0)}{d} \;=\; -\frac{qg^*}{d},$$

and similarly $G_0 - \kappa g = qf^*/d$. Since $F = F_0 - kf = (F_0 - \kappa f) + \varepsilon f$ and similarly for G, we obtain

$$(F, G) \;=\; \varepsilon \cdot (f, g) \;+\; \Big(-\frac{qg^*}{d},\; \frac{qf^*}{d}\Big). \tag{54}$$

We bound the norms of the two terms of (54) separately. For the first term, multiplication by x^i modulo $x^n + 1$ permutes coefficients up to sign, so $\|x^i f\| = \|f\|$ for all i. Writing $\varepsilon = \sum_i \varepsilon_i x^i$, the triangle inequality gives

$$\|\varepsilon f\| \;=\; \|\sum_i \varepsilon_i x^i f\| \;\le\; \sum_i |\varepsilon_i|\,\|x^i f\| \;\le\; \frac{n}{2}\,\|f\|,$$

and likewise $\|\varepsilon g\| \le \frac{n}{2}\|g\|$. Hence, by (N1),

$$\|\varepsilon \cdot (f, g)\|^2 \;=\; \|\varepsilon f\|^2 + \|\varepsilon g\|^2 \;\le\; \Big(\frac{n}{2}\Big)^2 \|(f, g)\|^2 \;\le\; \Big(\frac{n}{2}\Big)^2 c^2 q.$$

For the second term, negating a coordinate does not change the norm, so by (N3),

$$\Big\|\Big(-\frac{qg^*}{d},\; \frac{qf^*}{d}\Big)\Big\| \;=\; \Big\|\Big(\frac{qf^*}{d},\; \frac{qg^*}{d}\Big)\Big\| \;\le\; c\sqrt{q}.$$

Applying the triangle inequality to (54) gives

$$\|(F, G)\| \;\le\; \frac{n}{2}\, c\sqrt{q} + c\sqrt{q} \;=\; \Big(\frac{n}{2} + 1\Big) c\sqrt{q}. \qquad \square$$

Example 8.24 (*NTRU key generation*) Let $n = 4$ and $q = 99961$, so $R = \mathbb{Z}[x]/(x^4 + 1)$, $R_q = \mathbb{Z}_{99961}[x]/(x^4 + 1)$, and $Q = \mathbb{Q}[x]/(x^4 + 1)$. Let $c = 1.33$. Select short polynomials

$$f = -131 - 128x + 95x^2 + 177x^3 \in R \quad \text{and} \quad g = -171 + 158x - 46x^2 - 213x^3 \in R,$$

with $\|(f, g)\| \approx 419 \approx 1.325\sqrt{q}$. We have

$$d = ff^* + gg^* = 175589 - 16301x + 16301x^3 \in R,$$

$$d_{\text{inv}} = d^{-1} = \frac{175589}{30300051719} + \frac{16301}{30300051719}x - \frac{16301}{30300051719}x^3 \in Q.$$

Then $m_1 = qf^* d_{\text{inv}}$ and $m_2 = qg^* d_{\text{inv}}$ in Q are

$$m_1 = -\frac{2796305415404}{30300051719} - \frac{3474972132119}{30300051719}x - \frac{1747288691544}{30300051719}x^2 + \frac{329331910444}{4328578817}x^3 \in Q$$

and

$$m_2 = -\frac{2396869656128}{30300051719} + \frac{3534904049552}{30300051719}x + \frac{897014927689}{30300051719}x^2 - \frac{2419630475945}{30300051719}x^3 \in Q,$$

with $\|(m_1, m_2)\| \approx 240.63 \approx 0.761\sqrt{q}$.

Algorithm 8.23: NTRU key pair generation

Input: n a power of two, a prime q, and a constant c.
Output: Private key $(f, g, F, G) \in R^4$ and public key $h \in R_q$.

Ensure (N1)
  Randomly select $f, g \in R \setminus \{0\}$ so that $\|(f, g)\| \leq c\sqrt{q}$.
Ensure (N2)
  Compute $\gcd(f, x^n + 1)$ in $\mathbb{Z}_q[x]$; if the gcd is not 1 then go to step 2.
Ensure (N3)
  Compute $d = ff^* + gg^*$. // $d \in R$
  Compute $d_{\text{inv}} = d^{-1} \bmod (x^n + 1)$. // $d_{\text{inv}} \in Q$
  Compute $m_1 = qf^* d_{\text{inv}} \bmod (x^n + 1)$. // $m_1 \in Q$
  Compute $m_2 = qg^* d_{\text{inv}} \bmod (x^n + 1)$. // $m_2 \in Q$
  If $\|(m_1, m_2)\| > c\sqrt{q}$ then go to step 2.
Ensure (N4)
  Compute $f_{\text{inv}} = f^{-1} \bmod (x^n + 1)$. // $f_{\text{inv}} \in Q$
  Let R_f be the lcm of the denominators of the coefficients of f_{inv}. // $R_f \in \mathbb{Z}$
  Compute $s = R_f f_{\text{inv}}$. // $s \in R$
  Compute $g_{\text{inv}} = g^{-1} \bmod (x^n + 1)$. // $g_{\text{inv}} \in Q$
  Let R_g be the lcm of the denominators of the coefficients of g_{inv}. // $R_g \in \mathbb{Z}$
  Compute $t = R_g g_{\text{inv}}$. // $t \in R$
  Use EEA to find integers u, v such that $uR_f + vR_g = \gcd(R_f, R_g)$.
  If $\gcd(R_f, R_g) \neq 1$ then go to step 2.
Set $F_0 = -qvt$ and $G_0 = qus$. // $fG_0 - gF_0 = q$ in R
Compute $\kappa = (f^* F_0 + g^* G_0)/d$. // $\kappa \in Q$
Compute $k = \lfloor \kappa \rceil$. // $k \in R$
Set $F = F_0 - kf \bmod (x^n + 1)$ and $G = G_0 - kg \bmod (x^n + 1)$. // $F, G \in R$
Compute $h = g/f$ in R_q.
return (f, g, F, G, h)

Next we have

$$f_{\text{inv}} = f^{-1} = \frac{1308427}{493652667} - \frac{999421}{493652667}x - \frac{1611505}{493652667}x^2 + \frac{872568}{164550889}x^3 \in Q,$$

$R_f = 493652667$, and $s = R_f f_{\text{inv}} = 1308427 - 999421x - 1611505x^2 + 2617704x^3 \in R$; and

$$g_{\text{inv}} = g^{-1} = \frac{5208991}{2920556258} + \frac{14046095}{2920556258}x + \frac{8027845}{2920556258}x^2 - \frac{2849333}{2920556258}x^3 \in Q,$$

$R_g = 2920556258$, and $t = R_g g_{\text{inv}} = 5208991 + 14046095x + 8027845x^2 - 2849333x^3 \in R$. Since $\gcd(R_f, R_g) = 1$, EEA gives $uR_f + vR_g = 1$ with

$$u = -1142803437 \text{ and } v = 193164560.$$

Then

$$\begin{aligned} F_0 = -qvt = &- 100580003950168200560 - 271214960936664665200x \\ &- 155009037606580245200x^2 + 550175503077936992 80x^3 \in R \end{aligned}$$

and

$$\begin{aligned} G_0 = qus = &- 149469171546326019639 + 114169631852599110897x \\ &+ 184091521569611535285x^2 - 299035443500863102728x^3 \in R. \end{aligned}$$

One can verify that $fG_0 - gF_0 = q$ in R.

Finally, $\kappa = (f^* F_0 + g^* G_0)/d$ in Q is

$$\begin{aligned} \kappa = &\frac{2354182844966361616988829 2428}{30300051719} + \frac{1473628597520956914648078 7853}{30300051719}x \\ &+ \frac{5297366598859273357600093553}{30300051719}x^2 + \frac{2459353632673701287531593 9272}{30300051719}x^3 \in Q, \end{aligned}$$

and rounding gives

$$\begin{aligned} k = \lfloor \kappa \rceil = &776956708456752855 + 486345241647525293x \\ &+ 174830282403032929x^2 + 811666480137238504x^3 \in R, \end{aligned}$$

and

$$F = F_0 - kf = 49 - 64x - 14x^2 + 46x^3 \in R$$

and

$$G = G_0 - kg = 55 - 151x - 172x^2 + 267x^3 \in R.$$

One can verify that $fG - gF = q$ in R and $\|(F, G)\| \approx 368 \approx 1.164\sqrt{q}$.

Remark 8.25 (*efficiency of NTRU key pair generation*) Algorithm 8.23 is slow in practice. As illustrated in Example 8.24, the bottleneck is the enormous size of the coefficients of intermediate polynomials such as F_0 and G_0. Falcon instead uses the Pornin-Prest tower algorithm, which is roughly 100× faster for $n = 1024$ and uses only about 30 kilobytes of RAM. The idea is to project the NTRU equation to $\mathbb{Q}$ via the field norm, solve it there, and lift the solution through the tower of subfields

$$\mathbb{Q} \subseteq \mathbb{Q}[x]/(x^2+1) \subseteq \mathbb{Q}[x]/(x^4+1) \subseteq \cdots \subseteq \mathbb{Q}[x]/(x^{2^{\ell-1}}+1) \subseteq \mathbb{Q}[x]/(x^{2^\ell}+1)$$

where $n = 2^\ell$. At each level, Babai's reduction keeps coefficients small, and FFT-based floating-point arithmetic makes this efficient.

8.6 Falcon outline

Falcon follows the GPV framework for lattice-based hash-and-sign signature schemes. The key elements of Falcon's design are: (i) the use of NTRU lattices, which enable compact public keys and signatures; (ii) a discrete Gaussian sampler utilizing Fast Fourier Transform (FFT) arithmetic; and (iii) the specification of concrete parameters tailored to meet specific security levels.

The intricate implementation details of Falcon, particularly the mechanics of the FFT sampler, lie beyond our immediate scope. This section provides a high-level outline of Falcon, aiming to provide a solid foundation for a reader who wishes to embark on a detailed study of the full specification.

Notation. The Falcon specification and its related literature use a slightly modified definition of a Gaussian function compared to (45):

$$\rho_{\sigma,c}(x) = \exp\left(-\frac{\|x-c\|^2}{2\sigma^2}\right), \tag{55}$$

where σ is the standard deviation; see (47). For the remainder of this subsection, we adopt this modified definition (55).

Preliminaries. Falcon uses the 14-bit prime $q = 12289$, and $n = 512$ or $n = 1024$ for the 128-bit and 256-bit security levels, respectively (see Table 8.2). These parameters define the polynomial rings $R = \mathbb{Z}[x]/(x^n+1)$, $R_q = \mathbb{Z}_q[x]/(x^n+1)$ and $Q = \mathbb{Q}[x]/(x^n+1)$. Since $q - 1 = 2^{12} \times 3$ is divisible by $2n$, the NTT enables fast polynomial multiplication in R_q. The public key is a polynomial $h \in R_q$, whereas the corresponding private key is comprised of four short polynomials f, g, F and $G \in R$ with $h = g/f$ in R_q and $fG - gF = q$ in R. The polynomial h determines the matrix

$$A = [I_n \mid \mathcal{A}(h)] \in \mathbb{Z}^{n \times 2n},$$

which in turn defines the NTRU lattice

$$L_A^\perp = \{z \in \mathbb{Z}^{2n} \ : \ Az = 0 \pmod q\}.$$

The secret polynomials f, g, F and G yield a short basis of $L_A^\perp$:

$$T = \begin{bmatrix} \mathcal{A}(g) & \mathcal{A}(G) \\ -\mathcal{A}(f) & -\mathcal{A}(F) \end{bmatrix} \in \mathbb{Z}^{2n \times 2n}.$$

Parameter	Description	Falcon-512	Falcon-1024
λ	security level	128	256
q	prime modulus	12289	12289
n	$R_q = \mathbb{Z}_q[x]/(x^n + 1)$	512	1024
σ_1	std. deviation (for f_i, g_i)	$1.17\sqrt{q/2n}$	$1.17\sqrt{q/2n}$
σ	std. deviation (for z)	165.736617183	168.388571447
Q_s	max. signing queries	2^{64}	2^{64}
$\lfloor \beta^2 \rfloor$	squared acceptance bound	34,034,726	70,265,242

Table 8.2: Parameter sets for Falcon (FN-DSA).

The Falcon signature scheme is outlined in Algorithm 8.26. Below, we provide some clarifying notes regarding its implementation.

Key generation.

1. *Generating f and g.* In step 2 of key generation, the coefficients of f and g are sampled from the discrete Gaussian distribution $D_{\mathbb{Z},\sigma_1}$, where the standard deviation $\sigma_1 = 1.17\sqrt{q/2n}$ ensures that the expected value of $\|(f, g)\|$ is $\approx 1.17\sqrt{q}$. The public key $h = g/f$ is assumed to be computationally indistinguishable from a uniformly random element of R_q (the *NTRU assumption*).

2. *Bounding γ.* The bound $\gamma \leq 1.17\sqrt{q}$ ensures that f, g, F and G are short polynomials (see Lemma 8.22), but more importantly guarantees that $\tilde{T}$ is a short basis, i.e., $\|\tilde{T}\| \leq 1.17\sqrt{q}$ (Exercise 8.20).

3. *Solving the NTRU equation.* As mentioned in Remark 8.25, the NTRU equation in step 7 of key generation is solved using the Pornin-Prest tower algorithm, yielding $F, G \in R$ such that $\|(F, G)\| = O(n\sqrt{q})$.

Signature generation. To sign a message M, the signer finds two short polynomials $s_1, s_2 \in R_q$ that satisfy the equation $s_1 + s_2 h = u$ in R_q, where $u = H(h, r, M) \in R_q$ and r is a 320-bit salt.

1. *Salt length.* The 320-bit salt r ensures that the hash target u varies even for identical messages. This prevents multi-target attacks up to $Q_s = 2^{64}$ signatures, sustaining a security level of λ bits (see Table 8.2).

Algorithm 8.26: Falcon signature scheme

Domain parameters. $q = 12289$, $n \in \{512, 1024\}$, $\sigma_1 = 1.17\sqrt{q/2n}$, σ and β as per Table 8.2, $H : \{0,1\}^* \to R_q$.

Key generation. Alice does the following:

1 **for** $i = 0$ **to** $n-1$ **do**
2 $\quad f_i \leftarrow D_{\mathbb{Z},\sigma_1},\ g_i \leftarrow D_{\mathbb{Z},\sigma_1}$.
3 $f \leftarrow f_0 + f_1 x + \cdots + f_{n-1}x^{n-1}, \quad g \leftarrow g_0 + g_1 x + \cdots + g_{n-1}x^{n-1}$.
4 If $\gcd(f, x^n + 1) \neq 1$ in $\mathbb{Z}_q[x]$ then go to step 1.
5 Compute $\gamma = \max\left\{\|(f,g)\|,\ \|\left(\frac{qf^*}{ff^*+gg^*}, \frac{qg^*}{ff^*+gg^*}\right)\|\right\}$.
6 If $\gamma > 1.17\sqrt{q}$ then go to step 1.
7 Use the Pornin-Prest tower algorithm to compute $F, G \in R$ such that $fG - gF = q$ in R and $\|(F,G)\| = O(n\sqrt{q})$.
8 Compute $h = g/f$ in R_q.
9 Alice's verification (public) key is h; her signing (private) key is (f, g, F, G).

Signature generation. To sign a message $M \in \{0,1\}^*$, Alice does the following:

1 **repeat**
2 $\quad$ Select $r \in_R \{0,1\}^{320}$ and compute $u = H(h, r, M)$.
3 $\quad$ Set $t = (u, 0)$.
4 $\quad$ Sample $z \sim D_{L_A^\perp,\sigma,t}$ using the Ducas-Prest algorithm.
5 $\quad$ Compute $s = t - z$. // $s = (s_1, s_2)$
6 **until** $\|s\|^2 \leq \lfloor \beta^2 \rfloor$
7 **return** (sig $= (r, s_2)$).

Signature verification. To verify Alice's signature sig $= (r, s_2)$ on M, Bob does:

1 Obtain an authentic copy of Alice's verification key h.
2 Compute $u = H(h, r, M)$.
3 Compute $s_1 = u - s_2 h$ in R_q. // coefficients in symmetric mod q representation
4 Accept if and only if $\|(s_1, s_2)\|^2 \leq \lfloor \beta^2 \rfloor$.

2. *The target vector t.* The vector $t = (u, 0)$ in step 3 of signature generation satisfies $At = u \pmod q$ since

$$At = [I_n \mid \mathcal{A}(h)] \begin{bmatrix} u \\ 0 \end{bmatrix} = u + 0 = u.$$

3. *Standard deviation σ.* Let $\varepsilon = 1/\sqrt{Q_s \lambda}$ and

$$\sigma = \frac{1}{\pi} \sqrt{\frac{\ln(4n(1 + 1/\varepsilon))}{2}} \cdot 1.17\sqrt{q}, \tag{56}$$

so the Gaussian width is

$$\bar{s} = \sigma\sqrt{2\pi} = \sqrt{\frac{\ln(4n(1 + 1/\varepsilon))}{\pi}} \cdot 1.17\sqrt{q}.$$

Since $\|\tilde{T}\| \le 1.17\sqrt{q}$, Theorem 8.12 gives $\bar{s} \ge \eta_\varepsilon(L_A^\perp)$.

4. *Gaussian sampling.* Sampling from the discrete Gaussian distribution $D_{L_A^\perp, \sigma, t}$ in step 4 of signature generation uses the Ducas-Prest FFT-based sampler. This uses the *Falcon tree*, an FFT representation of the Gram-Schmidt data of the secret basis T precomputed during key generation (see §8.7).

5. *The acceptance bound β.* Falcon sets the bound on $\|(s_1, s_2)\|$ to be

$$\beta = 1.1 \times \sigma\sqrt{2n}. \tag{57}$$

The signature $s = t - z$ is a vector in $\mathbb{Z}^{2n}$ whose coordinates each have standard deviation approximately σ, so $\|s\|^2 \approx 2n\sigma^2$ on average. Moreover, since $\|s\|^2$ is a sum of $2n$ terms, it is tightly concentrated around this mean when n is large. Hence a sampled signature satisfies $\|s\|^2 \le \lfloor \beta^2 \rfloor$ with very high probability, and the signing loop typically requires only one iteration. Evaluating the condition against the floored integer $\lfloor \beta^2 \rfloor$ allows the check to be executed using integer arithmetic.

6. *Signature compression.* To minimize signature size, s_1 is omitted; the verifier reconstructs it as $s_1 = u - s_2 h$ with coefficients in symmetric mod q representation. Falcon also applies a specialized compression technique to s_2 (details omitted).

Signature verification.

1. *Verification check.* The verifier computes $s_1 = u - s_2 h \pmod q$. By checking that $\|(s_1, s_2)\|^2 \le \lfloor \beta^2 \rfloor$, the verifier simultaneously confirms that s is a short vector and that it is a valid solution to the inhomogeneous equation $As = u \pmod q$.

Table 8.3 compares the public key and signature sizes of Falcon against those of Dilithium, alongside their operational timings. Falcon has smaller public keys and significantly smaller signatures, but slower signing, greater implementation complexity, and more demanding side-channel protection. FFT is used in key generation and signing, but verification uses only integer arithmetic.

signature scheme	parameter set	security level	public key (bytes)	signature (bytes)	sign[a] (μs)	verify (μs)
Falcon	512	NIST 1	897	666	224.4	23.3
Dilithium	ML-DSA-44	NIST 2	1,312	2,420	61.9	27.2
Falcon	1024	NIST 5	1,793	1,280	448.2	44.9
Dilithium	ML-DSA-87	NIST 5	2,592	4,627	136.0	73.0

[a]Signature generation and verification times are taken from the PQShield NIST signatures zoo (https://pqshield.github.io/nist-sigs-zoo). The timings were obtained using a 12th Gen Intel(R) Core(TM) i7-12650H (8 cores, 2 threads/core, 4600 MHz max.)

Table 8.3: Size and performance comparison of Dilithium and Falcon (with floating-point arithmetic).

Security. Falcon has been proven existentially unforgeable under chosen-message attack (Definition A.7), assuming hardness of the multi-target inhomogeneous SIS problem over NTRU lattices[15], and modelling H as a random oracle. A concrete security analysis of Falcon against known lattice attacks is provided in §11.7.

8.7 The Fast Fourier Transform

Much of Falcon's computational efficiency stems from performing discrete Gaussian sampling with the Fast Fourier Transform (FFT) in floating-point arithmetic; this is also the root of the side-channel vulnerabilities mentioned in the introduction. This section overviews why and how Falcon uses the FFT.

FFT definition. The Number Theoretic Transform (NTT), presented in detail in §13, provides efficient polynomial multiplication in $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ when n is a power of two. In contrast, the FFT multiplies polynomials in the real algebra $\mathbb{R}[x]/(x^n + 1)$ and, by restriction, in the subrings $Q = \mathbb{Q}[x]/(x^n + 1)$ and $R = \mathbb{Z}[x]/(x^n + 1)$.

Definition 8.27 Let $n = 2^k$ and let $\zeta = e^{i\pi/n}$ be a complex primitive $2n$-th root of unity. The *Discrete Fourier Transform* (DFT) is the map $\mathrm{DFT} : \mathbb{R}[x]/(x^n + 1) \to \mathbb{C}^n$ defined by

$$\mathrm{DFT}(f) = (f(\zeta), f(\zeta^3), f(\zeta^5), \ldots, f(\zeta^{2n-1})).$$

Note that the odd powers of ζ are precisely the complex roots of $x^n + 1$. For notational convenience, we often write $\hat{f}$ in place of $\mathrm{DFT}(f)$. Addition, subtraction, and multiplication of polynomials in their DFT representation is componentwise, requiring $O(n)$ arithmetic operations. Since DFT is invertible on its image, polynomials can be converted between coefficient and DFT representations.

[15]In this problem, one is given an NTRU public key h and ℓ targets $u_1, \ldots, u_\ell \in_R R_q$, and has to find a pair of short polynomials $s_1, s_2 \in R$ and an index j such that $s_1 + s_2 h = u_j$ in R_q.

The Fast Fourier Transform is an $O(n \log n)$ algorithm for computing both DFT and DFT^{-1}. Combined with componentwise multiplication, this yields an overall $O(n \log n)$ algorithm for multiplying polynomials $f, g \in \mathbb{R}[x]/(x^n + 1)$:

$$f \cdot g = \mathrm{DFT}^{-1}\left(\mathrm{DFT}(f) \cdot \mathrm{DFT}(g)\right).$$

Since the DFT outputs complex numbers, their real and imaginary parts must be approximated in practice using floating-point numbers. Falcon utilizes standard IEEE 754 double-precision (64-bit) floating-point numbers for these computations.

FFT sampler. Discrete Gaussian sampling $z \sim D_{L_A^\perp, \sigma, t}$ in signature generation uses Klein's algorithm, which draws z one coordinate at a time along the Gram-Schmidt orthogonalization $\tilde{T}$ of T. Each coordinate is drawn from a one-dimensional discrete Gaussian with center and width determined by $\tilde{T}$. The difficulty is that Gram-Schmidt vectors, and hence per-coordinate centers and widths, are *real* numbers with no small exact representation. Thus $\tilde{T}$ is stored approximately as the Falcon tree, and the sampler uses IEEE 754 double-precision floating-point arithmetic.

8.8 Notes and further references

The security of the RSA full-domain hash (FDH) signature scheme in the random oracle model was analyzed by Bellare and Rogaway [28]. NTRUSign was designed by Hoffstein, Howgrave-Graham, Pipher, Silverman and Whyte [116]. Nguyen and Regev [173] demonstrated attacks on both NTRUSign and an earlier lattice-based signature scheme by Goldreich, Goldwasser and Halevi [103]. Their attack reduced the problem of recovering a secret parallelepiped from many random points uniformly distributed within it to a multivariate optimization problem, which they solved using gradient descent. Ducas and Nguyen [78] showed that two proposed countermeasures failed to adequately mask the geometric structure of the secret basis.

Micciancio and Regev [163] introduced the smoothing parameter. Theorems 8.7, 8.10 and 8.11, correspond to footnote 12, Lemma 4.1, and Lemma 4.4 of their paper, respectively. Theorem 8.12 is due to Gentry, Peikert and Vaikuntanathan [98, 99]. Algorithms for sampling from discrete Gaussian distributions on lattices were surveyed by Dwarakanath and Galbraith [82].

The notion of a sampleable trapdoor one-way function was formulated by Gentry, Peikert and Vaikuntanathan (GPV) [98, 99], who showed how to construct these primitives from lattices for use in hash-and-sign signature schemes and identity-based encryption schemes, and proved the claims in Remark 8.14. The GPV signature scheme uses Ajtai's algorithm [5] for selecting (A, T) and Klein's algorithm [131] for sampling from discrete Gaussians. Alwen and Peikert [15] improved Ajtai's algorithm, reducing the size of the Gram-Schmidt orthogonalization $\tilde{T}$ of the trapdoor basis T from $m^{2.5}$ to $O(\sqrt{m})$.

Gentry, Peikert and Vaikuntanathan gave a tight proof (cf. Remark 8.14) that their signature scheme is strongly secure in the random oracle model (ROM), where the hash

function H is modelled as a uniformly random oracle accessible to all parties. Boneh et al. [34] subsequently proved its security in the quantum random oracle model (QROM), where the adversary may query H in quantum superposition.

Micciancio and Peikert [162] introduced lattice-based trapdoor functions that are conceptually simpler and more efficient than the original GPV construction. Unlike GPV, which samples preimages via a randomized nearest-plane procedure, the Micciancio-Peikert method employs a public "gadget" $m \times n$ matrix G (with $m > n$) for which preimages of the SIS function $f_{G^T}(x) = G^T x \pmod q$ can be efficiently sampled from a discrete Gaussian distribution, and the LWE function $g_G(s, e) = Gs + e \pmod q$ can be efficiently inverted. The secret trapdoor is a short matrix R that relates a random-looking public matrix A to the structured gadget matrix G, enabling efficient preimage sampling or inversion while ensuring that the corresponding SIS problem for A^T and the LWE problem for A remain computationally hard. Three prominent applications are the attribute-based encryption scheme for circuits by Gorbunov, Vaikuntanathan and Wee [105], the predicate encryption scheme of Gorbunov, Vaikuntanathan and Wee [106], and short lattice-based signatures in the standard model by Ducas and Micciancio [77].

The NTRU public-key encryption scheme and the notion of an NTRU lattice were proposed in 1996 by Hoffstein, Pipher and Silverman [117]. In the original NTRU design, the secret polynomials f and g are chosen from the ring $\mathbb{Z}_q[x]/(x^n - 1)$ with coefficients in $\{-1, 0, 1\}$. Stehlé and Steinfeld [208] proposed instead using the power-of-two cyclotomic ring $R_q = \mathbb{Z}_q[x]/(x^n + 1)$, where the coefficients of $f, g \in \mathbb{Z}[x]/(x^n + 1)$ are sampled from a discrete Gaussian distribution of width $\mathrm{poly}(n) \cdot q^{1/2+\varepsilon}$. They proved that the resulting public key $h = g/f \in R_q$ is statistically indistinguishable from the uniform distribution over R_q. Algorithm 8.23 for NTRU key pair generation is derived from the NTRUSign paper by Hoffstein et al. [116].

Ducas, Lyubashevsky and Prest [76] realized the GPV signature scheme (and identity-based encryption scheme) using NTRU lattices, demonstrating for the first time that the GPV signature framework could be made practically efficient with respect to computational speed, public-key size, and signature size.

Ducas and Prest [80] showed how the Cooley-Tukey fast Fourier Transform (FFT) [61] could be adapted to substantially accelerate Klein's algorithm [131] for discrete Gaussian sampling. Klein's algorithm is a randomized variant of Babai's nearest plane algorithm [19] for finding a lattice vector close to a given target. The Ducas-Prest algorithm precomputes the Gram-Schmidt orthogonalization of the trapdoor lattice basis in the form of a tree. Using this tree, discrete Gaussian sampling can be performed in time $O(n \log n)$. A detailed exposition of the FFT-based Gaussian sampling method can be found in Prest's Ph.D. thesis [193].

Falcon was designed by Fouque et al. [89], building on the GPV framework. In Falcon, NTRU key pair generation uses the efficient algorithm of Pornin and Prest [191], which exploits the tower of subfields of $\mathbb{Q}[x]/(x^{2^\ell} + 1)$ as noted in Remark 8.25. This method reduces both computation time and RAM usage by two orders of magnitude compared to prior algebraic approaches. Falcon also employs the aforementioned Ducas-

Prest FFT-based discrete Gaussian sampling method; the FFT-tree representation of the Gram-Schmidt orthogonalization of the trapdoor lattice basis is called the *Falcon tree*. In 2024, Fouque, Gajland, de Groote, Janneck and Kiltz [87, 88] gave the first security proof of Falcon. Their proof required some modifications to the original Falcon specification, one of which was the inclusion of the public key h in the computation of the message digest u. The proof is in the random oracle model and assumes the hardness of the multi-target inhomogeneous SIS problem over NTRU lattices.

Hardware accelerators for Falcon have been designed by Beckwith, Nguyen and Gaj [27] as well as Ouyang et al. [185]. Meanwhile, high-throughput GPU implementations have been developed by Lee et al. [142] and Li et al. [147]. Notably, the latter achieved signing throughputs of 172,000 and 90,000 signatures per second for Falcon-512 and Falcon-1024, respectively, on an NVIDIA A100 (80GB PCIe) GPU.

Several authors have highlighted vulnerabilities of the discrete Gaussian samplers and floating-point arithmetic to side-channel attacks. Fouque, Kirchner, Tibouchi, Wallet and Yu [90] demonstrated timing attacks on Falcon; Guerreau, Martinelli, Ricosset and Rossi [108] alongside Zhang, Lin, Yu and Wang [219] introduced power analysis attacks; and Haidar, Payet and Tibouchi [110] investigated fault-injection attacks. These attacks all target discrete Gaussian sampling and recover the secret key. For instance, the power-analysis attack by Zhang et al. recovers a Falcon-512 secret key in roughly half an hour using power traces from approximately 45,000 signatures. Furthermore, Lin, Tibouchi, Yu and Zhang [148] observed that the sampler is highly sensitive to floating-point errors when executed twice with identical inputs; such a scenario might occur if a signer improperly reuses a salt on the same message.

NIST's proposal to use fixed-point arithmetic in FN-DSA was prompted in part by recent work by De Almeida Braga, Fouque, Lachguel and Prest [69] and Pornin [190]. The fixed-point implementation of De Almeida Braga et al. is roughly twice as slow as the original floating-point Falcon, though an order of magnitude faster than emulated floating point. Under the plan, FIPS 206 would require fixed-point arithmetic for both key generation and signing, initially in 32.32 and 64.64 formats respectively, and would not permit floating-point arithmetic. At the time of writing, NIST had not yet published a draft, so details may change.

8.9 Exercises

Exercise 8.1 (*LWE trapdoor one-way function*) Let $A \in_R \mathbb{Z}_q^{n\times m}$ (where $m \gg n$), and let B be a positive integer with $B \ll q/2$. Define the *LWE function*

$$g_A : \mathbb{Z}_q^n \times [-B, B]^m \longrightarrow \mathbb{Z}_q^m \text{ by } g_A(s, e) = A^T s + e \bmod q.$$

Let $S = [s_1, \ldots, s_m]$ be a short basis of the SIS lattice $L_A^\perp = \{z \in \mathbb{Z}^m : Az = 0 \pmod q\}$. Let $C = \max_i \|s_i\|_2$ denote the length of the longest basis vector, and assume that $BC\sqrt{m} \le (q-1)/2$ (as needed in part (c)).
(a) Show that, with overwhelming probability, g_A is injective but not surjective.
(b) Prove that $AS = 0 \pmod q$.

(c) Consider the following algorithm for inverting g_A using S.
Let $b = g_A(s, e)$ where $s \in \mathbb{Z}_q^n$ and $e \in [-B, B]^m$.

i) Compute $y = S^T b \bmod q$.

ii) Compute $\tilde{e} = (S^{-1})^T y$ where $S^{-1} \in \mathbb{Q}^{m \times m}$. (Note: there is no mod operation.)

iii) Compute $\tilde{s} \in \mathbb{Z}_q^n$ such that $A^T \tilde{s} = b - \tilde{e} \pmod q$.

Prove that $y = S^T e$ (note: there is no mod operation), and hence prove that $e = \tilde{e}$ and $s = \tilde{s}$. Conclude that S is a trapdoor for inverting g_A.

Exercise 8.2 Use the Gaussian integral $\int_{-\infty}^{\infty} e^{-t^2} dt = \sqrt{\pi}$ to show that

$$\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) dx = 1.$$

Exercise 8.3 Show that if $X \sim N(\mu, \sigma^2)$, then $E[X] = \mu$ and $\mathrm{Var}(X) = \sigma^2$.

Exercise 8.4 Show that if $X \sim N(\mu, \sigma^2)$ and $Z = (X - \mu)/\sigma$, then $Z \sim N(0, 1)$.

Exercise 8.5 Use the Poisson summation formula to show that $\rho_s(\mathbb{Z}^n) \approx s^n$.

Exercise 8.6 Let $L \subseteq \mathbb{R}^n$ be a lattice and let $c \in \mathbb{R}^n$ with $2c \in L$. Prove that $D_{L,s,c}(x) = D_{L,s,c}(2c - x)$ for all $x \in L$.

Exercise 8.7 Prove that if $X \sim D_{\mathbb{Z},s}$ then $aX \sim D_{a\mathbb{Z},as}$ for any $a > 0$.

Exercise 8.8 Prove that if $c \in L$ and $X \sim D_{L,s,c}$ then $X - c \sim D_{L,s}$.

Exercise 8.9 Let D and D' be countable sets, and let X, Y, Z be random variables on D.
(a) Prove that $0 \le \Delta(X, Y) \le 1$.
(b) Prove that $\Delta(X, Y) = 0$ if and only if X and Y have the same distribution.
(c) Prove that $\Delta(X, Y) = 1$ if and only if X and Y have disjoint support.
(d) Prove that $\Delta(X, Z) \le \Delta(X, Y) + \Delta(Y, Z)$.
(e) Let $f : D \to D'$ be any function. Prove that $\Delta(f(X), f(Y)) \le \Delta(X, Y)$.

Exercise 8.10 Prove that for any real number $c > 0$, $\eta_\varepsilon(cL) = c\eta_\varepsilon(L)$.

Exercise 8.11 Let L be a sublattice of L'. Prove that $\eta_\varepsilon(L') \le \eta_\varepsilon(L)$.

Exercise 8.12 Theorem 8.7 bounds the density of $D_{s,0} \bmod L$. In this exercise you will show that the same bound holds for an arbitrary center $c \in \mathbb{R}^n$. Let f_c denote the probability density function of $D_{s,c} \bmod L$ on $\mathbb{R}^n/L$, so that $f = f_0$.
(a) Show that $D_{s,c}(x) = D_{s,0}(x - c)$ for all $x \in \mathbb{R}^n$.
(b) Using part (a), show that $f_c(x) = f(x - c)$ for all $x \in \mathbb{R}^n/L$.
(c) Conclude that if $s \ge \eta_\varepsilon(L)$ and $c \in \mathbb{R}^n$, then

$$\frac{1-\varepsilon}{\mathrm{vol}(L)} \le f_c(x) \le \frac{1+\varepsilon}{\mathrm{vol}(L)}, \quad \forall x \in \mathbb{R}^n/L.$$

Exercise 8.13 Let $R = \mathbb{Z}[x]/(x^n+1)$. For $a \in R$, let $M_a : R \to R$ denote the multiplication-by-a map defined by $M_a(f) = a \times f$.
(a) Prove that M_a is $\mathbb{Z}$-linear, i.e., $M_a(f+g) = M_a(f) + M_a(g)$ and $M_a(cf) = cM_a(f)$ for all $f, g \in R$ and $c \in \mathbb{Z}$.
(b) Prove that the matrix of M_a with respect to the basis $[1, x, \ldots, x^{n-1}]$ of R over $\mathbb{Z}$ is the anti-circulant matrix $\mathcal{A}(a)$.
(c) Let $f, g \in R$. Prove that $\mathcal{A}(f \pm g) = \mathcal{A}(f) \pm \mathcal{A}(g)$.
(d) Let $f, g \in R$. Prove that $\mathcal{A}(f \times g) = \mathcal{A}(f) \times \mathcal{A}(g)$.
(e) Prove that $\mathcal{A}(1) = I_n$.
(f) Conclude that $\mathcal{A} : R \to \mathbb{Z}^{n \times n}$ is a ring homomorphism.
(g) Explain why (a)-(f) remain valid with R and $\mathbb{Z}$ replaced by $R_q = \mathbb{Z}_q[x]/(x^n+1)$ and $\mathbb{Z}_q$.

Exercise 8.14 (†) (*NTRU equation*) Let $q = 9929$, $n = 4$, and $R = \mathbb{Z}[x]/(x^4+1)$. For each $(f, g) \in R^2$, find $(F, G) \in R^2$ such that $fG - gF = q$ and $\|(F, G)\| \leq 120$.
(a) $f(x) = 22 + 51x - 26x^2 - 65x^3$, $g(x) = -27 + 29x - 33x^2 - 44x^3$.
(b) $f(x) = 16 + 27x - 18x^2 - 10x^3$, $g(x) = -24 - 43x - 68x^2 + 62x^3$.
(c) $f(x) = 25 - 60x - 48x^2 + 57x^3$, $g(x) = -13 - 4x + 50x^2 - 6x^3$.

Exercise 8.15 (†) (*Falcon-like keys*) Let $q = 13441$. For each of the following polynomials $h \in \mathbb{Z}_q[x]/(x^n+1)$, find polynomials $f, g \in R = \mathbb{Z}[x]/(x^n+1)$ such that $fh = g$ in R_q and $\|(f, g)\| \leq \beta$.
Hint: Use the LLL lattice-basis reduction algorithm.
(a) $n = 3$, $\beta = 100$, $h(x) = 5860 + 12121x + 6906x^2$.
(b) $n = 5$, $\beta = 100$, $h(x) = 11710 + 12199x + 2393x^2 + 12930x^3 + 13104x^4$.
(c) $n = 7$, $\beta = 150$, $h(x) = 9576 + 6768x + 5209x^2 + 7953x^3 + 182x^4 + 12396x^5 + 11303x^6$.
(d) $n = 10$, $\beta = 200$, $h(x) = 8208 + 8645x + 4403x^2 + 2384x^3 + 11990x^4 + 1953x^5 + 5624x^6 + 1236x^7 + 11400x^8 + 7016x^9$.

Exercise 8.16 (†) (*Falcon-like signatures*) Let $q = 13441$. For each pair of polynomials $h, u \in \mathbb{Z}_q[x]/(x^n+1)$, find polynomials $s_1, s_2 \in R = \mathbb{Z}[x]/(x^n+1)$ such that $s_1 + s_2 h = u$ in R_q and $\|(s_1, s_2)\| \leq \beta$.
Hint: Use Kannan's embedding technique with $Q \approx \sqrt{q}$ (Exercise 12.6(a)).
(a) $n = 3$, $\beta = 100$, $h(x) = 450 + 11772x + 12338x^2$, $u(x) = 3789 + 2286x + 9780x^2$.
(b) $n = 5$, $\beta = 120$, $h(x) = 11659 + 11710x + 2393x^2 + 13104x^3 + 9060x^4$,
$u(x) = 9234 + 12199x + 12930x^2 + 11440x^3 + 5476x^4$.
(c) $n = 7$, $\beta = 150$, $h(x) = 8393 + 8649x + 2529x^2 + 7918x^3 + 3368x^4 + 771x^5 + 780x^6$,
$u(x) = 1497 + 154x + 11891x^2 + 129x^3 + 3302x^4 + 4267x^5 + 10386x^6$.
(d) $n = 10$, $\beta = 200$, $h(x) = 1289 + 6905x + 1788x^2 + 4403x^3 + 12696x^4 + 13127x^5 + 6768x^6 + 7953x^7 + 12396x^8 + 8208x^9$,
$u(x) = 8712 + 13067x + 12266x^2 + 1279x^3 + 12022x^4 + 9576x^5 + 5209x^6 + 182x^7 + 11303x^8 + 8645x^9$.

Exercise 8.17 Let $Q = \mathbb{Q}[x]/(x^n + 1)$ and $f = f_0 + f_1 x + \cdots + f_{n-1}x^{n-1} \in Q$. Recall that the adjoint of f is $f^* = f(1/x) \bmod (x^n + 1)$.
(a) Using $x^{-i} = -x^{n-i}$ in Q, prove that $f^* = f_0 - f_{n-1}x - f_{n-2}x^2 - \cdots - f_1 x^{n-1}$.
(b) Prove that the adjoint is an involution, i.e., $(f^*)^* = f$ for every $f \in Q$.
(c) Prove that $f^* = f$ iff $f_k = -f_{n-k}$ for all $1 \le k \le n-1$ (such f are called *self-adjoint*).

Exercise 8.18 Let $f, g \in Q$, and let ζ be a complex root of $x^n + 1$.
(a) Prove that $f^*(\zeta) = \overline{f(\zeta)}$.
(b) Prove that the adjoint is additive and multiplicative,
i.e., $(f + g)^* = f^* + g^*$ and $(fg)^* = f^* g^*$.
(c) Prove that $c^* = c$ for all $c \in \mathbb{Q} \subseteq Q$, and deduce that the adjoint is $\mathbb{Q}$-linear.
(d) Prove that ff^* is self-adjoint for every $f \in Q$, and that $(ff^*)(\zeta) = |f(\zeta)|^2 \ge 0$.

Exercise 8.19 For polynomials $f, g \in Q$, let $\langle f, g \rangle$ denote the inner product of their coefficient vectors, and $(f)_0$ the constant coefficient of f.
(a) Prove that $\langle f, g \rangle = (f^* g)_0$ for all $f, g \in Q$.
(b) Prove that $\langle af, g \rangle = \langle f, a^* g \rangle$ for all $a, f, g \in Q$.

Exercise 8.20 Consider the Falcon trapdoor basis $T = [t_1, \ldots, t_{2n}]$ and its Gram-Schmidt orthogonalization $\tilde{T} = [\tilde{t}_1, \ldots, \tilde{t}_{2n}]$. The $2n$ columns of T are the rotations $x^i(g, -f)$ (the first block) and $x^i(G, -F)$ (the second block), $0 \le i \le n-1$. Let $d = ff^* + gg^*$, and

$$\gamma = \max\left\{\|(f, g)\|,\ \|(qf^*/d,\ qg^*/d)\|\right\}.$$

(a) Prove that the first block's leading Gram-Schmidt vector is $\tilde{t}_1 = (g, -f)$.
(b) Prove that the second block's leading Gram-Schmidt vector is $\tilde{t}_{n+1} = (qf^*/d,\ qg^*/d)$.
(c) Prove that within each block the Gram-Schmidt norms are at most that of the block's leading vector: $\|\tilde{t}_i\| \le \|\tilde{t}_1\|$ for $1 \le i \le n$, and $\|\tilde{t}_i\| \le \|\tilde{t}_{n+1}\|$ for $n + 1 \le i \le 2n$.
Hint: Use the fact that multiplication by x is an isometry and that the span of the first block is x-invariant.
(d) Conclude that $\|\tilde{T}\| = \gamma$, and hence $\|\tilde{T}\| \le 1.17\sqrt{q}$.

Exercise 8.21 Recall $\zeta = e^{i\pi/n}$ from Definition 8.27. Show that $(\zeta^j)^n = (-1)^j$, and conclude that ζ^j is a root of $x^n + 1$ if and only if j is odd. Deduce that the n evaluation points $\zeta, \zeta^3, \ldots, \zeta^{2n-1}$ are pairwise distinct and are *exactly* the roots of $x^n + 1$ in $\mathbb{C}$.

Exercise 8.22 Let $n = 2$, so $\zeta = i$. Write out $\mathrm{DFT}(f)$ explicitly for $f = a_0 + a_1 x$, and give a formula for DFT^{-1}. For $f = a_0 + a_1 x$ and $g = b_0 + b_1 x$, verify the identity

$$f \cdot g = \mathrm{DFT}^{-1}(\mathrm{DFT}(f) \cdot \mathrm{DFT}(g))$$

by computing both sides, and confirm the result agrees with reducing fg modulo $x^2 + 1$.

Exercise 8.23 Let $f \in \mathbb{R}[x]/(x^n + 1)$ where n is a power of two. Write $f = f_0(x^2) + x\, f_1(x^2)$ with f_0, f_1 of degree $< n/2$. Setting $\zeta' = \zeta^2$, show that ζ' is the primitive $2(n/2)$-th root of unity playing the role of ζ for transform size $n/2$, and that evaluating f at the odd powers of ζ reduces to evaluating f_0 and f_1 at the odd powers of ζ'. Using $\zeta^{2(i+n/2)+1} = -\zeta^{2i+1}$, derive the "butterfly" (cf. Figure 13.3(a)) combining the two half-size transforms. Hence derive the recurrence $T(n) = 2\,T(n/2) + O(n)$ for the running times of both DFT and DFT^{-1}, and conclude that $T(n) = O(n \log n)$.

9 FrodoKEM

Contents

9.1 The error distribution 125
9.2 Frodo public-key encryption 128
9.3 FrodoKEM 137
9.4 Notes and further references 140
9.5 Exercises 141

FrodoKEM is a quantum-safe key encapsulation mechanism (KEM) standardized by ISO in 2026. Like Kyber-KEM, FrodoKEM is based on the Lindner-Peikert encryption scheme (§4.2). However, its parameter choices are more conservative. In particular, FrodoKEM relies on the hardness of standard LWE in unstructured lattices, in contrast to Kyber-KEM which relies on the hardness of the module-LWE problem over algebraic lattices. Additionally, FrodoKEM samples its errors from a discrete approximation to a rounded Gaussian rather than from a centered binomial distribution, thereby matching more closely the Gaussian noise of Regev's worst-case to average-case reduction for LWE (§4.4.4).[16]

We begin in §9.1 with the error distribution used by FrodoKEM. We then introduce the underlying public-key encryption scheme, FrodoPKE, in §9.2. Finally, §9.3 presents the full FrodoKEM construction, derived via a suitably modified Fujisaki-Okamoto transformation.

9.1 The error distribution

Recall that the one-dimensional Gaussian function $\rho_s : \mathbb{R} \to \mathbb{R}$ of width $s > 0$ (Definition 8.1) is

$$\rho_s(x) = \exp\left(-\frac{\pi x^2}{s^2}\right).$$

The corresponding one-dimensional continuous Gaussian distribution $D_s : \mathbb{R} \to \mathbb{R}$ of width $s > 0$ (Definition 8.2) is

$$D_s(x) = \frac{\rho_s(x)}{s},$$

whereas the one-dimensional discrete Gaussian distribution $D_{\mathbb{Z},s} : \mathbb{Z} \to \mathbb{R}$ of width $s > 0$ over $\mathbb{Z}$ (Definition 8.3) is

$$D_{\mathbb{Z},s}(x) = \frac{\rho_s(x)}{S},$$

where $S = \rho_s(\mathbb{Z}) \approx s$ is a normalization factor. FrodoKEM's error distribution is a discrete approximation to the rounded Gaussian distribution, which is defined as follows.

[16] As noted in §4.4.4, Regev's reduction is asymptotic, and therefore does not by itself justify FrodoKEM's parameter choices. Gaussian noise is adopted as a design principle rather than as a basis for concrete security estimates.

Definition 9.1 The one-dimensional *rounded Gaussian distribution* $\Psi_s : \mathbb{Z} \to \mathbb{R}$ of width $s > 0$ is defined by

$$\Psi_s(x) = \int_{x-\frac{1}{2}}^{x+\frac{1}{2}} D_s(z)\, dz.$$

Example 9.2 (*one-dimensional rounded Gaussian distribution*) The rounded Gaussian distribution Ψ_5 is shown in Figure 9.1. Notice that Ψ_5 is symmetric about the origin. Observe also that Ψ_5 is not the same as the discrete Gaussian distribution $D_{\mathbb{Z},5}$ (Definition 8.3). For $x \in \mathbb{Z}$, the value $\Psi_5(x)$ is the area under the curve $y = D_5(z)$ between $x-0.5$ and $x + 0.5$, whereas $D_{\mathbb{Z},5}(x)$ is the value $\rho_5(x)/S$ where $S \approx 5$. The values $\Psi_5(x)$ and $D_{\mathbb{Z},5}(x)$ for $0 \le x \le 15$ are shown in Table 9.1.

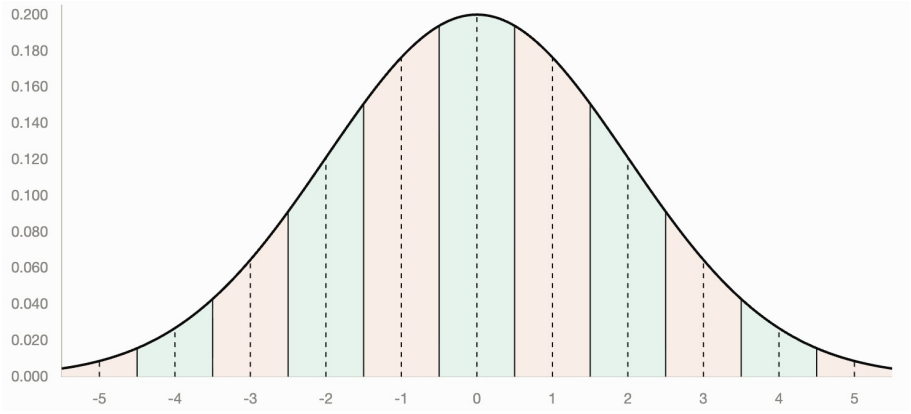


Figure 9.1: The one-dimensional rounded Gaussian distribution Ψ_5.

The approximation to Ψ_s employed by FrodoKEM is the following.

Definition 9.3 The *approximate rounded Gaussian distribution table* $\chi_{s,c}$ of width $s > 0$ and support $\{-c, -c+1, \ldots, 0, \ldots, c-1, c\}$ is defined by

$$\chi_{s,c}(i) = \lfloor 2^{16}\Psi_s(i) \rceil, \quad \text{for } -c \le i \le c. \tag{58}$$

Rounding, together with truncation of the tails beyond $\pm c$, will in general destroy the equality $\sum_{i=-c}^{c} \chi_{s,c}(i) = 2^{16}$. Individual values of $\chi_{s,c}(i)$ are therefore adjusted slightly so that the equality holds exactly and symmetry is preserved.

Thus $\chi_{s,c}$ is Ψ_s scaled by 2^{16}, rounded to integers and restricted to $\{-c, \ldots, c\}$. The associated probability distribution on $\{-c, \ldots, c\}$ is $\chi_{s,c}(i)/2^{16}$. For the sake of simplicity, we will also refer to $\chi_{s,c}$ as an (unnormalized) probability distribution. Note that $\chi_{s,c}$ inherits the symmetry of Ψ_s, so $\chi_{s,c}(-i) = \chi_{s,c}(i)$, and that the FrodoKEM parameters are chosen so that $\chi_{s,c}(0)$ is even. Table 9.2 shows the distributions χ used by the three FrodoKEM parameter sets listed in Table 9.3.

x	$\Psi_5(x)$	$D_{\mathbb{Z},5}(x)$
0	0.1979251973	0.2000000000
1	0.1750084087	0.1763822757
2	0.1209832899	0.1209845126
3	0.0653852325	0.0645437967
4	0.0276240727	0.0267811443
5	0.0091222756	0.0086427837
6	0.0023543380	0.0021693421
7	0.0004748077	0.0004234989
8	0.0000748127	0.0000643023
9	$0.9207909914 \times 10^{-5}$	$0.7593650578 \times 10^{-5}$
10	$0.8850937666 \times 10^{-6}$	$0.6974684712 \times 10^{-6}$
11	$0.6643119008 \times 10^{-7}$	$0.4982513029 \times 10^{-7}$
12	$0.3892421393 \times 10^{-8}$	$0.2768357654 \times 10^{-8}$
13	$0.1780095262 \times 10^{-9}$	$0.1196315814 \times 10^{-9}$
14	$0.6352633925 \times 10^{-11}$	$0.4020863978 \times 10^{-11}$
15	$0.1768745216 \times 10^{-12}$	$0.1051097035 \times 10^{-12}$

Table 9.1: Some values of the one-dimensional rounded Gaussian Ψ_5 and the discrete Gaussian $D_{\mathbb{Z},5}$.

Parameter set	σ	c	0	±1	±2	±3	±4	±5	±6	±7	±8	±9	±10	±11	±12
FrodoKEM-640	2.8	12	9288	8720	7216	5264	3384	1918	958	422	164	56	17	4	1
FrodoKEM-976	2.3	10	11278	10277	7774	4882	2545	1101	396	118	29	6	1		
FrodoKEM-1344	1.4	6	18286	14320	6876	2023	364	40	2						

Table 9.2: The error distributions $\chi = \chi_{s,c}$ used by the three FrodoKEM parameter sets. Each entry is the value $\chi(i)$, so the probability of i is $\chi(i)/2^{16}$, and $\chi(0)+2\sum_{i=1}^{c}\chi(i) = 2^{16}$. Here, σ is the standard deviation of the underlying Gaussian distribution, whose width is $s = \sigma\sqrt{2\pi}$.

To efficiently sample e from $\chi = \chi_{s,c}$, denoted $e \in_\chi \mathbb{Z}_q$, the following table $T = T_\chi$ is precomputed:

$$T(0) = \frac{\chi(0)}{2} - 1, \quad T(z) = T(0) + \sum_{i=1}^{z} \chi(i) \ \text{ for } 1 \le z \le c.$$

By the symmetry of χ, this is half the two-sided cumulative sum, shifted down by one:

$$T(z) = \frac{1}{2} \sum_{i=-z}^{z} \chi(i) - 1,$$

so T is an unnormalized cumulative table for $|i|$, with $T(c) = 2^{15} - 1$. A sample can be obtained by drawing a uniform 15-bit value $(r_1, \ldots, r_{15})$ and a uniform sign bit r_0, and comparing the former against T. This is depicted in Algorithm 9.4, whose main loop performs exactly c comparisons regardless of the value of t. This is deliberate, since a running time that depended on the sampled error would leak information about the secret through a timing side channel.

Algorithm 9.4: Sampling from $\chi_{s,c}$

Input: $T(0), \ldots, T(c-1)$.
Output: $e \in_\chi \mathbb{Z}_q$.

Select $r = (r_0, r_1, \ldots, r_{15}) \in_R \{0,1\}^{16}$.
Set $t = \sum_{i=1}^{15} r_i 2^{i-1}$ and $e = 0$.
for $z = 0$ **to** $c - 1$ **do**
    if $t > T(z)$ **then**
        $e \leftarrow e + 1$.
$e \leftarrow (-1)^{r_0} e$. // r_0 is the sign bit.
return (e).

9.2 Frodo public-key encryption

FrodoPKE encodes messages using a pair of functions Encode and Decode; the latter generalizes the Round_q function used in Kyber (Definition 4.12).

Definition 9.5 Let q be a power of two, and let b be a positive integer satisfying $2^b \le q$. Define the *encoding function* $\mathrm{Encode} : \mathbb{Z}_{2^b} \to \mathbb{Z}_q$ by

$$\mathrm{Encode}(x) = x \cdot \frac{q}{2^b},$$

and the corresponding *decoding function* $\mathrm{Decode} : \mathbb{Z}_q \to \mathbb{Z}_{2^b}$ by

$$\mathrm{Decode}(y) = \left\lfloor y \cdot \frac{2^b}{q} \right\rceil \bmod 2^b.$$

Since 2^b divides q, the quantity $q/2^b$ is an integer and both maps are well defined. Figure 9.2 illustrates the decoding function for the case $q = 16$, $b = 2$.

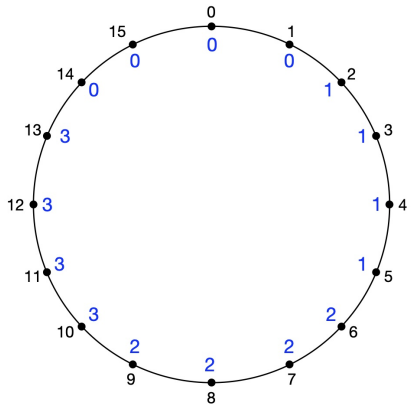


Figure 9.2: FrodoPKE decoding function Decode : $\mathbb{Z}_{16} \to \mathbb{Z}_4$.

The Decode function extends naturally to all integers y:

$$\text{Decode}(y) = \left\lceil (y \bmod q) \cdot \frac{2^b}{q} \right\rfloor \bmod 2^b.$$

The following lemma shows that decoding recovers an encoded value provided the additive error e is small enough.

Lemma 9.6 For all $0 \le x < 2^b$ and $-\frac{q}{2^{b+1}} \le e < \frac{q}{2^{b+1}}$, we have

$$\text{Decode}(\text{Encode}(x) + e) = x.$$

The Encode and Decode functions extend to matrices by applying them componentwise.

9.2.1 FrodoPKE

FrodoPKE is presented in Algorithm 9.7. It is structurally similar to the Kyber public-key encryption scheme (Algorithm 6.1), but is based on standard LWE, working with matrices whose entries are in $\mathbb{Z}_q$ rather than in a polynomial ring. The modulus q is a power of two, namely $q = 2^{15}$ for the FrodoKEM-640 parameter set and $q = 2^{16}$ for the FrodoKEM-976 and FrodoKEM-1344 parameter sets. This choice simplifies both modular reduction and the encoding of messages. See Table 9.3 for the three FrodoKEM parameter sets.

The pseudorandom bit generator ExpandA, instantiated using either SHAKE128 or AES128, produces a matrix in $\mathbb{Z}_q^{n \times n}$ from a 128-bit seed. The pseudorandom bit generator ExpandSE, instantiated using SHAKE128 for the FrodoKEM-640 parameter set and

Algorithm 9.7: Frodo public-key encryption scheme (FrodoPKE)

Domain parameters. q, n, $\overline{n}$, σ, c, b, ℓ.

Key generation. Bob does the following:

1 Select $\text{seed}_A \in_R \{0,1\}^{128}$ and $\text{seed}_{SE} \in_R \{0,1\}^{2\ell}$.
2 Compute $A = \text{ExpandA}(\text{seed}_A)$. // $A \in \mathbb{Z}_q^{n \times n}$
3 Compute $(S, E) = \text{ExpandSE}(\text{seed}_{SE})$. // $S, E \in \Lambda^{n \times \overline{n}}$
4 Compute $B = AS + E$. // $B \in \mathbb{Z}_q^{n \times \overline{n}}$
5 Bob's encryption (public) key is (seed_A, B); his decryption (private) key is S.

Encryption. To encrypt a message $m \in \{0,1\}^{\ell}$ for Bob, Alice does the following:

1 Obtain an authentic copy of Bob's encryption key (seed_A, B).
2 Compute $A = \text{ExpandA}(\text{seed}_A)$.
3 Select $\text{seed}_{RE} \in_R \{0,1\}^{2\ell}$.
4 Compute $(R, E_1, E_2) = \text{ExpandRE}(\text{seed}_{RE})$. // $R, E_1 \in \Lambda^{n \times \overline{n}}$, $E_2 \in \Lambda^{\overline{n} \times \overline{n}}$
5 Compute $U = A^T R + E_1$ and $V = B^T R + E_2 + \text{Encode}(m)$. // see Figure 9.3
6 **return** $(C = (U, V))$.

Decryption.

1 To decrypt $C = (U, V)$, Bob does the following:
2 Compute $M = V - S^T U$ and $m = \text{Decode}(M)$.
3 **return** (m).

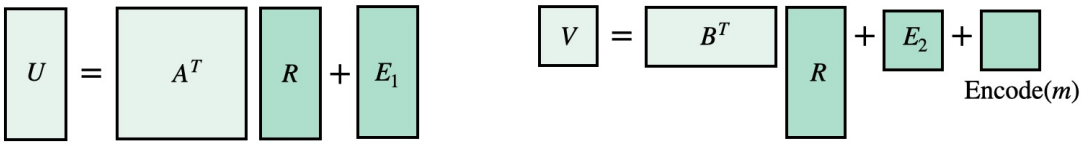


Figure 9.3: FrodoPKE encryption.

Parameter	Description	FrodoKEM -640	FrodoKEM -976	FrodoKEM -1344
q	modulus	$2^{15} = 32768$	$2^{16} = 65536$	$2^{16} = 65536$
n	LWE matrix dimension	640	976	1344
$\overline{n}$	plaintext matrix dimension	8	8	8
b	plaintext chunk bitlength	2	3	4
ℓ	plaintext bitlength	128	192	256
σ	error standard deviation	2.8	2.3	1.4
c	error sampling bound	12	10	6
λ	security level	≈ 128	≈ 192	≈ 256

Table 9.3: Parameter sets for FrodoKEM. The error distribution is $\chi = \chi_{s,c}$ of Definition 9.3, where $s = \sigma\sqrt{2\pi}$.

using SHAKE256 for the FrodoKEM-976 and FrodoKEM-1344 parameter sets, produces matrices $S, E \in \Lambda^{n\times\overline{n}}$ from a 2ℓ-bit seed where $\Lambda = \{-c, -c+1, \ldots, c-1, c\}$; each of the entries of these matrices is sampled independently from the distribution $\chi = \chi_{s,c}$. The pseudorandom bit generator ExpandRE likewise produces matrices $R, E_1 \in \Lambda^{n\times\overline{n}}$ and $E_2 \in \Lambda^{\overline{n}\times\overline{n}}$ from a 2ℓ-bit seed, with entries again sampled independently from χ. The reason for seed_{RE} being 2ℓ bits long is explained in Exercise 9.7(c); seed_{SE} is lengthened to match.

A plaintext message m is $\ell = \overline{n}^2 b$ bits long, divided into $\overline{n}^2$ chunks of b bits each, and interpreted as an $\overline{n}\times\overline{n}$ matrix with entries in $\mathbb{Z}_{2^b}$. Encryption encodes each entry of this matrix using the encoding function of Definition 9.5.

Example 9.8 (*FrodoPKE*)
Domain parameters are $q = 2^{10} = 1024$, $n = 10$, $\overline{n} = 3$, $c = 2$, $b = 5$, and $\ell = 45$. For simplicity, we omit the seeds used to generate A, S, E, R, E_1 and E_2. Also, the entries of S, E, R, E_1 and E_2 are sampled from the uniform distribution $\Lambda = \{-c, \ldots, c\}$ instead of the distribution χ.
Key generation. Bob selects

$$A = \begin{bmatrix} 931 & 214 & 277 & 261 & 351 & 558 & 965 & 558 & 684 & 289 \\ 45 & 361 & 67 & 564 & 912 & 507 & 607 & 75 & 784 & 423 \\ 558 & 515 & 177 & 741 & 779 & 597 & 170 & 469 & 873 & 306 \\ 96 & 883 & 496 & 965 & 727 & 532 & 52 & 277 & 337 & 197 \\ 139 & 985 & 210 & 86 & 170 & 644 & 947 & 695 & 946 & 130 \\ 53 & 250 & 92 & 651 & 486 & 727 & 439 & 904 & 978 & 969 \\ 129 & 176 & 722 & 577 & 497 & 196 & 210 & 126 & 825 & 680 \\ 1016 & 543 & 729 & 702 & 839 & 261 & 907 & 618 & 573 & 836 \\ 400 & 157 & 969 & 71 & 297 & 2 & 250 & 839 & 975 & 479 \\ 400 & 347 & 963 & 542 & 1001 & 652 & 674 & 960 & 374 & 653 \end{bmatrix} \in \mathbb{Z}_q^{10\times 10},$$

$$S = \begin{bmatrix} 0 & 2 & 0 \\ 0 & -1 & -2 \\ 0 & -1 & 2 \\ -1 & 2 & 2 \\ 2 & 1 & -1 \\ -1 & 1 & -2 \\ 2 & -1 & 1 \\ -2 & -2 & -1 \\ 2 & 0 & 1 \\ 0 & -2 & 2 \end{bmatrix} \in \Lambda^{10\times 3}, \quad \text{and} \quad E = \begin{bmatrix} 0 & 0 & -2 \\ -1 & 0 & -2 \\ 2 & 0 & -2 \\ 2 & -1 & 2 \\ -2 & 1 & 0 \\ -1 & 2 & -2 \\ -1 & -2 & -2 \\ 0 & 0 & -2 \\ 1 & -1 & -2 \\ 0 & -1 & 2 \end{bmatrix} \in \Lambda^{10\times 3},$$

and computes

$$B = AS + E = \begin{bmatrix} 17 & 143 & 848 \\ 312 & 606 & 774 \\ 346 & 538 & 17 \\ 183 & 1001 & 897 \\ 980 & 545 & 670 \\ 619 & 144 & 471 \\ 1014 & 407 & 552 \\ 391 & 473 & 899 \\ 270 & 300 & 759 \\ 984 & 374 & 383 \end{bmatrix} \in \mathbb{Z}_q^{10\times 3}.$$

Bob's encryption key is (A, B), and his decryption key is S.

Encryption. To encrypt a plaintext message in $\{0,1\}^{45}$ for Bob, Alice first represents it as a matrix in $\mathbb{Z}_{2^5}^{3\times 3}$:

$$m = \begin{bmatrix} 2 & 27 & 29 \\ 28 & 31 & 3 \\ 15 & 5 & 24 \end{bmatrix} \in \mathbb{Z}_{32}^{3\times 3}.$$

Alice then selects

$$R = \begin{bmatrix} 1 & 0 & 1 \\ -2 & 1 & 1 \\ 1 & 0 & 0 \\ -1 & 1 & 1 \\ 0 & -2 & -2 \\ 2 & -1 & 1 \\ 2 & -2 & 1 \\ -1 & -2 & 2 \\ 0 & 2 & -2 \\ 2 & 1 & 1 \end{bmatrix} \in \Lambda^{10\times 3}, \quad E_1 = \begin{bmatrix} -2 & 2 & 1 \\ 0 & 0 & 1 \\ 2 & 1 & -1 \\ 2 & 0 & 0 \\ 2 & -1 & -2 \\ 2 & 0 & 0 \\ -1 & 0 & -2 \\ 1 & -2 & 0 \\ 2 & 1 & -2 \\ -2 & 0 & 1 \end{bmatrix} \in \Lambda^{10\times 3}, \quad E_2 = \begin{bmatrix} -1 & 1 & 1 \\ 2 & -2 & 0 \\ 0 & 2 & 2 \end{bmatrix} \in \Lambda^{3\times 3},$$

and computes

$$U = A^T R + E_1 = \begin{bmatrix} 425 & 770 & 561 \\ 127 & 295 & 10 \\ 603 & 51 & 692 \\ 725 & 880 & 554 \\ 686 & 759 & 620 \\ 452 & 814 & 354 \\ 583 & 338 & 317 \\ 891 & 230 & 44 \\ 363 & 852 & 260 \\ 246 & 18 & 594 \end{bmatrix} \quad \text{and} \quad V = B^T R + E_2 + \text{Encode}(m) = \begin{bmatrix} 366 & 567 & 292 \\ 743 & 577 & 1003 \\ 813 & 45 & 563 \end{bmatrix}.$$

The ciphertext is $C = (U, V)$.
Decryption. To decrypt C, Bob uses his decryption key S to compute

$$M = V - S^T U = \begin{bmatrix} 61 & 871 & 942 \\ 892 & 980 & 94 \\ 478 & 164 & 770 \end{bmatrix} \text{ and } m = \text{Decode}(M) = \begin{bmatrix} 2 & 27 & 29 \\ 28 & 31 & 3 \\ 15 & 5 & 24 \end{bmatrix}.$$

9.2.2 Decryption error probability

Decryption succeeds precisely when each entry of the matrix M decodes to the corresponding entry of m. Now,

$$\begin{aligned} M = V - S^T U &= (B^T R + E_2 + \text{Encode}(m)) - S^T (A^T R + E_1) \\ &= (S^T A^T + E^T) R + E_2 - S^T (A^T R + E_1) + \text{Encode}(m) \\ &= E^T R - S^T E_1 + E_2 + \text{Encode}(m). \end{aligned}$$

Thus by Lemma 9.6, decryption succeeds provided that each entry d of the $\overline{n} \times \overline{n}$ matrix $D = E^T R - S^T E_1 + E_2$ satisfies

$$-\frac{q}{2^{b+1}} \le d < \frac{q}{2^{b+1}}. \tag{59}$$

For example, the FrodoKEM-640 parameters have $q = 2^{15}$, $n = 640$, $\overline{n} = 8$, $b = 2$ and $c = 12$. Since the entries of E, R, S, E_1, E_2 are drawn from the interval $[-c, c]$, each entry of D (before reduction modulo q) lies in the interval $[-2c^2 n - c,\ 2c^2 n + c] = [-184332, 184332]$. Decryption succeeds if each entry (after reduction modulo q) lies in $[-q/2^{b+1},\ q/2^{b+1} - 1] = [-4096, 4095]$.

Let X be a random variable following the distribution χ. The generating series of X is

$$\Phi_X(z) = \sum_{j=-c}^{c} p_j z^j, \quad \text{where } p_j = \Pr(X = j).$$

Let χ' be the distribution of each entry of D, and let X' be a random variable following the distribution χ'. Let $Y = X_1 X_2$, where X_1 and X_2 are independent random variables following the distribution χ. Since χ is symmetric about 0, negating a χ-distributed variable leaves its distribution unchanged. Hence,

$$X' = \sum_{i=1}^{n} Y_i + \sum_{i=1}^{n} Y_i' + X,$$

where X and the Y_i, Y_i' are independent, and each Y_i and Y_i' is distributed as Y. Hence, the generating series of X' is

$$\Phi_{X'}(z) = \Phi_X(z) \cdot \Phi_Y(z)^{2n}.$$

Then the probability π that any particular entry of D lies in the range (59) satisfies $\pi \geq \pi_0$ where

$$\pi_0 = \sum_{i=-q/2^{b+1}}^{q/2^{b+1}-1} [z^i]\Phi_{X'}(z),$$

since an entry falling outside this range may still be reduced modulo q into it. By the union bound[17], the probability p that at least one entry of D violates (59) satisfies

$$p \leq \overline{n}^2(1-\pi) \leq \overline{n}^2(1-\pi_0)$$

so

$$\overline{n}^2(1-\pi_0)$$

is an upper bound on the decryption error probability.

The Maple computer algebra system was used to extract coefficients from the generating series $\Phi_{X'}(z)$, yielding the upper bounds

$$p \leq 2^{-138.7}, \qquad p \leq 2^{-199.6}, \qquad p \leq 2^{-252.6}$$

on the decryption error probabilities for the FrodoKEM-640, FrodoKEM-976, and FrodoKEM-1344 parameter sets, respectively.

Example 9.9 (*decryption error probability for FrodoKEM-1344*) For the FrodoKEM-1344 parameter set, we have $q = 2^{16}$, $b = 4$, $n = 1344$, and $\overline{n} = 8$. The generating series of $X \sim \chi$ is

$$\Phi_X(z) = \frac{1}{2^{16}}\left(18286 + \delta(z) + \delta(1/z)\right),$$

where

$$\delta(z) = 14320z + 6876z^2 + 2023z^3 + 364z^4 + 40z^5 + 2z^6.$$

The generating series of $Y = X_1X_2$ where $X_1, X_2 \sim \chi$ is

$$\Phi_Y(z) = \frac{1}{2^{32}}\left(2062404796 + \gamma(z) + \gamma(1/z)\right),$$

where

$$\begin{aligned}\gamma(z) = {} & 410124800z + 393857280z^2 + 115877440z^3 + 115408672z^4 + 2291200z^5 \\ & + 55755152z^6 + 10011456z^8 + 8185058z^9 + 1100160z^{10} + 3000496z^{12} + 323680z^{15} \\ & + 264992z^{16} + 16184z^{18} + 58240z^{20} + 2912z^{24} + 3200z^{25} + 320z^{30} + 8z^{36}.\end{aligned}$$

Finally, an upper bound of the decryption error probability is $\overline{n}^2(1-\pi_0)$ where

$$\pi_0 = \sum_{i=-2048}^{2047} [z^i]\,\Phi_{X'}(z)$$

and

$$\Phi_{X'}(z) = \Phi_X(z) \cdot \Phi_Y(z)^{2688}.$$

[17]The union bound is used here because the $\overline{n}^2$ entries of D are not independent. Entries in the same column share a column of R and of E_1, and entries in the same row share a column of E and of S.

9.2.3 Security

The security of FrodoPKE rests on ss-D-MatrixLWE, which differs from the short-secret decisional LWE problem of Definition 4.10 in three respects:

(i) the modulus q is a power of two rather than a prime;

(ii) the secret and error terms are sampled from a discrete approximation to a rounded Gaussian distribution χ on (a subset of) $\mathbb{Z}_q$ rather than from the uniform distribution on a short interval centered at 0; and

(iii) the same matrix A is shared across $\overline{n}$ instances, one for each column of the secret matrix.

Differences (i) and (ii) are parameter choices, and we take the hardness of (general) ss-DLWE with these parameters (Definition 9.13) as our starting assumption. Difference (iii) is structural, and we show that it does not decrease security: ss-D-MatrixLWE is at least as hard as ss-DLWE. As a warm-up, we first prove the corresponding statement for the search problems, that MatrixLWE is at least as hard as LWE.

Definition 9.10 The (general) *Learning With Errors* problem, denoted $\mathrm{LWE}(m, n, q, \chi)$, is defined as follows. Choose a matrix $A \in_R \mathbb{Z}_q^{m\times n}$, a secret vector $s \in_R \mathbb{Z}_q^n$, and an error vector $e \in_\chi \mathbb{Z}_q^m$, and let $b = As + e \pmod q \in \mathbb{Z}_q^m$. Given the pair (A, b), the goal is to recover s.

Definition 9.11 The *Matrix Learning With Errors* problem, denoted $\mathrm{MatrixLWE}(m, n, \overline{n}, q, \chi)$, is defined as follows. Choose matrices $A \in_R \mathbb{Z}_q^{m\times n}$, $S \in_R \mathbb{Z}_q^{n\times\overline{n}}$, and $E \in_\chi \mathbb{Z}_q^{m\times\overline{n}}$, and let $B = AS + E \pmod q \in \mathbb{Z}_q^{m\times\overline{n}}$. Given the pair (A, B), the goal is to recover S.

Writing an instance of MatrixLWE by columns as $S = [s_1 \mid \cdots \mid s_{\overline{n}}]$, $E = [e_1 \mid \cdots \mid e_{\overline{n}}]$, and $B = [b_1 \mid \cdots \mid b_{\overline{n}}]$, we have $b_j = As_j + e_j$ for $1 \le j \le \overline{n}$, where the pairs (s_j, e_j) are independent and each distributed as the secret-error pair of an LWE instance. Each (A, b_j) is distributed exactly as an LWE instance, but the $\overline{n}$ instances are not independent, since they share the same matrix A.

The following shows that MatrixLWE is at least as hard as LWE.

Theorem 9.12 $\mathrm{LWE}(m, n, q, \chi) \le \mathrm{MatrixLWE}(m, n, \overline{n}, q, \chi)$.

Proof: Let (A, b) be an LWE instance. Set $b_1 = b$. For each $j = 2, \ldots, \overline{n}$, sample $s_j \in_R \mathbb{Z}_q^n$ and $e_j \in_\chi \mathbb{Z}_q^m$, and set $b_j = As_j + e_j$, reusing the given matrix A. Let $B = [b_1 \mid \cdots \mid b_{\overline{n}}]$. Then the pair (A, B) is distributed exactly as a MatrixLWE instance. If S is a solution to this MatrixLWE instance, then the first column s_1 of S is a solution to the original LWE instance. □

Theorem 9.15 uses a hybrid argument to show that ss-D-MatrixLWE is at least as hard as ss-DLWE.

Definition 9.13 The (general) *short-secret Decisional Learning With Errors* problem, denoted ss-DLWE(m, n, q, χ), is defined as follows. Choose a matrix $A \in_R \mathbb{Z}_q^{m\times n}$, a secret vector $s \in_\chi \mathbb{Z}_q^n$, an error vector $e \in_\chi \mathbb{Z}_q^m$, and a random vector $r \in_R \mathbb{Z}_q^m$, and let $b = As + e \pmod q \in \mathbb{Z}_q^m$. Let $z = b$ with probability $\frac{1}{2}$ and $z = r$ with probability $\frac{1}{2}$. Given A and z, decide (with success probability significantly greater than $\frac{1}{2}$) whether $z = b$ or $z = r$.

Definition 9.14 The *short-secret Decisional Matrix Learning With Errors* problem, denoted ss-D-MatrixLWE$(m, n, \overline{n}, q, \chi)$, is defined as follows. Choose matrices $A \in_R \mathbb{Z}_q^{m\times n}$, $S \in_\chi \mathbb{Z}_q^{n\times\overline{n}}$, $E \in_\chi \mathbb{Z}_q^{m\times\overline{n}}$, and $R \in_R \mathbb{Z}_q^{m\times\overline{n}}$, and let $B = AS + E \pmod q \in \mathbb{Z}_q^{m\times\overline{n}}$. Let $Z = B$ with probability $\frac{1}{2}$ and $Z = R$ with probability $\frac{1}{2}$. Given A and Z, decide (with success probability significantly greater than $\frac{1}{2}$) whether $Z = B$ or $Z = R$.

Theorem 9.15 ss-DLWE$(m, n, q, \chi) \le$ ss-D-MatrixLWE$(m, n, \overline{n}, q, \chi)$. More precisely, if ss-D-MatrixLWE$(m, n, \overline{n}, q, \chi)$ can be solved with success probability $\frac{1}{2} + \delta$ in time t, then ss-DLWE(m, n, q, χ) can be solved with success probability $\frac{1}{2} + \delta/\overline{n}$ in time $t + O(mn\overline{n}\log q)$.

Proof: Let $\mathcal{D}$ be an algorithm that solves ss-D-MatrixLWE$(m, n, \overline{n}, q, \chi)$ with success probability $\frac{1}{2} + \delta$ in time t, and suppose that $\mathcal{D}$ outputs its guess as either B or R.

Let $A \in_R \mathbb{Z}_q^{m\times n}$. For $0 \le j \le \overline{n}$, define the matrix

$$H_j = [\, r_1 \mid \cdots \mid r_j \mid b_{j+1} \mid \cdots \mid b_{\overline{n}} \,],$$

where $r_i \in_R \mathbb{Z}_q^m$, and $b_i = As_i + e_i$ with $s_i \in_\chi \mathbb{Z}_q^n$ and $e_i \in_\chi \mathbb{Z}_q^m$, all chosen independently and all reusing the same matrix A. Thus the first j columns of H_j are uniformly random and the remaining $\overline{n} - j$ columns are ss-DLWE samples. Writing an ss-D-MatrixLWE instance by columns as in Definition 9.14, we see that H_0 is distributed exactly as B and $H_{\overline{n}}$ is distributed exactly as R.

Let $p_j = \Pr(\mathcal{D}(A, H_j) = B)$. Since $\mathcal{D}$ is presented with B and with R each with probability $\frac{1}{2}$, its success probability is

$$\frac{1}{2} + \delta = \tfrac{1}{2}p_0 + \tfrac{1}{2}(1 - p_{\overline{n}}) = \tfrac{1}{2} + \tfrac{1}{2}(p_0 - p_{\overline{n}}),$$

and hence $p_0 - p_{\overline{n}} = 2\delta$.

We use $\mathcal{D}$ to solve ss-DLWE(m, n, q, χ) as follows. Given an instance (A, z), select $j \in_R \{1, \ldots, \overline{n}\}$, and form the matrix

$$Z = [\, r_1 \mid \cdots \mid r_{j-1} \mid z \mid b_{j+1} \mid \cdots \mid b_{\overline{n}} \,],$$

where the r_i and b_i are sampled as above using the given matrix A. Run $\mathcal{D}$ with input (A, Z), and answer $z = b$ if $\mathcal{D}$ outputs B, and $z = r$ otherwise.

If $z = b$ then Z is distributed as H_{j-1}, whereas if $z = r$ then Z is distributed as H_j. Averaging over the choice of j, the success probability of this algorithm is

$$\frac{1}{2} \cdot \frac{1}{\overline{n}} \sum_{j=1}^{\overline{n}} p_{j-1} + \frac{1}{2} \cdot \frac{1}{\overline{n}} \sum_{j=1}^{\overline{n}} (1 - p_j) = \frac{1}{2} + \frac{1}{2\overline{n}} \sum_{j=1}^{\overline{n}} (p_{j-1} - p_j) = \frac{1}{2} + \frac{p_0 - p_{\overline{n}}}{2\overline{n}} = \frac{1}{2} + \frac{\delta}{\overline{n}}.$$

Forming Z requires at most $\overline{n}$ column samples, each costing $O(mn)$ operations in $\mathbb{Z}_q$ for the product As_i, so the running time is $t + O(mn\overline{n} \log q)$. □

The security of FrodoPKE is captured in the following theorem.

Theorem 9.16 FrodoPKE is indistinguishable under chosen-plaintext attack (Definition A.2) assuming that ss-D-MatrixLWE is intractable.

Proof: The encryption operation can be written as

$$\begin{bmatrix} U \\ V \end{bmatrix} = \begin{bmatrix} A^T \\ B^T \end{bmatrix} R + \begin{bmatrix} E_1 \\ E_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \text{Encode}(m) \end{bmatrix}.$$

By the ss-D-MatrixLWE$(n, n, \overline{n}, q, \chi)$ intractability assumption,

$$\begin{bmatrix} A^T \\ B^T \end{bmatrix}$$

is indistinguishable from random. Next, by the ss-D-MatrixLWE$(n + \overline{n}, n, \overline{n}, q, \chi)$ intractability assumption,

$$\begin{bmatrix} A^T \\ B^T \end{bmatrix} R + \begin{bmatrix} E_1 \\ E_2 \end{bmatrix} = \begin{bmatrix} A^T R + E_1 \\ B^T R + E_2 \end{bmatrix} \tag{60}$$

is indistinguishable from random. Thus, from the adversary's perspective, V is the sum of the random matrix $B^T R + E_2$ and the encoded message Encode(m). Hence, the adversary learns nothing about m. □

9.3 FrodoKEM

As in the construction of Kyber-KEM from Kyber-PKE in §6.4, FrodoKEM is derived by applying a variant of the Fujisaki-Okamoto transform to FrodoPKE, though the variant here additionally incorporates a *salt*. This transformation converts a public-key encryption scheme that is only indistinguishable under chosen-plaintext attack (Definition A.2) into a KEM attaining a stronger notion of security, namely indistinguishability under chosen-ciphertext attack (Definition A.5). The guarantee holds in the (quantum) random oracle model.

FrodoKEM is shown in Algorithm 9.17. The pseudorandom bit generators ExpandA, ExpandSE and ExpandRE are the ones used in FrodoPKE. The variable-output length hash function H, which maps input $x \in \{0,1\}^*$ and a length parameter $d \geq 1$ to an output $H(x, d) \in \{0,1\}^d$, is instantiated using either SHAKE128 (for FrodoKEM-640) or SHAKE256 (for FrodoKEM-976 and FrodoKEM-1344).

Algorithm 9.17: Frodo key encapsulation mechanism (FrodoKEM)

Domain parameters. q, n, $\overline{n}$, σ, c, b, ℓ.

Key generation. Bob does the following:

1 Select $s \in_R \{0,1\}^\ell$, $\text{seed}_{SE} \in_R \{0,1\}^{2\ell}$, and $z \in_R \{0,1\}^{128}$.
2 Compute $\text{seed}_A = H(z, 128)$.
3 Compute $A = \text{ExpandA}(\text{seed}_A)$. // $A \in \mathbb{Z}_q^{n \times n}$
4 Compute $(S, E) = \text{ExpandSE}(\text{seed}_{SE})$. // $S, E \in \Lambda^{n \times \overline{n}}$
5 Compute $B = AS + E$. // $B \in \mathbb{Z}_q^{n \times \overline{n}}$
6 Compute $ekh = H(\text{seed}_A \| B, \ell)$.
7 Bob's encapsulation (public) key is $\text{EK} = (\text{seed}_A, B)$;
his decapsulation (private) key is $\text{DK} = (s, \text{seed}_A, B, S, ekh)$.

Encapsulation. To establish an ℓ-bit shared secret key with Bob, Alice does:

1 Obtain an authentic copy of Bob's encapsulation key $\text{EK} = (\text{seed}_A, B)$.
2 Select $m \in_R \{0,1\}^\ell$ and $\text{salt} \in_R \{0,1\}^{2\ell}$.
3 Compute $ekh = H(\text{seed}_A \| B, \ell)$.
4 Compute $A = \text{ExpandA}(\text{seed}_A)$.
5 Compute $(\text{seed}_{RE}, k) = H(ekh \| m \| \text{salt}, 3\ell)$. // $\text{seed}_{RE} \in \{0,1\}^{2\ell}$, $k \in \{0,1\}^\ell$
6 Compute $(R, E_1, E_2) = \text{ExpandRE}(\text{seed}_{RE})$. // $R, E_1 \in \Lambda^{n \times \overline{n}}$, $E_2 \in \Lambda^{\overline{n} \times \overline{n}}$
7 Compute $U = A^T R + E_1$ and $V = B^T R + E_2 + \text{Encode}(m)$.
8 Compute $K = H(U \| V \| \text{salt} \| k,\ \ell)$.
9 **return** $(K, C = (U, V, \text{salt}))$. // K is the secret key, C is the ciphertext

Decapsulation. To recover the secret key K from $C = (U, V, \text{salt})$ using $\text{DK} = (s, \text{seed}_A, B, S, ekh)$, Bob does the following:

1 Compute $A = \text{ExpandA}(\text{seed}_A)$.
2 Compute $M = V - S^T U$ and $m' = \text{Decode}(M)$.
3 Compute $(\text{seed}'_{RE}, k') = H(ekh \| m' \| \text{salt}, 3\ell)$.
4 Compute $(R', E'_1, E'_2) = \text{ExpandRE}(\text{seed}'_{RE})$.
5 Compute $U' = A^T R' + E'_1$ and $V' = B^T R' + E'_2 + \text{Encode}(m')$.
6 **if** $(U, V) = (U', V')$ **then**
7 $\quad \overline{k} \leftarrow k'$.
8 **else**
9 $\quad \overline{k} \leftarrow s$.
10 Compute $\overline{K} = H(U \| V \| \text{salt} \| \overline{k},\ \ell)$.
11 **return** $(\overline{K})$.

FrodoPKE is employed to encrypt a randomly chosen bit string m of length ℓ. During encapsulation, derandomization is applied as in Kyber-KEM, whereby the hashed encapsulation key *ekh*, the message m, and a freshly chosen salt are hashed to produce a random seed seed_{RE} and a secret k. The random matrices R, E_1 and E_2 required for FrodoPKE are derived from seed_{RE}, and the shared secret key K is derived by hashing the ciphertext components U, V, the salt, and the secret k. Unlike Kyber-KEM, however, encapsulation is not deterministic in m and EK alone, since a fresh salt is chosen each time. The salt is transmitted in the clear as part of the ciphertext $C = (U, V, \text{salt})$, so that decapsulation can reproduce seed_{RE} and k. The transform does not require determinism as such; it requires only that Bob be able to recompute the randomness used during encryption.

The salt was incorporated in response to an analysis showing that the multi-ciphertext security of lattice-based encryption schemes degrades as the number of ciphertexts under a single public key grows; see Exercise 9.7.

In decapsulation, the recipient decrypts the FrodoPKE ciphertext (U, V) to recover m', then hashes *ekh*, m' and the salt received in C to obtain seed'_{RE} and k'. He then re-encrypts m' using seed'_{RE} and compares the resulting (U', V') with the received (U, V). If the encryption was performed correctly, then m' will equal m with overwhelming probability and consequently $\text{seed}'_{RE} = \text{seed}_{RE}$ and $k' = k$. This ensures that the re-encryption of m results in (U, V). However, if decryption fails and $m' \neq m$, the re-encrypted ciphertext (U', V') will differ from (U, V) with overwhelming probability.

Therefore, if (U', V') equals (U, V), Bob sets $\overline{k} = k'$, so that the key $\overline{K}$ he computes is exactly Alice's key K. Otherwise, he sets $\overline{k} = s$, so that $\overline{K}$ is instead obtained by hashing C together with his secret s; in this case the decapsulation is deemed to have failed. As with Kyber-KEM, FrodoKEM exhibits plaintext awareness, which, as explained in §6.4, implies resistance to chosen-ciphertext attacks. Decapsulation can also fail when Alice and Bob both behave honestly, if the underlying FrodoPKE decryption fails; the probability of this is bounded in §9.2.2.

Security. FrodoKEM has been proven to be indistinguishable under chosen-ciphertext attack (Definition A.5) assuming that ss-D-MatrixLWE is intractable and that H is modelled as a random oracle, and also against a quantum adversary who is permitted to make queries to H in quantum superposition. Moreover, the salt and the inclusion of *ekh* in the encapsulation hash extend the guarantee to the *multi-target* setting, in which the adversary is given many encapsulation keys and many ciphertexts.

As with Kyber (§11.6), the parameter choices in Table 9.3 were justified by using the core-SVP methodology to assess the cost of the primal (§4.4.1) and dual (§4.4.2) attacks on the underlying LWE instances; see Exercise 11.11.

Performance. Table 9.4 compares the public key and ciphertext sizes, and the encapsulation and decapsulation times, for the three Kyber-KEM and FrodoKEM parameter sets. The comparison quantifies the price, in bandwidth and speed, of FrodoKEM's re-

liance on unstructured lattices.

KEM	parameter set	security level	public key (bytes)	ciphertext (bytes)	encaps. time	decaps. time
Kyber	ML-KEM-512	NIST 1	800	768	8.3μs	10.4μs
Frodo	FrodoKEM-640	NIST 1	9,616	9,752	378.1μs	373.1μs
Kyber	ML-KEM-768	NIST 3	1,184	1,088	12.8μs	16.0μs
Frodo	FrodoKEM-976	NIST 3	15,632	15,792	779.6μs	771.2μs
Kyber	ML-KEM-1024	NIST 5	1,568	1,568	18.8μs	23.3μs
Frodo	FrodoKEM-1344	NIST 5	21,520	21,696	1.13ms	1.11ms

Table 9.4: Size and performance comparison of Kyber-KEM and FrodoKEM (with AES128 used in ExpandA). Encapsulation and decapsulation times are taken from PQShield's website (https://pqshield.github.io/nist-sigs-zoo/kems/). The timings were obtained using a 12th Gen Intel(R) Core(TM) i7-12650H (8 cores, 2 threads/core, 4600 MHz max).

9.4 Notes and further references

Frodo was first proposed as an LWE-based key agreement protocol by Bos et al. [41] in 2016. The derived KEM, FrodoKEM, was submitted to the NIST Post-Quantum Cryptography project by Alkim et al. [12]. It advanced to Round 3 as an alternate candidate but was not selected for standardization, with NIST citing its bandwidth and performance costs relative to the structured-lattice finalists. Nevertheless, several agencies have endorsed FrodoKEM as a conservative alternative to structured-lattice schemes, including Germany's BSI, France's ANSSI, the Netherlands' NLNCSA, and the European Union Agency for Cybersecurity. In June 2026, ISO standardized FrodoKEM, together with ML-KEM and Classic McEliece, as part of ISO/IEC 18033-2 [124].

In 2023, the FrodoKEM designers added a salt to the Fujisaki-Okamoto transform [92, 118] in response to a multi-ciphertext security degradation observed by Bernstein [29]. At the same time they specified an *ephemeral* variant, eFrodoKEM, which omits the salt and shortens seed_{SE} and seed_{RE} to ℓ bits. Because it lacks these countermeasures, eFrodoKEM is intended only for settings in which a single encapsulation key is used to produce a small number of ciphertexts (e.g., fewer than 2^8). Glabush et al. [102] give a comprehensive description and analysis of FrodoKEM and eFrodoKEM, including details of the ExpandA, ExpandSE and ExpandRE pseudorandom bit generators.

The distribution $\chi_{s,c}$ is only an approximation to the rounded Gaussian Ψ_s, whereas the worst-case to average-case reductions underlying the security of LWE are stated for the exact distribution. Bounding the resulting security loss by the statistical distance between the two would require an approximation accurate to within $2^{-\lambda}$, where λ is the security level, and hence a sampler using far more than 16 bits of precision. The FrodoKEM designers instead bound the loss using the *Rényi divergence*, following Langlois, Stehlé and Steinfeld [140]; see also Bai et al. [21] for a systematic treatment. This

tolerates a much coarser approximation, at the cost of only a small loss in the concrete security estimate.

9.5 Exercises

Exercise 9.1 (†) Consider an instance of FrodoPKE with domain parameters $q = 1024$, $\overline{n} = 2$ and $b = 5$. Compute Encode(X) and Decode(Y), where

$$X = \begin{bmatrix} 20 & 6 \\ 7 & 18 \end{bmatrix} \text{ and } Y = \begin{bmatrix} 17 & 932 \\ 819 & 923 \end{bmatrix}.$$

Exercise 9.2 Prove Lemma 9.6.

Exercise 9.3 Consider an instance of FrodoPKE with domain parameters $q = 1024$, $n = 6$, $\overline{n} = 2$, $c = 3$ and $b = 5$ (and seeds are omitted). Bob's encryption key (A, B) and decryption key S are:

$$A = \begin{bmatrix} 606 & 351 & 903 & 489 & 922 & 170 \\ 198 & 367 & 575 & 803 & 528 & 347 \\ 435 & 913 & 932 & 751 & 148 & 413 \\ 427 & 360 & 130 & 651 & 169 & 373 \\ 792 & 1007 & 31 & 742 & 1004 & 531 \\ 518 & 31 & 964 & 233 & 615 & 893 \end{bmatrix}, \quad B = \begin{bmatrix} 422 & 901 \\ 463 & 711 \\ 429 & 754 \\ 271 & 499 \\ 894 & 249 \\ 833 & 158 \end{bmatrix}, \quad S = \begin{bmatrix} 1 & -3 \\ -3 & 2 \\ 1 & 0 \\ 0 & 3 \\ -3 & 3 \\ -2 & -1 \end{bmatrix}.$$

To encrypt the plaintext

$$m = \begin{bmatrix} 8 & 26 \\ 5 & 31 \end{bmatrix}$$

for Bob, Alice selects

$$R = \begin{bmatrix} -2 & 1 \\ -2 & 1 \\ 0 & -1 \\ 0 & 1 \\ -1 & -3 \\ 3 & 3 \end{bmatrix}, \quad E_1 = \begin{bmatrix} -1 & 2 \\ 0 & 2 \\ -1 & 3 \\ 3 & -2 \\ -3 & 3 \\ 3 & -3 \end{bmatrix}, \text{ and } E_2 = \begin{bmatrix} -3 & -2 \\ -2 & 0 \end{bmatrix}.$$

(a) Verify that the ciphertext that Alice computes is $C = (U, V)$ where

$$U = \begin{bmatrix} 177 & 1000 \\ 722 & 311 \\ 928 & 406 \\ 448 & 687 \\ 1010 & 307 \\ 93 & 536 \end{bmatrix} \text{ and } V = \begin{bmatrix} 88 & 350 \\ 231 & 28 \end{bmatrix}.$$

(b) Verify that decryption of C by Bob will recover m.

Exercise 9.4 Show that FrodoPKE is not indistinguishable under chosen-ciphertext attacks.

Exercise 9.5 (*converse of Theorem 9.12*) Suppose that you have an algorithm that solves $\mathrm{LWE}(m, n, q, \chi)$ with probability $1 - \varepsilon$. Show how this algorithm can be used to solve $\mathrm{MatrixLWE}(m, n, \overline{n}, q, \chi)$ with probability at least $1 - \overline{n}\varepsilon$. Conclude that if ε is negligible and $\overline{n}$ is not too large, then the resulting algorithm succeeds with overwhelming probability.

Exercise 9.6 (*converse of Theorem 9.15*) Prove that

$$\text{ss-D-MatrixLWE}(m, n, \overline{n}, q, \chi) \leq \text{ss-DLWE}(m, n, q, \chi).$$

Exercise 9.7 (*multi-ciphertext security*) In the eFrodoKEM variant of FrodoKEM, the salt is omitted, so encapsulation computes $(\mathrm{seed}_{RE}, k) = H(ekh \| m, 2\ell)$ where $\mathrm{seed}_{RE} \in \{0,1\}^\ell$ and $k \in \{0,1\}^\ell$. The ciphertext is therefore a deterministic function of EK and m.
(a) Suppose that an adversary obtains N ciphertexts, all produced under a single encapsulation key. Show that the adversary can recover the message underlying one of them at a cost of roughly $2^\ell/N$ encapsulation operations, whereas recovering the message underlying a prescribed ciphertext costs roughly 2^ℓ such operations.
(b) Explain why the salt in FrodoKEM defeats this attack, and why (a) motivates the restriction of eFrodoKEM to settings with fewer than 2^8 ciphertexts per encapsulation key.
(c) Show that an adversary can mount a variant of the attack in (a) by searching for seed_{RE} directly, at a cost unaffected by the salt. Explain why lengthening seed_{RE} to 2ℓ bits is a satisfactory fix, whereas the analogous remedy of enlarging the message space is not available.

10 The LLL lattice basis reduction algorithm

Contents

10.1 Gram-Schmidt orthogonalization 143
10.2 Gauss's algorithm 148
10.3 The LLL algorithm 153
10.4 LLL improvements 167
10.5 Notes and further references 172
10.6 Exercises 173

The LLL algorithm, also known as L^3, is a polynomial-time algorithm that finds a relatively short and nearly orthogonal basis of a lattice. It was discovered in 1982 by Arjen Lenstra, Hendrik Lenstra, and László Lovász. Its first major application was a polynomial-time algorithm for factoring polynomials with rational coefficients. Since then, LLL has become a fundamental tool in computational number theory and cryptography. Most notably, it is used to solve the approximate Shortest Vector Problem (approx-SVP) which, as we saw in §3 and §4, plays a central role in assessing the security of lattice-based cryptosystems.

The remainder of this section is organized as follows. In §10.1, we review the Gram-Schmidt orthogonalization process, a key ingredient of the LLL algorithm. Gauss's algorithm for computing a shortest basis of a two-dimensional integer lattice is presented in §10.2. The LLL algorithm itself is described and analyzed in §10.3, followed by an overview of several improvements in §10.4.

10.1 Gram-Schmidt orthogonalization

We begin by recalling some basic notions from Euclidean geometry and linear algebra. The *Euclidean length* of a vector $u = (u_1, \ldots, u_n) \in \mathbb{R}^n$ is

$$\|u\|_2 = \sqrt{u_1^2 + \cdots + u_n^2}.$$

To simplify the notation, we write $\|u\|$ instead of $\|u\|_2$. The *inner product* or *dot product* of vectors $u = (u_1, \ldots, u_n)$ and $v = (v_1, \ldots, v_n) \in \mathbb{R}^n$ is

$$\langle u, v \rangle = u_1 v_1 + \cdots + u_n v_n.$$

Notice that $\langle u, u \rangle = \|u\|^2$. The vectors u and v are said to be *orthogonal* if $\langle u, v \rangle = 0$.

Orthogonal complements and projections. Let V be a k-dimensional vector subspace of $\mathbb{R}^n$ with ordered basis $B = [b_1, \ldots, b_k]$. The *orthogonal complement* of V is the subspace

$$V^\perp = \{x \in \mathbb{R}^n \ : \ \langle x, v \rangle = 0 \text{ for all } v \in V\}.$$

The subspace $V^\perp$ has dimension $n-k$. The basis B can be extended by adding vectors $b_{k+1},\ldots,b_n$ to form a basis $[b_1,\ldots,b_k,\,b_{k+1},\ldots,b_n]$ of $\mathbb{R}^n$ such that $[b_{k+1},\ldots,b_n]$ is a basis of $V^\perp$.

Any vector $w\in\mathbb{R}^n$ can be written uniquely as a linear combination of the basis elements: $w=c_1b_1+\cdots+c_nb_n$ with real coefficients c_i. Define $w_1=c_1b_1+\cdots+c_kb_k\in V$ and $w_2=c_{k+1}b_{k+1}+\cdots+c_nb_n\in V^\perp$. Then $w=w_1+w_2$, expressing w as a sum of vectors in V and $V^\perp$.

Definition 10.1 Let $w\in\mathbb{R}^n$. The *projection of w onto a subspace V* is $\Pi_V(w)=w_1$, and the *projection of w onto $V^\perp$* is $\Pi_{V^\perp}(w)=w_2$.

Every vector w decomposes as $w=\Pi_V(w)+\Pi_{V^\perp}(w)$. Figure 10.1 illustrates this decomposition in the two-dimensional case, where $V=\langle v\rangle$ is a one-dimensional subspace. Theorem 10.4 shows that computing projections $\Pi_V(w)$ is particularly simple when the

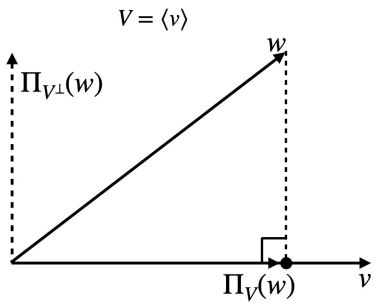


Figure 10.1: The projections of w onto V and $V^\perp$.

basis of V is orthogonal.

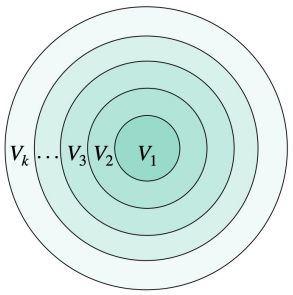
Definition 10.2 A basis $B=[b_1,\ldots,b_k]$ of a k-dimensional subspace $V\subseteq\mathbb{R}^n$ is called an *orthogonal basis* if $\langle b_i,b_j\rangle=0$ for all $1\le j<i\le k$.

A fundamental fact from linear algebra is that orthogonal bases always exist.

Theorem 10.3 Every subspace $V\subseteq\mathbb{R}^n$ admits an orthogonal basis.

Proof: Let $B=[b_1,\ldots,b_k]$ be any basis of V, and define $V_i=\mathrm{Span}(b_1,\ldots,b_i)$ for $1\le i\le k$. We construct vectors $b_1^*,\ldots,b_k^*$ inductively as follows. Set $b_1^*=b_1$, and for $2\le i\le k$, let $V_{i-1}^*=\mathrm{Span}(b_1^*,\ldots,b_{i-1}^*)$ and $b_i^*=\Pi_{(V_{i-1}^*)^\perp}(b_i)$; that is, b_i^* is the projection of b_i onto the orthogonal complement of the subspace spanned by the previously constructed vectors $b_1^*,\ldots,b_{i-1}^*$. Also, define $V_k^*=\mathrm{Span}(b_1^*,\ldots,b_k^*)$. To ease the notation, we will write Π_i instead of $\Pi_{(V_i^*)^\perp}$.

We claim that for each $1\le i\le k$, $V_i^*=V_i$ and $[b_1^*,\ldots,b_i^*]$ is an orthogonal basis of V_i^*. The claim is proved by induction on i.



Clearly, $V_1^* = V_1$ since $b_1^* = b_1$ and $[b_1^*]$ is an orthogonal basis of V_1^*. Let $i \geq 2$, and suppose that $V_{i-1}^* = V_{i-1}$ and that $[b_1^*, \ldots, b_{i-1}^*]$ is an orthogonal basis of V_{i-1}^*. Now,

$$b_i^* = \Pi_{i-1}(b_i) = b_i - \Pi_{V_{i-1}^*}(b_i) = b_i - \Pi_{V_{i-1}}(b_i) \in V_i,$$

and so $V_i^* \subseteq V_i$. Moreover, $b_i = b_i^* + \Pi_{V_{i-1}^*}(b_i) \in V_i^*$, and so $V_i \subseteq V_i^*$. Hence $V_i^* = V_i$ and so $[b_1^*, \ldots, b_i^*]$ is a basis of V_i^*. Finally, since $b_i^* \in (V_{i-1}^*)^\perp$, we have $\langle b_i^*, b_j^* \rangle = 0$ for all $1 \leq j \leq i-1$. Thus, $[b_1^*, \ldots, b_i^*]$ is an orthogonal basis of V_i^*, which completes the proof of the claim.

In conclusion, $B^* = [b_1^*, \ldots, b_k^*]$ is an orthogonal basis of $V_k^* = V_k = V$, which completes the proof of the theorem. □

The utility of orthogonal bases for computing projections is demonstrated in the next result.

Theorem 10.4 Let $B = [b_1, \ldots, b_k]$ be an orthogonal basis of a subspace $V \subseteq \mathbb{R}^n$, and let $w \in \mathbb{R}^n$. Then

$$\Pi_V(w) = \sum_{j=1}^{k} \frac{\langle w, b_j \rangle}{\|b_j\|^2} b_j. \tag{61}$$

Proof: Extend B to a basis $[b_1, \ldots, b_n]$ of $\mathbb{R}^n$ where $[b_{k+1}, \ldots, b_n]$ is a basis of $V^\perp$, and write $w = c_1 b_1 + \cdots + c_n b_n$ where $c_i \in \mathbb{R}$. For each $1 \leq j \leq k$, orthogonality implies that

$$\langle w, b_j \rangle = \langle c_1 b_1 + \cdots + c_n b_n, b_j \rangle = c_1 \langle b_1, b_j \rangle + \cdots + c_n \langle b_n, b_j \rangle = c_j \langle b_j, b_j \rangle = c_j \|b_j\|^2.$$

Hence, $c_j = \langle w, b_j \rangle / \|b_j\|^2$. Finally,

$$\Pi_V(w) = \Pi_V\left(\sum_{j=1}^{n} c_j b_j\right) = \sum_{j=1}^{n} c_j \Pi_V(b_j) = \sum_{j=1}^{k} c_j \Pi_V(b_j) = \sum_{j=1}^{k} c_j b_j,$$

and the result follows. □

Gram-Schmidt orthogonalization. Let $B = [b_1, \ldots, b_k]$ be a basis of a k-dimensional subspace $V \subseteq \mathbb{R}^n$. Applying the projection formula (61), we have

$$b_i^* = \Pi_{i-1}(b_i) = b_i - \Pi_{V_{i-1}^*}(b_i) = b_i - \sum_{j=1}^{i-1} \mu_{ij} b_j^*,$$

where the *Gram-Schmidt coefficients* μ_{ij}, for $1 \leq j < i \leq k$, are defined as follows:

$$\mu_{ij} = \frac{\langle b_i, b_j^* \rangle}{\|b_j^*\|^2}. \tag{62}$$

Algorithm 10.5: Gram-Schmidt orthogonalization process

Input: A basis $B = [b_1, \ldots, b_k]$ of a k-dimensional subspace $V \subseteq \mathbb{R}^n$.
Output: An orthogonal basis $B^* = [b_1^*, \ldots, b_k^*]$ of V.

```
b_1^* ← b_1.
for i = 2 to k do
    Compute μ_ij = ⟨b_i, b_j^*⟩/‖b_j^*‖^2 for 1 ≤ j < i.
    b_i^* ← b_i − Σ_{j=1}^{i−1} μ_ij b_j^*.
return (B^* = [b_1^*, ..., b_k^*]).
```

This yields the Gram-Schmidt process for computing an orthogonal basis of a vector space, which is summarized in Algorithm 10.5. Theorem 10.28 asserts that Algorithm 10.5 is a polynomial-time algorithm when B is a basis of a full-rank integer lattice.

Observe that each ordered basis determines a unique Gram-Schmidt basis. Two useful properties of Gram-Schmidt bases are that $\|b_i^*\| \leq \|b_i\|$ and $\langle b_i, b_i^* \rangle = \langle b_i^*, b_i^* \rangle$ for all $1 \leq i \leq k$ (Exercise 10.3).

Example 10.6 (*Gram-Schmidt orthogonalization*) Consider the basis

$$B = [(-5, -2, 5, 1),\ (3, 5, 2, 3),\ (0, 1, 2, -3),\ (-3, -4, -4, -5)]$$

for a full-rank integer lattice $L \subseteq \mathbb{Z}^4$. The squared lengths of the basis vectors are:

$$\|b_1\|^2 = 55, \quad \|b_2\|^2 = 47, \quad \|b_3\|^2 = 14, \quad \|b_4\|^2 = 66.$$

The Gram-Schmidt orthogonalization of B is

$$B^* = [(-5, -2, 5, 1),\ \left(\frac{21}{11}, \frac{251}{55}, \frac{34}{11}, \frac{177}{55}\right),\ \left(\frac{995}{2441}, \frac{2611}{2441}, \frac{3587}{2441}, -\frac{7738}{2441}\right),$$
$$\left(-\frac{35014}{32999}, \frac{31857}{32999}, -\frac{21525}{32999}, -\frac{3731}{32999}\right)].$$

The squared lengths of the Gram-Schmidt basis vectors are:

$$\|b_1^*\|^2 = 55, \quad \|b_2^*\|^2 \approx 44.4, \quad \|b_3^*\|^2 \approx 13.5, \quad \|b_4^*\|^2 \approx 2.5.$$

Notice that $\|b_i^*\|^2 \leq \|b_i\|^2$ for $i = 1, 2, 3, 4$. The lengths of the Gram-Schmidt basis vectors are:

$$\|b_1^*\| = \sqrt{55}, \quad \|b_2^*\| = \frac{\sqrt{134255}}{55}, \quad \|b_3^*\| = \frac{\sqrt{80550559}}{2441}, \quad \|b_4^*\| = \frac{287\sqrt{32999}}{32999}.$$

Even though these lengths are irrational numbers, their product is an integer:

$$\|b_1^*\| \cdot \|b_2^*\| \cdot \|b_3^*\| \cdot \|b_4^*\| = 287.$$

This is not unexpected since $\mathrm{vol}(L) = \prod_{i=1}^{4} \|b_i^*\|$, as we will prove in Theorem 10.8.

The matrix of Gram-Schmidt coefficients μ_{ij} is the following lower-triangular matrix, where $\mu_{ij} = 1$ if $i = j$ and $\mu_{ij} = 0$ if $i < j$:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ -\frac{12}{55} & 1 & 0 & 0 \\ \frac{1}{11} & \frac{60}{2441} & 1 & 0 \\ -\frac{2}{55} & -\frac{2884}{2441} & \frac{10913}{32999} & 1 \end{bmatrix}.$$

Applications to lattices. Let $B = [b_1, \ldots, b_n]$ be a basis of a full-rank lattice L, and let $B^* = [b_1^*, \ldots, b_n^*]$ be its *Gram-Schmidt orthogonalization*, so

$$b_1^* = b_1 \text{ and } b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{ij} b_j^* \text{ for } 2 \le i \le n \text{ where } \mu_{ij} = \frac{\langle b_i, b_j^* \rangle}{\|b_j^*\|^2}. \tag{63}$$

While B^* is a basis of $\mathbb{R}^n$, it is typically *not* a basis of the lattice L since the Gram-Schmidt coefficients μ_{ij} need not be integers. Nevertheless, as the next two theorems illustrate, B^* reveals useful information about L, namely a lower bound for the first successive minimum and a formula for the lattice volume.

Theorem 10.7 (*lower bound for the first successive minimum*) Let $B = [b_1, \ldots, b_n]$ be a basis of a full-rank lattice $L \subseteq \mathbb{Z}^n$, and let $B^* = [b_1^*, \ldots, b_n^*]$ be its Gram-Schmidt orthogonalization. Then $\lambda_1(L) \ge \min_i \|b_i^*\|$.

Proof: Let $v = \sum_{i=1}^{n} v_i b_i$ be a nonzero lattice vector, and let k be the largest index with $v_k \neq 0$ so $v = \sum_{i=1}^{k} v_i b_i$. Now,

$$\langle v, b_k^* \rangle = \langle \sum_{i=1}^{k} v_i b_i, b_k^* \rangle = \langle \sum_{i=1}^{k-1} v_i b_i, b_k^* \rangle + \langle v_k b_k, b_k^* \rangle = v_k \cdot \langle b_k^*, b_k^* \rangle = v_k \cdot \|b_k^*\|^2.$$

Hence the projection of v onto the one-dimensional subspace $\langle b_k^* \rangle$ is

$$\Pi_{\langle b_k^* \rangle}(v) = \frac{\langle v, b_k^* \rangle}{\|b_k^*\|^2} b_k^* = \frac{v_k \|b_k^*\|^2}{\|b_k^*\|^2} b_k^* = v_k b_k^*,$$

from which we conclude that

$$\|v\| \ge \|v_k b_k^*\| = |v_k| \cdot \|b_k^*\| \ge \min_i \|b_i^*\|$$

since $|v_k| \ge 1$. This shows that $\lambda_1(L) \ge \min_i \|b_i^*\|$. □

The proof of Theorem 10.7 can be modified to yield lower bounds on all the successive minima; see Exercise 10.5.

Theorem 10.8 (*volume of a lattice*) Let $B = [b_1, \ldots, b_n]$ be a basis of a full-rank lattice L, and let $B^* = [b_1^*, \ldots, b_n^*]$ be its Gram-Schmidt orthogonalization. Then

$$\mathrm{vol}(L) = \prod_{i=1}^{n} \|b_i^*\|. \tag{64}$$

Proof: Let B, B^* and Q be the $n \times n$ matrices whose i-th columns are b_i, b_i^* and $b_i^*/\|b_i^*\|$, respectively. Since B^* is an orthogonal basis, Q is an orthogonal matrix, meaning $Q^T Q = I_n$; hence $\det(Q) = \pm 1$. Let U be the following upper-triangular matrix:

$$U = \begin{bmatrix} 1 & \mu_{21} & \mu_{31} & \cdots & \mu_{n1} \\ & 1 & \mu_{32} & \cdots & \mu_{n2} \\ & & 1 & \cdots & \mu_{n3} \\ & & & \ddots & \\ & & & & 1 \end{bmatrix}_{n \times n}$$

and notice that $\det(U) = 1$. Let D be the $n \times n$ diagonal matrix with diagonal entries $\|b_1^*\|, \ldots, \|b_n^*\|$. By definition of Gram-Schmidt orthogonalization, we have $B = B^* U$. Furthermore, $B^* = QD$, so $B = QDU$. Taking the absolute values of the determinants of both sides gives

$$\mathrm{vol}(L) = |\det(B)| = |\det(QDU)| = |\det(Q)\ \det(D)\ \det(U)| = |\det(D)| = \prod_{i=1}^{n} \|b_i^*\|,$$

as required. □

Let $L = L(B)$ be an integer lattice, implying that $\mathrm{vol}(L) = |\det(B)|$ is a positive integer. By Theorem 10.8, the product $\prod_{i=1}^{n} \|b_i^*\|$ of the lengths of the Gram-Schmidt basis vectors is also a positive integer even though the $\|b_i^*\|$'s are typically not integers.

Remark 10.9 The Gram-Schmidt orthogonalization process can be interpreted geometrically as transforming the fundamental parallelepiped determined by n linearly independent vectors $[b_1, \ldots, b_n]$ in $\mathbb{R}^n$ into an orthogonal rectangular box (also called an *orthotope*) of equal volume. Each new edge b_i^* of the rectangular box is the height of the original parallelepiped in a new independent direction. Figures 10.2 and 10.3 show examples for $n = 2$ and $n = 3$.

10.2 Gauss's algorithm

Let $[u, v]$ be a basis of a two-dimensional integer lattice L, ordered so that $\|u\| \leq \|v\|$. Gauss's algorithm is a polynomial-time algorithm for finding a shortest possible basis of L, namely a basis $[u', v']$ satisfying $\|u'\| = \lambda_1(L)$ and $\|v'\| = \lambda_2(L)$. The geometric idea

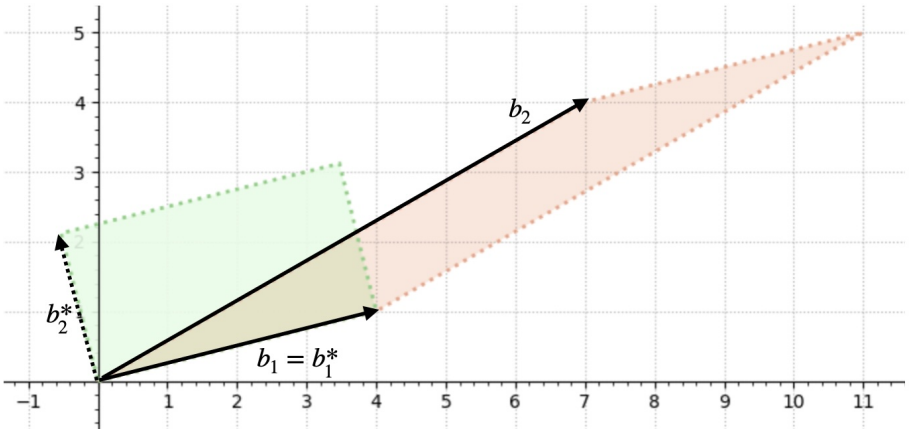


Figure 10.2: The fundamental parallelepipeds spanned by the lattice vectors $b_1 = (4, 1)$ and $b_2 = (7, 4)$ (shaded red), and by the Gram-Schmidt vectors $b_1^* = (4, 1)$ and $b_2^* = (-9/17, 36/17)$ (shaded green); both have area 9.

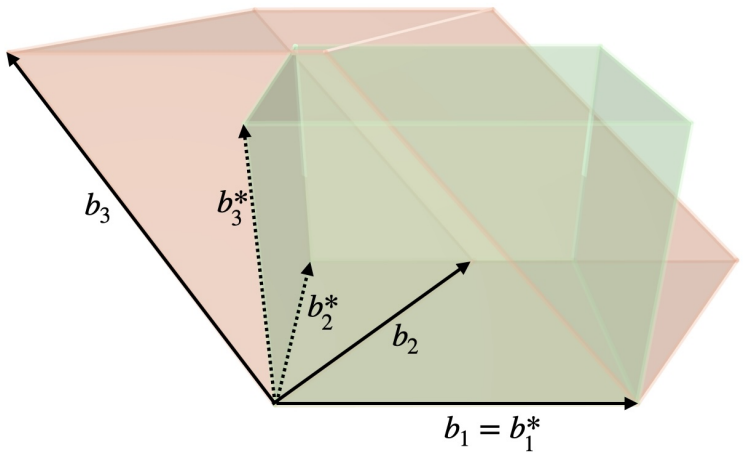


Figure 10.3: The fundamental parallelepipeds spanned by the lattice vectors $b_1 = (6, 1, 0)$, $b_2 = (3, 5, 0)$ and $b_3 = (-6, 3, 5)$ (shaded red), and by the Gram-Schmidt vectors $b_1^* = (6, 1, 0)$, $b_2^* = (-27/37, 162/37, 0)$ and $b_3^* = (0, 0, 5)$ (shaded green); both have volume 135.

behind Gauss's algorithm is the following; see Figure 10.4. Consider the projection of v onto the one-dimensional subspace spanned by u. Using formula (61), this projection is

$$\Pi_{\langle u\rangle}(v) = \mu u, \quad \text{where } \mu = \frac{\langle u, v\rangle}{\|u\|^2}.$$

Subtracting this component from v leaves the orthogonal component

$$\Pi_{\langle u\rangle^\perp}(v) = v - \mu u.$$

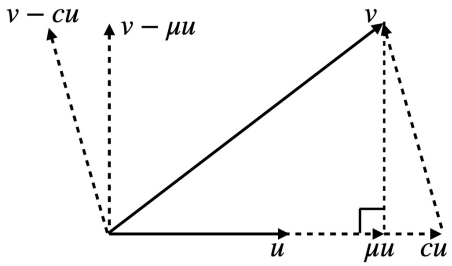


Figure 10.4: An iteration of Gauss's algorithm.

If μ were an integer, then $v - \mu u$ would itself be a lattice vector, and replacing v by this vector would immediately yield a shorter basis. In general, however, μ is rational but not integral, so $v - \mu u$ does not lie in the lattice. So, even though the vectors u and $v - \mu u$ are orthogonal, they do not form a basis of L. To address this, let[18] $c = \lfloor \mu \rceil$ be the closest integer to μ with ties broken by rounding towards 0 (so $\lfloor 3.5 \rceil = 3$ and $\lfloor -3.5 \rceil = -3$). Then $v - cu$ *is* a lattice vector, which closely approximates the orthogonal projection $v - \mu u$. As a result, $v - cu$ is typically shorter than v and nearly orthogonal to u. Consequently, the pair $[u, v - cu]$ is a new basis of L that is usually shorter than the original one.

Gauss's algorithm (Algorithm 10.10) repeatedly applies this operation: replace v by $v - cu$, and then swap u and the new v to maintain the ordering $\|u\| \le \|v\|$. This procedure terminates once $c = 0$, which occurs exactly when $|\mu| \le \frac{1}{2}$.

Example 10.11 (*Gauss's algorithm*) Consider the lattice $L \subseteq \mathbb{R}^2$ with basis vectors $u = (3, 5)$, $v = (4, 7)$. Here are the steps of Gauss's algorithm for finding a reduced basis of L.

iteration	u	$\|u\|^2$	v	$\|v\|^2$	μ	c	$v - cu$
1	$(3, 5)$	34	$(4, 7)$	65	$47/34$	1	$(1, 2)$
2	$(1, 2)$	5	$(3, 5)$	34	$13/5$	3	$(0, -1)$
3	$(0, -1)$	1	$(1, 2)$	5	-2	-2	$(1, 0)$
4	$(0, -1)$	1	$(1, 0)$	1	0	0	$(1, 0)$

[18] In §10, §11, and §12, the notation $\lfloor \cdot \rceil$ denotes rounding to the nearest integer with ties broken *towards* 0. This contrasts with the notation $\lceil \cdot \rfloor$ used in §4, §6, and §7, where ties are broken *away from* 0.

Algorithm 10.10: Gauss's algorithm

Input: A basis $[u, v]$ of an integer lattice $L \subseteq \mathbb{Z}^2$ with $\|u\| \leq \|v\|$.
Output: A basis $[u', v']$ of L with $\|u'\| = \lambda_1(L)$ and $\|v'\| = \lambda_2(L)$.

```
c ← 1.
while c ≠ 0 do
    Compute μ = ⟨u, v⟩/‖u‖² and c = ⌊μ⌉.
    Set v ← v − cu.
    if ‖u‖ > ‖v‖ then
        Swap u and v.
return ([u, v]).
```

A reduced basis of L is $[u', v']$ where $u' = (0, -1)$ and $v' = (1, 0)$.

Example 10.12 (*Gauss's algorithm*) Consider the lattice $L \subseteq \mathbb{R}^2$ with basis vectors $u = (47928, 63649)$, $v = (68827, 91412)$. Here are the steps of Gauss's algorithm for finding a reduced basis of L.

#	u	$\|u\|^2$	v	$\|v\|^2$	μ	c	$v - cu$
1	(47928, 63649)	6348288385	(68827, 91412)	13093309673	$\frac{9117022844}{6348288385}$	1	(20899, 27763)
2	(20899, 27763)	1207552370	(47928, 63649)	6348288385	$\frac{2768734459}{1207552370}$	2	(6130, 8123)
3	(6130, 8123)	103560029	(20899, 27763)	1207552370	$\frac{353629719}{103560029}$	3	(2509, 3394)
4	(2509, 3394)	17814317	(6130, 8123)	103560029	$\frac{42949632}{17814317}$	2	(1112, 1335)
5	(1112, 1335)	3018769	(2509, 3394)	17814317	$\frac{7320998}{3018769}$	2	(285, 724)
6	(285, 724)	605401	(1112, 1335)	3018769	$\frac{1283460}{605401}$	2	(542, -113)
7	(542, -113)	306533	(285, 724)	605401	$\frac{72658}{306533}$	0	(285, 724)

A reduced basis of L is $[u', v']$ where $u' = (542, -113)$ and $v' = (285, 724)$.

It is immediate that the output of Gauss's algorithm is a basis of L. One can also show that the algorithm always terminates after at most $O(\log X)$ iterations where $X = \|v\|$, and that all arithmetic operations involve integers that are $\leq X$. Consequently, the total running time of Gauss's algorithm is $O(\log^3 X)$ bit operations. We omit the proofs of these statements and instead present a proof that the algorithm produces a shortest possible basis.

Theorem 10.13 Let $[u, v]$ be the output of Gauss's algorithm. Then $\|u\| = \lambda_1(L)$ and $\|v\| = \lambda_2(L)$.

Proof: We will first prove that $\|u\| = \lambda_1(L)$. Let w be any nonzero lattice vector and write $w = au + bv$ where $a, b \in \mathbb{Z}$; note that a and b cannot both be 0 since $w \neq 0$. We have

$$\begin{aligned}
\|w\|^2 &= \|au + bv\|^2 \\
&= \langle au + bv, au + bv \rangle \\
&= a^2\|u\|^2 + 2ab\,\langle u, v\rangle + b^2\|v\|^2 \\
&\geq a^2\|u\|^2 + 2ab\langle u, v\rangle + b^2\|u\|^2 \quad \text{(since } \|v\| \geq \|u\|\text{)} \\
&\geq a^2\|u\|^2 - 2|ab| \cdot |\langle u, v\rangle| + b^2\|u\|^2 \\
&\geq a^2\|u\|^2 - |ab| \cdot \|u\|^2 + b^2\|u\|^2 \quad \text{(since } |\mu| = |\langle u, v\rangle|/\|u\|^2 \leq \frac{1}{2}\text{)} \\
&= (a^2 - |ab| + b^2)\|u\|^2.
\end{aligned}$$

Now, for $a, b \in \mathbb{R}$, define $f(a, b) = a^2 - |ab| + b^2$. We have $f(a, b) = (|a| - \frac{|b|}{2})^2 + \frac{3}{4}b^2 \geq 0$, with $f(a, b) = 0$ if and only if $a = b = 0$. Thus, if a and b are integers, not both 0, then $a^2 - |ab| + b^2 \geq 1$. It follows that $\|w\|^2 \geq \|u\|^2$, and thus $\|w\| \geq \|u\|$. This establishes that $\|u\| = \lambda_1(L)$.

We will next prove that $\|v\| = \lambda_2(L)$. Let w be any lattice vector that is not a multiple of u. Then we can write $w = au + bv$ where $a, b \in \mathbb{Z}$ with $b \neq 0$. We have

$$\begin{aligned}
\|w\|^2 &= \|au + bv\|^2 \\
&= \langle au + bv, au + bv \rangle \\
&= a^2\|u\|^2 + 2ab\langle u, v\rangle + b^2\|v\|^2 \\
&= b^2(\|v\|^2 - \|u\|^2) + (a^2 + b^2)\|u\|^2 + 2ab\langle u, v\rangle \\
&\geq b^2(\|v\|^2 - \|u\|^2) + (a^2 + b^2)\|u\|^2 - 2|ab| \cdot |\langle u, v\rangle| \\
&\geq b^2(\|v\|^2 - \|u\|^2) + (a^2 - |ab| + b^2)\|u\|^2 \quad \text{(since } |\mu| = |\langle u, v\rangle|/\|u\|^2 \leq \frac{1}{2}\text{)} \\
&\geq b^2(\|v\|^2 - \|u\|^2) + \|u\|^2 \quad \text{(since } a^2 - |ab| + b^2 \geq 1\text{)} \\
&= (b^2 - 1)(\|v\|^2 - \|u\|^2) + \|v\|^2 \\
&\geq \|v\|^2 \quad \text{(since } b \neq 0 \text{ and } \|v\| \geq \|u\|\text{)}.
\end{aligned}$$

Thus, $\|w\| \geq \|v\|$. Finally, suppose that $[u', v']$ are any two linearly independent lattice vectors with $\|u'\| \leq \|v'\|$. Then u' and v' cannot both be multiples of u. If v' is not a multiple of u, then $\|v'\| \geq \|v\|$ by what we showed above. On the other hand, if v' is a multiple of u, then u' is not a multiple of u, whence $\|u'\| \geq \|v\|$. We conclude that $\|v\| = \lambda_2(L)$. □

Definition 10.14 A basis $[u, v]$ of a lattice $L \subseteq \mathbb{Z}^2$ is called *Gauss-reduced* if $\|u\| \leq \|v\|$ and $|\mu| \leq \frac{1}{2}$ where $\mu = \langle u, v\rangle/\|u\|^2$.

Gauss's algorithm always produces a Gauss-reduced basis. As we have seen, this basis is a shortest possible basis of L. The notion of a Gauss-reduced basis extends naturally to integer lattices in higher dimensions. However, it turns out that such a reduced

lattice basis is not necessarily short. Instead, we will consider a stronger notion of reduced lattice basis due to Lovász. We will show that the vectors in such a reduced lattice basis are relatively short, and will present the LLL algorithm that efficiently finds such a basis.

10.3 The LLL algorithm

Let L be an n-dimensional integer lattice. As with Gauss's reduction in two dimensions, it is not difficult to find a lattice basis $B = [b_1, \ldots, b_n]$ so that the Gram-Schmidt coefficients μ_{ij} are small, specifically $|\mu_{ij}| \leq \frac{1}{2}$ for $1 \leq j < i \leq n$. This condition, known as *size reduction*, ensures that the basis vectors are reasonably close to being orthogonal. However, it does not guarantee that the basis vectors themselves are short.

Consider a randomly chosen lattice basis B. One expects that the Gram-Schmidt vectors $b_1^*, \ldots, b_n^*$ rapidly decrease in length. This is because each vector b_i^* is obtained by projecting b_i onto the orthogonal complement of the subspace spanned by the earlier basis vectors $b_1, \ldots, b_{i-1}$. As i increases, this orthogonal complement has smaller dimension, and consequently the projection of b_i typically tends to be shorter.

A key invariant of lattices comes into play here. Although the individual Gram-Schmidt vectors depend on the chosen lattice basis, the product of their lengths does not—it is equal to the volume of the lattice (see Theorem 10.8). The central idea behind the LLL algorithm is to exploit this fact by reshaping the lattice basis so that the lengths of the Gram-Schmidt vectors are distributed more evenly, rather than dropping off too quickly.

The mechanism that makes this possible is a sequence of local operations: *size reduction* and *swaps* of adjacent basis vectors. The swaps are designed to slow down the rate at which the Gram-Schmidt lengths decrease, thereby forcing earlier vectors to become relatively short. In particular, the first vector b_1^* is expected to be relatively short, and thus so will the first lattice basis vector b_1 (since $b_1 = b_1^*$). More precisely, we will prove that the first vector in the lattice basis produced by the LLL algorithm is guaranteed to have length no more than $2^{(n-1)/2}$ times the length of a shortest nonzero lattice vector.

10.3.1 LLL-reduced bases and the Lovász condition

We begin by formalizing what it means for a lattice basis to be reduced.

Definition 10.15 A lattice basis $[b_1, \ldots, b_n]$ is called *size-reduced* if $|\mu_{ij}| \leq \frac{1}{2}$ for all $1 \leq j < i \leq n$. A size-reduced basis is said to be *LLL-reduced* if it also satisfies the *Lovász condition*:[19]

$$\|b_k^*\|^2 \geq \left(\tfrac{3}{4} - \mu_{k,k-1}^2\right) \|b_{k-1}^*\|^2 \quad \text{for all } 2 \leq k \leq n. \tag{65}$$

[19] As further discussed in §10.4.2, the constant $\frac{3}{4}$ in the Lovász condition can be replaced by any constant δ where $\frac{1}{4} < \delta \leq 1$.

Since $0 \leq \mu_{k,k-1}^2 \leq \frac{1}{4}$, the Lovász condition implies that $\|b_k^*\|^2 \geq \frac{1}{2}\|b_{k-1}^*\|^2$. In other words, the Gram-Schmidt basis vectors are not allowed to shrink too quickly as we move along the basis.

Here are two equivalent ways to express the Lovász condition:

$$\|b_k^* + \mu_{k,k-1} b_{k-1}^*\|^2 \geq \tfrac{3}{4}\|b_{k-1}^*\|^2 \tag{66}$$

and

$$\|\Pi_{k-2}(b_k)\|^2 \geq \tfrac{3}{4}\|\Pi_{k-2}(b_{k-1})\|^2. \tag{67}$$

Recall from the proof of Theorem 10.3 that for $1 \leq i \leq n$, Π_i denotes the projection map $\Pi_{(V_i^*)^\perp} = \Pi_{V_i^\perp}$, where $V_i = \mathrm{Span}(b_1, \ldots, b_i)$, $V_i^* = \mathrm{Span}(b_1^*, \ldots, b_i^*)$, and $V_i = V_i^*$. The projection Π_0 is the identity map.

One of the most important guarantees offered by the LLL algorithm concerns the length of the first basis vector.

Theorem 10.16 Let $[b_1, \ldots, b_n]$ be an LLL-reduced basis of a lattice L. Then

$$\|b_1\| \leq 2^{(n-1)/2}\lambda_1(L). \tag{68}$$

Proof: For each $2 \leq i \leq n$, we have

$$\|b_i^*\|^2 \geq \left(\tfrac{3}{4} - \mu_{i,i-1}^2\right)\|b_{i-1}^*\|^2 \geq \left(\tfrac{3}{4} - \tfrac{1}{4}\right)\|b_{i-1}^*\|^2 = \tfrac{1}{2}\|b_{i-1}^*\|^2,$$

and so $\|b_{i-1}^*\|^2 \leq 2\|b_i^*\|^2$. Hence

$$\|b_1\|^2 = \|b_1^*\|^2 \leq 2^{i-1}\|b_i^*\|^2, \tag{69}$$

so

$$\|b_1\|^2 \leq 2^{n-1} \min_i \|b_i^*\|^2 \leq 2^{n-1}\lambda_1(L)^2$$

by Theorem 10.7. Taking square roots of both sides yields $\|b_1\| \leq 2^{(n-1)/2}\lambda_1(L)$. □

The following result gives an upper bound on the length of the first LLL-reduced basis vector in terms of the lattice volume. The proof is left as an exercise (Exercise 10.17).

Theorem 10.17 Let $[b_1, \ldots, b_n]$ be an LLL-reduced basis of a lattice L. Then

$$\|b_1\| \leq 2^{(n-1)/4}\,\mathrm{vol}(L)^{1/n}. \tag{70}$$

10.3.2 Size reduction

Algorithm 10.18 is a simple procedure for size reducing a lattice basis. The size reduction is performed in steps 2-5. In step 5, b_i is size-reduced with respect to b_j.

Theorem 10.19 Size reduction works, i.e., the basis $\tilde{B} = [\tilde{b}_1, \ldots, \tilde{b}_n]$ produced by Algorithm 10.18 is a size-reduced basis of $L(B)$. Moreover, size reduction leaves the Gram-Schmidt basis unchanged.

Algorithm 10.18: Size reduction

Input: Lattice basis $B = [b_1, \ldots, b_n]$.
Output: Size-reduced basis $\tilde{B} = [\tilde{b}_1, \ldots, \tilde{b}_n]$.

1 Compute the Gram-Schmidt orthogonalization $B^* = [b_1^*, \ldots, b_n^*]$.
2 **for** $i = 2$ **to** n **do**
3 **for** $j = i - 1$ **downto** 1 **do**
4 Compute $\mu_{ij} = \langle b_i, b_j^* \rangle / \|b_j^*\|^2$.
5 Set $b_i \leftarrow b_i - \lfloor \mu_{ij} \rceil b_j$. // size-reduce b_i with respect to b_j
6 **return** $(\tilde{B} = [b_1, \ldots, b_n])$.

Proof: Since each operation $b_i \leftarrow b_i - \lfloor \mu_{ij} \rceil b_j$ in the inner loop preserves the basis, $\tilde{B}$ is a basis of $L(B)$.

We claim that the Gram-Schmidt orthogonalization of $\tilde{B}$ is identical to that of B, i.e., $\tilde{b}_i^* = b_i^*$ for all $1 \le i \le n$. Let $V_i = \mathrm{Span}(b_1, \ldots, b_i)$ and $\tilde{V}_i = \mathrm{Span}(\tilde{b}_1, \ldots, \tilde{b}_i)$ for $1 \le i \le n$. Since each $\tilde{b}_i$ is obtained by subtracting integer multiples of earlier basis vectors $b_1, \ldots, b_{i-1}$ from b_i, we can write $\tilde{b}_i = b_i + c_i$ where $c_i \in V_{i-1}$. It follows that $V_i = \tilde{V}_i$ for all $1 \le i \le n$. Thus,

$$\tilde{b}_i^* = \Pi_{(\tilde{V}_{i-1})^\perp}(\tilde{b}_i) = \Pi_{(V_{i-1})^\perp}(\tilde{b}_i) = \Pi_{(V_{i-1})^\perp}(b_i + c_i) = \Pi_{(V_{i-1})^\perp}(b_i) = b_i^*,$$

which verifies the claim.

Lastly, we verify that the Gram-Schmidt coefficients for the output basis are bounded in absolute value by $\frac{1}{2}$. Consider the operation

$$\hat{b}_i \leftarrow b_i - \lfloor \mu_{ij} \rceil b_j$$

in step 5 of Algorithm 10.18. This operation does not change any $\mu_{i'j'}$ with $i' < i$ and $j' < i'$ that was previously computed in step 4 since $\mu_{i'j'} = \langle b_{i'}, b_{j'}^* \rangle / \|b_{j'}^*\|^2$ and neither $b_{i'}$ nor $b_{j'}^*$ depend on b_i. Also, the operation does not change any previously computed $\mu_{ij'}$ with $j' > j$ since it subtracts an integer multiple of b_j from b_i, and $\langle b_j, b_{j'}^* \rangle = 0$. Finally, the new value of μ_{ij} is

$$\hat{\mu}_{ij} = \frac{\langle \hat{b}_i, b_j^* \rangle}{\|b_j^*\|^2} = \frac{\langle b_i, b_j^* \rangle}{\|b_j^*\|^2} - \lfloor \mu_{ij} \rceil \frac{\langle b_j, b_j^* \rangle}{\|b_j^*\|^2} = \mu_{ij} - \lfloor \mu_{ij} \rceil$$

since $\langle b_j, b_j^* \rangle = \langle b_j^*, b_j^* \rangle$. Thus, $|\hat{\mu}_{ij}| \le \frac{1}{2}$. □

10.3.3 Algorithm description

The LLL lattice basis reduction algorithm (Algorithm 10.20) begins by size reducing the input basis. It then swaps two adjacent basis vectors b_{j-1} and b_j for which the Lovász

condition is violated. The purpose of the swap is to reduce the rate at which the Gram-Schmidt vectors decrease in length. If multiple violations occur, the algorithm is free to choose any one of them;[20] different choices may lead to different LLL-reduced bases, but all offer the same guarantees. After the swap, the new basis is size reduced. The process of swapping a pair of adjacent basis vectors and size reducing the new basis is repeated until all adjacent pairs satisfy the Lovász condition, at which point the basis is LLL-reduced.

Algorithm 10.20: LLL lattice basis reduction algorithm

Input: Lattice basis $B = [b_1, \ldots, b_n]$.
Output: LLL-reduced basis $[b_1, \ldots, b_n]$.

1 Size-reduce $[b_1, \ldots, b_n]$ using Algorithm 10.18.
2 **if** there exists an index $j \in [2, n]$ such that $\|b_j^*\|^2 < (\frac{3}{4} - \mu_{j,j-1}^2)\|b_{j-1}^*\|^2$ **then**
3 Swap b_{j-1} and b_j.
4 Go to step 1.
5 **return** $([b_1, \ldots, b_n])$.

Example 10.21 (*LLL algorithm*) In Example 10.6, we considered the basis

$$B = [(-5, -2, 5, 1),\ (3, 5, 2, 3),\ (0, 1, 2, -3),\ (-3, -4, -4, -5)]$$

for a full-rank integer lattice $L \subseteq \mathbb{Z}^4$ with $\mathrm{vol}(L) = 287$. The squared lengths of the basis vectors are:

$$\|b_1\|^2 = 55, \quad \|b_2\|^2 = 47, \quad \|b_3\|^2 = 14, \quad \|b_4\|^2 = 66.$$

An LLL-reduced basis of L is

$$\tilde{B} = [(0, 1, -2, -2),\ (0, 1, 2, -3),\ (-5, -1, -1, 0),\ (-2, 4, 1, 3)].$$

The squared lengths of the vectors in $\tilde{B}$ are:

$$\|\tilde{b}_1\|^2 = 9, \quad \|\tilde{b}_2\|^2 = 14, \quad \|\tilde{b}_3\|^2 = 27, \quad \|\tilde{b}_4\|^2 = 30.$$

Minkowski's Theorem (Theorem 2.15) guarantees that a shortest nonzero vector v in L satisfies

$$\|v\| = \lambda_1(L) \leq \sqrt{n}\ \mathrm{vol}(L)^{1/n} = \sqrt{4} \times 287^{1/4} \approx 8.23. \tag{71}$$

The lattice vector $\tilde{b}_1 = (0, 1, -2, -2)$ has length 3, showing that the bound in (71) is not tight in this case. An exact SVP solver confirms that $\lambda_1(L) = 3$, and hence the vector $\tilde{b}_1$ appearing in the LLL-reduced basis is indeed a shortest nonzero vector in the lattice.

[20] In the examples of the LLL algorithm, j was always chosen to be the *smallest* index for which the Lovász condition is violated.

Remark 10.22 To better appreciate the intricacies of the LLL algorithm, the reader is strongly encouraged to implement the algorithm independently using a mathematics software system. SageMath (`sagemath.org`) is particularly well suited for this purpose, as it is open source and based on a Python-like language. The reader can validate their own LLL implementation by comparing it with SageMath's built-in routine, and may also work through the computational exercises in §10.6 and §12.8.

Figure 10.5 shows SageMath script for computing the Gram-Schmidt orthogonalization and the associated Gram-Schmidt coefficients of a lattice basis (see Example 10.6), an LLL-reduced basis of the lattice (see Example 10.21), and a shortest nonzero vector in L (using the "enumeration" method that solves SVP exactly).

```
# The rows of matrix B are the basis vectors for an integer lattice L
B = Matrix(ZZ,[[-5,-2,5,1],[3,5,2,3],[0,1,2,-3],[-3,-4,-4,-5]])

print(det(B))            # Determine the volume of L

C,mu = B.gram_schmidt()  # Determine the Gram-Schmidt basis and coefficients
print(C)
print(mu)

D = B.LLL(0.75)          # Determine a (3/4)-LLL-reduced basis for L
print(D)

# Determine a shortest nonzero vector in L
from sage.modules.free_module_integer import IntegerLattice
L = IntegerLattice(B)
v = L.shortest_vector()
print(v)
```

Figure 10.5: SageMath scripts (see Remark 10.22).

Example 10.23 (*LLL algorithm*) Table 10.1 shows the intermediate basis matrices in one application of the LLL algorithm to a 5-dimensional lattice L. The components of the initial basis vectors were randomly selected from the interval $[0, 99]$. Note that the basis vectors are the *rows* of the matrix. The lengths of the vectors in the initial basis are 138.3, 66.4, 87.7, 164.8, and 110.7, respectively. The lengths of the vectors in the LLL-reduced basis are 34.8, 33.4, 42.0, 43.2, and 37.3, respectively. SageMath was used to confirm that the second vector $(16, -4, 21, -20, 2)$ is indeed a shortest nonzero lattice vector.

10.3.4 Termination

By the termination condition of Algorithm 10.20, the output of the LLL algorithm is an LLL-reduced lattice basis. Natural questions are whether the LLL algorithm must eventually terminate, and if so, how quickly. To show that the algorithm terminates, we introduce the important notion of a *potential function*.

Initial basis

$$\begin{bmatrix} 47 & 83 & 38 & 53 & 76 \\ 24 & 15 & 49 & 23 & 26 \\ 30 & 43 & 30 & 26 & 58 \\ 92 & 69 & 80 & 73 & 47 \\ 50 & 76 & 37 & 34 & 38 \end{bmatrix}$$

Size reduction

$$\longrightarrow \begin{bmatrix} 47 & 83 & 38 & 53 & 76 \\ 24 & 15 & 49 & 23 & 26 \\ -17 & -40 & -8 & -27 & -18 \\ 10 & -41 & -34 & -27 & -41 \\ 33 & 36 & 29 & 7 & 20 \end{bmatrix}$$

Swap(1,2) + size reduction

$$\longrightarrow \begin{bmatrix} 24 & 15 & 49 & 23 & 26 \\ -1 & 53 & -60 & 7 & 24 \\ 7 & -25 & 41 & -4 & 8 \\ 34 & -26 & 15 & -4 & -15 \\ 9 & 21 & -20 & -16 & -6 \end{bmatrix}$$

Swap(2,3) + size reduction

$$\longrightarrow \begin{bmatrix} 24 & 15 & 49 & 23 & 26 \\ 7 & -25 & 41 & -4 & 8 \\ -11 & -12 & -27 & -24 & 14 \\ 27 & -1 & -26 & 0 & -23 \\ 16 & -4 & 21 & -20 & 2 \end{bmatrix}$$

Swap(1,2) + size reduction

$$\longrightarrow \begin{bmatrix} 7 & -25 & 41 & -4 & 8 \\ 17 & 40 & 8 & 27 & 18 \\ -11 & -12 & -27 & -24 & 14 \\ 27 & -1 & -26 & 0 & -23 \\ 16 & -4 & 21 & -20 & 2 \end{bmatrix}$$

Swap(2,3) + size reduction

$$\longrightarrow \begin{bmatrix} 7 & -25 & 41 & -4 & 8 \\ -11 & -12 & -27 & -24 & 14 \\ 6 & 28 & -19 & 3 & 32 \\ 27 & -1 & -26 & 0 & -23 \\ 16 & -4 & 21 & -20 & 2 \end{bmatrix}$$

Swap(1,2) + size reduction

$$\longrightarrow \begin{bmatrix} -11 & -12 & -27 & -24 & 14 \\ 7 & -25 & 41 & -4 & 8 \\ 6 & 28 & -19 & 3 & 32 \\ 27 & -1 & -26 & 0 & -23 \\ 9 & 21 & -20 & -16 & -6 \end{bmatrix}$$

Swap(4,5) + size reduction

$$\longrightarrow \begin{bmatrix} -11 & -12 & -27 & -24 & 14 \\ 7 & -25 & 41 & -4 & 8 \\ 6 & 28 & -19 & 3 & 32 \\ 9 & 21 & -20 & -16 & -6 \\ 18 & -22 & -6 & 16 & -17 \end{bmatrix}$$

Swap(3,4) + size reduction

$$\longrightarrow \begin{bmatrix} -11 & -12 & -27 & -24 & 14 \\ 7 & -25 & 41 & -4 & 8 \\ 9 & 21 & -20 & -16 & -6 \\ 6 & 28 & -19 & 3 & 32 \\ 18 & -22 & -6 & 16 & -17 \end{bmatrix}$$

Swap(2,3) + size reduction

$$\longrightarrow \begin{bmatrix} -11 & -12 & -27 & -24 & 14 \\ 9 & 21 & -20 & -16 & -6 \\ 16 & -4 & 21 & -20 & 2 \\ -3 & 7 & 1 & 19 & 38 \\ 18 & -22 & -6 & 16 & -17 \end{bmatrix}$$

Swap(1,2) + size reduction

$$\longrightarrow \begin{bmatrix} 9 & 21 & -20 & -16 & -6 \\ -11 & -12 & -27 & -24 & 14 \\ 16 & -4 & 21 & -20 & 2 \\ -3 & 7 & 1 & 19 & 38 \\ 18 & -22 & -6 & 16 & -17 \end{bmatrix}$$

Swap(2,3) + size reduction

$$\longrightarrow \begin{bmatrix} 9 & 21 & -20 & -16 & -6 \\ 16 & -4 & 21 & -20 & 2 \\ -11 & -12 & -27 & -24 & 14 \\ -3 & 7 & 1 & 19 & 38 \\ 18 & -22 & -6 & 16 & -17 \end{bmatrix}$$

Table 10.1: Intermediate bases in an execution of the LLL algorithm (see Example 10.23). The order of the steps is left to right, top to bottom.

Consider a lattice L with basis $B = [b_1, \ldots, b_n]$ and associated Gram-Schmidt orthogonalization $B^* = [b_1^*, \ldots, b_n^*]$. For each $1 \leq \ell \leq n$, define

$$B_\ell = [b_1, \ldots, b_\ell] \text{ and } L_\ell = L(B_\ell). \tag{72}$$

Observe that L_ℓ is a rank-ℓ sublattice of L.

Remark 10.24 (*lattices that do not have full rank*) The notion of volume of a full-rank lattice (Definition 2.11) extends naturally to low-rank lattices. Let $L_\ell = L(B_\ell) \subseteq \mathbb{R}^n$ be a lattice of rank ℓ, where $1 \leq \ell \leq n$. The volume of L_ℓ is defined to be

$$\text{vol}(L_\ell) = \sqrt{\det(B_\ell^T B_\ell)}. \tag{73}$$

Note that B_ℓ is an $n \times \ell$ matrix, so $B_\ell^T B_\ell$ is an $\ell \times \ell$ matrix. Let the Gram-Schmidt orthogonalization of B_ℓ be $B_\ell^* = [b_1^*, \ldots, b_\ell^*]$. Theorem 10.8 extends naturally to low-rank lattices, so

$$\text{vol}(L_\ell) = \prod_{i=1}^{\ell} \|b_i^*\|. \tag{74}$$

Definition 10.25 Let $B = [b_1, \ldots, b_n]$ be a basis of a full-rank lattice. For each $1 \leq \ell \leq n$, define

$$d_\ell = \text{vol}(L_\ell)^2 = \prod_{i=1}^{\ell} \|b_i^*\|^2. \tag{75}$$

The *potential function* is

$$D(b_1, \ldots, b_n) = \prod_{\ell=1}^{n} d_\ell = \prod_{i=1}^{n} \|b_i^*\|^{2(n+1-i)}. \tag{76}$$

The potential function assigns greater weight to the earlier Gram-Schmidt vectors. Since we are only considering integer lattices, $d_\ell = \text{vol}(L_\ell)^2 = \det(B_\ell^T B_\ell)$ is a positive integer, and hence $D = D(b_1, \ldots, b_n)$ is also a positive integer.

Theorem 10.26 The LLL algorithm terminates after $O(n^2 \log X)$ swaps, where X is the maximum length of the original basis vectors.

Proof: The only operation in the LLL algorithm that changes any sublattice L_ℓ is the Swap operation. Consider $\text{Swap}(b_{j-1}, b_j)$ for some $2 \leq j \leq n$, which only changes L_{j-1}. After this swap, the new lattice basis is $C = [c_1, \ldots, c_n]$, where $c_{j-1} = b_j$, $c_j = b_{j-1}$, and $c_i = b_i$ for all $i \neq j-1, j$. The new Gram-Schmidt orthogonalization is $C^* = [c_1^*, \ldots, c_n^*]$, where $c_i^* = b_i^*$ for all $i \neq j-1, j$. Now,

$$\|c_{j-1}^*\|^2 = \|\Pi_{j-2}(c_{j-1})\|^2 = \|\Pi_{j-2}(b_j)\|^2 = \|b_j^* + \mu_{j,j-1} b_{j-1}^*\|^2 < \tfrac{3}{4}\|b_{j-1}^*\|^2.$$

Thus,

$$\|c_{j-1}^*\|^2 < \tfrac{3}{4}\|b_{j-1}^*\|^2, \tag{77}$$

say $\|c_{j-1}^*\|^2 = \varepsilon\|b_{j-1}^*\|^2$ for some $\varepsilon < \frac{3}{4}$. Let $\tilde{d}_i = \prod_{k=1}^{i} \|c_k^*\|^2$ for $1 \le i \le n$. Then we have $\tilde{d}_{j-1} = \varepsilon d_{j-1}$, whereas $\tilde{d}_i = d_i$ for all $i \ne j-1$. It follows that the potential of the new lattice basis C is

$$\tilde{D} = D(c_1, \ldots, c_n) = \prod_{\ell=1}^{n} \tilde{d}_\ell = \varepsilon \prod_{\ell=1}^{n} d_\ell = \varepsilon D.$$

Consequently, the potential D decreases by a multiplicative factor greater than $\frac{4}{3}$ after each swap.

Now, the potential of the initial basis is

$$D = \prod_{i=1}^{n} \|b_i^*\|^{2(n+1-i)} \le \prod_{i=1}^{n} \|b_i\|^{2(n+1-i)} \le \left(\max_i \|b_i\|\right)^{2(1+2+\cdots+n)} = X^{n^2+n}. \tag{78}$$

Since the potential is a positive integer, the maximum number of swaps before the LLL algorithm terminates is

$$\log_{4/3} X^{n^2+n} = O(n^2 \log X),$$

which concludes the proof. □

10.3.5 Bit complexity

By Theorem 10.26, the total number of iterations executed by the LLL algorithm is polynomial in n and $\log X$. Each iteration consists of a swap followed by a size reduction, the number of arithmetic operations needed being bounded by a polynomial in n. It remains to show that the sizes of the integers arising during the execution of the algorithm do not grow exponentially.

We begin by proving that the Gram-Schmidt orthogonalization process can be carried out in polynomial time, and then show the same for the full LLL algorithm. We will use the following elementary observation to bound the bitlengths of integers appearing in the Gram-Schmidt orthogonalization process and in the LLL algorithm.

Lemma 10.27 Let $\alpha \in \mathbb{Q}$, and let $t \le X^{2n}$ be a positive integer such that $t\alpha \in \mathbb{Z}$ and $|t\alpha| \le \sqrt{n}X^{2n}$. Then $\alpha = s/t$, where both $|s|$ and t have bitlength $O(n \log X)$.

Proof: Since $t\alpha \in \mathbb{Z}$, we can write $\alpha = s/t$ for some $s \in \mathbb{Z}$. Moreover, $|s| = |t\alpha| \le \sqrt{n}X^{2n}$. Consequently, both $|s|$ and t have bitlength $O(n \log X)$. □

We now establish that the Gram-Schmidt orthogonalization process (Algorithm 10.5) runs in polynomial time.

Theorem 10.28 Let $B = [b_1, \ldots, b_n]$ be a basis of an integer lattice $L \subseteq \mathbb{Z}^n$, and let $X = \max_i \|b_i\|$. The Gram-Schmidt orthogonalization process for computing the Gram-Schmidt basis $B^* = [b_1^*, \ldots, b_n^*]$ is a polynomial-time algorithm.[21]

[21] Since $X = \max_i \|b_i\|$, every coordinate of each b_i has absolute value at most X. Therefore, each vector b_i can be represented using $O(n \log X)$ bits. Consequently, the total input size, i.e., the number of bits needed to represent the basis B, is $O(n^2 \log X)$. It follows that an algorithm taking B as input is considered polynomial-time if its running time is bounded by a polynomial in n and $\log X$.

Proof: Recall that $b_1^* = b_1$ and

$$b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{ij} b_j^* \text{ for } 2 \le i \le n, \text{ where } \mu_{ij} = \frac{\langle b_i, b_j^* \rangle}{\|b_j^*\|^2}.$$

Each μ_{ij} lies in $\mathbb{Q}$, and each vector b_i^* belongs to $\mathbb{Q}^n$. Our goal is to show that the bitlengths of the numerators and denominators appearing in the μ_{ij} and in the coordinates of the b_i^* do not grow exponentially.

For each $1 \le \ell \le n$, let B_ℓ, L_ℓ and d_ℓ be as defined in (72) and (75), and let $d_0 = 1$. Recall that each d_ℓ is a positive integer. Since $\|b_i^*\| \le \|b_i\| \le X$, it follows from (74) that $1 \le d_\ell \le X^{2\ell} \le X^{2n}$ for $1 \le \ell \le n$.

Fix $i \in [2, n]$. Because $b_i^* - b_i \in \text{Span}(b_1, \ldots, b_{i-1})$, there exist unique coefficients $x_{ij} \in \mathbb{Q}$ such that

$$b_i^* = b_i + \sum_{j=1}^{i-1} x_{ij} b_j. \tag{79}$$

We next derive a formula for the x_{ij}.

For each $1 \le k \le i-1$, the orthogonality condition $\langle b_i^*, b_k \rangle = 0$ implies

$$\langle b_i^*, b_k \rangle = \langle b_i, b_k \rangle + \sum_{j=1}^{i-1} x_{ij} \langle b_j, b_k \rangle = 0,$$

and hence

$$\sum_{j=1}^{i-1} x_{ij} \langle b_j, b_k \rangle = -\langle b_i, b_k \rangle \text{ for } 1 \le k \le i-1. \tag{80}$$

This system of $i-1$ linear equations in $i-1$ unknowns can be written compactly as $Ax = c$ where $A = B_{i-1}^T B_{i-1}$, and where the j-th entries of vectors x and c are x_{ij} and $-\langle b_i, b_j \rangle$, respectively. Using Cramer's rule yields

$$x_{ij} = \frac{\det(A_j)}{\det(A)} = \frac{\det(A_j)}{\det(B_{i-1}^T B_{i-1})} = \frac{\det(A_j)}{d_{i-1}} \text{ for } 1 \le j \le i-1,$$

where A_j is obtained from A by replacing the j-th column with c. Both $\det(A_j)$ and d_{i-1} are integers. Writing $e_j = \det(A_j)$, we have $x_{ij} = e_j / d_{i-1}$ for $1 \le j \le i-1$. Substituting this expression into (79) and multiplying both sides by d_{i-1} gives

$$d_{i-1} b_i^* = d_{i-1} b_i + \sum_{j=1}^{i-1} e_j b_j,$$

and therefore

$$d_{i-1} b_i^* \in \mathbb{Z}^n. \tag{81}$$

Furthermore,

$$\|d_{i-1} b_i^*\| = d_{i-1} \|b_i^*\| \le d_{i-1} \|b_i\| \le X^{2i-1} \le X^{2n}. \tag{82}$$

Thus, each coordinate of $d_{i-1}b_i^*$ is an integer with absolute value at most X^{2n}. It follows by Lemma 10.27 that the numerator and denominator of each coordinate of b_i^* have bitlength $O(n \log X)$.

We now bound the sizes of the Gram-Schmidt coefficients μ_{ij}. Observe that for each $1 \leq j \leq i-1$, we have

$$d_j \mu_{ij} = d_j \frac{\langle b_i, b_j^* \rangle}{\|b_j^*\|^2} = d_{j-1} \langle b_i, b_j^* \rangle = \langle b_i, d_{j-1} b_j^* \rangle \in \mathbb{Z}. \tag{83}$$

Moreover,

$$|d_j \mu_{ij}| = |\langle b_i, d_{j-1} b_j^* \rangle| \leq \|b_i\| \cdot \|d_{j-1} b_j^*\| \leq X \cdot X^{2j-1} = X^{2j} \leq X^{2n},$$

where the first inequality follows from the Cauchy-Schwarz inequality. Hence,

$$|d_j \mu_{ij}| \leq X^{2n} \text{ for } 1 \leq j \leq i-1. \tag{84}$$

Applying Lemma 10.27 once again, we conclude that each μ_{ij} has numerator and denominator of bitlength $O(n \log X)$.

We have therefore shown that all the integers arising during the Gram-Schmidt orthogonalization process have bitlength $O(n \log X)$, which completes the proof. □

We now prove that the LLL algorithm runs in polynomial time. Throughout, we use $\text{poly}(n, \log X)$ to denote a quantity of the form $O((n \log X)^c)$ for some constant c.

Theorem 10.29 Let $B = [b_1, \ldots, b_n]$ be a basis of a lattice $L \subseteq \mathbb{Z}^n$, and define $X = \max_i \|b_i\|$. The LLL algorithm outputs an LLL-reduced basis of L in time $\text{poly}(n, \log X)$.

Proof: The LLL algorithm (Algorithm 10.20) starts with an initial size reduction (Algorithm 10.18), followed by a sequence of $O(n^2 \log X)$ additional size reductions each preceded by a swap. To prove that LLL runs in polynomial time, it is therefore sufficient to show that each invocation of the size reduction procedure executes in $\text{poly}(n, \log X)$ time.

Claim 1. Throughout the execution of the LLL algorithm, $\|b_i^*\| \leq X$ for all $1 \leq i \leq n$.
Proof of Claim 1: As observed in the proof of Theorem 10.26, the only operation in the LLL algorithm that modifies the Gram-Schmidt vectors is a swap. Consider the operation $\text{Swap}(b_{j-1}, b_j)$ for some $2 \leq j \leq n$, and let $C = [c_1, \ldots, c_n]$ denote the new lattice basis. Then $c_{j-1} = b_j$, $c_j = b_{j-1}$, and $c_i = b_i$ for all $i \neq j-1, j$. Let $C^* = [c_1^*, \ldots, c_n^*]$ be the Gram-Schmidt orthogonalization of C. We have $c_i^* = b_i^*$ for all $i \neq j-1, j$, and $\|c_{j-1}^*\|^2 < \frac{3}{4}\|b_{j-1}^*\|^2$ (see equation (77)).

Now,

$$\begin{aligned} c_j^* &= c_j - \sum_{k=1}^{j-1} \frac{\langle c_j, c_k^* \rangle}{\|c_k^*\|^2} c_k^* \\ &= b_{j-1} - \sum_{k=1}^{j-2} \frac{\langle b_{j-1}, b_k^* \rangle}{\|b_k^*\|^2} b_k^* - \frac{\langle b_{j-1}, c_{j-1}^* \rangle}{\|c_{j-1}^*\|^2} c_{j-1}^* \\ &= b_{j-1}^* - \frac{\langle b_{j-1}, c_{j-1}^* \rangle}{\|c_{j-1}^*\|^2} c_{j-1}^*. \end{aligned} \tag{85}$$

Since

$$b_{j-1} = b_{j-1}^* + \sum_{k=1}^{j-2} \frac{\langle b_{j-1}, b_k^* \rangle}{\|b_k^*\|^2} b_k^*,$$

we have

$$\langle b_{j-1}, c_{j-1}^* \rangle = \langle b_{j-1}^*, c_{j-1}^* \rangle + \sum_{k=1}^{j-2} \frac{\langle b_{j-1}, b_k^* \rangle}{\|b_k^*\|^2} \langle b_k^*, c_{j-1}^* \rangle = \langle b_{j-1}^*, c_{j-1}^* \rangle$$

because $\langle b_k^*, c_{j-1}^* \rangle = \langle c_k^*, c_{j-1}^* \rangle = 0$ for $1 \le k \le j-2$. Thus, equation (85) can be written as

$$c_j^* = b_{j-1}^* - \frac{\langle b_{j-1}^*, c_{j-1}^* \rangle}{\|c_{j-1}^*\|^2} c_{j-1}^*.$$

This means that c_j^* is the projection of b_{j-1}^* onto the orthogonal complement of $\langle c_{j-1}^* \rangle$, implying that $\|c_j^*\| \le \|b_{j-1}^*\|$. Altogether, this shows that $\max_i \|c_i^*\| \le \max_i \|b_i^*\|$. Since initially $\|b_i^*\| \le X$, it follows that every Gram-Schmidt vector computed during the algorithm satisfies $\|b_i^*\| \le X$. □

Claim 2. At the conclusion of each invocation to Algorithm 10.18, the basis vectors satisfy $\|b_i\| \le \sqrt{n}\, X$ for all $1 \le i \le n$.

Proof of Claim 2: Since Algorithm 10.18 produces a size-reduced basis, we have

$$b_i = b_i^* + \sum_{j=1}^{i-1} \mu_{ij} b_j^* \quad \text{with } |\mu_{ij}| \le \tfrac{1}{2}.$$

Therefore,

$$\|b_i\|^2 = \|b_i^*\|^2 + \sum_{j=1}^{i-1} \mu_{ij}^2 \|b_j^*\|^2 \le X^2 + \frac{n-1}{4} X^2 \le nX^2,$$

which implies that $\|b_i\| \le \sqrt{n}\, X$. □

Claim 3. The Gram-Schmidt orthogonalization performed in each invocation of Algorithm 10.18 runs in time $\text{poly}(n, \log X)$.

Proof of Claim 3: Let $B = [b_1, \ldots, b_n]$ be the input basis to an invocation of Algorithm 10.18, and let $B^* = [b_1^*, \ldots, b_n^*]$ denote its Gram-Schmidt orthogonalization. For $1 \le i \le n$, let d_i be as defined in (75), and let $d_0 = 1$. Note that $d_i \le X^{2i} \le X^{2n}$.

As shown in the proof of Theorem 10.28, $d_{i-1} b_i^* \in \mathbb{Z}^n$ for each $1 \le i \le n$. Consequently,

$$\|d_{i-1} b_i^*\| = d_{i-1} \|b_i^*\| \le X^{2i-2} X \le X^{2n}$$

for $2 \le i \le n$. Hence, Lemma 10.27 implies that the numerators and denominators of the components of the b_i^* have bitlength $O(n \log X)$.

The proof of Theorem 10.28 also established that $d_j \mu_{ij} = \langle b_i, d_{j-1} b_j^* \rangle \in \mathbb{Z}$ for all $1 \le j < i \le n$. Thus

$$|d_j \mu_{ij}| = |\langle b_i, d_{j-1} b_j^* \rangle| \le \|b_i\| \cdot d_{j-1} \cdot \|b_j^*\| \le \sqrt{n}\, X \cdot X^{2j-2} \cdot X \le \sqrt{n}\, X^{2n},$$

where the first inequality follows from the Cauchy-Schwarz inequality. Hence, we can deduce by Lemma 10.27 that the numerators and denominators of each μ_{ij} also have bitlength $O(n \log X)$. Finally, since the input basis vectors satisfy $\|b_i\| \le \sqrt{n}\, X$, it follows that the entire Gram-Schmidt orthogonalization can be computed in $\mathrm{poly}(n, \log X)$ time. □

Claim 4. The computations $\mu_{ij} = \langle b_i, b_j^* \rangle / \|b_j^*\|^2$ and $b_i \leftarrow b_i - \lfloor \mu_{ij} \rceil b_j$ in the inner loop of Algorithm 10.18 can be performed in time $\mathrm{poly}(n, \log X)$.

Proof of Claim 4: It suffices to show that the updated vector $v_{ij} = b_i - \lfloor \mu_{ij} \rceil b_j$ has bitlength $\mathrm{poly}(n, \log X)$. Indeed,

$$|\lfloor \mu_{ij} \rceil| = \left| \left\lfloor \frac{\langle b_i, b_j^* \rangle}{\|b_j^*\|^2} \right\rceil \right| \le 1 + \frac{\|b_i\| \cdot \|b_j^*\|}{\|b_j^*\|^2} = 1 + \frac{\|b_i\|}{\|b_j^*\|} \le 1 + X^{2n-2} \|b_i\| \le 2X^{2n-2} \|b_i\|, \quad (86)$$

where the first inequality follows from Cauchy-Schwarz, and the second uses the fact that $d_{j-1} b_j^* \in \mathbb{Z}^n$, implying $\|d_{j-1} b_j^*\| \ge 1$, whence $\|b_j^*\| \ge 1/d_{j-1} \ge 1/X^{2j-2} \ge 1/X^{2n-2}$. During the size reduction of b_i, all vectors b_j with $j < i$ have already been size-reduced. Therefore, applying (86) gives

$$\begin{aligned} \|v_{ij}\| &= \|b_i - \lfloor \mu_{ij} \rceil b_j\| \\ &\le \|b_i\| + |\lfloor \mu_{ij} \rceil| \cdot \|b_j\| \\ &\le \|b_i\| + 2X^{2n-2} \|b_i\| \cdot \|b_j\| \\ &\le (1 + 2X^{2n-2} \sqrt{n}\, X)\ \|b_i\| \\ &\le 4\sqrt{n}\, X^{2n} \|b_i\|. \end{aligned}$$

This shows that each execution of step 5 increases the length of b_i by a factor of at most $4\sqrt{n}\, X^{2n}$. During each invocation of Algorithm 10.18, b_i is updated exactly $i - 1 \le n - 1$ times. Since the input basis satisfies $\|b_i\| \le \sqrt{n}\, X$, we have

$$\|v_{ij}\| \le (4\sqrt{n}\, X^{2n})^{n-1}\, \sqrt{n}\, X,$$

which implies that the bitlength of v_{ij} is $\mathrm{poly}(n, \log X)$. This completes the proof. □

After incorporating some minor optimizations into the LLL algorithm, a more refined analysis (which we omit) yields the following explicit running-time bound.

Theorem 10.30 *(Lenstra, Lenstra, and Lovász)* The running time of the LLL algorithm is $O(n^6 \log^3 X)$ bit operations.

Remark 10.31 The notion of an LLL-reduced basis extends naturally to lattices that do not have full rank. The LLL algorithm can be applied to find LLL-reduced bases in such lattices while preserving its polynomial-time complexity bound.

Remark 10.32 Up to this point, we have assumed that the input to the LLL algorithm is an integer lattice. However, the algorithm extends naturally to rational lattices $L \subseteq \mathbb{Q}^n$, while preserving its polynomial-time complexity bound.

10.3.6 A dimension-15 example

We conclude this section with a more substantial illustration of the LLL algorithm.

Example 10.33 (*LLL algorithm*) We start with a randomly-chosen basis of a full-rank integer lattice L in fifteen dimensions. Each coordinate of the basis vectors was sampled uniformly at random from the interval $[-2, 2]$. This initial basis is displayed in Figure 10.6. By construction, the basis vectors are quite short, with lengths ranging from 4.47 to 6.7.

Next, we deliberately distorted the basis to obtain a much more skewed basis with significantly longer vectors. This is done by repeating the following operation 100 times: choose distinct indices $i, j \in_R [1, 15]$ and a multiplier $t \in_R [-25, 25]$, and replace basis vector b_i by $b_i - t \cdot b_j$. The resulting basis, which still spans the same lattice L, is shown in Figure 10.7.

Applying the LLL algorithm to this skewed basis produces the reduced basis shown in Figure 10.8. The resulting vectors are very short—indeed, even shorter than those of the original randomly generated basis. The shortest vector

$$v = (-2, -1, 1, -1, -1, 0, 1, 1, -1, 0, 0, 0, 0, 0, 1)$$

has length $\sqrt{12} \approx 3.46$, while the longest has length 5.0 (see also Table 10.3). Moreover, each coordinate of every basis vector lies in the interval $[-3, 3]$. SageMath was used to confirm that v is indeed a shortest nonzero vector in L.

In total, the algorithm performed 1157 swaps. Table 10.2 lists the first 66 swaps and the last 66 swaps. The table also records $\log_2 D$, the base-2 logarithm of the potential D, as well as $\log_2(\Delta D)$, the base-2 logarithm of the ratio of the new potential to the old one. Notice that the potential is strictly decreasing, and that $\log_2(\Delta D)$ is always less than $\log_2(3/4)$, i.e., $\log_2(\Delta D) < -0.415037$.

Table 10.3 reports the lengths of the original (distorted) lattice basis vectors b_i and their corresponding Gram-Schmidt basis vectors b_i^*. As expected, the Gram-Schmidt

Swap		$\log_2 D$	$\log_2(\Delta D)$	Swap		$\log_2 D$	$\log_2(\Delta D)$	Swap		$\log_2 D$	$\log_2(\Delta D)$
1	2	2550.7	-2.447960	2	3	2415.6	-1.859969	7	8	458.9	-0.964809
1	2	2547.6	-3.147762	1	2	2413.8	-1.837203	6	7	457.9	-1.036352
1	2	2536.8	-10.758942	2	3	2413.1	-0.683711	5	6	456.9	-0.969374
2	3	2533.0	-3.751924	3	4	2411.8	-1.301468	4	5	456.0	-0.946371
2	3	2529.8	-3.202972	2	3	2411.3	-0.540166	7	8	455.5	-0.527695
2	3	2520.1	-9.709751	4	5	2409.4	-1.895998	8	9	454.5	-0.931297
1	2	2517.8	-2.294698	3	4	2408.7	-0.657851	9	10	453.8	-0.773329
1	2	2514.1	-3.688553	5	6	2406.2	-2.456322	10	11	453.3	-0.451904
2	3	2512.0	-2.125418	4	5	2404.8	-1.441955	11	12	452.6	-0.701915
2	3	2505.5	-6.547939	3	4	2404.2	-0.599990	10	11	452.0	-0.607623
1	2	2502.2	-3.251262	2	3	2403.4	-0.753708	13	14	451.2	-0.842960
3	4	2499.4	-2.841144	1	2	2403.0	-0.448939	12	13	450.1	-1.084172
2	3	2498.6	-0.733080	3	4	2402.2	-0.830693	11	12	449.4	-0.708663
3	4	2494.7	-3.987387	2	3	2401.7	-0.494969	14	15	446.4	-2.963863
2	3	2491.0	-3.627501	5	6	2396.2	-5.509475	13	14	444.0	-2.389888
1	2	2487.9	-3.161642	4	5	2390.9	-5.266025	12	13	442.4	-1.615325
3	4	2484.5	-3.353147	3	4	2386.5	-4.412512	11	12	441.2	-1.191619
2	3	2482.2	-2.339682	2	3	2383.4	-3.086960	10	11	440.3	-0.911681
3	4	2481.3	-0.917240	1	2	2381.3	-2.071812	9	10	439.2	-1.075492
4	5	2477.4	-3.824909	5	6	2379.9	-1.460864	8	9	438.7	-0.528563
3	4	2475.1	-2.296750	4	5	2379.0	-0.858862	7	8	437.9	-0.781122
2	3	2473.5	-1.666312			.....		6	7	436.6	-1.307679
1	2	2472.4	-1.085987			.....		5	6	434.9	-1.681067
4	5	2470.0	-2.381498			.....		4	5	434.4	-0.531945
3	4	2467.1	-2.943220	9	10	484.6	-1.130753	3	4	433.9	-0.521572
2	3	2464.7	-2.340614	8	9	482.9	-1.669808	6	7	433.4	-0.432818
1	2	2462.9	-1.846870	7	8	482.2	-0.691385	7	8	432.4	-1.042172
4	5	2456.1	-6.781332	6	7	481.5	-0.728582	6	7	431.9	-0.519962
3	4	2450.8	-5.270088	5	6	480.4	-1.077897	8	9	430.2	-1.689900
2	3	2447.0	-3.855391	4	5	478.8	-1.630391	7	8	429.3	-0.891293
1	2	2445.7	-1.223059	3	4	477.2	-1.611077	6	7	428.6	-0.653534
4	5	2441.1	-4.619216	2	3	476.1	-1.087169	5	6	428.0	-0.677866
3	4	2439.1	-1.997948	1	2	475.2	-0.943416	4	5	427.3	-0.636691
5	6	2435.6	-3.564133	9	10	473.8	-1.344804	9	10	426.5	-0.818747
4	5	2434.1	-1.448619	10	11	473.0	-0.771700	10	11	425.2	-1.287720
3	4	2433.2	-0.957347	11	12	472.6	-0.474191	9	10	424.4	-0.857999
2	3	2432.1	-1.055315	12	13	471.5	-1.054905	8	9	423.4	-1.004275
3	4	2431.6	-0.534558	11	12	470.7	-0.836913	7	8	422.4	-1.004637
5	6	2428.7	-2.848068	10	11	470.2	-0.463727	6	7	421.8	-0.587346
4	5	2427.2	-1.510119	13	14	467.0	-3.184673	11	12	420.8	-0.950476
3	4	2426.3	-0.899297	12	13	464.6	-2.411948	10	11	420.2	-0.579370
4	5	2425.4	-0.918421	11	12	463.2	-1.433110	9	10	419.7	-0.575864
5	6	2421.4	-3.949843	10	11	462.1	-1.084950	12	13	418.8	-0.867230
4	5	2419.4	-2.057158	9	10	460.9	-1.161409	13	14	417.8	-0.941581
3	4	2417.5	-1.909732	8	9	459.9	-1.018782	14	15	417.2	-0.655858

Table 10.2: The first 66 and the last 66 swaps executed by the LLL algorithm in Example 10.33. The order of the swaps is top to bottom, left to right.

$$\begin{bmatrix}
2 & -1 & 2 & -1 & 0 & 1 & 1 & 2 & 0 & -2 & 2 & -1 & 0 & -1 & -1 \\
1 & 0 & 0 & -2 & -1 & 2 & 1 & 2 & -1 & 0 & -1 & 1 & -2 & 2 & 0 \\
-2 & 2 & 1 & 0 & 2 & 0 & 2 & 1 & 0 & -1 & 0 & -2 & -2 & 0 & 2 \\
2 & -1 & -1 & -1 & 0 & -1 & 1 & -1 & -2 & 2 & -2 & 1 & 1 & 2 & 0 \\
-1 & -2 & -1 & 1 & 2 & 0 & 1 & -2 & -2 & -2 & -1 & -1 & 0 & 1 & 2 \\
-2 & 1 & -2 & -1 & -2 & 2 & -2 & -2 & 1 & 0 & 2 & 1 & 1 & 1 & -1 \\
-1 & -1 & 0 & 2 & 2 & 1 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 1 & 0 \\
-1 & 2 & -1 & 0 & 0 & 1 & 0 & -2 & 1 & -1 & 0 & -1 & -2 & -2 & -2 \\
2 & -1 & 1 & 2 & 1 & -2 & 0 & -2 & 0 & -1 & 1 & -2 & 2 & -1 & 0 \\
-1 & 1 & -2 & -2 & -1 & -2 & 2 & 0 & -1 & 2 & 2 & 0 & -2 & 2 & 1 \\
1 & 2 & 0 & 0 & -2 & 2 & 0 & -2 & -1 & -2 & 1 & 2 & -2 & -2 & -2 \\
0 & -2 & -2 & 0 & 0 & -2 & 1 & 2 & 1 & -2 & -1 & 2 & 2 & -2 & -1 \\
-2 & 2 & 1 & 1 & 2 & 1 & -1 & -2 & 1 & -2 & 2 & -2 & -2 & 2 & -2 \\
1 & 2 & 1 & 2 & -2 & 2 & -1 & 2 & 2 & 2 & -1 & 0 & -2 & 1 & 1 \\
-1 & -1 & -1 & 1 & -2 & 2 & -1 & 2 & -1 & 2 & 2 & -2 & 2 & -1 & -1
\end{bmatrix}_{15\times 15}$$

Figure 10.6: A randomly-selected short basis of a 15-dimensional lattice (Example 10.33). The basis vectors are the rows of the matrix.

lengths $\|b_i^*\|$ decrease rapidly. The table also includes the lengths of the LLL-reduced basis vectors $\tilde{b}_i$ and their Gram-Schmidt counterparts $\tilde{b}_i^*$. Notice that the lengths of the $\tilde{b}_i^*$ are approximately uniformly distributed. One can verify that $\prod_{i=1}^{15} \|b_i^*\|$ and $\prod_{i=1}^{15} \|\tilde{b}_i^*\|$ both equal the integer 23677332, which is the lattice volume.

Finally, Table 10.4 displays the Gram-Schmidt coefficients μ_{ij} of the LLL-reduced basis. As expected, these coefficients satisfy $|\mu_{ij}| \leq \frac{1}{2}$ for $1 \leq j < i \leq 15$.

10.4 LLL improvements

Since its introduction in 1982, the LLL algorithm has undergone numerous enhancements, which we briefly discuss below. The most impactful is the block Korkine-Zolotarev (BKZ) algorithm, a blockwise generalization of LLL that will be described in §11.

10.4.1 Scheduling the Gram-Schmidt computations

The efficiency of LLL can be improved by carefully scheduling the Gram-Schmidt computations. Rather than recomputing the entire Gram-Schmidt orthogonalization after each swap, one can organize the updates so that only those Gram-Schmidt vectors and coefficients μ_{ij} that are affected by the swap are recomputed.

For example, suppose that the 9th and 10th basis vectors are swapped. In this case, the first eight Gram-Schmidt vectors b_i^* remain unchanged, as do all the Gram-Schmidt

$$
\begin{bmatrix}
-115876334270 & 95334076546 & 44582453025 & 57364088199 & 86794272932 \\
48376818744 & -62515108200 & -91698909830 & 60554412628 & -99007760874 \\
102291706978 & -91931888448 & -92286074031 & 67369918728 & -85178843944 \\
-413892649229 & 340531262846 & 159244076714 & 204897045599 & 310035529595 \\
172790730091 & -223283060344 & -327547371938 & 216281025848 & -353634883086 \\
365359051013 & -328376636917 & -329645272684 & 240667714177 & -304246036603 \\
-70725072756 & 58187121125 & 27210878264 & 35012148934 & 52974857343 \\
29526771769 & -38156066064 & -55968401462 & 36959352394 & -60429337352 \\
62433695384 & -56110596809 & -56326771578 & 41119227151 & -51988878529 \\
29344864262 & -24142613972 & -11290390935 & -14527068224 & -21980057784 \\
-12251045046 & 15831330296 & 23221766669 & -15334932342 & 25073054801 \\
-25904628042 & 23281234464 & 23370848279 & -17060537533 & 21570553617 \\
-65530319211 & 53712416149 & 25219705032 & 32425631689 & 48794843177 \\
27407490127 & -35485962608 & -51594189366 & 34386678123 & -56085446912 \\
58012967496 & -51869083935 & -51998985316 & 37521291477 & -48085362525 \\
1791158 & -1685872 & -655465 & -899941 & -1619153 \\
-707577 & 852136 & 1721096 & -804495 & 1436360 \\
-1417528 & 1529799 & 1618272 & -1627284 & 1445896 \\
-624001707 & 511464496 & 240151794 & 308767430 & 464636954 \\
260983533 & -337910637 & -491292191 & 327444016 & -534066865 \\
552421320 & -493913013 & -495149493 & 357279871 & -457881957 \\
-6428054113 & 5288506100 & 2473139854 & 3182181089 & 4814774407 \\
2683626559 & -3467925004 & -5086851386 & 3359158186 & -5492296131 \\
5674467972 & -5099775047 & -5119422800 & 3737241249 & -4725160676 \\
37833052667 & -31126018421 & -14556208209 & -18729114989 & -28337929097 \\
-15794737663 & 20410643195 & 29938810218 & -19770658960 & 32325595186 \\
-33397706287 & 30015479398 & 30131014582 & -21995406398 & 27809973328 \\
9360025257 & -7671966556 & -3602277252 & -4631511281 & -6969553956 \\
-3914752705 & 5068659349 & 7369381469 & -4911659962 & 8011002647 \\
-8286319777 & 7408694854 & 7427241597 & -5359197761 & 6868228288 \\
-794491835566 & 653641371951 & 305679836659 & 393310077243 & 595090601893 \\
331689187293 & -428624161845 & -628708919380 & 415184677459 & -678836947464 \\
701352321466 & -630321510956 & -632746558308 & 461893725042 & -584006153880 \\
1771941120788 & -1457815890550 & -681739794315 & -877191962578 & -1327228248029 \\
-739761653728 & 955959364476 & 1402227862677 & -925977235323 & 1513992730651 \\
-1564209634396 & 1405791028402 & 1411207019720 & -1030197188514 & 1302525192609 \\
-94600 & 208951 & 22318 & 55729 & 246209 \\
12652 & 27277 & -253968 & -34969 & -22976 \\
-20392 & -148178 & -197825 & 408721 & -136922 \\
-7747249494485 & 6373816330920 & 2980742346251 & 3835247462429 & 5802885174723 \\
3234362272426 & -4179583696992 & -6130705304876 & 4048531348870 & -6619463205782 \\
6839005255361 & -6146407224125 & -6170064964319 & 4504093360564 & -5694775864394 \\
586897205399 & -482852088933 & -225807785821 & -290541314978 & -439600904682 \\
-245020902701 & 316626651405 & 464435111645 & -306698701299 & 501461091480 \\
-518092596048 & 465624546756 & 467416782913 & -341210320554 & 431410969263
\end{bmatrix}_{15\times 15}
$$

Figure 10.7: The randomized, initial long basis of the 15-dimensional lattice in Example 10.33. The basis vectors are the rows of the matrix. Due to space constraints, each basis vector spans three lines.

$$\begin{bmatrix}
-1 & 1 & 0 & 2 & 0 & 0 & -1 & -1 & 1 & -1 & 0 & -1 & -1 & 0 & 1 \\
2 & 0 & 1 & 0 & -2 & 1 & 0 & 0 & -2 & -1 & 1 & 3 & 0 & 0 & 0 \\
0 & 0 & -1 & -2 & -1 & -2 & 1 & 1 & 0 & 0 & -1 & -1 & -1 & -1 & -2 \\
2 & -1 & 0 & 1 & -2 & -1 & 1 & 0 & -2 & 0 & -1 & 1 & 0 & -1 & 0 \\
0 & 0 & 1 & -1 & -1 & -1 & -1 & 2 & 0 & -1 & 1 & 1 & 0 & 2 & 1 \\
-1 & -1 & 1 & -1 & 1 & 0 & 0 & 0 & 1 & 0 & -1 & -1 & -1 & -2 & -2 \\
0 & 0 & -1 & 1 & -1 & -1 & 1 & 2 & 0 & 0 & 1 & -1 & 1 & -1 & 1 \\
-1 & -1 & -1 & 0 & 1 & -1 & 1 & 1 & 2 & 0 & -1 & 1 & -1 & 0 & -2 \\
1 & 2 & 1 & 0 & -1 & 2 & -1 & -1 & 1 & -1 & 1 & 0 & 2 & 0 & 1 \\
1 & 1 & 1 & 1 & -1 & 1 & -1 & 2 & -2 & 0 & 1 & -1 & -1 & 1 & 0 \\
3 & -1 & 1 & 0 & 0 & 0 & 1 & 0 & -1 & -1 & 0 & -1 & 0 & 1 & 0 \\
0 & 0 & -1 & -1 & 1 & 1 & 0 & 0 & 1 & -1 & 0 & 0 & -3 & -1 & 0 \\
-2 & -1 & 1 & -1 & -1 & 0 & 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & -1 & 2 & -1 & -2 & -2 & 1 & 0 & 0 & 1 & 0 & -1 & -1 & -1 & 1 \\
0 & -2 & 1 & 0 & -1 & 1 & -2 & -1 & 1 & -1 & 1 & -1 & 1 & 2 & 0
\end{bmatrix}_{15\times 15}$$

Figure 10.8: The LLL-reduced basis of Example 10.33. The basis vectors are the rows of the matrix.

i	$\|b_i\|$	$\|b_i^*\|$	$\|\tilde{b}_i\|$	$\|\tilde{b}_i^*\|$
1	320436945197.4	320436945197.359360255541512	3.61	3.61
2	1144568732312.6	46732111.664529318780066	5.00	4.71
3	195578567373.0	12819.739043033123886	4.47	3.81
4	81148130911.1	397137.610643187052484	4.36	3.15
5	180908523242.6	14000.367117458200531	4.12	3.57
6	5340229.8	1676.733789170875920	4.12	2.88
7	1722666532.5	0.080881385402629	3.74	3.02
8	17775727743.2	0.276952900412341	4.24	3.35
9	104620743288.4	0.000045519492794	4.58	2.74
10	25839996203.4	2.172605704403767	4.36	3.09
11	2197024540111.8	0.574351076442320	4.00	2.97
12	4900011449631.2	0.019296310688108	4.00	2.44
13	656730.3	0.000000000176541	3.46	2.28
14	21423670206853.1	0.002453963889001	4.47	2.52
15	1622962205744.5	0.000000001243810	4.58	3.15

Table 10.3: The lengths of the original (randomized) basis vectors b_i, the LLL-reduced basis vectors $\tilde{b}_i$, and the corresponding Gram-Schmidt basis vectors b_i^* and $\tilde{b}_i^*$ (Example 10.33).

1.00														
-0.46	1.00													
-0.46	-0.35	1.00												
-0.38	0.48	0.37	1.00											
-0.15	0.27	0.01	-0.48	1.00										
-0.08	-0.43	0.36	-0.18	-0.43	1.00									
0.00	-0.09	0.16	0.41	0.35	-0.36	1.00								
-0.15	-0.40	0.34	-0.13	-0.05	0.33	-0.09	1.00							
0.31	0.40	-0.42	-0.45	-0.15	-0.28	0.26	-0.49	1.00						
0.08	0.38	0.03	-0.08	0.35	-0.09	0.24	-0.45	-0.46	1.00					
-0.31	0.23	-0.00	0.40	0.15	0.05	-0.19	-0.29	0.17	0.38	1.00				
0.23	-0.07	0.33	-0.49	-0.35	0.24	-0.06	0.23	-0.42	-0.06	0.36	1.00			
-0.23	-0.02	0.03	-0.02	0.36	0.29	0.24	-0.19	-0.41	-0.31	-0.24	-0.09	1.00		
-0.15	-0.04	0.47	0.42	0.38	0.46	0.09	-0.48	-0.05	-0.37	0.00	-0.01	0.38	1.00	
0.23	0.11	-0.40	-0.29	0.36	0.33	-0.25	-0.26	0.41	0.01	0.32	-0.38	0.05	-0.11	1.00

Table 10.4: Matrix of Gram-Schmidt coefficients μ_{ij} for the LLL-reduced basis in Example 10.33 (and where $\mu_{ii} = 1$).

coefficients μ_{ij} for $1 \le j < i \le 8$. By updating only the quantities that actually change, the algorithm avoids unnecessary recomputation and achieves a noticeable improvement in running time.

10.4.2 The parameter δ

The constant $\frac{3}{4}$ appearing in the Lovász condition (65) can be replaced by an arbitrary constant δ satisfying $\frac{1}{4} < \delta \le 1$.

Definition 10.34 Let $\frac{1}{4} < \delta \le 1$. A lattice basis $B = [b_1, \dots, b_n]$ is called *δ-LLL-reduced* if it is size-reduced and satisfies

$$\|b_k^*\|^2 \ge \left(\delta - \mu_{k,k-1}^2\right) \|b_{k-1}^*\|^2 \quad \text{for all } 2 \le k \le n. \tag{87}$$

Choosing δ closer to 1 improves the approximation factor achieved by LLL. More precisely, we have the following generalization of Theorem 10.16 (Exercise 10.21(a)).

Theorem 10.35 Let $\frac{1}{4} < \delta \le 1$. If $B = [b_1, \dots, b_n]$ is δ-LLL-reduced, then

$$\|b_1\| \le \left(\frac{4}{4\delta - 1}\right)^{(n-1)/2} \lambda_1(L). \tag{88}$$

For $\frac{1}{4} < \delta < 1$ the LLL algorithm runs in polynomial time, although the number of iterations increases with δ. Setting $\delta = 1$ gives the approximation factor $(2/\sqrt{3})^{n-1}$. However, it remains an open problem to determine whether or not the LLL algorithm runs in polynomial time for this choice of δ. In practice, $\delta = 0.99$ is often used.

The first vector b_1 of a δ-LLL-reduced basis satisfies

$$\|b_1\| \le \left(\frac{4}{4\delta - 1}\right)^{(n-1)/4} \mathrm{vol}(L)^{1/n}; \tag{89}$$

see Exercise 10.21(b). Taking $\delta = 1 - \varepsilon'$ for small $\varepsilon' > 0$, (89) can be written as

$$\|b_1\| \leq \left(\frac{4}{3} + \varepsilon\right)^{(n-1)/4} \operatorname{vol}(L)^{1/n} \tag{90}$$

for some small $\varepsilon > 0$ depending on ε'. Thus, the LLL algorithm can be viewed as an algorithmic version of Hermite's upper bound (5) on $\lambda_1(L)$.

10.4.3 Using floating point arithmetic

Recall that the running time of LLL is $O(n^6 \log^3 X)$ bit operations, where n is the lattice dimension and X denotes the maximum length of the input basis vectors. During its execution, the algorithm performs arithmetic operations on integers whose bitlengths are on the order of $O(n \log X)$. In practical settings, the value of X can be quite large. For instance, in applications to the analysis of RSA implementations, X may be 2048 bits or more. For such large values of X, the $O(\log^3 X)$ dependence in the running time becomes a major computational bottleneck. As a result, the original LLL algorithm is typically practical only for lattices of relatively small dimensions.

When working with higher-dimensional lattices, it is therefore common to replace the exact integer arithmetic with floating point arithmetic in the computation of the Gram-Schmidt orthogonalization B^* and coefficients μ_{ij}. While this modification substantially improves efficiency, it also introduces the possibility of rounding errors which, in some cases, may prevent the algorithm from terminating or may result in a basis that is not LLL-reduced.

To address these difficulties, Nguyen and Stehlé proposed a floating-point version of LLL, which they called L^2. In their algorithm, the Gram-Schmidt computations are performed using floating-point arithmetic with a precision of approximately $1.58n$ bits. Notably, this precision is independent of the length X of the input basis vectors. The running time of L^2 has the important feature that it depends only on $\log^2 X$, rather than $\log^3 X$ as in the original LLL algorithm.

10.4.4 Deep insertion

The Lovász condition $\|b_k^*\|^2 \geq (\frac{3}{4} - \mu_{k,k-1}^2)\, \|b_{k-1}^*\|^2$, which can equivalently be written as $\|\Pi_{k-2}(b_k)\|^2 \geq \frac{3}{4}\|b_{k-1}^*\|^2$, guarantees that the Gram-Schmidt basis vectors do not shrink too rapidly. When this condition fails at some index k, that is, when $\|\Pi_{k-2}(b_k)\|^2 < \frac{3}{4}\|b_{k-1}^*\|^2$, the LLL algorithm swaps the adjacent basis vectors b_{k-1} and b_k. The reordered basis then becomes $[\ldots, b_{k-2}, b_k, b_{k-1}, b_{k+1}, \ldots]$.

In 1994, Schnorr and Euchner introduced a refinement of LLL, called *deep insertion*, which permits the swapping of non-adjacent basis vectors. Specifically, when the Lovász condition is violated at index k, then instead of simply swapping b_{k-1} and b_k, the vector b_k is inserted deeper into the basis: it is inserted between b_{i-1} and b_i, where i is the smallest index satisfying

$$\|\Pi_{i-1}(b_k)\|^2 < \frac{3}{4}\|b_i^*\|^2.$$

Observe that choosing $i = k - 1$ recovers the standard LLL swap.

Experiments have demonstrated that incorporating deep insertions often yields significantly shorter lattice bases. This improvement, however, comes at a computational cost: the running time typically increases, and unlike the standard LLL algorithm, LLL with deep insertions is not known to have a polynomial running-time bound.

10.5 Notes and further references


Carl Friedrich Gauss

Gauss described his lattice basis reduction algorithm in 1801. The algorithm was, in fact, first discovered by Lagrange in 1773, but this was not well known until quite recently, and so the algorithm is commonly called Gauss's algorithm. The polynomial-time complexity of Gauss's algorithm was established by Lagarias [137]. Vallée [213] derived the best possible upper bound on the number of iterations, and explored the connection between Gauss's algorithm and Euclid's algorithm for computing the greatest common divisor of two integers. Nguyen and Stehlé [178] showed that the running time of Gauss's algorithm can be improved to $O(\log^2 X)$ bit operations, where X is the length of the longer of the two input basis vectors.

The LLL algorithm was introduced by Lenstra, Lenstra and Lovász [144] in their landmark 1982 paper, which also derived its $O(n^6 \log^3 X)$ bit complexity stated in Theorem 10.30, and presented the first polynomial-time algorithm for factoring polynomials over the rational numbers. The book *The LLL Algorithm: Survey and Applications*, edited by Nguyen and Vallée [181], contains a collection of articles covering algorithmic aspects of lattice reduction, applications to algorithmic number theory, integer programming, and cryptography, as well as the complexity of lattice problems. Chapter 1 provides an interesting historical account of the development of the LLL algorithm.

Stehlé [207] studied practical and theoretical aspects of floating-point variants of LLL. The L^2 variant of LLL was introduced by Nguyen and Stehlé [177]. Nguyen and Stehlé [176] reported extensive experimental results on the average-case behaviour of L^2, aiming to explain the significant gap between its practical performance and the proven worst-case bounds. Specifically, they studied the algorithm's behaviour on rank-d integer lattices $L \subseteq \mathbb{R}^n$ with $n = \Theta(d)$, whose bases are randomly generated with vector lengths bounded by X, where $d = O(\log X)$. Their experiments indicate that the average-case estimate of the *Hermite factor*, defined as $\|b_1\| / \operatorname{vol}(L)^{1/d}$ for the first vector b_1 of the output basis, is approximately 1.02^d. This represents a substantial improvement over the worst-case bound of $(4/3)^{d/4} \approx 1.075^d$ for $\delta \approx 1$; see equation (89). This gap is a feature of the algorithm rather than of the reduction conditions. Indeed, Kim and Venkatesh [129] showed that for almost all lattices, most bases satisfying Siegel reduction, a slightly weaker notion than LLL-reduction, come very close to attaining the worst-case bound. Furthermore, while the worst-case analysis requires about $1.58d$ bits of numerical precision, the average-case analysis suggests that about $0.25d$ bits suffice in practice. These experimental findings were further corroborated by Gama and Nguyen

[94]. A recent LLL implementation incorporating several theoretical improvements is described by Ducas, Pulles and Stevens [81].

10.6 Exercises

Exercise 10.1 Let $w, w_1, w_2 \in \mathbb{R}^n$, and suppose that $w = w_1 + w_2$ and $\langle w_1, w_2 \rangle = 0$. Prove that $\|w\|^2 = \|w_1\|^2 + \|w_2\|^2$.

Exercise 10.2 Let V be a subspace of $\mathbb{R}^n$.
(a) Prove that $\Pi_V(w) = w$ for all $w \in V$, and $\Pi_V(w) = 0$ for all $w \in V^\perp$.
(b) Prove that $\|w\| \geq \|\Pi_V(w)\|$ for all $w \in \mathbb{R}^n$.
(c) Prove that $\Pi_{V^\perp}(w + u) = \Pi_{V^\perp}(w)$ for all $w \in \mathbb{R}^n$ and $u \in V$.

Exercise 10.3 Let $B = [b_1, \ldots, b_k]$ be a basis of a vector space in $\mathbb{R}^n$, and let $B^* = [b_1^*, \ldots, b_k^*]$ be its Gram-Schmidt orthogonalization.
(a) Prove that $\|b_i^*\| \leq \|b_i\|$ for all $1 \leq i \leq k$.
(b) Prove that $\langle b_i, b_i^* \rangle = \langle b_i^*, b_i^* \rangle$ for all $1 \leq i \leq k$.

Exercise 10.4 For each of the following ordered bases B, verify that B^* is indeed the Gram-Schmidt orthogonalization.
(a) $B = [(0, 0, 2),\ (5, 2, 4),\ (-5, 1, -3)]$, $\quad B^* = [(0, 0, 2),\ (5, 2, 0),\ (-30/29, 75/29, 0)]$.
(b) $B = [(-3, 1, 5, 0),\ (2, -4, -1, -2),\ (5, 0, 4, 5),\ (-3, 1, -2, 5)]$,
$B^* = [(-3, 1, 5, 0),\ (5/7, -25/7, 8/7, -2),\ (11/2, -1/2, 17/5, 24/5),$
$\quad (-19673/8463, -20819/8463, -7640/8463, 8595/2821)]$.
(c) $B = [(2, -2, 0, -1, 1),\ (0, 0, 0, 0, -1),\ (2, -2, -2, -2, 1),\ (-1, 1, 2, 0, -2),$
$\quad (-1, 2, -1, 0, 2)]$,
$B^* = [(2, -2, 0, -1, 1),\ (1/5, -1/5, 0, -1/10, -9/10),\ (-2/9, 2/9, -2, -8/9, 0),$
$\quad (-3/11, 3/11, 6/11, -12/11, 0),\ (1/2, 1/2, 0, 0, 0)]$.

Exercise 10.5 Let $B = [b_1, \ldots, b_n]$ be a basis of a lattice $L \subseteq \mathbb{R}^n$, and let $B^* = [b_1^*, \ldots, b_n^*]$ be its Gram-Schmidt orthogonalization. For each i, $1 \leq i \leq n$, prove that

$$\lambda_i(L) \geq \min_{i \leq j \leq n} \|b_j^*\|.$$

Exercise 10.6 Consider the three-dimensional integer lattice L with basis $b_1 = (1, 2, 1)$, $b_2 = (2, 1, 1)$, $b_3 = (1, 1, 2)$. Compute the Gram-Schmidt orthogonalization $[b_1^*, b_2^*, b_3^*]$ and verify that $\mathrm{vol}(L) = \|b_1^*\| \cdot \|b_2^*\| \cdot \|b_3^*\|$.

Exercise 10.7 (†) For each of the following two-dimensional lattices L with basis vectors u and v, determine $\lambda_1(L)$ and $\lambda_2(L)$.
(a) $u = (4, 7)$, $v = (6, 11)$.
(b) $u = (4, 7)$, $v = (5, 11)$.
(c) $u = (4321, 8765)$, $v = (1234, 5678)$.

Exercise 10.8 Let p be a prime that is congruent to 1 modulo 4, and let $i \in \mathbb{Z}_p$ with $i^2 \equiv -1 \pmod{p}$; such an i exists and can be efficiently found. We are interested in efficiently writing p as the sum of two squares.
Consider the lattice $L \subseteq \mathbb{Z}^2$ with basis matrix

$$B = \begin{bmatrix} 1 & 0 \\ i & p \end{bmatrix}.$$

(a) Prove that every $v \in L$ satisfies $\|v\|^2 \equiv 0 \pmod{p}$.
(b) Using Exercise 10.17(d), prove that $\lambda_1(L) = \sqrt{p}$.
(c) Describe an efficient algorithm which, on input a prime $p \equiv 1 \pmod 4$ and $i \in \mathbb{Z}_p$ with $i^2 \equiv -1 \pmod{p}$, finds integers a and b such that $p = a^2 + b^2$.
(d) (†) Write each of the following primes p as the sum of two squares:
(i) $p = 12373$ ($i = 3243$); (ii) $p = 555557$ ($i = 202235$); (iii) $p = 44444453$ ($i = 11833757$).

Exercise 10.9 Explain why the index j in the inner loop in size reduction (Algorithm 10.18) is decremented from $i - 1$ to 1, instead of being incremented from 1 to $i - 1$.

Exercise 10.10 (†) Write a program to find LLL-reduced bases of the lattices $L(B)$ where B is the following basis:
(a) $[(7, 8, -6, -6, -4),\ (2, 2, -1, -1, -1),\ (-2, 1, 0, -2, -4),\ (-5, -10, 7, 7, 6),$
$(0, -3, 0, -3, -3)]$.
(b) $[(-11, 11, -13, -6, 2),\ (36, -18, 54, 44, -11),\ (-1, -13, -6, -9, -1),$
$(-11, 5, -16, -13, 3),\ (-179, 91, -267, -223, 62)]$.
(c) $[(-2403, -1860, 2556, -698, -1275),\ (146, 109, -155, 37, 72),\ (-10, 23, 13, 40, 35),$
$(-482, -411, 510, -193, -305),\ (-143, -106, 152, -35, -70)]$.

Exercise 10.11 (†) The E_8 lattice is a full-rank lattice in $\mathbb{Q}^8$ with basis matrix

$$\begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 & 0 & 1/2 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1/2 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 \end{bmatrix}.$$

Construct a basis of E_8 in which every basis vector has length $\sqrt{2}$.
Hint: Use the LLL algorithm.

Exercise 10.12 (†) Construct a basis of the Leech lattice Λ_{24} (Example 2.13) in which every basis vector has length 2. Conclude that $\lambda_i(\Lambda_{24}) = 2$ for all $1 \le i \le 24$.
Hint: The only column of the basis matrix B in Example 2.13 which doesn't have length 2 is the first column.

Exercise 10.13 Prove that (66) and (67) are equivalent formulations of the Lovász condition.

Exercise 10.14 Each iteration of the LLL algorithm produces a basis of the input lattice L. Find examples of lattices for which two consecutive bases B and B' produced by the LLL algorithm satisfy $\max_{b\in B} \|b\| < \max_{b'\in B'} \|b'\|$. This shows that the basis vectors can increase in length during the LLL algorithm.

Exercise 10.15 Let $B = [b_1, \ldots, b_n]$ and $C = [c_1, \ldots, c_n]$ be two lattice bases produced in successive iterations of the LLL algorithm. Prove that $\min_i \|c_i^*\| \geq \min_i \|b_i^*\|$. (This shows that the minimum length of a Gram-Schmidt basis vector never decreases.)

Exercise 10.16 Let $B = [b_1, \ldots, b_n]$ be an LLL-reduced basis of a lattice $L \subseteq \mathbb{R}^n$, and let $B^* = [b_1^*, \ldots, b_n^*]$ be its Gram-Schmidt orthogonalization. Suppose that $\|b_1\| \leq \|b_i^*\|$ for all $1 \leq i \leq n$. Prove that $\lambda_1(L) = \|b_1\|$.

Exercise 10.17 Let $B = [b_1, \ldots, b_n]$ be an LLL-reduced basis of a lattice $L \subseteq \mathbb{R}^n$, and let $B^* = [b_1^*, \ldots, b_n^*]$ be its Gram-Schmidt orthogonalization.
(a) Prove that $\|b_j^*\|^2 \leq 2^{i-j}\|b_i^*\|^2$ for all $1 \leq j \leq i \leq n$.
(b) Prove that $\|b_i\| \leq 2^{(i-1)/2}\|b_i^*\|$ for all $1 \leq i \leq n$.
(c) Prove that $\|b_j\| \leq 2^{(i-1)/2}\|b_i^*\|$ for all $1 \leq j \leq i \leq n$.
(d) Prove that $\|b_1\| \leq 2^{(n-1)/4} \operatorname{vol}(L)^{1/n}$.

Exercise 10.18 Let $B = [b_1, \ldots, b_n]$ be an LLL-reduced basis of a lattice $L \subseteq \mathbb{R}^n$. Prove that $\|b_i\| \leq 2^{(n-1)/2}\lambda_i(L)$ for $1 \leq i \leq n$.

Exercise 10.19 Let $L \subseteq \mathbb{Z}^n$ be a full-rank integer lattice. Prove that the number of LLL-reduced bases of L is finite.

Exercise 10.20 Let $B = [b_1, \ldots, b_n]$ be a basis of a lattice $L \subseteq \mathbb{R}^n$, and let $B^* = [b_1^*, \ldots, b_n^*]$ be its Gram-Schmidt orthogonalization. For each $\ell \in [1, n]$, let L_ℓ be the sublattice of L with basis $B_\ell = [b_1, \ldots, b_\ell]$. The volume of L_ℓ is defined to be $\operatorname{vol}(L_\ell) = \sqrt{\det(B_\ell^T B_\ell)}$. Prove that $\operatorname{vol}(L_\ell) = \prod_{i=1}^{\ell} \|b_i^*\|$.

Exercise 10.21 Let $\frac{1}{4} < \delta \leq 1$, and let $B = [b_1, \ldots, b_n]$ be a δ-LLL reduced basis of a lattice $L \subseteq \mathbb{R}^n$ (see Definition 10.34).
(a) Prove that

$$\|b_1\| \leq \left(\frac{4}{4\delta - 1}\right)^{(n-1)/2} \lambda_1(L).$$

(b) Prove that

$$\|b_1\| \leq \left(\frac{4}{4\delta - 1}\right)^{(n-1)/4} \operatorname{vol}(L)^{1/n}.$$

Exercise 10.22 Explain why the proof that LLL terminates (Theorem 10.26) works for any $\frac{1}{4} < \delta < 1$, but not for $\delta = 1$.

Exercise 10.23 Let $B = [b_1, \ldots, b_n]$ be a basis of a lattice $L \subseteq \mathbb{R}^n$, and let $B^* = [b_1^*, \ldots, b_n^*]$ be its Gram-Schmidt orthgonalization. Suppose that B is size-reduced, and let $D = \max_i \|b_i^*\|$. Prove that $\|b_i\| \leq \sqrt{n}\, D$ for all $1 \leq i \leq n$.

Exercise 10.24 Design a polynomial-time algorithm that, on input a basis $B = [b_1, \ldots, b_n]$ of a lattice $L \subseteq \mathbb{Z}^n$, computes $\mathrm{vol}(L)$.

Exercise 10.25 Let $B = [b_1, \ldots, b_n]$ be a basis of a lattice $L \subseteq \mathbb{R}^n$. For each $j \in [1, n]$, define $B_j = [b_1, \ldots, b_{j-1}, 2b_j, b_{j+1}, \ldots, b_n]$ and $L_j = L(B_j)$.
(a) Prove that L_j is a sublattice of L for each $j \in [1, n]$.
(b) Let v be a shortest nonzero vector in L, and let $v = \sum_{i=1}^n c_i b_i$ where $c_i \in \mathbb{Z}$. Prove that at least one of the c_i's must be odd.
(c) Let $w = \sum_{i=1}^n d_i b_i \in L$, and suppose that d_j is odd for some $j \in [1, n]$. Let $u = w + b_j$. Prove that $u \in L_j$ and $\|u - b_j\| = \|w\|$. Conclude, using (b), that $\mathrm{dist}(b_k, L_k) \leq \lambda_1(L)$ for at least one index $k \in [1, n]$.
(d) Let $j \in [1, n]$. Prove that $u - b_j \in L \setminus \{0\}$ for every $u \in L_j$, and conclude that $\mathrm{dist}(b_j, L_j) \geq \lambda_1(L)$.
(e) Let $\gamma \geq 1$. Prove that $\mathrm{SVP}_\gamma \leq \mathrm{CVP}_\gamma$.

11 The BKZ lattice basis reduction algorithm

Contents

11.1 BKZ-reduced bases . . . 177
11.2 Algorithm description . . . 180
11.3 Complexity analysis . . . 181
11.4 Enumeration . . . 183
11.5 Sieving . . . 185
11.6 Estimating the security level of Kyber . . . 192
11.7 Estimating the security level of Falcon . . . 196
11.8 Notes and further references . . . 199
11.9 Exercises . . . 201

Block Korkine-Zolotarev (BKZ) is a *blockwise* generalization of the LLL algorithm, introduced by Schnorr and Euchner in 1994. Rather than operating on the entire lattice at once, BKZ repeatedly solves SVP instances in certain low-dimensional lattices called *blocks*. These blocks have dimension β, called the *blocksize*, where β is typically much smaller than the full lattice dimension n. Each SVP instance is solved by an *SVP oracle*, usually implemented via either *enumeration* or *sieving*, both of which have exponential running times; enumeration is described in §11.4, and sieving in §11.5. There is a fundamental tradeoff in the choice of the blocksize: larger values of β typically produce higher-quality reduced bases, but at the cost of a substantial increase in running time. When $\beta = 2$, the BKZ algorithm is essentially the LLL algorithm with parameter $\delta = 1$.

Although precise cost estimates for BKZ remain difficult to obtain, extensive experimental evidence shows that BKZ significantly outperforms LLL in practice. In particular, BKZ has been used to solve some of the hardest SVP challenges, and BKZ running-time estimates play a central role in assessing the hardness of SVP and approx-SVP instances that arise in the security analysis of Kyber, Dilithium, Falcon and FrodoKEM. These estimates, in turn, underpin the selection and justification of the standardized parameter sets for these schemes.

11.1 BKZ-reduced bases

Let $B = [b_1, \ldots, b_n]$ be a basis of a lattice $L \subseteq \mathbb{R}^n$, and let $B^* = [b_1^*, \ldots, b_n^*]$ denote its Gram-Schmidt orthogonalization. Recall that for $1 \le i \le n$, Π_i denotes the projection map onto $\langle b_1, \ldots, b_i \rangle^\perp = \langle b_1^*, \ldots, b_i^* \rangle^\perp$. Also, Π_0 is the identity map on $\mathbb{R}^n$.

For integers $1 \le i < j \le n$, define the *local projected block*

$$B[i,j] = [\Pi_{i-1}(b_i),\ \Pi_{i-1}(b_{i+1}),\ \ldots,\ \Pi_{i-1}(b_j)],$$

and let $L[i,j]$ be the lattice spanned by $B[i,j]$. Since the vectors in $B[i,j]$ are linearly independent, $L[i,j]$ is a lattice of dimension $j - i + 1$. Observe that the first vector of the local projected block $B[i,j]$ is precisely the Gram-Schmidt vector b_i^*.

Definition 11.1 Let $2 \leq \beta \leq n$. A basis $B = [b_1, \ldots, b_n]$ of a lattice $L \subseteq \mathbb{R}^n$ is said to be *β-BKZ-reduced* if B is size-reduced and

$$\|b_j^*\| = \lambda_1(L[j,k]) \text{ for each } 1 \leq j \leq n-1, \tag{91}$$

where $k = \min(j+\beta-1, n)$. The number β is called the *blocksize*.

Remark 11.2 The notion of a β-BKZ-reduced basis extends naturally to lattices that do not have full rank.

A β-BKZ-reduced basis $[b_1, \ldots, b_n]$ is a size-reduced basis whose Gram-Schmidt vectors are shortest nonzero vectors in the lattices generated by the corresponding projected blocks:

$$\|b_i^*\| = \begin{cases} \lambda_1(L[i, i+\beta-1]), & \text{if } 1 \leq i \leq n-\beta+1, \\ \lambda_1(L[i,n]), & \text{if } n-\beta+2 \leq i \leq n-1. \end{cases}$$

Two extreme cases are worth noting. If $\beta = n$, then $\|b_1^*\| = \lambda_1(L)$, so the first basis vector b_1 solves SVP in L. At the other extreme, when $\beta = 2$, a β-BKZ-reduced basis is 1-LLL-reduced (Definition 10.34), as shown in the next result; the converse also holds (Exercise 11.3).

Theorem 11.3 Let $B = [b_1, \ldots, b_n]$ be a 2-BKZ-reduced basis of a lattice $L \subseteq \mathbb{R}^n$. Then B is 1-LLL-reduced.

Proof: For each $1 \leq i \leq n-1$, the projected block corresponding to $L[i, i+1]$ is $B[i, i+1] = [\Pi_{i-1}(b_i),\ \Pi_{i-1}(b_{i+1})]$. We have $\Pi_{i-1}(b_i) = b_i^*$ and $\Pi_{i-1}(b_{i+1}) = b_{i+1}^* + \mu_{i+1,i} b_i^*$. Since B is 2-BKZ-reduced, b_i^* is a shortest nonzero vector in $L[i, i+1]$. Therefore,

$$\|b_i^*\|^2 \leq \|\Pi_{i-1}(b_{i+1})\|^2 = \|b_{i+1}^*\|^2 + \mu_{i+1,i}^2 \|b_i^*\|^2,$$

which implies

$$\|b_{i+1}^*\|^2 \geq (1 - \mu_{i+1,i}^2)\|b_i^*\|^2 \quad \text{for all } 1 \leq i \leq n-1.$$

This is precisely the Lovász condition with parameter $\delta = 1$. □

The next theorem provides an upper bound on the length of the first vector in a β-BKZ-reduced basis. Its proof relies on the following inequality for Hermite constants (Definition 2.14), whose proof is left as an exercise (Exercise 2.21).

Lemma 11.4 For all $1 \leq i \leq j$, we have $\gamma_i^i \leq \gamma_j^j$.

Theorem 11.5 Let $2 \leq \beta \leq n$, and let $B = [b_1, \ldots, b_n]$ be a β-BKZ-reduced basis of a lattice L. Then

$$\|b_1\| \leq \gamma_\beta^{(n-1)/(\beta-1)} \lambda_1(L). \tag{92}$$

Proof: For simplicity, we assume that $n \geq 2\beta - 2$. For $1 \leq i \leq n - \beta + 1$, since $\|b_i^*\| = \lambda_1(L[i, i+\beta-1])$, the Hermite bound[22] gives

$$\|b_i^*\| \leq \sqrt{\gamma_\beta} \cdot \mathrm{vol}\left(L[i, i+\beta-1]\right)^{1/\beta}.$$

Using the volume formula (Remark 10.24), we obtain

$$\|b_i^*\| \leq \sqrt{\gamma_\beta} \left(\prod_{j=i}^{i+\beta-1} \|\Pi_{i-1}(b_j)^*\| \right)^{1/\beta}.$$

Since $\Pi_{i-1}(b_j)^* = b_j^*$ for all $i \leq j \leq i+\beta-1$, we have

$$\|b_i^*\| \leq \sqrt{\gamma_\beta} \left(\prod_{j=i}^{i+\beta-1} \|b_j^*\| \right)^{1/\beta}. \tag{93}$$

Raising both sides of (93) to the power β gives

$$\|b_i^*\|^\beta \leq \gamma_\beta^{\beta/2} \prod_{j=i}^{i+\beta-1} \|b_j^*\|. \tag{94}$$

Multiplying (94) over $i = 1, \ldots, n-\beta+1$ yields the inequality

$$\prod_{i=1}^{n-\beta+1} \|b_i^*\|^\beta \leq \gamma_\beta^{(n-\beta+1)\beta/2} \left(\prod_{i=1}^{\beta-1} \|b_i^*\|^i \right) \left(\prod_{i=\beta}^{n-\beta+1} \|b_i^*\|^\beta \right) \left(\prod_{i=n-\beta+2}^{n} \|b_i^*\|^{n-i+1} \right),$$

which after cancelling terms simplifies to

$$\prod_{i=1}^{\beta-1} \|b_i^*\|^{\beta-i} \leq \gamma_\beta^{(n-\beta+1)\beta/2} \prod_{i=n-\beta+2}^{n} \|b_i^*\|^{n-i+1}. \tag{95}$$

Next, since $L[1,i] \subseteq L[1,\beta]$ for $2 \leq i \leq \beta$, the vector b_1^* being a shortest nonzero vector in $L[1,\beta]$, is also a shortest nonzero vector in $L[1,i]$. Applying the Hermite bound (3) and the volume formula yields

$$\|b_1^*\|^i \leq \gamma_i^{i/2} \ \mathrm{vol}(L[1,i]) = \gamma_i^{i/2} \prod_{j=1}^{i} \|b_j^*\| \ \text{ for all } 2 \leq i \leq \beta - 1. \tag{96}$$

Multiplying the $\beta - 2$ inequalities (96) and $\|b_1^*\| \leq \|b_1^*\|$, we obtain

$$\prod_{i=1}^{\beta-1} \|b_1^*\|^i \leq \left(\prod_{i=2}^{\beta-1} \gamma_i^{i/2} \right) \left(\prod_{i=1}^{\beta-1} \|b_i^*\|^{\beta-i} \right).$$

[22] The Hermite bound (3) is also valid for lattices that do not have full rank: if $L \subseteq \mathbb{R}^n$ is a rank-β lattice, then $\lambda_1(L) \leq \sqrt{\gamma_\beta}\ \mathrm{vol}(L)^{1/\beta}$.

Applying Lemma 11.4 and substituting (95) gives

$$\|b_1^*\|^{\beta(\beta-1)/2} \le \gamma_\beta^{\beta(\beta-2)/2} \gamma_\beta^{(n-\beta+1)\beta/2} \prod_{i=n-\beta+2}^{n} \|b_i^*\|^{n-i+1} = \gamma_\beta^{\beta(n-1)/2} \prod_{i=n-\beta+2}^{n} \|b_i^*\|^{n-i+1}. \tag{97}$$

Suppose that there is a shortest nonzero vector $v \in L$ such that $\Pi_{n-1}(v) \neq 0$. Then $\Pi_j(v) \neq 0$ for all $j \le n-1$. Since B is β-BKZ-reduced, we have $\|b_i^*\| = \lambda_1(L[i,n])$ for each $i \in [n-\beta+2, n-1]$. Hence

$$\|b_i^*\| \le \|\Pi_{i-1}(v)\| \le \|v\| = \lambda_1(L) \text{ for } n-\beta+2 \le i \le n-1. \tag{98}$$

Let $v = c_1 b_1 + \cdots + c_n b_n$ where $c_j \in \mathbb{Z}$, whence $\Pi_{n-1}(v) = c_n b_n^*$. Since $\Pi_{n-1}(v) \neq 0$, we have $c_n \neq 0$. Thus

$$\|b_n^*\| \le \|\Pi_{n-1}(v)\| \le \|v\| = \lambda_1(L). \tag{99}$$

Substituting (98) and (99) into (97) gives

$$\|b_1^*\|^{\beta(\beta-1)/2} \le \gamma_\beta^{\beta(n-1)/2} \prod_{i=n-\beta+2}^{n} \lambda_1(L)^{n-i+1} = \gamma_\beta^{\beta(n-1)/2} \lambda_1(L)^{\beta(\beta-1)/2}.$$

Raising both sides to the power $2/\beta(\beta-1)$ and substituting $b_1 = b_1^*$ gives

$$\|b_1\| \le \gamma_\beta^{(n-1)/(\beta-1)} \lambda_1(L).$$

If $\Pi_{n-1}(v) = 0$ for all shortest nonzero vectors $v \in L$, then let $k < n-1$ be the largest index for which one of the shortest nonzero vectors $v \in L$ satisfies $\Pi_k(v) \neq 0$. Here, we also assume for simplicity that $k+1 \ge 2\beta-2$. Let $B' = [b_1, \ldots, b_{k+1}]$ and $L' = L(B')$, and note that $\lambda_1(L') = \lambda_1(L)$. Then B' is a β-BKZ-reduced basis of L'. The result established above also holds for lattices which do not have full rank. Thus,

$$\|b_1\| \le \gamma_\beta^{k/(\beta-1)} \lambda_1(L') \le \gamma_\beta^{(n-1)/(\beta-1)} \lambda_1(L),$$

which completes the proof of the theorem. □

11.2 Algorithm description

No efficient algorithm for computing a β-BKZ-reduced basis of a lattice $L \subseteq \mathbb{R}^n$ is known. This is not surprising since, as noted above, the first vector of an n-BKZ-reduced basis solves SVP in L, which is NP-hard. Moreover, by Theorem 11.3, a 2-BKZ-reduced basis is a 1-LLL-reduced basis, and no polynomial-time algorithm is known for computing the latter.

The BKZ algorithm (Algorithm 11.6) begins with an LLL-reduced basis $B = [b_1, \ldots, b_n]$ and repeatedly performs *tours* until a β-BKZ-reduced basis is obtained. A tour consists of a sequence of $n-1$ *block operations*. The first $n-\beta+1$ block operations act on the

Algorithm 11.6: BKZ lattice basis reduction algorithm

Input: Basis $B = [b_1, \ldots, b_n]$ for a lattice L, blocksize $\beta \in [2, n]$.
Output: β-BKZ-reduced basis B.

$B \leftarrow \mathrm{LLL}(B)$. // compute an LLL-reduced basis of L
repeat
  Perform block operations on $B[1, \beta]$, $B[2, \beta + 1], \ldots, B[n - \beta + 1, n]$.
  Perform block operations on $B[n - \beta + 2, n]$, $B[n - \beta + 3, n], \ldots, B[n - 1, n]$.
until the basis B doesn't change.
return (B).

length-β blocks $B[i, i + \beta - 1]$ for $1 \leq i \leq n - \beta + 1$. The remaining $\beta - 2$ block operations act on the smaller blocks $B[i, n]$ for $n - \beta + 2 \leq i \leq n - 1$, whose lengths decrease from $\beta - 1$ down to 2.

Like the LLL swap operation, the BKZ block operation (Algorithm 11.7) is a local modification of a lattice basis, designed to prevent the Gram-Schmidt vectors from shrinking too rapidly. However, while the LLL swap acts on two adjacent basis vectors, the BKZ block operation works on a block of up to β consecutive basis vectors.

The BKZ block operation on $B[i, j]$ is the following. First, an SVP oracle (either enumeration or sieving) is used to find a shortest nonzero vector v' in the projected lattice $L[i, j]$. Writing $v' = \sum_{k=i}^{j} a_k \Pi_{i-1}(b_k)$ where $a_k \in \mathbb{Z}$, we define the corresponding lattice vector $v = \sum_{k=i}^{j} a_k b_k \in L$ so that $\Pi_{i-1}(v) = v'$. Next, the vector v is inserted between b_{i-1} and b_i producing the ordered set

$$B' = [b_1, \ldots, b_{i-1}, v, b_i, \ldots, b_n].$$

Of course, B' is not a lattice basis since it has one vector too many. Nonetheless, notice that the Gram-Schmidt orthogonalization of the first i vectors in B' is $[b_1^*, \ldots, b_{i-1}^*, v^*]$ where

$$\|v^*\| = \|\Pi_{i-1}(v)\| = \|v'\| \leq \|\Pi_{i-1}(b_i)\| = \|b_i^*\|,$$

and where b_i^* is the i-th Gram-Schmidt vector of basis B. Thus the i-th vector v^* in the Gram-Schmidt orthogonalization of B' is at most as long as the i-th vector b_i^* in the Gram-Schmidt orthogonalization of B (and the first $i - 1$ Gram-Schmidt vectors remain unchanged). Let $h = \min(j + 1, n)$, so the next BKZ block operation is on $B[i + 1, h]$. If indeed $\|v^*\| < \|b_i^*\|$, then linear independence of B' can be restored by applying a modified version of LLL (not described here) to the linearly dependent vectors $[b_1, \ldots, b_{i-1}, v, b_i, \ldots, b_h]$ yielding the linearly independent (and LLL-reduced) vectors $[\hat{b}_1, \ldots, \hat{b}_h]$. On the other hand, if $\|v^*\| = \|b_i^*\|$, then LLL is applied to $[b_1, \ldots, b_h]$.

11.3 Complexity analysis

No satisfactory upper bound is known on the number of tours performed by Algorithm 11.6, and it remains an open question whether the number of tours can be bounded

Algorithm 11.7: BKZ block operation

Input: Basis $B = [b_1, \ldots, b_n]$ for a lattice L, indices $1 \le i < j \le n$.

Output: Basis B after performing the BKZ block operation on $B[i, j]$.

Find a shortest nonzero vector v' in $L[i, j]$ using enumeration or sieving
Compute $v = \sum_{k=i}^{j} a_k b_k$ where $v' = \sum_{k=i}^{j} a_k \Pi_{i-1}(b_k)$.
Set $h \leftarrow \min(j + 1, n)$.
if $\|v^*\| < \|b_i^*\|$ **then**
    $\hat{B} \leftarrow$ modified-LLL$([b_1, \ldots, b_{i-1}, v, b_i, \ldots, b_h])$.
else
    $\hat{B} \leftarrow \mathrm{LLL}([b_1, \ldots, b_h])$.
$B \leftarrow \hat{B} \cup [b_{h+1}, \ldots, b_n]$.
return (B).

by a polynomial in the lattice dimension n. In practical implementations, BKZ is terminated after successive tours no longer produce a significant improvement in the quality of the basis. The complexity analysis presented here assumes such early-abort strategies are employed.

The blocksize β is the central parameter controlling both the running time and quality of the output basis. As β increases, BKZ tends to produce shorter basis vectors. This improvement, however, comes at a computational cost since the SVP solvers employed (enumeration or sieving) require exponential time in β. Roughly speaking, the running time of BKZ can be written as $O(2^{c\beta}\mathrm{poly}(n))$ for some constant $c > 0$, where the $2^{c\beta}$ term reflects the exponential cost of solving SVP in each block, and $\mathrm{poly}(n)$ reflects the total number of SVP oracle calls and overhead per call.

In terms of quality of the output basis B, BKZ achieves a Hermite factor of roughly $\beta^{O(n/\beta)}$, where the *Hermite factor* of B is defined as

$$\delta(B) = \|b_1\| / \mathrm{vol}(L)^{1/n}, \tag{100}$$

b_1 being the first basis vector. At one extreme, when the blocksize is $\beta = 2$, BKZ essentially coincides with the LLL algorithm with $\delta = 1$. In practice, LLL is run with δ slightly less than 1, in which case it runs in polynomial time and achieves a Hermite factor of approximately $(\frac{4}{3})^{n/4}$ (equation 89 with $\delta \approx 1$). At the opposite extreme, when the blocksize is $\beta = n$, BKZ reduces to an SVP solver, achieving a Hermite factor of roughly $\sqrt{n}$.

Between these two extremes lies the general case of BKZ. When the blocksize is $\beta = n^\alpha$ for some $\alpha \in (0, 1)$, BKZ exhibits subexponential running time $2^{\tilde{O}(n^\alpha)}$ and achieves a Hermite factor $2^{\tilde{O}(n^{1-\alpha})}$. (Here, the tilde indicates that polylogarithm factors in n are suppressed.) For example, if $\alpha = \frac{1}{2}$ so $\beta = \sqrt{n}$, both the running time and the Hermite factor of BKZ are approximately $2^{\sqrt{n}}$. Table 11.1 compares the performance of the LLL,

Algorithm	Approximate running time	Performance	Notes
LLL	$\text{poly}(n)$	Hermite factor $\approx 2^{n/4} \approx 1.19^n$ with LLL parameter $\delta = \frac{3}{4}$	proven (Theorem 10.17)
	$\text{poly}(n)$	Hermite factor $\approx (4/3)^{n/4} \approx 1.075^n$ with LLL parameter $\delta \approx 1$	proven (Theorem 10.35)
	$\text{poly}(n)$	Hermite factor $\approx 1.02^n$ with LLL parameter $\delta \approx 1$	experimental (Nguyen-Stehlé)
BKZ-β	$\text{poly}(n) \cdot 2^{0.292\beta}$	Hermite factor $\approx (\beta/2\pi e)^{n/2\beta}$ for $\beta \geq 50$	see §11.6
Enumeration	$2^{0.184n \log_2 n}$	Exact: finds a vector of length $\lambda_1(L)$	see §11.4
Sieving	$2^{0.292n}$	Exact: finds a vector of length $\lambda_1(L)$	see §11.5

Table 11.1: Comparison of the lengths of the shortest lattice vectors produced by LLL, BKZ, enumeration, and sieving.

BKZ, enumeration, and sieving algorithms according to the length of the shortest vector returned by each method.

11.4 Enumeration

Enumeration is a method for solving SVP, i.e., finding a shortest nonzero vector in a lattice $L(B) \subseteq \mathbb{Z}^n$. Although its running time is superexponential in the lattice dimension n, enumeration has the advantage of requiring very little memory.

The algorithm begins with an upper bound D on the first successive minimum $\lambda_1(L)$. It then systematically enumerates integer vectors $x \in \mathbb{Z}^n$ satisfying $0 < \|x\| \leq D$, and outputs a lattice vector of shortest length found during the search. By first applying the LLL algorithm, we may assume without loss of generality that the lattice basis $B = [b_1, \ldots, b_n]$ is LLL-reduced. Minkowski's Theorem provides a natural upper bound for $\lambda_1(L)$:

$$D = \min(\|b_1\|, \sqrt{n}\ \text{vol}(L)^{1/n}). \tag{101}$$

11.4.1 Naive enumeration

A naive approach to finding a shortest nonzero lattice vector involves enumerating all candidate vectors $v = (v_1, \ldots, v_n) \in [-D, D]^n$ one coordinate at a time and testing each for membership in L. This can be done systematically using a *depth-first search tree* of height n. The root, at level 0, is denoted by R. For each $1 \leq i \leq n$, nodes at level i are labelled by the admissible integer values of v_i. Consequently, each internal node has $2D + 1$ children, and the leaves correspond precisely to the vectors in $[-D, D]^n$. A node z at level i represents the partial vector $(v_1, \ldots, v_i)$, whose coordinates are the labels of nodes along the unique path from R to z. During the depth-first traversal, the search tree can be *pruned* by maintaining the length of the current partial vector. If this partial

length exceeds either the bound D or the length of the shortest nonzero lattice vector found so far, the corresponding branch is discarded and the algorithm backtracks, since no extension of the current partial vector can yield a shorter lattice vector. In the worst case, the running time of this naive approach is $(2D+1)^n$.

11.4.2 Basic enumeration

For integers $x_1, \ldots, x_n$, let $x = x_1 b_1 + \cdots + x_n b_n$ denote the lattice vector determined by using the x_i's as the coefficients of a linear combination of the basis vectors. The basic enumeration strategy consists of enumerating all integer tuples $(x_1, \ldots, x_n)$ such that $\|x\| \leq D$, and selecting a nonzero x of shortest length. Bounds on the coefficients x_i are derived by expressing x in terms of the Gram-Schmidt orthogonalization $B^* = [b_1^*, \ldots, b_n^*]$ of the basis B.

Since $b_1 = b_1^*$ and $b_i = b_i^* + \sum_{j=1}^{i-1} \mu_{ij} b_j^*$ for $2 \leq i \leq n$, we can write

$$x = \sum_{i=1}^{n} x_i b_i = \sum_{i=1}^{n} x_i \left(b_i^* + \sum_{j=1}^{i-1} \mu_{ij} b_j^* \right) = \sum_{j=1}^{n} \left(x_j + \sum_{i=j+1}^{n} \mu_{ij} x_i \right) b_j^*.$$

Recall that for $0 \leq i \leq n-1$, Π_i denotes the projection onto the orthogonal complement of $\langle b_1^*, \ldots, b_i^* \rangle$. Since the vectors $b_1^*, \ldots, b_n^*$ form an orthogonal basis, we have $\Pi_k(b_j^*) = 0$ for all $1 \leq j \leq k \leq n-1$, and $\Pi_k(b_j^*) = b_j^*$ for all $0 \leq k < j \leq n$. Therefore, for all $0 \leq k \leq n-1$ we have

$$\Pi_k(x) = \sum_{j=k+1}^{n} \left(x_j + \sum_{i=j+1}^{n} \mu_{ij} x_i \right) b_j^* \tag{102}$$

and thus

$$\|\Pi_k(x)\|^2 = \sum_{j=k+1}^{n} \left(x_j + \sum_{i=j+1}^{n} \mu_{ij} x_i \right)^2 \|b_j^*\|^2. \tag{103}$$

For any lattice vector x of length at most D, we have the chain of inequalities

$$\|\Pi_{n-1}(x)\| \leq \|\Pi_{n-2}(x)\| \leq \cdots \leq \|\Pi_2(x)\| \leq \|\Pi_1(x)\| \leq \|\Pi_0(x)\| = \|x\| \leq D.$$

Applying (103) to the inequality $\|\Pi_{n-1}(x)\| \leq D$, we obtain $x_n^2 \|b_n^*\|^2 \leq D^2$, which yields the bound

$$|x_n| \leq D / \|b_n^*\|. \tag{104}$$

From (102), we note that $\Pi_{k-1}(x)$ depends only on the coefficients $x_k, x_{k+1}, x_{k+2}, \ldots, x_n$. Proceeding recursively, if integers $x_n, x_{n-1}, \ldots, x_{k+1}$ are fixed such that

$$\|\Pi_{n-1}(x)\| \leq \|\Pi_{n-2}(x)\| \leq \cdots \leq \|\Pi_k(x)\| \leq D,$$

the admissible values for x_k are determined by the condition $\|\Pi_{k-1}(x)\| \leq D$. Using (103), this condition can be written as

$$\left(x_k + \sum_{i=k+1}^{n} \mu_{ik} x_i\right)^2 \|b_k^*\|^2 \leq D^2 - \sum_{j=k+1}^{n} \left(x_j + \sum_{i=j+1}^{n} \mu_{ij} x_i\right)^2 \|b_j^*\|^2.$$

This gives the bound

$$|x_k + \Delta| \leq \frac{\sqrt{D^2 - \Omega}}{\|b_k^*\|}, \tag{105}$$

where

$$\Delta = \Delta(x_{k+1}, \ldots, x_n) = \sum_{i=k+1}^{n} \mu_{ik} x_i$$

and

$$\Omega = \Omega(x_{k+1}, \ldots, x_n) = \sum_{j=k+1}^{n} \left(x_j + \sum_{i=j+1}^{n} \mu_{ij} x_i\right)^2 \|b_j^*\|^2.$$

One proceeds to find a shortest nonzero lattice vector by systematically enumerating all admissible integer tuples $(x_1, \ldots, x_n)$ where the x_i are restricted by (104) and (105). The search is structured as a depth-first search on a tree of height n. The root of the tree is denoted by R. For each $1 \leq i \leq n$, nodes at level i are labelled by the admissible integer values of x_{n-i+1}.

11.4.3 Analysis

The complexity of the basic enumeration procedure is bounded by the number of nodes in the search tree. From (104) and (105), we see that the admissible values for x_k lie in an interval of length at most $2D/\|b_k^*\|$. As a rough heuristic, suppose that for each k the number of admissible values for x_k is about $2D/\|b_k^*\|$. Under this assumption, the total number of admissible tuples $(x_1, \ldots, x_n)$ is roughly

$$\prod_{k=1}^{n} \frac{2D}{\|b_k^*\|} = \frac{2^n D^n}{\prod_{k=1}^{n} \|b_k^*\|} \leq \frac{2^n (\sqrt{n}\ \mathrm{vol}(L)^{1/n})^n}{\mathrm{vol}(L)} = 2^n n^{n/2} = 2^{\frac{1}{2} n \log_2 n + O(n)}.$$

Thus, heuristically, the running time of the basic enumeration procedure is

$$2^{\frac{1}{2} n \log_2 n + O(n)}.$$

11.5 Sieving

Sieving is a probabilistic method for solving SVP in an integer lattice $L \subseteq \mathbb{Z}^n$. Whereas sieving algorithms are conceptually quite simple, their analysis can be very complicated. In this section, we will outline the Nguyen-Vidick sieving algorithm and give a simplified heuristic analysis.

Overview. The algorithm begins with a set S_0 of $N_0 \approx n^3(4/3)^{n/2}$ randomly sampled lattice vectors each of length approximately R_0, for some $R_0 \leq 2^{O(n)}\lambda_1(L)$. Let $\gamma < 1$ be a *shrinking factor*; for the sake of concreteness, we will take $\gamma = 1 - \frac{1}{n}$. For $k = 1, 2, 3, \ldots$, the algorithm iteratively applies a sieving step to S_{k-1} to produce a set S_k of lattice vectors each of which has length approximately $R_k = \gamma^k R_0$; see Figure 11.1. Notably, the sets do not significantly decrease in size, so S_k has size $N_k \approx N_0$. However, once $R_k \approx \lambda_1(L)$, the sets S_k rapidly decrease in size and inevitably collapse, namely $S_\ell = \emptyset$. In that event, the shortest vectors in $S_{\ell-1}$ are expected to have lengths close to $\lambda_1(L)$.

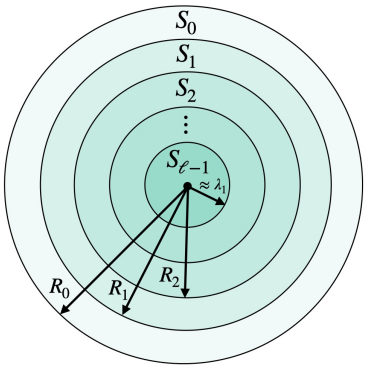


Figure 11.1: The sieving method.

11.5.1 Preliminaries

We collect here some notation and geometric facts about balls and spheres that will be used in the algorithm and its analysis.

Definition 11.8 The *ball* $B_n(v, r)$ of radius r centered at $v \in \mathbb{R}^n$ is defined by

$$B_n(v, r) = \{x \in \mathbb{R}^n \ : \ \|x - v\| \leq r\}.$$

If $v = 0$, we will denote $B_n(v, r)$ by $B_n(r)$. The *unit sphere* is

$$S = \{x \in \mathbb{R}^n \ : \ \|x\| = 1\},$$

and its elements are called *unit vectors*.

Theorem 11.9 The volume of the ball $B_n(R)$ is

$$\mathrm{vol}(B_n(R)) = \frac{\pi^{n/2}}{\Gamma(\frac{n}{2} + 1)} R^n,$$

where Γ is the gamma function.

We next prove an upper bound on the number of lattice points that can be contained in a ball centered at the origin.

Theorem 11.10 Let $L \subseteq \mathbb{Z}^n$ be a lattice with $\lambda_1 = \lambda_1(L)$, and let $R > 0$. The number of lattice points v in $B_n(R)$ satisfies

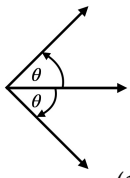
$$|L \cap B_n(R)| \leq \left(1 + \frac{2R}{\lambda_1}\right)^n .$$

Proof: By definition of $\lambda_1(L)$, the balls $B_n(v, \frac{\lambda_1}{2})$ centered at lattice points $v \in B_n(R)$ do not overlap, and are all contained in $B_n(R + \frac{\lambda_1}{2})$. Thus,

$$|L \cap B_n(R)| \leq \frac{\mathrm{vol}(B_n(R + \frac{\lambda_1}{2}))}{\mathrm{vol}(B_n(\frac{\lambda_1}{2}))} = \frac{\left(R + \frac{\lambda_1}{2}\right)^n}{\left(\frac{\lambda_1}{2}\right)^n} = \left(1 + \frac{2R}{\lambda_1}\right)^n . \qquad \square$$

The *angle* $\theta \in [0, \pi]$ between nonzero vectors $u, v \in \mathbb{R}^n$ is defined to be

$$\cos^{-1}\left(\frac{\langle u, v\rangle}{\|u\|\,\|v\|}\right).$$



The *law of cosines* states that

$$\|u - v\|^2 = \|u\|^2 + \|v\|^2 - 2\|u\|\,\|v\|\,\cos\theta. \tag{106}$$

If u and v are unit vectors, then

$$\|u - v\| = \sqrt{2(1 - \cos\theta)}.$$

The following gives the probability density function of the angle between two random unit vectors.

Theorem 11.11 Fix $c \in S$. Let $\theta \in [0, \pi]$ be the angle between c and v, where v is selected uniformly at random from S. Then the density function of θ is

$$f_n(\theta) = \frac{\Gamma(\frac{n}{2})}{\sqrt{\pi}\,\Gamma(\frac{n-1}{2})}\,\sin^{n-2}\theta.$$

Remark 11.12 Observe that if $n = 2$, then

$$f_n(\theta) = \frac{\Gamma(1)}{\sqrt{\pi}\,\Gamma(\frac{1}{2})} = \frac{1}{\sqrt{\pi}\,\sqrt{\pi}} = \frac{1}{\pi},$$

so θ is uniformly distributed over $[0, \pi]$, as expected. On the other hand, if n is large then $\Gamma(\frac{n}{2})/\Gamma(\frac{n-1}{2}) \approx \sqrt{\frac{n}{2}}$, so

$$f_n(\theta) \approx \sqrt{\frac{n}{2\pi}}\,\sin^{n-2}\theta.$$

Since $\sin^{n-2}\theta$ is maximized at $\theta = \pi/2$, and the peak sharpens rapidly with n, $f_n(\theta)$ is sharply concentrated around $\theta = \pi/2$. Consequently, two random unit vectors in $\mathbb{R}^n$ (n large) are nearly orthogonal with high probability, illustrating the counterintuitive geometry of high-dimensional spaces.

Lemma 11.14 estimates the probability that two random unit vectors are close to each other. Its proof will use the following technical result.

Lemma 11.13 For large n,

$$\int_0^{\pi/3} \sin^n\theta\, d\theta \approx \frac{\sqrt{3}}{n}\left(\frac{\sqrt{3}}{2}\right)^n.$$

Proof: Substituting $\theta = \pi/3 - t$, we have

$$\int_0^{\pi/3} \sin^n\theta\, d\theta = \int_0^{\pi/3} \sin^n(\pi/3 - t)\, dt.$$

Now,

$$\begin{aligned}\sin(\pi/3 - t) &= \sin(\pi/3)\cos(t) - \cos(\pi/3)\sin(t)\\ &= \frac{\sqrt{3}}{2}\left(1 - \frac{t^2}{2!} + O(t^4)\right) - \frac{1}{2}\left(t - \frac{t^3}{3!} + O(t^5)\right)\\ &= \frac{\sqrt{3}}{2}\left(1 - \frac{t}{\sqrt{3}}\right) + O(t^2).\end{aligned}$$

Therefore

$$\sin^n(\pi/3 - t) \approx \left(\frac{\sqrt{3}}{2}\right)^n e^{-nt/\sqrt{3}}.$$

Finally, by Laplace's method we get

$$\int_0^{\pi/3} \sin^n(\pi/3 - t)\, dt \approx \left(\frac{\sqrt{3}}{2}\right)^n \int_0^{\infty} e^{-nt/\sqrt{3}}\, dt \sim \frac{\sqrt{3}}{n}\left(\frac{\sqrt{3}}{2}\right)^n. \qquad \square$$

Lemma 11.14 Suppose that n is large. Fix a unit vector $c \in S$. The probability that a randomly selected unit vector $v \in S$ lies in $B_n(c, 1)$ is

$$\Pr(\|v - c\| \le 1) \approx \frac{1}{\sqrt{n}}\left(\frac{3}{4}\right)^{n/2}.$$

Proof: By the law of cosines (106), $\|v-c\|^2 = 2(1-\cos\theta)$, where θ is the angle between v and c. Thus, $\|v - c\| \le 1$ if and only if $\cos\theta \ge 1/2$, i.e. $\theta \in [0, \pi/3]$. By Theorem 11.11,

$$\Pr(\theta \in [0, \pi/3]) = \frac{\int_0^{\pi/3} f_n(\theta)\, d\theta}{\int_0^{\pi} f_n(\theta)\, d\theta} \approx \frac{\int_0^{\pi/3} \sqrt{\frac{n}{2\pi}}\, \sin^{n-2}\theta\, d\theta}{1} = \sqrt{\frac{n}{2\pi}} \int_0^{\pi/3} \sin^{n-2}\theta\, d\theta.$$

Using Lemma 11.13, we have

$$\Pr(\theta \in [0, \pi/3]) \approx \sqrt{\frac{n}{2\pi}} \left(\frac{\sqrt{3}}{n-2}\right) \left(\frac{\sqrt{3}}{2}\right)^{n-2} \approx \frac{1}{\sqrt{n}} \left(\frac{3}{4}\right)^{n/2}. \qquad \square$$

11.5.2 The Nguyen-Vidick sieving algorithm

A simplified version of the sieving method is given in Algorithm 11.15. The algorithm begins by applying LLL to the input basis B to obtain an LLL-reduced basis. This ensures that the basis vectors are relatively short, which in turn allows us to sample[23] lattice vectors of length less than $2^n \lambda_1(L)$. The initial radius R_0 is then computed as the maximum length of the vectors in S_0.

The main loop iteratively applies a sieving step to the current set S_k and produces a new set S_{k+1} of lattice vectors of length approximately γR_k. Each sieving step maintains a set C of "centers", which are vectors in S_k possessing the property that the distance between any two centers is greater than γR_k. The sieving step processes every vector $v \in S_k$. If v has length at most γR_k, it is inserted in S_{k+1}. Otherwise, either $v - c$ is inserted in S_{k+1} if $v \in B_n(c, \gamma R_k)$ for some $c \in C$, or else v is identified as a center. After processing all $v \in S_k$, the radius R_{k+1} is computed as the maximum length of the vectors in S_{k+1}.

The sieving continues until the set S_k is empty, meaning no further reductions are possible. As we will discuss in the analysis below, this occurs once $R_k \approx \lambda_1(L)$. The algorithm then returns the shortest vector in the final nonempty set, which is expected to have length close to $\lambda_1(L)$.

11.5.3 Analysis

We now give a heuristic analysis of the sieving algorithm. The analysis rests on the following key heuristic assumption.

Heuristic assumption 1 For each $k \geq 0$, the unit vectors $\{v/\|v\| : v \in S_k\}$ are independently and uniformly distributed on the unit sphere S.

While this assumption is not provably true, it is supported by experiment and is standard in the lattice sieving literature. It allows us to treat the normalized vectors in each set S_k as random points on the unit sphere.

We first justify that the size N_0 of the initial set S_0 is sufficient to sustain the sieve, meaning that $N_k \approx N_0$, where $N_k = |S_k|$. We'll assume that each vector in S_k has size approximately $R_k \approx \gamma^k R_0$. Since the sieve is scale-invariant, we may normalize all

[23] The Nguyen-Vidick algorithm uses Klein's algorithm to select the initial lattice vectors according to a discrete Gaussian distribution. The vectors selected have length $2^{O(n)}\lambda_1(L)$, which we have simplified to $2^n \lambda_1(L)$.

Algorithm 11.15: Nguyen-Vidick sieving method for solving SVP (simplified)

Input: A basis of a lattice $L \subseteq \mathbb{Z}^n$.
Output: A vector in $L = L(B)$ of length $\approx \lambda_1(L)$.

$[b_1, \ldots, b_n] \leftarrow \text{LLL}(B)$. // compute an LLL-reduced basis of L
Randomly select small integer linear combinations of $b_1, \ldots, b_n$ to obtain a set S_0 of size $N_0 \approx n^3(4/3)^{n/2}$. // Klein's algorithm is used in practice
Compute $R_0 = \max\{\|v\| : v \in S_0\}$.
$\gamma \leftarrow 1 - \frac{1}{n}$. // γ is the shrinking factor
$k \leftarrow 0$.
while $S_k \neq \emptyset$ **do**
    $C \leftarrow \emptyset$. // C is the set of centers
    $S_{k+1} \leftarrow \emptyset$. // S_{k+1} is the new set of lattice vectors
    for each vector $v \in S_k$ **do**
        if $\|v\| \leq \gamma R_k$ **then**
            $S_{k+1} \leftarrow S_{k+1} \cup \{v\}$.
        else
            if there is a $c \in C$ such that $\|v - c\| \leq \gamma R_k$ **then**
                $S_{k+1} \leftarrow S_{k+1} \cup \{v - c\}$.
            else
                $C \leftarrow C \cup \{v\}$.
    Compute $R_{k+1} = \max\{\|v\| : v \in S_{k+1}\}$.
    $k \leftarrow k + 1$.
Select $v \in S_{k-1}$ of minimum length.
return (v).

vectors in S_k to lie on the unit sphere S without loss of generality. Fix a vector $c \in S_k$. Since $\gamma \approx 1$, we can use Lemma 11.14 to conclude that the probability a vector $v \in S$ satisfies $\|v - c\| \leq \gamma$ is approximately $\frac{1}{\sqrt{n}}(3/4)^{n/2}$. In other words, each vector $c \in S_k$ "covers" a $\frac{1}{\sqrt{n}}(3/4)^{n/2}$ fraction of the sphere, in the sense that any unit vector v falling within this fraction will yield a shorter vector $v - c$. For the sieve to be sustainable, every vector on the sphere must be covered by at least one vector in S_k. Assuming the covered regions have negligible overlap, a set of size N covers a $\frac{N}{\sqrt{n}}(3/4)^{n/2}$ fraction of the sphere. Full coverage therefore requires

$$N \cdot \frac{1}{\sqrt{n}} \left(\frac{3}{4}\right)^{n/2} \geq 1,$$

which gives

$$N \geq \sqrt{n} \left(\frac{4}{3}\right)^{n/2}.$$

The factor n^3 in the initial set size $N_0 = n^3(4/3)^{n/2}$ is a polynomial correction that ensures the coverage condition holds with overwhelming probability.

The set S_{k+1} will be smaller than S_k for two reasons: (i) vectors v in the center $C \subseteq S_k$ are discarded; and (ii) collisions may arise wherein an eligible vector $v - c$ is already in S_{k+1} and thus doesn't contribute to expanding S_{k+1}. One expects that the number of centers selected during a sieving step to be on the order of $\sqrt{n}(4/3)^{n/2}$, corresponding to the size of a random covering. Thus, if the number of collisions is negligible, then one expects that

$$N_k = |S_k| \approx |S_0| - k\sqrt{n}\left(\frac{4}{3}\right)^{n/2} \approx (n^3 - k\sqrt{n})\left(\frac{4}{3}\right)^{n/2} \approx n^3\left(\frac{4}{3}\right)^{n/2} \approx |S_0| \text{ for } k \leq n^2.$$

By the birthday paradox, collisions are only expected when $|S_k|$ exceeds the square root of the number of available lattice vectors, namely the lattice vectors in $B_n(R_{k+1})$. By Theorem 11.10, this occurs when

$$N_k \geq \left(1 + \frac{2R_{k+1}}{\lambda_1(L)}\right)^{n/2} \approx \left(\frac{2R_{k+1}}{\lambda_1(L)}\right)^{n/2},$$

which (roughly) yields the condition $R_{k+1} \approx \lambda_1(L)$.

Suppose that a (scaled) set S_k contains a covering set of the unit sphere S. Then every other vector v in S_k will be within distance γ of a vector c in the cover. This vector $v - c$ will be added to S_{k+1} except if it collides with a vector already in S_{k+1}. The algorithm must terminate by the time $R_k = \lambda_1(L)$. Since $R_k \approx \gamma^k R_0 \approx \gamma^k 2^n \lambda_1(L)$, the maximum number of sieving steps is determined by solving

$$\gamma^k 2^n \lambda_1(L) \approx \lambda_1(L),$$

yielding

$$k \approx -\frac{n \ln 2}{\ln \gamma} = -\frac{n \ln 2}{\ln(1 - \frac{1}{n})} \approx \frac{-n \ln 2}{-1/n} \approx 0.69n^2.$$

Thus, the algorithm is expected to terminate after at most n^2 sieving steps.

Each sieving step processes all N_k vectors in the current set S_k. Using a naive search through the centers C, each vector may need to be compared against all the centers, resulting in a cost of $O(N_k^2)$ per sieving step. Since there are at most n^2 sieving steps, the total running time is at most

$$n^2 \cdot N_0^2 = n^8 \left(\frac{4}{3}\right)^n \approx 2^{0.415n+o(n)},$$

while the space complexity is

$$N_0 = n^3 \left(\frac{4}{3}\right)^{n/2} \approx 2^{0.2075n+o(n)}.$$

In summary, the sieving algorithm solves SVP in heuristic time $2^{0.415n+o(n)}$, which is asymptotically faster than the best enumeration algorithms. A major drawback is the exponential space requirements.

11.6 Estimating the security level of Kyber

Recall the parameters in Kyber key generation (Algorithm 6.10): $A \in_R R_q^{k\times k}$, $s \in_{CBD} S_{\eta_1}^k$, $e \in_{CBD} S_{\eta_1}^k$, and $t = As + e \pmod q$, where $\in_{CBD}$ denotes sampling from a centered binomial distribution. We are interested in assessing the difficulty of recovering the Kyber decryption key s from an encryption key (A, t); this is an instance of the short-secret MLWE problem (Definition 5.17).

From MLWE to LWE. Since no attacks on MLWE are known to be substantially faster than corresponding attacks on LWE, the complexity is analyzed by solving the standard short-secret LWE problem (Definition 4.7) with matching dimensions. Accordingly, let $A \in_R \mathbb{Z}_q^{m\times n}$, $s \in_{CBD} [-\eta_1, \eta_1]^n$, $e \in_{CBD} [-\eta_1, \eta_1]^m$, and $t = As + e \pmod q \in \mathbb{Z}_q^m$. In Kyber key generation, these parameters are $m = 256k$ and $n = 256k$.

11.6.1 Primal attack

The primal attack constructs a lattice where the ss-LWE solution (s, e) corresponds to a nonzero vector x of unusually small length. This transforms the problem to an instance of the Unique Shortest Vector Problem (uSVP). Applying the BKZ lattice reduction algorithm produces a basis containing sufficiently short vectors, from which the target vector x, and hence the ss-LWE solution (s, e), can be extracted.

Definition 11.16 The lattice associated with the ss-LWE instance (A, t) is defined as:

$$L = \{x \in \mathbb{Z}^{m+n+1} : [I_m \mid A \mid -t]x = 0 \pmod q\}. \tag{107}$$

Theorem 11.17 L is a full-rank lattice of dimension $d = m + n + 1$ and volume q^m.

Proof: L is a discrete additive subgroup, and therefore a lattice. We will show that

$$B = \begin{bmatrix} qI_m & -A & t \\ 0 & I_n & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{108}$$

is a basis matrix of L, from which it follows that L has full rank and volume q^m.

First note that the columns of B are in L, since

$$[I_m \mid A \mid -t]\, B = [qI_m \mid -A + A \mid t - t] = [qI_m \mid 0 \mid 0] = 0 \pmod{q}.$$

Hence $L(B) \subseteq L$. Next, let $x \in L$ and write $x = (x_1, x_2, x_3)$ where $x_1 \in \mathbb{Z}^m$, $x_2 \in \mathbb{Z}^n$, and $x_3 \in \mathbb{Z}$. By the definition of L, we have

$$x_1 + Ax_2 - tx_3 = 0 \pmod{q},$$

whence

$$x_1 = -Ax_2 + tx_3 + qu$$

for some $u \in \mathbb{Z}^m$. Thus,

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} qI_m & -A & t \\ 0 & I_n & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ x_2 \\ x_3 \end{bmatrix}$$

which shows that B is a basis matrix of L. □

The vector $x = (e, s, 1)$ belongs to this lattice L. By Theorem 6.6, each component e_i of e and each component s_j of s satisfies $E(e_i^2) = E(s_j^2) = \frac{\eta_1}{2}$. Hence, the expected length of x is

$$E(\|x\|) = E(\sqrt{\|x\|^2}) \approx \sqrt{E(\|x\|^2)} = \sqrt{(m+n)\frac{\eta_1}{2} + 1} \approx \sqrt{(m+n)\frac{\eta_1}{2}}.$$

Here the first approximation holds since $\|x\|$ concentrates tightly around $\sqrt{E(\|x\|^2)}$ for large $m+n$. On the other hand, the Gaussian Heuristic (8) predicts the length of a shortest nonzero vector v in L to be:

$$\|v\| \approx \sqrt{\frac{m+n+1}{2\pi e}} q^{\frac{m}{m+n+1}} \approx \sqrt{\frac{m+n}{2\pi e}} q^{\frac{1}{2}},$$

where the latter approximation uses $m \approx n$. Consequently, if

$$\eta_1 \ll q/(\pi e), \tag{109}$$

then x is expected to be an unusually short vector, and thus may reasonably be expected to be a shortest nonzero vector of L. For the ML-KEM-768 parameter set where $q = 3329$ and $\eta_1 = 2$, this condition is satisfied (since $2 \ll 389.8$).

11.6.2 Geometric Series Assumption (GSA)

The GSA is used to estimate the performance of the BKZ lattice basis reduction algorithm. Let $L \subseteq \mathbb{Z}^n$ be a random full-rank lattice. The GSA asserts that the Gram-Schmidt norms $\|b_i^*\|$ of a β-BKZ-reduced basis $B = [b_1, \ldots, b_n]$ of L approximate a decreasing geometric series, i.e.,

$$\frac{\|b_{i+1}^*\|}{\|b_i^*\|} \approx r \quad \text{for } 1 \leq i \leq n-1 \tag{110}$$

for some $0 < r < 1$. Equivalently,

$$\|b_i^*\| \approx r^{i-1}\|b_1\| \quad \text{for } 1 \leq i \leq n. \tag{111}$$

Now by Theorem 10.8,

$$\operatorname{vol}(L) = \prod_{i=1}^{n} \|b_i^*\| \approx \prod_{i=1}^{n} r^{i-1}\|b_1\| = \|b_1\|^n r^{n(n-1)/2}.$$

Thus

$$\operatorname{vol}(L)^{1/n} \approx \|b_1\| r^{(n-1)/2}.$$

Recalling the definition (100) of the Hermite factor δ of lattice basis B, we obtain

$$r \approx \delta^{-2/(n-1)}. \tag{112}$$

Define the *root Hermite factor* of B by

$$\delta_0 = \delta_0(B) = \delta^{1/n}. \tag{113}$$

Then (112) becomes

$$r \approx \delta_0^{-2n/(n-1)} \approx \delta_0^{-2},$$

where the latter approximation holds for large n. Substituting into (111) yields the following estimate for $\|b_i^*\|$:

$$\|b_i^*\| \approx \delta_0^{-2(i-1)}\|b_1\| = \delta_0^{-2(i-1)}\delta_0^n \operatorname{vol}(L)^{1/n} = \delta_0^{n-2i+2} \operatorname{vol}(L)^{1/n} \quad \text{for } 1 \leq i \leq n. \tag{114}$$

Assuming both the Gaussian heuristic (8) and the GSA, it has been shown that for β-BKZ-reduced bases B of random full-rank lattices,

$$\lim_{n\to\infty} \delta_0(B) = \left(\frac{\beta}{2\pi e}(\pi\beta)^{\frac{1}{\beta}}\right)^{\frac{1}{2(\beta-1)}}. \tag{115}$$

Experimental evidence suggests that for fixed $\beta \geq 50$, this asymptotic estimate quickly converges as n gets larger. Consequently, we have the following approximation for the lengths of the Gram-Schmidt basis vectors of B:

$$\|b_i^*\| \approx \delta_0^{n-2i+2} \operatorname{vol}(L)^{1/n} \text{ for } 1 \leq i \leq n, \tag{116}$$

where

$$\delta_0 = \left(\frac{\beta}{2\pi e}(\pi\beta)^{\frac{1}{\beta}}\right)^{\frac{1}{2(\beta-1)}} \approx \left(\frac{\beta}{2\pi e}\right)^{\frac{1}{2\beta}}. \tag{117}$$

As one would expect, for fixed dimension n, larger values of β produce a smaller root Hermite factor, and hence a more reduced lattice basis, but at the cost of increased running time. Table 11.2 shows the root Hermite factors for some values of β.

β	50	100	200	300	400	500	600	700
δ_0	1.0121	1.0093	1.0063	1.0048	1.0040	1.0034	1.0030	1.0027

Table 11.2: Root Hermite factors for β-BKZ-reduced bases.

11.6.3 Primal attack analysis

Let $d = m + n + 1$ denote the dimension of the lattice L in (107). Recall that the lattice vector we seek is $x = (e, s, 1)$, whose expected length is

$$\ell = \sqrt{(m+n)\frac{\eta_1}{2} + 1}.$$

The attack begins by selecting the smallest blocksize $\beta \geq 50$ for which

$$\sqrt{\frac{\beta}{d}}\,\ell \leq \delta_0^{2\beta-d} q^{m/d}. \tag{118}$$

The BKZ algorithm with blocksize β is then applied to the lattice basis (108). After sufficiently many tours, one expects to obtain a basis $[b_1, \ldots, b_d]$ satisfying the GSA, so that the Gram-Schmidt lengths $\|b_i^*\|$ are approximated by (116).

Now consider the local projected block

$$B[d-\beta+1,\ d] = [\Pi_{d-\beta}(b_{d-\beta+1}),\ \Pi_{d-\beta}(b_{d-\beta+2}),\ \ldots,\ \Pi_{d-\beta}(b_d)],$$

whose vectors span the β-dimensional projected lattice $L' = L[d-\beta+1, d]$. Recall that

$$b^*_{d-\beta+1} = \Pi_{d-\beta}(b_{d-\beta+1}).$$

Hence, by (116), we have

$$\|\Pi_{d-\beta}(b_{d-\beta+1})\| = \|b^*_{d-\beta+1}\| \approx \delta_0^{2\beta-d} q^{m/d}.$$

Let $x' = \Pi_{d-\beta}(x)$ denote the projection of the target vector. Assuming that x behaves independently of the reduced basis $[b_1, \ldots, b_d]$, one expects that

$$\|x'\| \approx \sqrt{\beta\frac{\eta_1}{2}} \approx \sqrt{\frac{\beta}{d}}\,\ell.$$

Since condition (118) holds, it is expected that x' is a shortest nonzero vector in L'. Consequently, an SVP oracle applied to L' should recover x' (or $-x'$). Once x' is known, x can then be reconstructed (details omitted).

As discussed in §11.3, no satisfactory upper bound is known for the number of tours performed by the BKZ algorithm. To sidestep this issue, the complexity of the primal attack is typically estimated by the running time of the best algorithm known for solving SVP in β-dimensional lattices, where β is selected as described above. Currently, the fastest known classical approach is *sieving*, with a running time of approximately $2^{0.292\beta}$. This estimate is called the *classical core-SVP hardness* of Kyber. Similarly, the *quantum core-SVP hardness* is estimated as $2^{0.256\beta}$, corresponding to the fastest known quantum algorithm for solving SVP in β-dimensional lattices. These security estimates should be regarded as conservative lower bounds on the actual security of Kyber.

The smallest feasible BKZ blocksize β was obtained by determining, for each $m' \leq m$, the smallest $\beta \geq 50$ satisfying (118), where m' denotes the number of LWE samples. The minimum such β is then used to estimate the core-SVP hardness of Kyber. The resulting estimates for the standardized Kyber parameter sets are presented in Table 11.3. In the table, $d' = m' + n + 1$.

	ML-KEM-512	ML-KEM-768	ML-KEM-1024
(k, η_1)	(2,3)	(3,2)	(4,2)
lattice dimension d'	1025	1483	1950
BKZ blocksize β	406	623	873
classical core-SVP hardness	119	182	255
quantum core-SVP hardness	104	159	223

Table 11.3: Core-SVP hardness for the standardized Kyber parameter sets.

Remark 11.18 (*state-of-the-art security analysis*) We presented a simplified security analysis of Kyber key generation. A comprehensive security analysis of Kyber-KEM would also include: (i) the use of a simulator to better predict the lengths of the Gram-Schmidt vectors after each BKZ tour; (ii) an analysis of the MLWE instance arising from the encryption operation (21), in which e_1 and e_2 may be drawn from distributions different from those of r; (iii) an analysis of dual attacks on LWE (see §4.4); and (iv) an analysis of recent improvements to hybrid dual attacks on MLWE.

11.7 Estimating the security level of Falcon

This section derives concrete security estimates for the Falcon-512 and Falcon-1024 parameter sets (Table 8.2) against key recovery and signature forgery attacks.

11.7.1 Key recovery

Given a public key $h \in R_q$, the Falcon key recovery problem asks to find polynomials $f, g \in R$ such that $g - fh = 0$ in R_q subject to the size constraint $\|(f, g)\| \leq \tau$, where $\tau = 1.17\sqrt{q}$. The corresponding target vector $(g, -f)$ belongs to the NTRU lattice

$$L_A^\perp = \{(z_1, z_2) \in R \times R \ : \ z_1 + z_2 h = 0 \text{ in } R_q\},$$

which has a volume of q^n. According to the Gaussian Heuristic (8), the expected length of a shortest nonzero vector $v \in L_A^\perp$ is

$$\|v\| \approx \sqrt{\frac{2n}{2\pi e}} q^{n/2n} = \sqrt{\frac{nq}{\pi e}}.$$

For dimensions $n = 512$ and $n = 1024$, this heuristic yields $\|v\| \approx 7.74\sqrt{q}$ and $10.95\sqrt{q}$, respectively. Because these expected lengths are significantly larger than the bound τ, the private key vector $(g, -f)$ is exceptionally short. Consequently, key recovery can be modelled as an instance of the Unique Shortest Vector Problem and analyzed via the primal attack methodology described in §11.6.

To execute the attack, the BKZ lattice basis reduction algorithm is applied to find a β-BKZ-reduced basis of $L_A^\perp$. Adhering to the conditions in (117) and (118), we determine the minimal blocksize $\beta \geq 50$ satisfying

$$\sqrt{\frac{\beta}{2n}} \tau \leq \delta_0^{2\beta - 2n} q^{n/2n},$$

where the root Hermite factor δ_0 is parameterized by the blocksize β as

$$\delta_0 = \left(\frac{\beta}{2\pi e} (\pi\beta)^{1/\beta} \right)^{1/2(\beta-1)}. \tag{119}$$

The resulting estimates for the standardized Falcon parameter sets are presented in Table 11.4.

11.7.2 Signature forgery

Given a public key $h \in R_q$ and an arbitrary message $M \in \{0,1\}^*$, the signature forgery problem asks an adversary to produce a valid signature (r, s) for M. The adversary can proceed by first sampling a random salt $r \in_R \{0,1\}^{320}$ and computing the hash target $u = H(h, r, M) \in R_q$. The adversary must then find a pair of polynomials $s_1, s_2 \in R$ that satisfy $s_1 + s_2 h = u$ in R_q, subject to the size constraint[24] $\|(s_1, s_2)\| \leq \overline{\beta}$, where $\overline{\beta} = 1.1 \times \sigma\sqrt{2n}$ and σ is defined in (56).

[24] We denote Falcon's acceptance bound (57) by $\overline{\beta}$ (rather than the standard β) to avoid a notation conflict with the BKZ blocksize β.

	Falcon-512	Falcon-1024
Key recovery		
BKZ blocksize β	481	973
classical core-SVP hardness	140	284
quantum core-SVP hardness	123	249
Signature forgery		
BKZ blocksize β	416	956
classical core-SVP hardness	121	279
quantum core-SVP hardness	106	245

Table 11.4: Core-SVP hardness for the Falcon parameter sets.

Kannan's embedding technique (Exercise 12.6) can be deployed to find a vector in the NTRU lattice $L_A^\perp$ close to $(u, 0)$. To do so, consider the related lattice L generated by the columns of the basis matrix

$$B' = \begin{bmatrix} qI_n & -\mathcal{A}(h) & u \\ 0 & I_n & 0 \\ 0 & 0 & K \end{bmatrix} \in \mathbb{Z}^{(2n+1)\times(2n+1)}$$

where the scaling factor is $K \approx \sqrt{q}$. According to the Gaussian Heuristic (8), the expected length of a shortest nonzero vector $v \in L$ is

$$\|v\| \approx \sqrt{\frac{2n+1}{2\pi e}}\sqrt{q} \approx \sqrt{\frac{nq}{\pi e}}. \tag{120}$$

Let $s = (s_1, s_2)$ be any valid signature for M with respect to the chosen salt r, and define $w = (s_1, s_2, K) \in \mathbb{Z}^{2n+1}$. Note that s is not expected to be unique. Because $u = s_1 + s_2 h$ in R_q, there exists a polynomial $a \in R$ such that $s_1 = qa - s_2 h + u$. It follows that $w \in L$ because

$$B' \cdot \begin{bmatrix} a \\ s_2 \\ 1 \end{bmatrix} = \begin{bmatrix} qI_n & -\mathcal{A}(h) & u \\ 0 & I_n & 0 \\ 0 & 0 & K \end{bmatrix} \begin{bmatrix} a \\ s_2 \\ 1 \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \\ K \end{bmatrix} = w.$$

For modulus $q = 12289$ and dimension $n = 512$, the estimate (120) yields an expected shortest vector length of $\|v\| \approx 858$. In contrast, a valid vector w satisfies $\|w\| \approx \|(s_1, s_2)\| \approx \overline{\beta} \approx 5833$ (see Table 8.2). Because $\|w\|$ exceeds the Gaussian Heuristic estimate for the shortest nonzero vector length by a large factor, w is unlikely to be a unique shortest vector. Thus, we cannot model the search for w as an instance of uSVP.

Instead, the forging strategy is to find a β-BKZ-reduced basis of L, hoping that a valid signature w appears as the first basis vector b_1. Using the Geometric Series Assumption estimate (114), the length of the first basis vector is estimated as

$$\|b_1\| = \|b_1^*\| \approx \delta_0^{2n+1} q^{(n+0.5)/(2n+1)} = \delta_0^{2n+1}\sqrt{q}$$

where the root Hermite factor δ_0 is parameterized by the blocksize β according to (119). We can therefore expect the reduction to successfully recover a valid w when $\|b_1\| \le \overline{\beta}$ or

$$\delta_0^{2n+1}\sqrt{q} \le \overline{\beta}.$$

The resulting estimates for the standardized Falcon parameter sets are presented in Table 11.4.

Remark 11.19 (*state-of-the-art security analysis*) We have presented a simplified security analysis of Falcon. The more refined analysis provided in the Falcon specification yields slightly lower security levels; for example, the classical core-SVP hardness against key recovery and signature forgery are 133 and 120 bits for Falcon-512, and 273 and 277 bits for Falcon-1024, respectively.

11.8 Notes and further references

Yasuda [217] surveys algorithms for solving SVP, including LLL, BKZ, sieving, and enumeration. A more recent survey that covers the latest advances in LLL, BKZ, enumeration, and sieving is provided by Shen and Li [203].

The notion of a β-BKZ-reduced basis was introduced by Schnorr [198]. The upper bound in Theorem 11.5 on the length of the first vector in a β-BKZ-reduced basis $B = [b_1, \ldots, b_n]$ was proved in full generality by Schnorr [199]; the proof given in §11.1 makes some simplifying assumptions on n and β. Schnorr also showed that $\|b_i\| \le \sqrt{(i+3)/4} \cdot \gamma_\beta^{(n-1)/(\beta-1)} \lambda_i(L)$ for $2 \le i \le n$. The BKZ algorithm is due to Schnorr and Euchner [201]. The modified-LLL algorithm, which produces an LLL-reduced basis when the input is a linearly dependent spanning set rather than a true basis, was first described by Pohst [188]; see also Section 2.6.4 of Cohen [58].

For a version of BKZ that differs slightly from Algorithm 11.6, Gama and Nguyen [94] stated a bound of $(n\beta)^n$ on the number of calls to the SVP oracle; a proof of a similar $O(\beta^n)$ bound is given in Appendix A of Hanrot, Pujol and Stehlé [111]. In 2011, Chen and Nguyen [54] introduced BKZ 2.0, a refined version of BKZ designed with early abort. See also the article by Albrecht, Player and Scott [10], which performs a detailed analysis of BKZ 2.0 and gives concrete estimates for various families of LWE instances. The first rigorous analysis of the original (unmodified) BKZ algorithm was given by Li and Nguyen [146]. Running-time estimates for BKZ were used by Bos et al. [42] to assess the security of Kyber, and by Ducas et al. [75] to assess the security of Dilithium.

The basic enumeration algorithm is due to Pohst [187] and Fincke and Pohst [86]. Kannan [126] showed that by first computing a strongly reduced basis, the running time can be lowered to $n^{O(n)}$. Hanrot and Stehlé [112] refined the analysis of Kannan's algorithm to obtain the running time bound $O(2^{0.184n \log_2 n + o(n \log n)})$. Although enumeration algorithms are asymptotically slower than sieving, they require only modest amounts of memory. Schnorr and Euchner [201] introduced a method for pruning the search tree, and later an improved pruning method was proposed by Gama, Nguyen and Regev [95].

Sieving methods are the fastest algorithms known for solving SVP exactly. Sieving was originally proposed by Ajtai, Kumar and Sivakumar [7], who also showed that approx-SVP with approximation factor $\gamma > 2^{(n \log\log n)/\log n}$ is solvable in polynomial time. Theorem 11.10 and the sieving algorithm presented in §11.5 are due to Nguyen and Vidick [182]. In Nguyen and Vidick's sieving algorithm, the vectors in the original list S_0 are selected according to a Gaussian distribution using Klein's algorithm [131]. Nguyen and Vidick give a more careful analysis than the one given in §11.5, and also provide experimental evidence for the validity of Heuristic Assumption 1. Theorem 11.11 appears in Cai, Fan and Jiang [46].

Micciancio and Voulgaris [164] proposed GaussSieve, which performs well in practice. The fastest classical sieving algorithm, due to Becker, Ducas, Gama and Laarhoven [26], uses *locality sensitive hashing* to speed up the search for nearby lattice vectors, and runs in time $2^{0.292n+o(n)}$. The fastest quantum variant, described by Bonnetain, Chailloux, Schrottenloher and Shen [39], runs in time $2^{0.256n+o(n)}$. While these running times are heuristic, the fastest known classical SVP solvers with rigorously proven running times are the $2^{0.73134n+o(n)}$-time randomized algorithm of Gao, Feng and Hu [97] which returns a shortest nonzero lattice vector with probability $1-2^{-\Omega(n \log n)}$, the $2^{0.63269n+o(n)}$-time algorithm of Pouly and Shen [192] which succeeds for most random lattices, and the $2^{0.6039n+o(n)}$-time randomized algorithm of Hhan [115]. Aggarwal and Stephens-Davidowitz [1] described a conceptually simple sieving algorithm that provably runs in time $2^{n+o(n)}$ and space $2^{n+o(n)}$.

A notable drawback of sieving-based methods is that they require both exponential time and exponential space. Zhao, Ding and Yang [221] report on their solution in 2024 of a dimension-183 SVP challenge using sieving; their computation required 30 days on a 112-core server with 0.87 Terabytes of RAM. More recently, Zhao and Ding [220] report on their solution of a dimension-200 SVP challenge using sieving; their computation required 30 days on a server with eight RTX 4090 GPUs, 1 Terabyte of RAM, and 30 Terabytes of disk space.

The strategy of using the primal attack together with BKZ reduction to evaluate concrete security was first proposed by Alkim, Ducas, Pöppelmann and Schwabe [13] in their work introducing the Ring-LWE-based key agreement protocol NewHope. This paper also introduced the notion of core-SVP hardness. The primal attack strategy was subsequently adapted to Kyber and developed further by Bos et al. [42] in the CRYSTALS-Kyber submission to the NIST post-quantum standardization process. A method for recovering the target vector x from the projected vector x' was described and analyzed by Albrecht, Göpfert, Virdia and Wunderer [9]. They also gave experimental evidence supporting the accuracy of the security estimates, showing that the practical behaviour of the primal attack closely matches the theoretical predictions.

The Geometric Series Assumption (110) was introduced by Schnorr [200]. The asymptotic limit (115) for the root Hermite factor of β-BKZ-reduced bases of random lattices is due to Chen [53]. An efficient algorithm for simulating the lengths of the Gram-Schmidt basis vectors after each tour in the BKZ algorithm was presented by Chen and Nguyen [54].

Module-BKZ is a generalization of BKZ to module lattices, such as those arising in the study of MLWE. It was introduced by Lee, Pellet-Mary, Stehlé and Wallet [141], and by Mukherjee and Stephens-Davidowitz [165]. The average-case behaviour of module-BKZ was later analyzed by Ducas, Engelberts and de Perthuis [74] and by de Perthuis and Trenkić [71]. These works did not find any improvement in primal attacks on MLWE compared to the corresponding attacks on LWE. More recently, Hou, Jiang and Ogilvie [119] presented an enhanced primal hybrid attack, as well as an improvement to the dual hybrid attack of Carrier, Meyer-Hilfiger, Shen and Tillich [47]. They showed that these attacks perform slightly better against MLWE than against unstructured LWE. Based on this analysis, they estimate that the core-SVP hardness of standardized Kyber is approximately 1 bit lower than previously believed.

11.9 Exercises

Exercise 11.1 Prove that a $(\beta + 1)$-BKZ-reduced lattice basis is also β-BKZ-reduced.

Exercise 11.2 Prove that a β-BKZ-reduced lattice basis is also LLL-reduced.

Exercise 11.3 Prove that a 1-LLL-reduced lattice basis is 2-BKZ-reduced.

Exercise 11.4 Let L be the lattice with basis $b_1 = (2, 0)$, $b_2 = (1, 2)$.
(a) Verify that $[b_1, b_2]$ is LLL-reduced, and compute b_1^*, b_2^*, μ_{21} and the bound D in (101).
(b) Run the basic enumeration algorithm by hand. Draw the search tree, and use it to determine $\lambda_1(L)$ and all shortest nonzero vectors of L.

Exercise 11.5 Implement the naive enumeration algorithm and use it to find all shortest nonzero vectors in the lattices defined in Exercise 10.10.

Exercise 11.6 Implement the basic enumeration algorithm and use it to find all shortest nonzero vectors in the lattices defined in Exercise 10.10. Compare the number of nodes visited with that of the naive enumeration algorithm.

Exercise 11.7 Prove that $|\mathbb{Z}^n \cap B_n(1)| = 2n + 1$ and $|\mathbb{Z}^n \cap B_n(\sqrt{2})| = 2n^2 + 1$.

Exercise 11.8 Assume the GSA estimate (114). Show that $\|b_1\|/\|b_n^*\| \approx \delta_0^{2(n-1)}$. Using Table 11.2, estimate this ratio for $n = 500$ and $\beta = 100$.

Exercise 11.9 Using (117), verify the entries of Table 11.2.

Exercise 11.10 The core-SVP hardness estimates for Dilithium are obtained by analyzing the performance of BKZ on the MLWE instance arising in key generation and the MSIS instance arising in signature forgery (see equation 37). Study the methods used to assess core-SVP hardness in Sections C.2 and C.3 of the Dilithium specification [75] and then verify the core-SVP hardness values reported in Table 1 of [75].

Exercise 11.11 This exercise estimates the classical security of FrodoKEM against the primal lattice attack of §11.6.
(a) Let $e \in_\chi \{-c, \ldots, c\}$, where χ is the error distribution specified in Table 9.2. Show that the variance $E(e^2)$ for the three parameter sets is approximately 7.92175 for FrodoKEM-640, 5.37219 for FrodoKEM-976, and 2.04245 for FrodoKEM-1344.
(b) Let (A, B) be a FrodoKEM public key, and let b denote the first column of B, forming an LWE instance (A, b). Show that the optimal BKZ blocksize β required to solve this LWE instance via the primal attack is $\beta = 487, 708, 934$ for the three FrodoKEM parameter sets, respectively.
(c) Using the classical core-SVP model, deduce that the security of the three FrodoKEM parameter sets against the primal attack is approximately 142.2, 206.7, 272.7 bits, respectively.

12 LLL applications

Contents

12.1 Primal attack on short-secret LWE 203
12.2 Dual attack on SIS_2 205
12.3 Finding integer relations 206
12.4 Simultaneous Diophantine approximation 209
12.5 Side-channel attack on ECDSA 211
12.6 Finding small roots of a polynomial modulo N 215
12.7 Notes and further references 222
12.8 Exercises 223

The general methodology for applying the LLL algorithm proceeds as follows. One first constructs a lattice L in which the (unknown) solution to the problem at hand is encoded by a nonzero lattice vector v that is significantly shorter than the Gaussian heuristic (8) predicts for a shortest nonzero lattice vector. The LLL algorithm is then applied to compute a reduced basis B of L. Although LLL does not guarantee recovery of a shortest nonzero lattice vector, in practice its performance is often substantially better than suggested by the worst-case bounds, and in many practical instances the target vector v appears among the vectors of the reduced basis. One then inspects the basis B to determine whether v has been recovered; if so, the original problem can be solved efficiently.

This section illustrates this methodology through six applications of the LLL algorithm. The first application is the primal attack on short-secret LWE that was presented in §11.6. The second application is the dual attack on SIS_2 discussed previously in §3.3. The third application, discussed in §12.3, concerns the discovery of integer relations among real numbers. This problem is mainly of theoretical interest and may be skipped by readers whose primary focus is cryptographic applications. The fourth application, simultaneous Diophantine approximation, presented in §12.4, arises naturally in numerous cryptanalytic settings. The fifth application, described in §12.5, is a concrete side-channel attack on the ECDSA signature scheme. Lastly, §12.6 introduces a technique based on LLL for finding small roots of a univariate polynomial modulo a composite integer with unknown factorization; this technique has found numerous applications to problems related to RSA and integer factorization. Some other LLL applications are explored in the Exercises including solving approximate CVP (Exercises 12.5 and 12.6), solving the subset sum problem (Exercise 12.13), and writing a prime as a sum of four squares (Exercise 12.14).

12.1 Primal attack on short-secret LWE

Consider an ss-LWE(m, n, q, B) instance (A, t) (Definition 4.7), where $A \in_R \mathbb{Z}_q^{m\times n}$, $s \in_R [-B, B]^n$, $e \in_R [-B, B]^m$, and $t = As + e \pmod q$. The goal is to recover s and e.

Recall the primal attack on ss-LWE from §11.6. The vector $x = (e, s, 1) \in \mathbb{Z}^{m+n+1}$

belongs to the $(m+n+1)$-dimensional lattice L_A of volume q^m with basis matrix

$$\begin{bmatrix} qI_m & -A & t \\ 0 & I_n & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The expected value of the square of a component of either e or s is

$$\frac{1}{2B+1}\sum_{i=-B}^{B} i^2 = \frac{2B(B+1)(2B+1)}{6(2B+1)} = \frac{B(B+1)}{3} \approx \frac{B^2}{3}.$$

Hence, the expected length of the target vector $x = (e, s, 1)$ is

$$\|x\| \approx \sqrt{(m+n)B^2/3 + 1} \approx B\sqrt{(m+n)/3}.$$

On the other hand, by the Gaussian heuristic (8), the expected length of a shortest nonzero vector in L_A is approximately

$$\sqrt{\frac{m+n+1}{2\pi e}} q^{\frac{m}{m+n+1}}.$$

Therefore, if

$$B\sqrt{\frac{m+n}{3}} \ll \sqrt{\frac{m+n+1}{2\pi e}} q^{\frac{m}{m+n+1}} \tag{121}$$

then $\pm x$ are expected to be the shortest nonzero vectors in L_A, and so can likely be determined by finding a sufficiently reduced basis of L_A.

Example 12.1 (*primal attack on short-secret LWE*) Let $m = 5$, $n = 3$, $q = 7919$, and $B = 100$,[25] and consider the ss-LWE(m, n, q, B) instance (A, t) where

$$A = \begin{bmatrix} 306 & 7354 & 1517 \\ 5470 & 4491 & 670 \\ 6522 & 3601 & 5578 \\ 7427 & 917 & 607 \\ 3855 & 7914 & 896 \end{bmatrix} \quad \text{and} \quad t = \begin{bmatrix} 5891 \\ 4669 \\ 6299 \\ 4043 \\ 5973 \end{bmatrix}.$$

A basis matrix for the associated 9-dimensional lattice L_A is

$$\begin{bmatrix} 7919 & 0 & 0 & 0 & 0 & 7613 & 565 & 6402 & 5891 \\ 0 & 7919 & 0 & 0 & 0 & 2449 & 3428 & 7249 & 4669 \\ 0 & 0 & 7919 & 0 & 0 & 1397 & 4318 & 2341 & 6299 \\ 0 & 0 & 0 & 7919 & 0 & 492 & 7002 & 7312 & 4043 \\ 0 & 0 & 0 & 0 & 7919 & 4064 & 5 & 7023 & 5973 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

[25] In this example, the parameters m, n, q and B do not satisfy the Gaussian heuristic condition (121)

The LLL algorithm finds the following LLL-reduced basis of L_A:

$$\begin{aligned}
b_1 &= (-19, 33, -80, -20, -8, 53, 48, -62, 1)\\
b_2 &= (-14, 30, 6, 44, -11, -11, 89, 25, 39)\\
b_3 &= (-52, -94, 87, 15, 12, -29, 30, -17, 45)\\
b_4 &= (43, 6, -19, -114, -30, -28, 29, -65, 57)\\
b_5 &= (101, -34, 18, -31, -17, 21, 91, 107, -54)\\
b_6 &= (-46, 31, 91, -69, -126, -14, 31, 61, 12)\\
b_7 &= (-20, -6, -119, 16, 3, -111, -76, 47, 19)\\
b_8 &= (-83, 46, 13, -28, 80, -61, -20, 34, -135)\\
b_9 &= (-11, -57, 89, 4, -72, -76, 58, -35, -118).
\end{aligned}$$

The basis vector b_1 is a potential choice for the target vector x, since its last coordinate is 1. The ss-LWE solution (s, e) is then extracted from b_1:

$$s = \begin{bmatrix} 53 \\ 48 \\ -62 \end{bmatrix}, \quad e = \begin{bmatrix} -19 \\ 33 \\ -80 \\ -20 \\ -8 \end{bmatrix}.$$

12.2 Dual attack on SIS_2

Consider an $\mathrm{SIS}_2(n, m, q, B)$ instance (Definition 3.13) defined by a uniformly random matrix $A \in_R \mathbb{Z}_q^{n\times m}$. The goal is to find a vector $z \in \mathbb{Z}^m$ satisfying $0 < \|z\|_2 \le B$ and $Az = 0 \pmod q$.

Recall the dual attack on SIS_2 from §3.3.1. The associated SIS lattice is

$$L_A^\perp = \{z \in \mathbb{Z}^m : Az = 0 \pmod q\}.$$

Let $[I_n|\overline{A}]$ denote the reduced row echelon form of A over $\mathbb{Z}_q$. A basis matrix for $L_A^\perp$ is

$$C = \begin{bmatrix} qI_n & -\overline{A} \\ 0 & I_{m-n} \end{bmatrix} \in \mathbb{Z}^{m\times m}.$$

The dual attack proceeds by computing a reduced basis of $L_A^\perp$, and then identifying any of its vectors whose length does not exceed B.

Example 12.2 (*dual attack on* SIS_2) Let $n = 5$, $m = 9$, $q = 97$, and $B = 12$, and consider the $\mathrm{SIS}_2(n, m, q, B)$ instance

$$A = \begin{bmatrix} 77 & 96 & 10 & 86 & 58 & 36 & 80 & 22 & 44 \\ 39 & 60 & 39 & 43 & 12 & 55 & 2 & 24 & 71 \\ 45 & 29 & 21 & 48 & 7 & 33 & 57 & 65 & 16 \\ 93 & 96 & 71 & 44 & 70 & 58 & 25 & 29 & 73 \\ 57 & 40 & 78 & 88 & 25 & 44 & 67 & 88 & 5 \end{bmatrix}.$$

The reduced row echelon form of A over $\mathbb{Z}_q$ is

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 24 & 63 & 3 & 15 \\ 0 & 1 & 0 & 0 & 0 & 50 & 42 & 20 & 13 \\ 0 & 0 & 1 & 0 & 0 & 14 & 75 & 9 & 40 \\ 0 & 0 & 0 & 1 & 0 & 17 & 43 & 7 & 25 \\ 0 & 0 & 0 & 0 & 1 & 39 & 4 & 40 & 4 \end{bmatrix},$$

and so a basis matrix for the associated lattice $L_A^\perp$ is

$$C = \begin{bmatrix} 97 & 0 & 0 & 0 & 0 & 73 & 34 & 94 & 82 \\ 0 & 97 & 0 & 0 & 0 & 47 & 55 & 77 & 84 \\ 0 & 0 & 97 & 0 & 0 & 83 & 22 & 88 & 57 \\ 0 & 0 & 0 & 97 & 0 & 80 & 54 & 90 & 72 \\ 0 & 0 & 0 & 0 & 97 & 58 & 93 & 57 & 93 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The LLL algorithm finds the following LLL-reduced basis of $L_A^\perp$:

$$\begin{aligned} b_1 &= (-4, 1, -2, 9, -2, 1, 2, 1, 3) & b_2 &= (1, 1, 0, -5, -4, 4, 9, 5, 0) \\ b_3 &= (3, 0, -8, 2, -5, -1, -9, 2, 0) & b_4 &= (1, -9, 5, 5, -2, 4, 9, 0, 1) \\ b_5 &= (-7, -4, -3, 1, 3, 5, 2, 0, -3) & b_6 &= (-8, -8, -1, 2, -5, -8, 1, -4, -3) \\ b_7 &= (7, -1, -4, 0, 6, -1, 4, -11, -7) & b_8 &= (0, 3, 11, 6, 0, 0, -3, 1, -7) \\ b_9 &= (-3, 4, 10, -3, -7, 8, 0, -6, 8). \end{aligned}$$

The basis vectors b_1 and b_5 have lengths 11 and $\sqrt{122}$, respectively, and so are SIS_2 solutions.

12.3 Finding integer relations

Let $\beta = (\beta_1, \beta_2, \ldots, \beta_n)$ be a vector of real numbers. An *integer relation for* β is a linear combination

$$c_1\beta_1 + c_2\beta_2 + \cdots + c_n\beta_n = 0, \tag{122}$$

where $c_1, c_2, \ldots, c_n$ are integers. An integer relation algorithm attempts to find such a nonzero vector $c = (c_1, c_2, \ldots, c_n)$, preferably with small coordinates, or determines that no nontrivial relation exists.

In practice, computations must be performed using finite-precision arithmetic. Consequently, integer relation algorithms operate on rational approximations to the real numbers β_i. Specifically, one is given a vector $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathbb{Q}^n$ satisfying $\alpha_i \approx \beta_i$ for each i. If the algorithm produces a nonzero vector $c \in \mathbb{Z}^n$ such that

$$c_1\alpha_1 + c_2\alpha_2 + \cdots + c_n\alpha_n = 0,$$

this suggests that the corresponding integer relation (122) holds for the original real numbers. The validity of the relation can then be verified using high-precision computations, after which one may seek a mathematical proof.

In this section, we explain how the LLL algorithm can be applied to the problem of finding integer relations. As an illustrative example, we also discuss its use in discovering *Machin-like formulas*, which are integer relations involving π and values of the arctangent function.

Given a rational approximation $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{Q}^n$ to $\beta = (\beta_1, \ldots, \beta_n) \in \mathbb{R}^n$, select a *scaling factor* $Q \in \mathbb{Z}$ and define a lattice $L \subseteq \mathbb{Q}^{n+1}$ with basis matrix

$$B = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ Q\alpha_1 & Q\alpha_2 & Q\alpha_3 & \cdots & Q\alpha_n \end{bmatrix}_{(n+1)\times n}. \tag{123}$$

The lattice L has rank n and is therefore *not* a full-rank lattice. Its vectors are of the form

$$v = (x_1, x_2, \ldots, x_n, Q(\alpha_1 x_1 + \cdots + \alpha_n x_n)),$$

where $x_1, \ldots, x_n \in \mathbb{Z}$. If such a vector v has small length, then the coordinates x_i must themselves be small. Furthermore, for a sufficiently large scaling factor Q, the rational number $|\sum_{i=1}^n \alpha_i x_i|$ is expected to be very small, so that the vector $x = (x_1, \ldots, x_n)$ is a promising candidate for an integer relation $x_1\beta_1 + \cdots + x_n\beta_n = 0$.

The LLL algorithm is applied to compute an LLL-reduced basis of L. One then examines the reduced basis vectors to determine whether any of them provide plausible candidates for an integer relation. The effectiveness of this approach depends on the choice of scaling factor Q and on the accuracy of the approximations $\alpha_i \approx \beta_i$, and typically requires substantial experimentation.

Example 12.3 (*discovering a Machin-like integer relation*) Consider the real numbers

$$\beta_1 = \arctan\left(\frac{1}{18}\right), \quad \beta_2 = \arctan\left(\frac{1}{57}\right), \quad \beta_3 = \arctan\left(\frac{1}{239}\right), \quad \beta_4 = \pi,$$

and their rational approximations

$$\alpha_1 = \frac{2432}{43821}, \quad \alpha_2 = \frac{11971}{682417}, \quad \alpha_3 = \frac{717}{171364}, \quad \alpha_4 = \frac{104348}{33215}.$$

Choose the scaling factor $Q = 100000000$. The associated lattice $L \subseteq \mathbb{Q}^5$ is defined by

the following basis matrix

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ Q\alpha_1 & Q\alpha_2 & Q\alpha_3 & Q\alpha_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & \frac{243200000000}{43821} \\ 0 & 1 & 0 & 0 & \frac{1197100000000}{682417} \\ 0 & 0 & 1 & 0 & \frac{17925000000}{42841} \\ 0 & 0 & 0 & 1 & \frac{2086960000000}{6643} \end{bmatrix}^T.$$

Applying the LLL algorithm yields the following reduced basis:

$$\tilde{B} = \begin{bmatrix} -48 & -32 & 20 & 1 & -\frac{436321916060000000}{2836839193980133797} \\ -40 & 112 & 61 & 0 & -\frac{108699521075000000}{1281125633289237} \\ -73 & 75 & -97 & 1 & \frac{766445843743945000000}{8510517581940401391} \\ -34 & 83 & 103 & 0 & \frac{209280728875000000}{1281125633289237} \end{bmatrix}^T.$$

The first basis vector is

$$\begin{aligned} v &= (x_1, x_2, x_3, x_4, Q(x_1\alpha_1 + x_2\alpha_2 + x_3\alpha_3 + x_4\alpha_4)) \\ &= \left(-48, -32, 20, 1, -\frac{436321916060000000}{2836839193980133797}\right) \end{aligned}$$

with

$$x_1\alpha_1 + x_2\alpha_2 + x_3\alpha_3 + x_4\alpha_4 \approx -0.1538 \times 10^{-8}.$$

Hence, $c = (-48, -32, 20, 1)$ is a plausible candidate solution vector. A high-precision floating point computation confirms that

$$x_1\beta_1 + x_2\beta_2 + x_3\beta_3 + x_4\beta_4 \approx 0.$$

Consequently, a candidate integer relation is

$$\frac{\pi}{4} = 12\arctan\left(\frac{1}{18}\right) + 8\arctan\left(\frac{1}{57}\right) - 5\arctan\left(\frac{1}{239}\right). \tag{124}$$

This relation was originally discovered and proven by Gauss using properties of the Gaussian integers.

12.4 Simultaneous Diophantine approximation

Simultaneous Diophantine approximation concerns the problem of finding good rational approximations to several real numbers, using a small common denominator. More precisely, a simultaneous Diophantine approximation of the vector of positive real numbers $\beta = (\beta_1, \beta_2, \ldots, \beta_n)$ is a vector of rational numbers $(p_1/q, p_2/q, \ldots, p_n/q)$ that share the same (small) denominator q, and such that $p_i/q \approx \beta_i$ for $1 \leq i \leq n$.

The LLL algorithm can be used to efficiently find simultaneous Diophantine approximations that are good in the following sense. Let α_i be a close rational number approximation to β_i for $1 \leq i \leq n$, and let $\varepsilon \in \mathbb{Q}$ with $0 < \varepsilon < 1$. We will show how LLL can be used to efficiently find integers $p_1, \ldots, p_n, q$ such that

$$1 \leq q \leq 2^{n(n+1)/4}/\varepsilon^n \tag{125}$$

(this is the sense in which q is small) and

$$|p_i - q\alpha_i| \leq \varepsilon \text{ for } 1 \leq i \leq n \tag{126}$$

(and so p_i/q is a close approximation to α_i).

Let $L \subseteq \mathbb{Q}^{n+1}$ be the lattice with basis matrix

$$B = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & -\alpha_1 \\ 0 & 1 & 0 & \cdots & 0 & -\alpha_2 \\ 0 & 0 & 1 & \cdots & 0 & -\alpha_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & -\alpha_n \\ 0 & 0 & 0 & \cdots & 0 & \varepsilon^{n+1}/2^{n(n+1)/4} \end{bmatrix}_{(n+1)\times(n+1)}, \tag{127}$$

and let $\tilde{B} = [b_1, \ldots, b_n, b_{n+1}]$ be an LLL-reduced basis of L. Note that

$$\mathrm{vol}(L) = \varepsilon^{n+1}/2^{n(n+1)/4}.$$

By Theorem 10.17 we have

$$\|b_1\| \leq 2^{n/4}\,\mathrm{vol}(L)^{1/(n+1)} = \varepsilon.$$

Since $b_1 \in L$, we can write

$$b_1 = (p_1 - q\alpha_1,\ p_2 - q\alpha_2,\ \ldots,\ p_n - q\alpha_n,\ q \cdot \varepsilon^{n+1}/2^{n(n+1)/4}), \tag{128}$$

where $p_1, p_2, \ldots, p_n, q \in \mathbb{Z}$. Hence

$$\|b_1\|^2 = (p_1 - q\alpha_1)^2 + (p_2 - q\alpha_2)^2 + \cdots + (p_n - q\alpha_n)^2 + q^2(\varepsilon^{n+1}/2^{n(n+1)/4})^2 \leq \varepsilon^2. \tag{129}$$

It follows that

$$|p_i - q\alpha_i| \leq \varepsilon \ \text{ for } \ 1 \leq i \leq n,$$

and

$$|q| \le 2^{n(n+1)/4}/\varepsilon^n.$$

Since $\|b_1\| \neq 0$ and $\varepsilon < 1$, the inequality (129) implies that $q \neq 0$. Finally, we may replace b_1 by $-b_1$ if necessary and thereby assume that $q \ge 1$.

Example 12.4 (*simultaneous Diophantine approximation*) Let us find small rational approximations with a common denominator to the irrational numbers $\beta_1 = \sqrt{2}$, $\beta_2 = \sqrt{3}$, $\beta_3 = e$ and $\beta_4 = \pi$ with $n = 4$ and $\varepsilon = \frac{1}{3}$. We begin with the following rational approximations obtained from the decimal number representations of the β_i's:

$$\begin{aligned}
\sqrt{2} &\approx \alpha_1 = 1414213562/1000000000 = 707106781/500000000,\\
\sqrt{3} &\approx \alpha_2 = 1732050808/1000000000 = 216506351/125000000,\\
e &\approx \alpha_3 = 2718281828/1000000000 = 679570457/250000000,\\
\pi &\approx \alpha_4 = 3141592654/1000000000 = 1570796327/500000000.
\end{aligned}$$

The associated lattice L has basis matrix

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 & -\alpha_1 \\ 0 & 1 & 0 & 0 & -\alpha_2 \\ 0 & 0 & 1 & 0 & -\alpha_3 \\ 0 & 0 & 0 & 1 & -\alpha_4 \\ 0 & 0 & 0 & 0 & \varepsilon^{n+1}/2^{n(n+1)/4} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & -707106781/500000000 \\ 0 & 1 & 0 & 0 & -216506351/125000000 \\ 0 & 0 & 1 & 0 & -679570457/250000000 \\ 0 & 0 & 0 & 1 & -1570796327/500000000 \\ 0 & 0 & 0 & 0 & 1/7776 \end{bmatrix}.$$

The LLL algorithm produces the following basis:

$$\tilde{B} = \begin{bmatrix} -\frac{2362823}{62500000} & \frac{50252533}{500000000} & -\frac{11147309}{125000000} & -\frac{18723299}{250000000} & -\frac{21681981}{500000000} \\ -\frac{1277133}{15625000} & -\frac{15544457}{125000000} & \frac{954961}{31250000} & -\frac{1461329}{62500000} & -\frac{11425551}{125000000} \\ \frac{1902069}{31250000} & -\frac{6993199}{250000000} & -\frac{5179273}{62500000} & \frac{17116697}{125000000} & -\frac{23204857}{250000000} \\ -\frac{1095141}{62500000} & \frac{4425711}{500000000} & -\frac{16607703}{125000000} & -\frac{24379033}{250000000} & \frac{66265273}{500000000} \\ \frac{83}{972} & \frac{7}{7776} & -\frac{37}{648} & -\frac{107}{1296} & -\frac{799}{7776} \end{bmatrix}.$$

The first basis vector is

$$b_1 = \left(-\frac{2362823}{62500000}, -\frac{1277133}{15625000}, \frac{1902069}{31250000}, -\frac{1095141}{62500000}, \frac{83}{972}\right).$$

Using (128), we obtain (after adjusting signs) $q = 664$, $p_1 = 939$, $p_2 = 1150$, $p_3 = 1805$, $p_4 = 2086$. The resulting simultaneous Diophantine approximations to the real numbers $\sqrt{2}$, $\sqrt{3}$, e and π are compared below.

real number	rational approximation	decimal rep.	smaller rational approximation	decimal rep.
$\sqrt{2}$	707106781/500000000	1.414213562	939/664	1.414156627
$\sqrt{3}$	216506351/125000000	1.732050808	1150/664	1.731927711
e	679570457/250000000	2.718281828	1805/664	2.718373494
π	1570796327/500000000	3.141592654	2086/664	3.141566265

Example 12.5 With $n = 5$ and $\varepsilon = \frac{1}{2}$, the LLL algorithm was used to find the following small rational approximations with a common denominator:

$$\sqrt{2} \approx \frac{481}{340}, \quad \sqrt{13} \approx \frac{1226}{340}, \quad \sqrt{29} \approx \frac{1831}{340}, \quad \sqrt{47} \approx \frac{2331}{340}, \quad \sqrt{71} \approx \frac{2865}{340}.$$

12.5 Side-channel attack on ECDSA

The LLL lattice basis reduction algorithm, together with refinements such as BKZ, has played a central role in the cryptanalysis of many public-key cryptosystems. One of the earliest and most striking applications of LLL was the successful cryptanalysis of knapsack-based public-key encryption schemes (Exercise 12.13). These schemes, based on the subset sum problem, were among the very first public-key proposals of the 1970s. Since then, LLL-based techniques have also been applied to RSA, DSA, and ECDSA, especially in settings where partial information about secret key material is exposed. Such leakage can result from flawed implementations, poor parameter selection, or side-channel attacks. This section presents a side-channel attack on ECDSA.

ECDSA. The *domain parameters* for the Elliptic Curve Digital Signature Algorithm (ECDSA) consist of an elliptic curve E defined over a prime field $\mathbb{Z}_p$, chosen so that the number n of points in $E(\mathbb{Z}_p)$ is prime, together with a generator point $P \in E(\mathbb{Z}_p)$. A user, Alice, selects a secret integer $a \in_R [1, n-1]$ and computes the point $A = aP$. The point A serves as Alice's *long-term public key*, while a is her corresponding *long-term private key*.

To sign a message M, Alice first computes its hash $m = H(M)$. She then chooses a *per-message secret* $k \in_R [1, n-1]$, and computes the point $R = kP$. Let $r \in [1, n-1]$ denote the x-coordinate of R reduced modulo n. Alice also computes the integer $s \in [1, n-1]$ as

$$s = k^{-1}(m + ar) \bmod n. \tag{130}$$

Alice's signature on M is the pair of integers (r, s). Since the side-channel attack of interest targets the signature generation process, the verification algorithm is omitted.

For ECDSA to be secure, both the long-term private key a and each per-message secret key k must be chosen uniformly at random from $[1, n-1]$, and a fresh value of k must be used for every signature. In practice, however, partial information about some

per-message secrets k may leak during the signing process. Such leakage can arise from weaknesses including poor random number generation, timing variations, electromagnetic emanations, and so on. In some cases, the leaked information is sufficient to recover the long-term private key a. Once a is compromised, the entire ECDSA implementation is effectively broken.

Leakage with two signatures. We will first consider a setting in which an attacker learns substantial partial information about the per-message secrets used in two ECDSA signatures. Specifically, suppose the attacker discovers that somewhat more than half the most significant bits of each per-message secret k are zero. Although this scenario is rather contrived, it provides a clean illustration of how lattice techniques can be applied in side-channel attacks. We therefore assume that the secrets satisfy $k_1, k_2 \leq K$, where $K \ll n$ is a parameter to be specified later.

Assume the adversary obtains two signed messages (M_1, r_1, s_1) and (M_2, r_2, s_2). From the two ECDSA signature equations (130), the adversary can derive the linear congruences:

$$k_1 s_1 \equiv m_1 + a r_1 \pmod{n} \quad \text{and} \quad k_2 s_2 \equiv m_2 + a r_2 \pmod{n}.$$

Solving each congruence for a and equating the results yields the congruence

$$k_1 s_1 r_2 - m_1 r_2 \equiv k_2 s_2 r_1 - m_2 r_1 \pmod{n}.$$

Rearranging terms gives the linear congruence

$$k_1 - (s_1 r_2)^{-1} s_2 r_1 k_2 + (s_1 r_2)^{-1}(m_2 r_1 - m_1 r_2) \equiv 0 \pmod{n},$$

in which the only quantities that are not known to the adversary are the two per-message secrets k_1 and k_2. This may be written more compactly as

$$k_1 + t k_2 + u \equiv 0 \pmod{n}$$

where

$$t = -(s_1 r_2)^{-1} s_2 r_1 \bmod n \quad \text{and} \quad u = (s_1 r_2)^{-1}(m_2 r_1 - m_1 r_2) \bmod n.$$

Equivalently, the congruence can be expressed as the equation

$$k_1 + t k_2 + u + n x = 0,$$

where $x \in \mathbb{Z}$. The values t and u can be computed by the adversary.

Now, consider the integer lattice $L \subseteq \mathbb{Z}^3$ with basis matrix

$$B = \begin{bmatrix} n & t & u \\ 0 & 1 & 0 \\ 0 & 0 & K \end{bmatrix}.$$

Observe that

$$v = B[x, k_2, 1]^T = [xn + t k_2 + u, k_2, K]^T = [-k_1, k_2, K]^T,$$

so $v = (-k_1, k_2, K) \in L$. Moreover, the length of v satisfies

$$\|v\| = \sqrt{k_1^2 + k_2^2 + K^2} \leq \sqrt{3K^2} = \sqrt{3}K.$$

The lattice L has volume $\det(B) = nK$. By the Gaussian heuristic (8),

$$\lambda_1(L) \approx \sqrt{3/2\pi e}\,(nK)^{1/3} \approx (nK)^{1/3}.$$

Consequently, if $\|v\| \ll (nK)^{1/3}$, which reduces to the condition $K \ll \sqrt{n}$, then v is expected to be an unusually short lattice vector and, in particular, to be a shortest nonzero vector of L. In this case, the adversary can plausibly recover v by applying the LLL algorithm to compute a reduced basis of L, and checking whether $\pm v$ appears among the basis vectors. Once v is known, the values of k_1 and k_2 are immediately revealed, and the long-term private key a can then be recovered.

Leakage with many signatures. We now turn to a more realistic setting in which an adversary obtains several ECDSA signed messages (M_i, r_i, s_i), $1 \leq i \leq N$, and also learns the ℓ least significant bits of each of the corresponding per-message secrets k_i. Writing $k_i = \alpha_i 2^\ell + \beta_i$ where $\beta_i < 2^\ell$, we assume the values β_i are known to the adversary, while the α_i remain unknown. Let $K = \lceil n/2^\ell \rceil$, so $\alpha_i < K$.

For each i, $1 \leq i \leq N-1$, eliminate a from the two ECDSA signing equations (130) corresponding to message M_i and the last message M_N. This yields a linear congruence of the form

$$k_i + t_i' k_N + u_i' \equiv 0 \pmod{n},$$

where the only quantities that are not known to the adversary are k_i and k_N. Substituting $k_i = \alpha_i 2^\ell + \beta_i$ and $k_N = \alpha_N 2^\ell + \beta_N$ gives

$$(\alpha_i 2^\ell + \beta_i) + t_i'(\alpha_N 2^\ell + \beta_N) + u_i' \equiv 0 \pmod{n}.$$

Multiplying both sides by the inverse of $2^\ell \bmod n$, simplifying, and eliminating the "mod n" operator, results in $N-1$ equations

$$\alpha_i + t_i \alpha_N + u_i + x_i n = 0,$$

where the unknown quantities are the α_i for $1 \leq i \leq N$ and the integers x_i for $1 \leq i \leq N-1$.

Next, consider the $(N+1)$-dimensional integer lattice $L \subseteq \mathbb{Z}^{N+1}$ with basis matrix

$$B = \begin{bmatrix} n & 0 & \cdots & 0 & t_1 & u_1 \\ 0 & n & \cdots & 0 & t_2 & u_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & n & t_{N-1} & u_{N-1} \\ 0 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & \cdots & 0 & 0 & K \end{bmatrix}_{(N+1)\times(N+1)}.$$

Observe that

$$v = B[x_1, x_2, \ldots, x_{N-1}, \alpha_N, 1]^T = [-\alpha_1, -\alpha_2, \ldots, -\alpha_{N-1}, \alpha_N, K]^T$$

is a lattice vector of length

$$\|v\| = \sqrt{\alpha_1^2 + \cdots + \alpha_N^2 + K^2} \leq \sqrt{N+1}\, K \approx \sqrt{N} K.$$

The lattice L has volume $\det(B) = n^{N-1}K$. By the Gaussian heuristic (8),

$$\lambda_1(L) \approx \sqrt{(N+1)/2\pi e}\, (n^{N-1}K)^{1/(N+1)} \approx N^{1/2} n^{(N-1)/(N+1)} K^{1/(N+1)}.$$

For the attack to succeed, we require that $\|v\|$ be significantly smaller than this estimate for $\lambda_1(L)$, meaning

$$N^{1/2}K \ll N^{1/2} n^{(N-1)/(N+1)} K^{1/(N+1)},$$

which simplifies to the condition $K^N \ll n^{N-1}$.

Let t denote the bitlength of n, so that $\log_2 K \approx t - \ell$. By taking logarithms of both sides, the inequality $K^N \ll n^{N-1}$ then becomes $N(t-\ell) \ll Nt - t$, or equivalently, $N \gg t/\ell$. Hence, if the adversary obtains signed messages for which the ℓ least significant bits of the per-message secrets are known across $N \gg t/\ell$ messages, one can expect v to be an exceptionally short lattice vector and, in particular, to be a shortest nonzero vector in L. In this case, applying LLL to compute a reduced basis of L and checking whether $\pm v$ appears among the basis vectors is likely to recover v. From this, the adversary can determine the values of k_i and subsequently recover the long-term private key a.

Example 12.6 (*side-channel attack on ECDSA*) Select $n = 935878837$ which is a $t = 30$-bit prime number, $\ell = 5$, and $N = 10$. We have $K = \lceil n/2^\ell \rceil = 29246214$. Suppose that the adversary has obtained 10 ECDSA-signed messages and knows the 5 least significant bits of each of the corresponding per-message secrets. Assume that the unknown portions α_i of the per-message secrets are:

$\alpha_1 = 9550684$ $\alpha_2 = 11443958$ $\alpha_3 = 8533292$ $\alpha_4 = 505823$ $\alpha_5 = 9809889$
$\alpha_6 = 14944555$ $\alpha_7 = 16279651$ $\alpha_8 = 1484325$ $\alpha_9 = 21735620$ $\alpha_{10} = 3788058$.

Also, assume that the quantities t_i that the adversary computed are:

$t_1 = 201354591$ $t_2 = 891235047$ $t_3 = 922939225$ $t_4 = 934992634$ $t_5 = 676943009$
$t_6 = 889775274$ $t_7 = 471852626$ $t_8 = 206613081$ $t_9 = 215952988$.

Let $u_i = -\alpha_i - t_i\alpha_{10}$ for $1 \leq i \leq 9$, so:

$u_1 = 245087712$ $u_2 = 884448799$ $u_3 = 274011166$ $u_4 = 926348469$ $u_5 = 199874056$
$u_6 = 477691203$ $u_7 = 847522394$ $u_8 = 33208359$ $u_9 = 802959243$.

As shown above, the adversary is able to compute the u_i without knowledge of the α_i. The adversary's task is to determine the α_i, given n, t_i and u_i.

The associated lattice is $L = L(B) \subseteq \mathbb{Z}^{11}$ where

$$B = \begin{bmatrix} n & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 201354591 & 245087712 \\ 0 & n & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 891235047 & 884448799 \\ 0 & 0 & n & 0 & 0 & 0 & 0 & 0 & 0 & 922939225 & 274011166 \\ 0 & 0 & 0 & n & 0 & 0 & 0 & 0 & 0 & 934992634 & 926348469 \\ 0 & 0 & 0 & 0 & n & 0 & 0 & 0 & 0 & 676943009 & 199874056 \\ 0 & 0 & 0 & 0 & 0 & n & 0 & 0 & 0 & 889775274 & 477691203 \\ 0 & 0 & 0 & 0 & 0 & 0 & n & 0 & 0 & 471852626 & 847522394 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & n & 0 & 206613081 & 33208359 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & n & 215952988 & 802959243 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 29246214 \end{bmatrix}_{11\times 11}.$$

Applying the LLL algorithm to this basis produces the reduced basis shown in Figure 12.1. The first vector of the basis is the desired one $(-\alpha_1, \ldots, -\alpha_9, \alpha_{10}, K)$. The adversary can identify this vector by noticing that K appears in its last component, then extracting the α_i, computing the k_i (using the known values β_i) and thereafter the long-term private key a, and verifying that $A = aP$.

Extensive experimental results indicate that the attack can succeed even when as few as $\ell = 2$ bits of per-message secrets are leaked. Moreover, this approach can be adapted to settings in which the adversary learns a small number of most significant bits, or even a few middle bits, of the per-message secrets. In practice, these techniques have proven effective against poor implementations of ECDSA systems, including real-world Bitcoin applications.

12.6 Finding small roots of a polynomial modulo N

Univariate polynomials with integer coefficients can be efficiently factored over the integers or, equivalently, over the rational numbers. In fact, the first major application of the LLL lattice basis reduction algorithm was a polynomial-time method for this task. There are also randomized algorithms for factoring polynomials modulo a prime p that run in expected polynomial time. In contrast, no efficient algorithm is known for factoring polynomials modulo a composite number N; indeed, this problem is as hard as factoring N itself.

In 1996, Coppersmith demonstrated that the LLL algorithm can be used to find all sufficiently small roots of a polynomial $f(x) \in \mathbb{Z}_N[x]$, where N is a composite number whose factorization is unknown. More precisely, for a degree-d polynomial $f(x)$, the method recovers all roots x_0 satisfying $|x_0| \leq N^{1/d}$.

In §12.6.1, we present Howgrave-Graham's formulation of what is still widely called Coppersmith's method. Then, in §12.6.2 and §12.6.3, we describe two applications that exploit weaknesses in some padding mechanisms for the RSA public-key encryption scheme.

$$\left[\begin{array}{rrrrrr}
-9550684 & -11443958 & -8533292 & -505823 & -9809889 & -14944555 \\
-16279651 & -1484325 & -21735620 & 3788058 & 29246214 & \\
\hline
-41311850 & 16900714 & 41413030 & -93099186 & -8553329 & -2542146 \\
1419659 & -25764384 & -3240651 & -584960 & 29246214 & \\
\hline
34135239 & -6867183 & 36587811 & -33449287 & 27568406 & 2322804 \\
-67573114 & 12645502 & 13904351 & 73069518 & 0 & \\
\hline
44355001 & -3193014 & 12572852 & 227503 & 3773040 & -81867128 \\
32886850 & -8674033 & 39863888 & 71775813 & 0 & \\
\hline
-15272374 & 42386155 & -35766423 & -13087977 & -32724161 & -33155395 \\
-42424262 & 30657732 & 19663684 & 69776738 & -58492428 & \\
\hline
12219651 & -89373135 & 42673924 & -64744357 & -40234085 & 19955255 \\
-22397523 & -34546996 & -11644606 & 6163219 & -29246214 & \\
\hline
19729402 & -7687782 & -2384161 & 55173765 & -81415633 & 31377161 \\
77473667 & -55455163 & 5685632 & 38448777 & 0 & \\
\hline
55358855 & 34048060 & -48308186 & -38930222 & 13619789 & 7485867 \\
-30183366 & 34740289 & -83396757 & -32176865 & -58492428 & \\
\hline
8548959 & 4082825 & -10754428 & -1310998 & 15376068 & 24531739 \\
-33765648 & -89629917 & 44660893 & -32612045 & 58492428 & \\
\hline
-16512091 & 5472975 & 62360251 & 45786710 & -32495195 & -49558677 \\
20009308 & -113304986 & 15649897 & -34101103 & -29246214 & \\
\hline
-9597542 & 29413244 & 20704559 & -36375836 & 55150417 & -6234725 \\
-18905490 & -64921672 & -60601488 & 55864289 & -29246214 &
\end{array}\right]$$

Figure 12.1: The LLL-reduced basis in Example 12.6. The basis vectors are the *rows* of the displayed matrix. Due to space constraints, each row of the matrix spans two lines.

12.6.1 The Coppersmith/Howgrave-Graham method

Let $f(x) \in \mathbb{Z}[x]$ be a monic polynomial of degree d, and let N be a composite number with unknown factorization. Suppose that x_0 is a small root of $f(x)$ modulo N. Coppersmith's method begins by applying the LLL algorithm to construct a polynomial $G(x)$ with small integer coefficients such that

$$G(x_0) \equiv 0 \pmod{N^\alpha}$$

for some $\alpha \geq 1$. If both x_0 and the coefficients of G are sufficiently small, then one expects that

$$-N^\alpha < G(x_0) < N^\alpha,$$

which forces $G(x_0) = 0$. Consequently, x_0 can be recovered by computing all integer roots β of $G(x)$ and checking which of them satisfy $f(\beta) \equiv 0 \pmod{N}$.

Algorithm description. Fix an integer $h \geq 2$ (the choice of h will be discussed later), and set $n = dh$. Define the solution bound

$$X = \left\lceil 2^{-1/2}\, N^{(h-1)/(n-1)}\, n^{-1/(n-1)} \right\rceil - 1. \tag{131}$$

Note that $X \longrightarrow N^{1/d}/\sqrt{2}$ as $h \longrightarrow \infty$. The goal is to efficiently determine all integers $x_0 \in [-X, X]$ such that $f(x_0) \equiv 0 \pmod{N}$.

For each $i \in [0, h-1]$ and $j \in [0, d-1]$, define the polynomial

$$g_{i,j}(x) = N^{h-1-i} x^j f(x)^i. \tag{132}$$

Then $\deg(g_{i,j}) = di + j \leq n-1$, and any root x_0 of $f(x)$ modulo N satisfies

$$g_{i,j}(x_0) \equiv 0 \pmod{N^{h-1}}.$$

Index these polynomials as $G_\ell(x)$ for $1 \leq \ell \leq n$ by setting

$$G_\ell(x) = g_{i,j}(x), \quad \text{where } j = (\ell - 1) \bmod d \ \text{ and } \ i = \lfloor (\ell-1)/d \rfloor.$$

Next, let B be the $n \times n$ matrix whose ℓth row consists of the coefficients of $G_\ell(xX)$. That is, if $G_\ell(x) = \gamma_0 + \gamma_1 x + \cdots + \gamma_{n-1} x^{n-1}$, then the ℓth row of B is

$$[\gamma_0,\ \gamma_1 X,\ \gamma_2 X^2,\ \ldots,\ \gamma_{n-1} X^{n-1}].$$

Consider the integer lattice $L = L(B) \subseteq \mathbb{Z}^n$ spanned by the rows[26] of B. Since B is a lower triangular matrix with diagonal entries

$$\begin{array}{cccc}
N^{h-1}, & N^{h-1}X, & \ldots, & N^{h-1}X^{d-1}, \\
N^{h-2}X^{d}, & N^{h-2}X^{d+1}, & \ldots, & N^{h-2}X^{2d-1}, \\
N^{h-3}X^{2d}, & N^{h-3}X^{2d+1}, & \ldots, & N^{h-3}X^{3d-1}, \\
\cdots & \cdots & \cdots & \cdots \\
X^{(h-1)d}, & X^{(h-1)d+1}, & \ldots, & X^{hd-1},
\end{array}$$

[26] In this subsection, the lattice L is spanned by the *rows* of B instead of its columns.

we have

$$\mathrm{vol}(L) = |\det(B)| = N^{n(h-1)/2} X^{n(n-1)/2}. \tag{133}$$

Each row of B corresponds to a polynomial $G_\ell(x)$ that vanishes at x_0 modulo N^{h-1}. More precisely, if the ℓ-th row of B is $b_\ell = (r_0, r_1 X, r_2 X^2, \ldots, r_{n-1} X^{n-1})$, then the corresponding polynomial is $G_\ell(x) = \sum_{i=0}^{n-1} r_i x^i$. Therefore, since every lattice vector is an integer linear combination of the rows of B, it follows that every lattice vector corresponds to a polynomial $G(x)$ satisfying

$$G(x_0) \equiv 0 \pmod{N^{h-1}}.$$

Applying the LLL algorithm yields a nonzero such polynomial with small coefficients.

Now, let $\tilde{B} = [\tilde{b}_1, \ldots, \tilde{b}_n]$ be an LLL-reduced basis of L. By Theorem 10.17, we have

$$\begin{aligned}
\|\tilde{b}_1\| &\le 2^{(n-1)/4} \,\mathrm{vol}(L)^{1/n} \\
&= 2^{(n-1)/4} N^{(h-1)/2} X^{(n-1)/2} \quad \text{by (133)} \\
&< 2^{(n-1)/4} N^{(h-1)/2} \left(2^{-1/2}\, N^{(h-1)/(n-1)}\, n^{-1/(n-1)}\right)^{(n-1)/2} \quad \text{by (131)} \\
&= N^{h-1}/\sqrt{n}.
\end{aligned} \tag{134}$$

Let $G(x) = \sum_{i=0}^{n-1} r_i x^i$ be the polynomial corresponding to $\tilde{b}_1$. Then for all integers x_0 with $|x_0| \le X$, we have

$$|G(x_0)| = \left|\sum_{i=0}^{n-1} r_i x_0^i\right| \le \sum_{i=0}^{n-1} |r_i x_0^i| \le \sum_{i=0}^{n-1} |r_i| X^i. \tag{135}$$

By the Cauchy-Schwarz inequality,[27] we have

$$\left|\sum_{i=0}^{n-1} |r_i| X^i\right|^2 \le n \cdot \|G(xX)\|^2,$$

where the length of a polynomial $a(x) = \sum_{i=0}^{n-1} a_i x^i$ is $\|a(x)\| = (\sum_{i=0}^{n-1} a_i^2)^{1/2}$. Taking square roots of both sides yields

$$\sum_{i=0}^{n-1} |r_i| X^i \le \sqrt{n} \cdot \|G(xX)\| = \sqrt{n}\, \|\tilde{b}_1\|. \tag{136}$$

Combining the bounds (134), (135) and (136) gives

$$|G(x_0)| \le \sum_{i=0}^{n-1} |r_i| X^i \le \sqrt{n}\, \|\tilde{b}_1\| < N^{h-1}.$$

[27] Let $u = (|r_0|, |r_1| X, \ldots, |r_{n-1}| X^{n-1})$ and $v = (1, 1, \ldots, 1)$, and use the Cauchy-Schwarz inequality to conclude that $|\langle u, v\rangle|^2 \le \|u\|^2 \cdot \|v\|^2$.

Finally, for each x_0 satisfying $|x_0| \leq X$ and $f(x_0) = 0 \pmod{N}$, we have $G(x_0) \equiv 0 \pmod{N^{h-1}}$. Since $|G(x_0)| < N^{h-1}$, we must have $G(x_0) = 0$.

In summary, the small roots of $f(x)$ modulo N can be found by first computing all integer roots x_0 of $G(x)$ with $|x_0| \leq X$, and then checking which of them satisfy $f(x_0) \equiv 0 \pmod{N}$.

Example 12.7 (*finding small roots of a polynomial modulo N*) Consider the degree-4 polynomial

$$f(x) = x^4 + 41383461x^3 + 67085051x^2 + 66463060x + 67200429 \in \mathbb{Z}[x].$$

We wish to determine all small roots of f modulo the composite number

$$N = 77123141.$$

Select $h = 2$, whereby $n = 8$ and $X = 7$. The associated lattice is $L = L(B) \subseteq \mathbb{Z}^8$ where the matrix B is given in Figure 12.2. Applying the LLL algorithm to this basis produces the

$$\left[\begin{array}{rrrr}
77123141 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\hline
0 & 539861987 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\hline
0 & 0 & 3779033909 & 0 \\
0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 26453237363 \\
0 & 0 & 0 & 0 \\
\hline
67200429 & 465241420 & 3287167499 & 14194527123 \\
2401 & 0 & 0 & 0 \\
\hline
0 & 470403003 & 3256689940 & 23010172493 \\
99361689861 & 16807 & 0 & 0 \\
\hline
0 & 0 & 3292821021 & 22796829580 \\
161071207451 & 695531829027 & 117649 & 0 \\
\hline
0 & 0 & 0 & 23049747147 \\
159577807060 & 1127498452157 & 4868722803189 & 823543
\end{array}\right]$$

Figure 12.2: The 8×8 basis matrix B in Example 12.7.

reduced basis shown in Figure 12.3. Dividing the entries of the first row of $\tilde{B}$ by powers of X yields the polynomial

$$g(x) = 9x^6 + 12x^5 + 450x^4 - 3259x^3 + 19504x^2 + 13424x - 606720 \in \mathbb{Z}[x],$$

whose factorization over $\mathbb{Z}$ is

$$g(x) = (x - 5)(x + 4)(9x^4 + 21x^3 + 651x^2 - 2188x + 30336).$$

$$\left[\begin{array}{rrrr}
-606720 & 93968 & 955696 & -1117837 \\
1080450 & 201684 & 1058841 & 0 \\
\hline
164520 & 112462 & -395626 & 266168 \\
-21609 & -1731121 & 235298 & 1647086 \\
\hline
-801700 & -718235 & 1949171 & 434581 \\
-770721 & 941192 & 235298 & 1647086 \\
\hline
-107360 & -337456 & 170814 & 1630965 \\
-7203 & -1613472 & 941192 & -1647086 \\
\hline
-722160 & -1016596 & 683648 & 899003 \\
2100875 & 1462209 & -941192 & 823543 \\
\hline
57460 & -1119909 & -1428644 & 1638168 \\
825944 & 1361367 & 2470629 & 823543 \\
\hline
64720521 & -6353277 & 23837373 & 1704710 \\
11505592 & 1378174 & 5647152 & 1647086 \\
\hline
20683901 & 423093853 & 69889778 & 183500198 \\
53331012 & 82068581 & 34471157 & 37882978
\end{array}\right]$$

Figure 12.3: The LLL-reduced basis $\tilde{B}$ in Example 12.7.

The roots of $g(x)$ in $[-X, X]$ are $x_0 = 5$ and $x_0 = -4$, which one can check are also roots of $f(x)$ modulo N.

Analysis. The overall complexity of the Coppersmith/Howgrave-Graham method is governed primarily by the cost of the LLL reduction step. For a lattice $L \subseteq \mathbb{Z}^n$ with basis $B = [b_1, \ldots, b_n]$, the LLL algorithm computes an LLL-reduced basis in time $O(n^6 \log^3 Y)$ where $Y = \max_i \|b_i\|$ (Theorem 10.30). By reducing the coefficients of $f(x)$ modulo N, we may assume without loss of generality that they lie in the range $[0, N-1]$.

Now, the length of the longest basis vector b_i is

$$\max_\ell \|G_\ell(xX)\| \le \max_\ell \sqrt{n}\, X^{n-1} \|G_\ell(x)\|_\infty \le \sqrt{n}\, N^{h-1} \max_\ell \|G_\ell(x)\|_\infty,$$

where $\|a(x)\|_\infty$ denotes the maximum of the absolute values of the coefficients of $a(x)$. One can verify that $\|f(x)^i\|_\infty \le d^{i-1} N^i$ for $i \ge 1$. Hence, we deduce from (132) that

$$\|G_\ell(x)\|_\infty \le N^{h-1} d^{h-2} N^{h-1},$$

and so

$$\max_\ell \|G_\ell(xX)\| \le \sqrt{n}\, d^{h-2} N^{3h-3},$$

which implies that $\log Y = O(h \log N)$. Consequently, the running time of the Coppersmith/ Howgrave-Graham method is

$$O(d^6 h^9 \log^3 N). \tag{137}$$

Recall that $X \to N^{1/d}/\sqrt{2}$ as $h \to \infty$. Increasing h is therefore beneficial, as it enlarges the bound X within which solutions may be found. However, this comes at the cost of a rapidly increasing running time, as indicated by (137). In practice, one typically starts with $h = 2$ and incrementally increases h until the sought-after solution is found.

Remark 12.8 The Coppersmith/Howgrave-Graham method can be refined to find all the roots x_0 with $|x_0| \leq N^{1/d}$ of a degree-d polynomial $f(x)$ over $\mathbb{Z}_N$ in polynomial time.

12.6.2 RSA encryption with fixed padding

When utilizing RSA with a small public exponent e, one must be careful when encrypting short messages. Consider the case of encryption exponent $e = 3$. Suppose the scheme is used to encrypt a symmetric key K whose bitlength is much smaller than that of the RSA modulus N. For example, N might be 2048 bits long whereas K is only 256 bits. The resulting ciphertext is $c = K^3 \bmod N$. But since $K^3 < N$, reduction modulo N does not occur, so $c = K^3$. Consequently, K can be recovered simply by computing the integer cube root of c.

To mitigate this weakness, suppose that K is padded on the left with a string of bits, all of which are 1, to form a message m whose bitlength is just one less than that of N. The ciphertext is then $c = m^3 \bmod N$.

However, if the bitlength of K is less than one-third that of N, the key can still be efficiently recovered from c. Writing $m = P + K$, we have $c = (P + K)^3 \bmod N$. Thus, K is a root of the polynomial

$$f(x) = x^3 + 3Px^2 + 3P^2x + P^3 - c \pmod{N}.$$

Since $K < N^{1/3}$, it can be recovered using the Coppersmith/Howgrave-Graham method.

12.6.3 RSA encryption with random padding

To achieve security against chosen-ciphertext attacks, a plaintext message should be randomized prior to applying the RSA encryption function. One way to introduce this randomness is through random padding. The following example illustrates an attack on RSA with random right-padding.

Let N be an RSA modulus and let $e = 3$ be the encryption exponent. Suppose a message M is encrypted by first padding it on the right with a randomly selected bitstring R of length k. Thus, the ciphertext corresponding to M is $c = m^3 \bmod N$ where $m = 2^k M + R$.

Now suppose a second encryption of the same message M is performed using independent randomness $R' \in_R \{0,1\}^k$, yielding $m' = 2^k M + R'$ and $c' = (m')^3 \bmod N$. Assume an adversary obtains N, c and c'. We show that the adversary can efficiently recover M.

Let $r = R' - R$, so that $m' = m + r$. Consider the bivariate polynomials

$$f_1(x, y) = y^3 - c \quad \text{and} \quad f_2(x, y) = (x + y)^3 - c' \in \mathbb{Z}_N[x, y].$$

The resultant of f_1 and f_2 with respect to y, denoted $\text{Res}_y(f_1, f_2)$, is a degree-9 polynomial in $\mathbb{Z}_N[x]$:

$$f(x) = \text{Res}_y(f_1, f_2) = x^9 + (3c - 3c')x^6 + (3c^2 + 21cc' + 3(c')^2)x^3 + (c - c')^3.$$

The resultant has the following property: if, for some $\alpha \in \mathbb{Z}_N$, the univariate polynomials $f_1(\alpha, y)$ and $f_2(\alpha, y)$ have a common root in $\mathbb{Z}_N$, then $f(\alpha) = 0$. Now, $f_1(r, y)$ and $f_2(r, y)$ share the root $y = m$ and hence $f(r) \equiv 0 \pmod{N}$. If $|r| < N^{1/9}$, then the Coppersmith/Howgrave-Graham method can be used to efficiently determine r. Once r is known, one can easily recover m and m', and hence M (Exercise 12.11).

12.7 Notes and further references

Early applications of lattice basis reduction in cryptography are surveyed by Nguyen and Stern [180]. Lattice-based attacks on the subset sum problem were first devised by Lagarias and Odlyzko [138] and later improved by Coster et al. [64]. Exercise 12.13 uses the Lagarias-Odlyzko attack. In 1997, Ajtai and Dwork [6] introduced a public-key encryption scheme and proved its security based solely on the worst-case hardness of the unique shortest vector problem (uSVP). Shortly thereafter, Nguyen and Stern [179] presented lattice attacks showing that any secure implementation of the Ajtai-Dwork scheme would require prohibitively large keys, rendering it impractical. Also in 1997, Goldreich, Goldwasser and Halevi [103] proposed another lattice-based public-key encryption scheme. This scheme was broken by Nguyen [171] in 1999, who showed that decryption could be reduced to solving easy instances of the Closest Vector Problem. In 1985, Odlyzko and te Riele [184] used the LLL algorithm to disprove the Mertens conjecture, which posited that $|\sum_{n \le x} \mu(n)| < x^{1/2}$ for $x \ge 2$, where μ denotes the Möbius function. This conjecture was of interest in part because its validity would have implied the Riemann hypothesis.

Lenstra, Lenstra and Lovász [144] were the first to propose the use of lattice basis reduction for discovering integer relations. Borwein and Lisoněk [40] presented a number of examples of relations obtained using integer relation-finding algorithms, including LLL and PSLQ. Håstad, Just, Lagarias and Schnorr [113] introduced an integer relation-finding algorithm, together with a detailed complexity analysis. Lehmer [143] compiled a list of Machin-like formulae. The lattice-based method for finding simultaneous Diophantine approximations described in §12.4 is also due to Lenstra, Lenstra and Lovász [144].

Boneh and Venkatesan [37] showed that computing roughly $\sqrt{\log p}$ of the most significant bits of a Diffie-Hellman shared secret is as hard as computing the entire secret, by introducing the *hidden number problem* and solving it with lattice reduction and Babai's nearest-plane algorithm. This technique later became the basis for lattice attacks on

DSA and ECDSA with partially known or biased nonces. The first lattice-based attack on (EC)DSA that exploited the partial leakage of per-message secrets was proposed by Howgrave-Graham and Smart [122] and further analyzed by Nguyen and Shparlinski [174, 175]. The exposition of the ECDSA side-channel attack in §12.5 is largely based on the work of De Micheli and Heninger [70], which offers a comprehensive overview of lattice-based attacks on both RSA and elliptic curve cryptosystems. Breitner and Heninger [44] and Rowe, Breitner and Heninger [197] study weak implementations of ECDSA signing in cryptocurrencies, including Bitcoin. Aranha et al. [16] devised a side-channel attack targeting weaknesses in an implementation of the Montgomery ladder used for ECDSA point multiplication. Remarkably, their technique succeeded even under extremely limited leakage—an average of less than one bit of information about each per-message secret.

May [157] surveyed applications of the Coppersmith [62] and Howgrave-Graham [120] techniques for finding small solutions to polynomial equations modulo a composite number. Among these applications are an attack by Boneh and Durfee [35] that breaks RSA when the private exponent d has bitlength less than 0.292 times that of the RSA modulus N; an algorithm by Boneh, Durfee and Howgrave-Graham [36] for factoring integers of the form $p^r q$ with large r; Shoup's [204] security proof for RSA-OAEP with encryption exponent $e = 3$; and the deterministic polynomial-time algorithm of Coron and May [63] for factoring an RSA modulus N given the private exponent d. The flaws in specific RSA padding mechanisms described in §12.6.2 and §12.6.3 were demonstrated by Coppersmith [62].

12.8 Exercises

Exercise 12.1 (†) Let $m = 6$, $n = 4$, $q = 98947$, and $B = 300$. Use the primal attack to solve the ss-LWE(m, n, q, B) instance (A, t) where

$$A = \begin{bmatrix} 24487 & 21871 & 9749 & 28102 \\ 90929 & 26089 & 60954 & 46363 \\ 87262 & 40841 & 30905 & 55441 \\ 8122 & 70592 & 74652 & 26958 \\ 83978 & 14387 & 91339 & 17846 \\ 27139 & 89719 & 90428 & 81851 \end{bmatrix} \quad \text{and} \quad t = \begin{bmatrix} 32578 \\ 45126 \\ 74973 \\ 32418 \\ 89962 \\ 6941 \end{bmatrix}.$$

Exercise 12.2 (†) Consider the polynomial ring $R_q = \mathbb{Z}_{97}[x]/(x^4 + 1)$. Let $k = 4$, $\ell = 3$ and $B = 2$. Use the primal attack to solve the ss-MLWE instance (A, t) where:

$$A = \begin{bmatrix} 77 + 96x + 10x^2 + 86x^3 & 58 + 36x + 80x^2 + 22x^3 & 44 + 39x + 60x^2 + 39x^3 \\ 43 + 12x + 55x^2 + 2x^3 & 24 + 71x + 45x^2 + 29x^3 & 21 + 48x + 7x^2 + 33x^3 \\ 57 + 65x + 16x^2 + 93x^3 & 96 + 71x + 44x^2 + 70x^3 & 58 + 25x + 29x^2 + 73x^3 \\ 57 + 40x + 78x^2 + 88x^3 & 25 + 44x + 67x^2 + 88x^3 & 5 + 83x + 90x^2 + 33x^3 \end{bmatrix} \in R_q^{4\times 3}$$

and

$$t = \begin{bmatrix} 31 + 38x + 7x^2 + 71x^3 \\ 95 + 55x + 12x^2 + 87x^3 \\ 79 + 34x + 6x^2 + 34x^3 \\ 17 + 39x + 74x^2 + 43x^3 \end{bmatrix} \in R_q^4.$$

Exercise 12.3 (†) Let $n = 7$, $m = 14$, $q = 997$, and $B = 35$. Use the dual attack to solve the following $\mathrm{SIS}_2(n, m, q, B)$ instance:

$$A = \begin{bmatrix} 301 & 352 & 400 & 269 & 585 & 916 & 438 & 560 & 510 & 108 & 14 & 157 & 265 & 620 \\ 886 & 348 & 376 & 178 & 581 & 111 & 717 & 789 & 212 & 238 & 542 & 288 & 585 & 272 \\ 751 & 758 & 760 & 641 & 438 & 400 & 702 & 205 & 81 & 615 & 816 & 455 & 237 & 521 \\ 28 & 4 & 376 & 755 & 213 & 145 & 375 & 559 & 122 & 538 & 756 & 659 & 453 & 281 \\ 474 & 349 & 139 & 507 & 8 & 223 & 584 & 948 & 45 & 124 & 640 & 843 & 985 & 292 \\ 499 & 740 & 268 & 274 & 195 & 834 & 897 & 835 & 443 & 686 & 337 & 139 & 414 & 162 \\ 352 & 523 & 981 & 765 & 540 & 403 & 473 & 920 & 724 & 836 & 523 & 663 & 277 & 194 \end{bmatrix}.$$

Exercise 12.4 (†) Consider the polynomial ring $R_q = \mathbb{Z}_{97}[x]/(x^5 + 1)$. Let $k = 3$, $\ell = 4$ and $B = 50$. Use the dual attack to solve the following MSIS_2 instance:

$$A = \left[\begin{array}{cc} 77 + 96x + 10x^2 + 86x^3 + 58x^4 & 36 + 80x + 22x^2 + 44x^3 + 39x^4 \\ 60 + 39x + 43x^2 + 12x^3 + 55x^4 & 2 + 24x + 71x^2 + 45x^3 + 29x^4 \\ \hline 21 + 48x + 7x^2 + 33x^3 + 57x^4 & 65 + 16x + 93x^2 + 96x^3 + 71x^4 \\ 44 + 70x + 58x^2 + 25x^3 + 29x^4 & 73 + 57x + 40x^2 + 78x^3 + 88x^4 \\ \hline 25 + 44x + 67x^2 + 88x^3 + 5x^4 & 83 + 90x + 33x^2 + 88x^3 + 56x^4 \\ 81 + 85x + 3x^2 + 65x^3 + 36x^4 & 52 + 28x + 21x^2 + 51x^3 + 68x^4 \end{array}\right] \in R_q^{3\times 4}.$$

Note: The size of a polynomial $a = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1} \in \mathbb{Z}_q[x]/(x^n + 1)$ is $\|a\|_2 = \sqrt{a_0^2 + a_1^2 + \cdots + a_{n-1}^2}$. The size of a polynomial vector $b = (b_1, b_2, \ldots, b_k) \in R_q^k$ is $\|b\|_2 = \sqrt{\|b_1\|_2^2 + \|b_2\|_2^2 + \cdots + \|b_k\|_2^2}$.

Exercise 12.5 (*Babai's rounding algorithm*) Let $L = L(B) \subseteq \mathbb{Z}^n$ be a full-rank integer lattice, and let $t \in \mathbb{Z}^n$ be a target vector. Babai's rounding algorithm for approximate CVP proceeds as follows. First, compute an LLL-reduced basis $\tilde{B} = [\tilde{b}_1, \ldots, \tilde{b}_n]$ of L. Next, express $t = \sum_{i=1}^n t_i \tilde{b}_i$ with $t_i \in \mathbb{R}$. The algorithm then outputs the lattice vector $x = \sum_{i=1}^n \lfloor t_i \rceil \tilde{b}_i$.

(a) Suppose that L is given with an orthogonal basis. Show that in this case, Babai's rounding algorithm solves CVP exactly, i.e., with approximation factor $\gamma = 1$.

(b) (†) For each of the following pairs (B, t), apply Babai's rounding algorithm to find a lattice vector $x \in L(B)$ that is close to t.

(i)

$$B = \begin{bmatrix} -84 & 7 & 41 & -80 \\ 85 & -57 & -23 & 36 \\ -4 & 86 & -53 & -80 \\ 81 & -69 & 1 & 57 \end{bmatrix}, \quad t = \begin{bmatrix} -1576 \\ 1520 \\ -1276 \\ 1718 \end{bmatrix}.$$

(ii)

$$B = \begin{bmatrix} 10 & 14 & 14 & -81 & 93 \\ 72 & 43 & 71 & -61 & 39 \\ -30 & -35 & 24 & 6 & 29 \\ -49 & 25 & 36 & 36 & -83 \\ -22 & -34 & 29 & 64 & -27 \end{bmatrix}, \quad t = \begin{bmatrix} 275 \\ 209 \\ 26 \\ 204 \\ -311 \end{bmatrix}.$$

(iii)

$$B = \begin{bmatrix} 2 & -14 & -83 & 99 & 49 & -11 \\ -15 & 87 & 73 & 76 & 94 & -56 \\ 95 & -28 & 88 & 16 & -41 & -92 \\ -88 & -19 & 83 & 2 & -36 & -49 \\ -35 & -59 & 48 & 1 & 15 & -32 \\ -11 & -43 & 47 & 50 & 22 & -32 \end{bmatrix}, \quad t = \begin{bmatrix} -608 \\ -1323 \\ 686 \\ -809 \\ -408 \\ -530 \end{bmatrix}.$$

Exercise 12.6 (*Kannan's embedding technique*) Let $L = L(B) \subseteq \mathbb{Z}^n$ be a full-rank integer lattice, and let $t \in \mathbb{Z}^n$ be a target vector. Let $v \in L$ be a lattice vector that is closest to t, and define $d = v - t$. Let $Q = \|d\|$ and suppose that $Q < \lambda_1(L)/\sqrt{2}$.
(a) Prove that (d, Q) is a shortest nonzero vector in the lattice $L' \subseteq \mathbb{Z}^{n+1}$ with basis matrix

$$B' = \begin{bmatrix} B & -t \\ 0 & Q \end{bmatrix}.$$

(b) The result in (a) suggests the following algorithm which, given B and t, attempts to find a vector $v \in L(B)$ that is close to t.
For $M = 1, 2, 3, \ldots$, let L_M be the lattice with basis matrix

$$B_M = \begin{bmatrix} B & -t \\ 0 & M \end{bmatrix}.$$

Apply the LLL algorithm to find a reduced basis $\tilde{B}_M$ for L_M. If $\tilde{B}_M$ has a vector of the form (d, M), then output $v = d + t$ (and stop).
Show that $v \in L$, and explain why v can be expected to be close to t when the algorithm terminates.
(c) (†) For the (B, t) pairs listed in Exercise 12.5(b), use the method in (b) to find a vector $v \in L(B)$ that is close to t. Compare your results with the vectors found using Babai's rounding method.

Exercise 12.7 Use the lattice method described in §12.3 to "discover" the following known Machin-like formulae. You will need to experiment with increasing values of scaling factor Q and precision for the real number approximations.
(a) $\frac{\pi}{4} = 4 \arctan\left(\frac{1}{5}\right) - \arctan\left(\frac{1}{239}\right)$.
(b) $\frac{\pi}{4} = \arctan\left(\frac{1}{2}\right) + \arctan\left(\frac{1}{5}\right) + \arctan\left(\frac{1}{8}\right)$.
(c) $\frac{\pi}{4} = 12 \arctan\left(\frac{1}{38}\right) + 20 \arctan\left(\frac{1}{57}\right) + 7 \arctan\left(\frac{1}{239}\right) + 24 \arctan\left(\frac{1}{268}\right)$.

Exercise 12.8 (†) A real number α is said to be *algebraic* if there exists a nonzero polynomial $f(x) \in \mathbb{Z}[x]$ such that $f(\alpha) = 0$. Such a polynomial of degree n immediately yields an integer relation for $\{1, \alpha, \alpha^2, \ldots, \alpha^n\}$. For the following algebraic numbers α, use the lattice method described in §12.3 to find a polynomial $f(x) \in \mathbb{Z}[x]$ of degree n such that $f(\alpha) = 0$. You will need to experiment with increasing values of scaling factor Q and precision for the real number approximations.
(a) $\alpha = \sqrt{2}$, $n = 2$.
(b) $\alpha = \sqrt{2} + \sqrt{3}$, $n = 4$.
(c) $\alpha = \sqrt{2} + \sqrt{3} + \sqrt{5}$, $n = 8$.
(d) $\alpha = \sqrt{2} + \sqrt[3]{3}$, $n = 6$.
(e) $\alpha = \sqrt{2} + \sqrt[5]{3}$, $n = 10$.

Exercise 12.9 (†) Use the LLL algorithm to find small rational approximations with common denominator for the following lists of irrational numbers (see §12.4).
(a) $\sqrt{2}$, $\sqrt{3}$, $\sqrt{5}$, $\sqrt{7}$, $\sqrt{11}$ with $\varepsilon = \frac{1}{2}$.
(b) $\sqrt{2}$, $\sqrt{3}$, $\sqrt{5}$, $\sqrt{7}$, $\sqrt{11}$, $\sqrt{13}$, $\sqrt{17}$, $\sqrt{19}$, $\sqrt{23}$, $\sqrt{29}$ with $\varepsilon = \frac{2}{3}$.

Exercise 12.10 (†) Consider the RSA encryption scheme with public exponent $e = 3$ and 128-bit RSA modulus

$$N = 340282360475865017927401647640365095477.$$

A 25-bit symmetric key K is encrypted as follows: first, K is padded on the left with 102 bits, all equal to 1, and then the resulting integer is encrypted using RSA. The resulting ciphertext is

$$c = 450286776050842601917572720419479461 43.$$

Determine K.

Exercise 12.11 Let N be an RSA modulus and let the public exponent be $e = 3$. Consider two plaintext messages $m_1, m_2 \in [0, N-1]$, and define $r = m_2 - m_1$. Their corresponding ciphertexts are $c_1 = m_1^3 \bmod N$ and $c_2 = m_2^3 \bmod N$. Show that

$$m_1 = \frac{r(c_2 + 2c_1 - r^3)}{c_2 - c_1 + 2r^3} \bmod N.$$

Deduce that both m_1 and m_2 can be efficiently recovered given N, c_1, c_2 and r.

Exercise 12.12 (†) Consider the RSA encryption scheme with public exponent $e = 3$ and the following 1024-bit RSA modulus N:
179769313486231590772930519078902473361797697894230657273430081
157732675805500963122285464175907708342203798918733278441903442
174533480039066292309567731160938363406010393266636450860954680
068032681621010214994094051742659664966953120773373042312984042
921694668239847794914047597261248074460050141767190617789.

A plaintext M is encrypted by first padding it on the right with a randomly selected bitstring of length 64, and then applying the RSA encryption function. Suppose that a plaintext M is encrypted twice, yielding the following two ciphertexts:
9728155551827669716260410515847753897943974122807701291732618976953936425367993997986368652505647505445551016158158347063652583151364394888827892833502492235327260199094915120382332466537441754967687963783269036663366247669945624881011947482732855912092142012120959387488261956244656395893699473773989692179468139198670134075394263843025945117533834586201395253535669608529312295302448519258963302818017010719579132814215476307266883267717547160401474188428484208486563243946042047936676957893110117386344048776180629336629358304407181735901753369050731937809872861064476406164346918801600160175569256700755592467.
Determine M.

Exercise 12.13 The *subset sum problem* is the following. Fix positive integers k and n. Select $a_i \in_R [1, 2^k - 1]$ and $e_i \in_R \{0, 1\}$ for $1 \leq i \leq n$, and define $s = \sum_{i=1}^{n} e_i a_i$. Given $a = (a_1, \ldots, a_n)$ and s, the problem is to determine $x_1, \ldots, x_n \in \{0, 1\}$ such that $\sum_{i=1}^{n} a_i x_i = s$.
Note: The *density* of a is $d = n/k$. If $d \ll 1$, then one expects that the subset sum problem has a unique solution, namely $e = (e_1, \ldots, e_n)$.
Suppose now that $d < 0.6463$. Then the subset sum problem can be solved (with high probability) using the lattice-based method outlined below.
We will suppose, without loss of generality, that $0 < \sum_{i=1}^{n} e_i \leq \frac{1}{2}n$. Select an integer $N > \sqrt{n}$, and define the $(n+1)$-dimensional integer lattice $L = L(B)$ with basis matrix

$$B = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ Na_1 & Na_2 & Na_3 & \cdots & Na_n & Ns \end{bmatrix}_{(n+1)\times(n+1)}.$$

(a) Let $v = (e_1, \ldots, e_n, 0)$. Prove that $v \in L$ and $\|v\| \leq \sqrt{n/2}$.
(b) Using the Gaussian Heuristic, give an informal argument that v can be expected to be a lattice vector of shortest nonzero length. Hence, one can expect that v appears as a basis vector in an LLL-reduced basis of L.
(c) (†) Use this LLL-based method to solve the following instances of subset sum:
(i) $a = (491761, 618427, 742119, 526658, 100492, 607921, 945509, 570458, 654371, 381711)$, $s = 1899948$.
(ii) $a = (551012, 435750, 764101, 642331, 28000, 882705, 994024, 530149, 192346, 711419)$, $s = 2584681$.

Exercise 12.14 Let p be an odd prime. Lagrange's Theorem states that p can be written as a sum of four squares (and thus, so can every positive integer). This exercise develops an efficient method to compute this representation.
(a) Prove that there exist integers $u, v \in [0, p-1]$ such that $u^2 + v^2 \equiv -1 \pmod{p}$.
Hint: Show that the sets $U = \{u^2 + 1 : u \in \mathbb{Z}_p\}$ and $V = \{v^2 : v \in \mathbb{Z}_p\}$ have a non-trivial intersection.
(b) Let $u, v \in [0, p-1]$ with $u^2 + v^2 \equiv -1 \pmod{p}$. Consider the lattice $L \subseteq \mathbb{Z}^4$ with basis matrix

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ u & v & p & 0 \\ -v & u & 0 & p \end{bmatrix}.$$

Prove that every lattice vector $w \in L$ satisfies $\|w\|^2 \equiv 0 \pmod{p}$.
(c) Let $\tilde{B} = \{\tilde{b}_1, \tilde{b}_2, \tilde{b}_3, \tilde{b}_4\}$ be an $\frac{8}{9}$-LLL-reduced basis of B. Use (89) to prove that $\|\tilde{b}_1\|^2 = p$.
(d) Describe and analyze a polynomial-time algorithm for expressing a prime p as a sum of four squares.
(e) (†) Write each of the following primes as the sum of four squares.
(i) $p = 7919 \quad (u = 4, v = 3211)$.
(ii) $p = 59359 \quad (u = 4, v = 41632)$.
(iii) $p = 104743 \quad (u = 2, v = 59657)$.

13 The Number-Theoretic Transform

Contents

13.1 NTT definition 229
13.2 NTT computation 231
13.3 Dilithium NTT 236
13.4 Kyber NTT 240
13.5 Notes and further references 244
13.6 Exercises 244

The Number-Theoretic Transform (NTT) is a finite-field analogue of the Discrete Fourier Transform (DFT). It enables the multiplication of two polynomials in the ring $R_q = \mathbb{Z}_q[x]/(x^n+1)$ using $O(n \log n)$ $\mathbb{Z}_q$-operations, in contrast to the $O(n^2)$ $\mathbb{Z}_q$-operations required by naive polynomial multiplication. The NTT is used in Kyber, Dilithium and Falcon to accelerate polynomial multiplication in R_q, which is the dominant operation in Kyber and Dilithium. The NTT is introduced in §13.1 and §13.2, while the specific NTT variants used in Dilithium and Kyber are described in §13.3 and §13.4, respectively.

13.1 NTT definition

Let q be a prime number and let $n \geq 2$ be an integer. Denote by $\mathbb{Z}_q[x]$ the set of all polynomials in the indeterminate x with coefficients in $\mathbb{Z}_q$. The quotient ring $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ consists of all polynomials in $\mathbb{Z}_q[x]$ of degree at most $n-1$, with multiplication performed modulo the polynomial x^n+1. A polynomial $a(x) = a_0+a_1x+\cdots+a_{n-1}x^{n-1} \in R_q$ can be identified with its coefficient vector $(a_0, a_1, \ldots, a_{n-1})$, giving a natural one-to-one correspondence between R_q and the vector space $\mathbb{Z}_q^n$.

Classical polynomial multiplication. Let $a(x) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$ and $b(x) = b_0 + b_1x + \cdots + b_{n-1}x^{n-1}$ be elements of R_q. The classical method for computing their product $c(x) = a(x)b(x) \bmod (x^n + 1)$ in R_q proceeds as follows:

1. Compute the ordinary polynomial product in $\mathbb{Z}_q[x]$:
$$d(x) = a(x) \cdot b(x) = d_0 + d_1x + \cdots + d_{2n-2}x^{2n-2}.$$
2. Reduce $d(x)$ modulo $x^n + 1$ by substituting $x^n = -1$, $x^{n+1} = -x$, …, and $x^{2n-2} = -x^{n-2}$, yielding
$$(d_0 + d_1x + \cdots + d_{n-1}x^{n-1}) + (-d_n - d_{n+1}x - \cdots - d_{2n-2}x^{n-2}).$$
3. Add the two polynomials above to obtain $c(x) = c_0 + c_1x + \cdots + c_{n-1}x^{n-1}$.

Steps 1 and 3 require $O(n^2)$ and $O(n)$ arithmetic operations,[28] respectively, for an overall running time of $O(n^2)$. In contrast, the NTT allows multiplication in R_q to be carried out in time $O(n \log n)$.

[28] An arithmetic operation is an addition, subtraction, multiplication, or division in $\mathbb{Z}_q$.

Definition 13.1 Let $n = 2^k$, and let q be a prime such that $q - 1$ is divisible by $2n$. Let $\zeta \in \mathbb{Z}_q^*$ be an element of order[29] $2n$, so that $\zeta^n = -1$. The *Number-Theoretic Transform* (NTT) is the map $\text{NTT} : R_q \longrightarrow \mathbb{Z}_q^n$ defined by

$$\text{NTT}(a) = (a(\zeta), a(\zeta^3), a(\zeta^5), \ldots, a(\zeta^{2n-1})).$$

For notational convenience, we will often write $\hat{a}$ in place of $\text{NTT}(a)$.

The NTT of $a(x)$ evaluates the polynomial at the odd powers of ζ. These odd powers are precisely the elements of order $2n$ in $\mathbb{Z}_q^*$, also known as the primitive $(2n)$-th roots of unity modulo q. A key property of the NTT is that it is invertible; its inverse function $\text{NTT}^{-1} : \mathbb{Z}_q^n \longrightarrow R_q$ can be interpreted as polynomial interpolation. Consequently, $\text{NTT}^{-1}(\text{NTT}(a)) = a$ for all $a \in R_q$.

Addition and subtraction in R_q. Let $a(x), b(x) \in R_q$ and let $c(x) = a(x) \pm b(x)$. For any odd integer i, we have

$$c(\zeta^i) = a(\zeta^i) \pm b(\zeta^i),$$

and so $\hat{c} = \text{NTT}(c) = \hat{a} \pm \hat{b}$. Thus, addition and subtraction in NTT form is componentwise.

Fast multiplication in R_q. Let $a(x), b(x) \in R_q$ and let $c(x)$ be their product in R_q, i.e., $c(x) = a(x)b(x) \bmod (x^n + 1)$. There exists a polynomial $\ell(x) \in \mathbb{Z}_q[x]$ such that

$$c(x) = a(x)b(x) + \ell(x)(x^n + 1).$$

For any odd integer i, we have

$$c(\zeta^i) = a(\zeta^i)b(\zeta^i) + \ell(\zeta^i)(\zeta^{in} + 1) = a(\zeta^i)b(\zeta^i),$$

since $\zeta^{in} = (\zeta^n)^i = (-1)^i = -1$. Therefore

$$\text{NTT}(c) = \hat{c} = (a(\zeta)b(\zeta),\ a(\zeta^3)b(\zeta^3),\ \ldots,\ a(\zeta^{2n-1})b(\zeta^{2n-1})) = \hat{a} \circ \hat{b},$$

where $\hat{a} \circ \hat{b}$ denotes the componentwise multiplication of vectors in $\mathbb{Z}_q^n$.

In §13.2, we show that both NTT and NTT^{-1} can be computed in time $O(n \log n)$. This leads to an alternative procedure for multiplying $a(x)$ and $b(x)$ in R_q:

1. Compute $\hat{a} = \text{NTT}(a)$ and $\hat{b} = \text{NTT}(b)$.
2. Compute the componentwise product $\hat{c} = \hat{a} \circ \hat{b}$.
3. Recover $c = \text{NTT}^{-1}(\hat{c})$.

Since the componentwise multiplication requires only $O(n)$ operations, the overall running time of this method is $O(n \log n)$, a substantial improvement over the $O(n^2)$ running time of the classical method.

[29] The *order* of a nonzero element α in $\mathbb{Z}_q$ is the smallest positive integer t such that $\alpha^t \equiv 1 \pmod{q}$. For each positive divisor t of $q - 1$, there exists an element of order t in $\mathbb{Z}_q^*$.

13.2 NTT computation

Recall that $\mathrm{NTT}(a) = (a(\zeta), a(\zeta^3), \ldots, a(\zeta^{2n-1}))$. We can view the value $a(\zeta^i)$ as the remainder obtained by dividing $a(x)$ by the linear polynomial $x - \zeta^i$. The algorithm for computing the NTT uses the following result to recursively compute these remainders by first reducing modulo higher-degree factors.

Theorem 13.2 Let a, f, g be polynomials in $\mathbb{Z}_q[x]$. If f divides g, then

$$a(x) \bmod f(x) = (a(x) \bmod g(x)) \bmod f(x).$$

In other words, the remainder upon dividing $a(x)$ by $f(x)$ can be obtained by first dividing $a(x)$ by $g(x)$, and then dividing the resulting remainder by $f(x)$.

NTT computation for $n = 8$. To illustrate the algorithm, consider the case $n = 8$. Let q be a prime such that $q - 1$ is divisible by $2n = 16$, and let $\zeta \in \mathbb{Z}_q^*$ be an element of order 16. Then $\zeta^8 = -1$ and the polynomial ring of interest is

$$R_q = \mathbb{Z}_q[x]/(x^8 + 1).$$

The NTT computation is based on the identities (138), (139) and (140) derived as follows:

$$\begin{aligned}
x^8 + 1 &= x^8 - \zeta^8 \\
&= (x^4 - \zeta^4)(x^4 + \zeta^4) && (138)\\
&= (x^4 - \zeta^4)(x^4 - \zeta^{12}) \\
&= (x^2 - \zeta^2)(x^2 + \zeta^2)(x^2 - \zeta^6)(x^2 + \zeta^6) && (139)\\
&= (x^2 - \zeta^2)(x^2 - \zeta^{10})(x^2 - \zeta^6)(x^2 - \zeta^{14}) \\
&= (x - \zeta)(x + \zeta)(x - \zeta^5)(x + \zeta^5)(x - \zeta^3)(x + \zeta^3)(x - \zeta^7)(x + \zeta^7). && (140)
\end{aligned}$$

Let $a \in R_q$, so $a(x)$ is a polynomial in $\mathbb{Z}_q[x]$ of degree ≤ 7. The computation $\mathrm{NTT}(a)$ proceeds recursively down the levels of the *NTT computation tree* shown in Figure 13.1. More specifically, one first computes

$$a_0 = a \bmod (x^4 - \zeta^4) \quad \text{and} \quad a_1 = a \bmod (x^4 + \zeta^4), \tag{141}$$

then

$$a_{00} = a_0 \bmod (x^2 - \zeta^2), \quad a_{01} = a_0 \bmod (x^2 + \zeta^2), \tag{142}$$
$$a_{10} = a_1 \bmod (x^2 - \zeta^6), \quad a_{11} = a_1 \bmod (x^2 + \zeta^6), \tag{143}$$

and finally

$$a_{000} = a_{00} \bmod (x - \zeta), \quad a_{001} = a_{00} \bmod (x + \zeta), \tag{144}$$
$$a_{010} = a_{01} \bmod (x - \zeta^5), \quad a_{011} = a_{01} \bmod (x + \zeta^5), \tag{145}$$
$$a_{100} = a_{10} \bmod (x - \zeta^3), \quad a_{101} = a_{10} \bmod (x + \zeta^3), \tag{146}$$
$$a_{110} = a_{11} \bmod (x - \zeta^7), \quad a_{111} = a_{11} \bmod (x + \zeta^7). \tag{147}$$

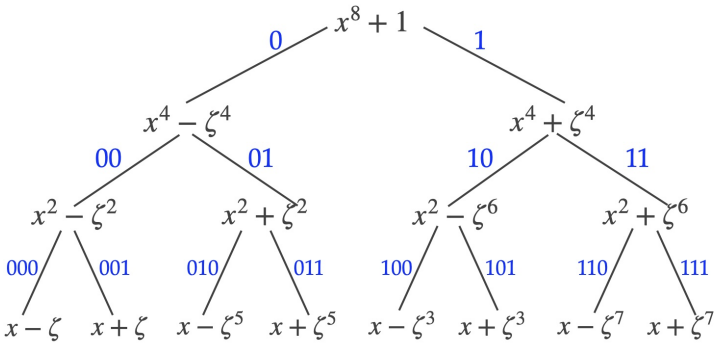


Figure 13.1: The NTT computation tree ($n = 8$).

The subscripts i in the polynomials a_i correspond to the labellings of the branches in Figure 13.1.

The polynomials a_i are computed as follows. Let $a_L(x)$ and $a_R(x)$ denote the left and right portions of $a(x)$. More precisely, we have $a(x) = a_L(x) + a_R(x)x^4$ where $a_L(x), a_R(x)$ each have degree ≤ 3. Then

$$a_0(x) = a(x) \bmod (x^4 - \zeta^4) = a_L(x) + \zeta^4 a_R(x), \tag{148}$$

and

$$a_1(x) = a(x) \bmod (x^4 + \zeta^4) = a_L(x) - \zeta^4 a_R(x), \tag{149}$$

so $a_0(x)$ and $a_1(x)$ can be computed simply by multiplying the right half $a_R(x)$ by ζ^4 and adding the result to $a_L(x)$ or subtracting it from $a_L(x)$, respectively.

Similarly, writing $a_0(x) = a_{0L}(x) + a_{0R}(x)x^2$ and $a_1(x) = a_{1L}(x) + a_{1R}(x)x^2$ where $a_{0L}, a_{0R}, a_{1L}, a_{1R}$ each have degree ≤ 1, we have

$$a_{00}(x) = a_0(x) \bmod (x^2 - \zeta^2) = a_{0L}(x) + \zeta^2 a_{0R}(x), \tag{150}$$
$$a_{01}(x) = a_0(x) \bmod (x^2 + \zeta^2) = a_{0L}(x) - \zeta^2 a_{0R}(x), \tag{151}$$
$$a_{10}(x) = a_1(x) \bmod (x^2 - \zeta^6) = a_{1L}(x) + \zeta^6 a_{1R}(x), \tag{152}$$
$$a_{11}(x) = a_1(x) \bmod (x^2 + \zeta^6) = a_{1L}(x) - \zeta^6 a_{1R}(x). \tag{153}$$

Finally, writing $a_{00}(x) = a_{00L} + a_{00R}x$, $a_{01}(x) = a_{01L} + a_{01R}x$, $a_{10}(x) = a_{10L} + a_{10R}x$, and $a_{11}(x) = a_{11L} + a_{11R}x$, where each a_{ijL} and a_{ijR} is in $\mathbb{Z}_q$, we have

$$a_{000} = a_{00L} + \zeta a_{00R}, \quad a_{001} = a_{00L} - \zeta a_{00R}, \tag{154}$$
$$a_{010} = a_{01L} + \zeta^5 a_{01R}, \quad a_{011} = a_{01L} - \zeta^5 a_{01R}, \tag{155}$$
$$a_{100} = a_{10L} + \zeta^3 a_{10R}, \quad a_{101} = a_{10L} - \zeta^3 a_{10R}, \tag{156}$$
$$a_{110} = a_{11L} + \zeta^7 a_{11R}, \quad a_{111} = a_{11L} - \zeta^7 a_{11R}. \tag{157}$$

Thus,

$$\mathrm{NTT}(a) = (a_{000}, a_{100}, a_{010}, a_{110}, a_{001}, a_{101}, a_{011}, a_{111}). \tag{158}$$

NTT running time. Let $n = 2^k$, and let $T(n)$ denote the time required to evaluate NTT. We have the recursive formula

$$T(n) = 2T(n/2) + 3n/2.$$

Indeed, at each level of the NTT computation tree, a total of $\frac{n}{2}$ multiplications and n additions are required to carry out the reduction steps. Moving to the next level doubles the number of reduction steps, but each reduction operates on polynomials of half the degree. Solving this recurrence yields $T(n) = O(n \log n)$, which is the running time of the NTT algorithm.

NTT^{-1} computation for $n = 8$. The inverse NTT is computed by traversing the NTT computation tree in reverse, starting from the leaves and working back to the root. Thus, given

$$\text{NTT}(a) = (a_{000}, a_{100}, a_{010}, a_{110}, a_{001}, a_{101}, a_{011}, a_{111}),$$

we first combine a_{000} and a_{001} to recover $a_{00}(x)$, then combine a_{010} and a_{011} to recover $a_{01}(x)$, and so on, until the original polynomial $a(x)$ is reconstructed from $a_0(x)$ and $a_1(x)$.

As an illustration, recall that $a_0(x) = a_L(x) + \zeta^4 a_R(x)$ and $a_1(x) = a_L(x) - \zeta^4 a_R(x)$. Solving for $a_L(x)$ and $a_R(x)$ gives

$$a_L(x) = \frac{a_0(x) + a_1(x)}{2} \quad \text{and} \quad a_R(x) = \frac{a_0(x) - a_1(x)}{2\zeta^4},$$

and hence $a(x) = a_L(x) + a_R(x)x^4$ can be recovered from $a_0(x)$ and $a_1(x)$. Applying this procedure recursively yields the full set of inverse NTT formulas, which are listed below. For notational simplicity, the argument "(x)" is omitted.

$$\begin{aligned}
a_{00} &= a_{00L} + a_{00R}x, \quad \text{where } a_{00L} = (a_{000} + a_{001})/2 \text{ and } a_{00R} = (a_{000} - a_{001})/(2\zeta),\\
a_{01} &= a_{01L} + a_{01R}x, \quad \text{where } a_{01L} = (a_{010} + a_{011})/2 \text{ and } a_{01R} = (a_{010} - a_{011})/(2\zeta^5),\\
a_{10} &= a_{10L} + a_{10R}x, \quad \text{where } a_{10L} = (a_{100} + a_{101})/2 \text{ and } a_{10R} = (a_{100} - a_{101})/(2\zeta^3),\\
a_{11} &= a_{11L} + a_{11R}x, \quad \text{where } a_{11L} = (a_{110} + a_{111})/2 \text{ and } a_{11R} = (a_{110} - a_{111})/(2\zeta^7),\\
a_0 &= a_{0L} + a_{0R}x^2, \quad \text{where } a_{0L} = (a_{00} + a_{01})/2 \text{ and } a_{0R} = (a_{00} - a_{01})/(2\zeta^2),\\
a_1 &= a_{1L} + a_{1R}x^2, \quad \text{where } a_{1L} = (a_{10} + a_{11})/2 \text{ and } a_{1R} = (a_{10} - a_{11})/(2\zeta^6),\\
a &= a_L + a_R x^4, \quad \text{where } a_L = (a_0 + a_1)/2 \text{ and } a_R = (a_0 - a_1)/(2\zeta^4).
\end{aligned}$$

The inverse NTT map can also be computed in time $O(n \log n)$.

Example 13.3 (*NTT computation*) Consider $q = 241$ and $n = 8$, whence the polynomial ring of interest is $R_q = \mathbb{Z}_{241}[x]/(x^8 + 1)$. Note that $2n = 16$ divides $q - 1 = 240 = 16 \times 15$. The element $\zeta = 76 \in \mathbb{Z}_{241}^*$ has order 16. Let

$$a(x) = 71 + 152x + 4x^3 + 74x^5 + 133x^6 + 200x^7 \in R_q$$

and

$$b(x) = 199 + x^2 + x^3 + 11x^4 + 44x^5 + 165x^6 + 114x^7 \in R_q.$$

We wish to compute $c(x) = a(x)b(x) \bmod (x^8 + 1)$ using NTT.

Let us first compute $\hat{a} = \text{NTT}(a)$. We have

$$a_L(x) = 71 + 152x + 4x^3 \quad \text{and} \quad a_R(x) = 74x + 133x^2 + 200x^3.$$

We compute

$$a_0(x) = a_L(x) + \zeta^4 a_R(x) = 71 + 68x + 77x^2 + 31x^3$$

and

$$a_1(x) = a_L(x) - \zeta^4 a_R(x) = 71 + 236x + 164x^2 + 218x^3,$$

then

$$a_{00}(x) = 178 + 61x,\ a_{01}(x) = 205 + 75x,\ a_{10}(x) = 212 + 203x,\ a_{11}(x) = 171 + 28x,$$

and finally

$$\begin{aligned} a_{000}(x) = 235,\ \ a_{001}(x) = 121,\ \ a_{010}(x) = 131,\ \ a_{011}(x) = 38,\\ a_{100}(x) = 180,\ \ a_{101}(x) = 3,\ \ a_{110}(x) = 196,\ \ a_{111}(x) = 146. \end{aligned}$$

Thus,

$$\hat{a} = (235, 180, 131, 196, 121, 3, 38, 146).$$

Similarly, we compute

$$b_0(x) = b_L(x) + \zeta^4 b_R(x) = 180 + 165x + 198x^2 + 67x^3$$

and

$$b_1(x) = b_L(x) - \zeta^4 b_R(x) = 218 + 76x + 45x^2 + 176x^3,$$

then

$$b_{00}(x) = 42 + 111x,\ b_{01}(x) = 77 + 219x,\ b_{10}(x) = 73 + 98x,\ b_{11}(x) = 122 + 54x,$$

and finally

$$\begin{aligned} b_{000}(x) = 43,\ \ b_{001}(x) = 41,\ \ b_{010}(x) = 73,\ \ b_{011}(x) = 81\\ b_{100}(x) = 16,\ \ b_{101}(x) = 130,\ \ b_{110}(x) = 153,\ \ b_{111}(x) = 91. \end{aligned}$$

Thus,

$$\hat{b} = (43, 16, 73, 153, 41, 130, 81, 91),$$

and

$$\hat{c} = \hat{a} \circ \hat{b} = (224, 229, 164, 104, 141, 149, 186, 31).$$

Finally, we compute (details omitted)

$$c(x) = \text{NTT}^{-1}(\hat{c}) = 33 + 3x + 239x^2 + 37x^3 + 71x^4 + 43x^5 + 143x^6 + 69x^7.$$

The general case. Let $n = 2^k$. Let q be a prime such that $q - 1$ is divisible by $2n$, and let $\zeta \in \mathbb{Z}_q^*$ be an element of order $2n$. The NTT tree (cf. Figure 13.2) is a complete binary tree of height k, where the root is at height 0. Each node at height $d \in \{0, \ldots, k\}$ in the tree is indexed by its path $b_1 b_2 \ldots b_d \in \{0,1\}^d$ from the root, where 0 means left and 1 means right.

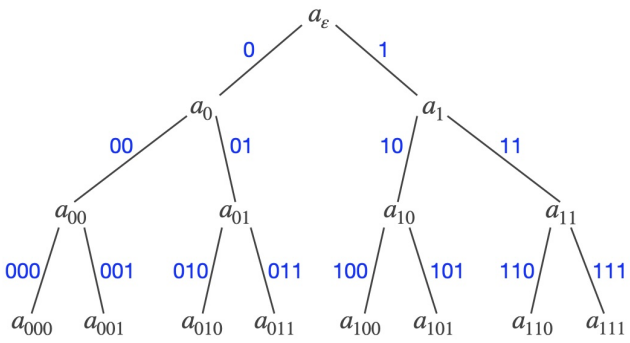


Figure 13.2: The NTT tree ($n = 8$).

For $a \in R_q$, $\mathrm{NTT}(a)$ can be computed by a top-to-bottom, left-to-right traversal of the NTT tree; see Figure 13.3(a). The root a_ε is defined to be $a(x)$. The children a_{i0} and a_{i1} of a node a_i at height $d \in \{0, 1, \ldots, k-1\}$ are computed as follows. Let $a_i(x) = a_{iL}(x) + x^{2^{k-d-1}} a_{iR}(x)$ where $a_{iL}(x)$ and $a_{iR}(x)$ are polynomials of degree less than 2^{k-d-1}. Furthermore, define

$$j = \mathrm{bitrev}_{k+1}(2^{d+1} + 2i), \tag{159}$$

where the binary string i is interpreted as an integer and $\mathrm{bitrev}_s(t)$ denotes the integer obtained by reversing the s-bit binary representation of $t \in [0, 2^s - 1]$. For example, $\mathrm{bitrev}_8(113) = 142$ since the 8-bit binary representation of 113 is 01110001 whose reversal is 10001110. Then we have

$$a_{i0}(x) = a_i(x) \bmod (x^{2^{k-d-1}} - \zeta^j) = a_{iL}(x) + \zeta^j a_{iR}(x) \tag{160}$$

and

$$a_{i1}(x) = a_i(x) \bmod (x^{2^{k-d-1}} + \zeta^j) = a_{iL}(x) - \zeta^j a_{iR}(x). \tag{161}$$

Similarly, NTT^{-1} can be computed by a bottom-to-top, left-to-right traversal of the NTT tree; see Figure 13.3(b). The formulas for computing the node $a_i(x) = a_{iL}(x) + x^{2^{k-d-1}} a_{iR}(x)$ at height $d \in \{0, 1, \ldots, k-1\}$ from its two children a_{i0} and a_{i1} are

$$a_{iL}(x) = (a_{i0} + a_{i1})/2 \quad \text{and} \quad a_{iR}(x) = (a_{i0} - a_{i1})/(2\zeta^j), \tag{162}$$

where j is as defined in (159).

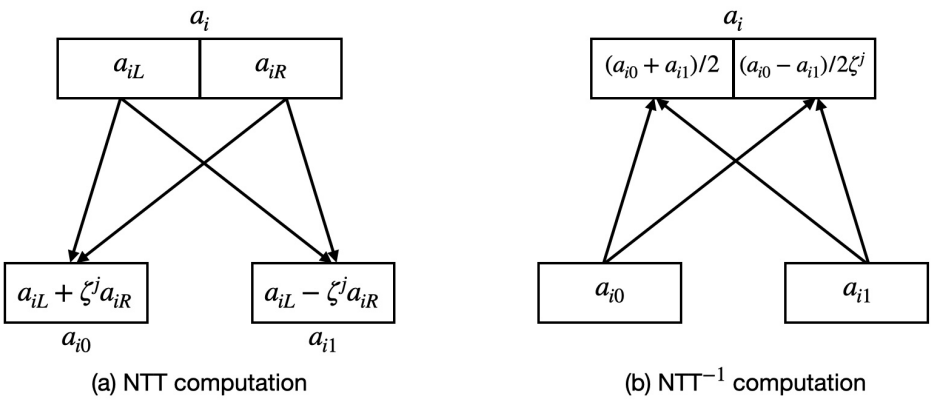


Figure 13.3: The "butterfly" formulas for computing NTT and NTT^{-1}.

13.3 Dilithium NTT

The FIPS 204 standard that specifies the Dilithium signature scheme fixes the parameters $n = 256$ and $q = 2^{23} - 2^{13} + 1 = 8,380,417$, chosen so that $2n$ divides $q-1$, and the order-$2n$ element $\zeta = 1753$ in $\mathbb{Z}_q^*$. The associated Dilithium ring $R_q = \mathbb{Z}_q[x]/(x^{256} + 1)$ consists of polynomials of degree at most 255 with coefficients in $\mathbb{Z}_{8380417}$.

Dilithium NTT definition. The NTT used in Dilithium, as specified in FIPS 204, differs slightly from the transform introduced in Definition 13.1. In that earlier definition, the components of $\hat{a} = \text{NTT}(a)$ correspond to the consecutive odd powers of ζ, namely $\hat{a}_i = a(\zeta^{2i-1})$ for $1 \le i \le n$. In contrast, the Dilithium NTT permutes these powers of ζ so that they appear in the same order as the leaves of the underlying NTT computation tree. This avoids having to permute the values corresponding to the leaves of the NTT tree as was done in (158).

This ordering of leaves can be described succinctly using bit reversal. Suppose that $n = 2^k$. For an integer $v \in [0, n-1]$, let $\text{brv}(v) = \text{bitrev}_k(v)$ denote the k-bit integer obtained by reversing the k-bit binary representation of v.

Definition 13.4 Let $n = 2^k$, and let q be a prime such that $q - 1$ is divisible by $2n$. Let $\zeta \in \mathbb{Z}_q^*$ be an element of order $2n$, and define

$$\zeta_i = \zeta^{2\text{brv}(i)+1} \bmod q \quad \text{for } 0 \le i \le n-1.$$

The *Dilithium NTT* is the map $\text{NTT} : R_q \longrightarrow \mathbb{Z}_q^n$ given by

$$\text{NTT}(a) = (a(\zeta_0), a(\zeta_1), a(\zeta_2), \ldots, a(\zeta_{n-2}), a(\zeta_{n-1})).$$

A notable feature of the NTT and its inverse is that both can be computed *in-place*, requiring only minimal auxiliary storage. This is especially advantageous for hardware

Algorithm 41 NTT(w)

Computes the NTT.

Input: Polynomial $w(X) = \sum_{j=0}^{255} w_j X^j \in R_q$.
Output: $\hat{w} = (\hat{w}[0], \ldots, \hat{w}[255]) \in T_q$.

1: **for** j **from** 0 **to** 255 **do**
2: $\quad \hat{w}[j] \leftarrow w_j$
3: **end for**
4: $m \leftarrow 0$
5: $len \leftarrow 128$
6: **while** $len \geq 1$ **do**
7: $\quad start \leftarrow 0$
8: $\quad$ **while** $start < 256$ **do**
9: $\quad\quad m \leftarrow m + 1$
10: $\quad\quad z \leftarrow \text{zetas}[m]$ // $z = \zeta^{\text{brv}(m)} \bmod q$
11: $\quad\quad$ **for** j **from** $start$ **to** $start + len - 1$ **do**
12: $\quad\quad\quad t \leftarrow (z \cdot \hat{w}[j + len]) \bmod q$
13: $\quad\quad\quad \hat{w}[j + len] \leftarrow (\hat{w}[j] - t) \bmod q$
14: $\quad\quad\quad \hat{w}[j] \leftarrow (\hat{w}[j] + t) \bmod q$
15: $\quad\quad$ **end for**
16: $\quad\quad start \leftarrow start + 2 \cdot len$
17: $\quad$ **end while**
18: $\quad len \leftarrow \lfloor len/2 \rfloor$
19: **end while**
20: **return** $\hat{w}$

Algorithm 42 NTT$^{-1}(\hat{w})$

Computes the inverse of the NTT.

Input: $\hat{w} = (\hat{w}[0], \ldots, \hat{w}[255]) \in T_q$.
Output: Polynomial $w(X) = \sum_{j=0}^{255} w_j X^j \in R_q$.

1: **for** j **from** 0 **to** 255 **do**
2: $\quad w_j \leftarrow \hat{w}[j]$
3: **end for**
4: $m \leftarrow 256$
5: $len \leftarrow 1$
6: **while** $len < 256$ **do**
7: $\quad start \leftarrow 0$
8: $\quad$ **while** $start < 256$ **do**
9: $\quad\quad m \leftarrow m - 1$
10: $\quad\quad z \leftarrow -zetas[m]$ // $z = -\zeta^{\text{brv}(m)} \bmod q$
11: $\quad\quad$ **for** j **from** $start$ **to** $start + len - 1$ **do**
12: $\quad\quad\quad t \leftarrow w_j$
13: $\quad\quad\quad w_j \leftarrow (t + w_{j+len}) \bmod q$
14: $\quad\quad\quad w_{j+len} \leftarrow (t - w_{j+len}) \bmod q$
15: $\quad\quad\quad w_{j+len} \leftarrow (z \cdot w_{j+len}) \bmod q$
16: $\quad\quad$ **end for**
17: $\quad\quad start \leftarrow start + 2 \cdot len$
18: $\quad$ **end while**
19: $\quad len \leftarrow 2 \cdot len$
20: **end while**
21: $f \leftarrow 8347681$ // $f = -256^{-1} \bmod q$
22: **for** j **from** 0 **to** 255 **do**
23: $\quad w_j \leftarrow (f \cdot w_j) \bmod q$
24: **end for**
25: **return** w

Figure 13.4: NTT and NTT^{-1} pseudocode from the FIPS 204 Dilithium specification.

implementations. The in-place feature and compactness of the NTT and NTT^{-1} code are evident from the pseudocode in Figure 13.4.

We next illustrate the in-place computation for the case $n = 8$.

In-place NTT computation. Let $n = 8$ and recall that $\zeta^8 = -1$. The brv function is:

i	0	1	2	3	4	5	6	7
$\text{brv}(i)$	0	4	2	6	1	5	3	7

The Dilithium NTT of a polynomial $A(x) = A_0 + A_1x + \cdots + A_7x^7 \in R_q$ is

$$\text{NTT}(A) = (A(\zeta), A(\zeta^9), A(\zeta^5), A(\zeta^{13}), A(\zeta^3), A(\zeta^{11}), A(\zeta^7), A(\zeta^{15})).$$

Here are the steps of the computation of $\text{NTT}(A)$:

1. A length-8 array is initialized with the coefficients of $A(x)$:
$$[A_0, A_1, A_2, A_3, A_4, A_5, A_6, A_7].$$
2. The array entries are divided into two halves $[a_L, a_R]$, and then updated to $[a_0, a_1]$ where $a_0 = a_L + t$ and $a_1 = a_L - t$ and $t = \zeta^4 a_R$; see equations (148) and (149).
3. Next, the array entries are divided into four pieces and updated to $[a_{00}, a_{01}, a_{10}, a_{11}]$; see equations (150)–(153).
4. Finally the array entries are updated to $[a_{000}, a_{001}, a_{010}, a_{011}, a_{100}, a_{101}, a_{110}, a_{111}]$ according to equations (154)–(157), yielding $\text{NTT}(A)$.

Each step updates the array in place, requiring no additional arrays beyond the original length-n input.

Dilithium key generation. To generate a Dilithium key pair (Algorithm 7.9), a user randomly selects a $k \times \ell$ matrix of polynomials $A \in_R R_q^{k\times\ell}$, together with two polynomial vectors $s_1 \in_R S_\eta^\ell$ and $s_2 \in_R S_\eta^k$. Here, S_η denotes the set of all polynomials in R_q whose coefficients lie in the interval $[-\eta, \eta]$. The user then computes the $k\times 1$ polynomial vector $t = As_1 + s_2$. Her public key is (A, t), while her private key is (s_1, s_2).

NTT is used to accelerate the $k\ell$ polynomial multiplications needed to compute the matrix-vector product As_1. To minimize the number of NTT and NTT^{-1} applications, the matrix A is generated directly in NTT form. Specifically, each entry of $\hat{A}$ is a randomly selected vector $\hat{a} \in_R \mathbb{Z}_q^n$; the corresponding polynomial $a = \text{NTT}^{-1}(\hat{a}) \in R_q$ is not needed and so is never explicitly computed.

Key generation proceeds as follows.

1. Select $\hat{A} \in_R (\mathbb{Z}_q^n)^{k\times\ell}$.
2. Select $s_1 \in_R S_\eta^\ell$ and $s_2 \in_R S_\eta^k$ and compute $\hat{s}_1 = \text{NTT}(s_1)$.
(Here, $\text{NTT}(s_1)$ means applying the NTT to each polynomial in s_1.)

3. Compute $\hat{b} = \hat{A} \circ \hat{s}_1$.
   (Here, $\hat{A} \circ \hat{s}_1$ means the matrix-vector product of $\hat{A}$ and $\hat{s}_1$, where each multiplication within the product is performed componentwise.)
4. Compute $b = \text{NTT}^{-1}(\hat{b})$.
5. Compute $t = b + s_2$.

Overall, NTT is applied only ℓ times (to compute $\hat{s}_1$ in step 2), while NTT^{-1} is applied k times (to compute b in step 4). The $k\ell$ multiplications needed in step 3 are fast componentwise multiplications in $\mathbb{Z}_q^n$.

Dilithium signature generation. The dominant computation in signature generation (Algorithm 7.9) is the evaluation of $w = Ay$, where A is the user's public key matrix and y is a randomly selected $\ell \times 1$ polynomial vector in $\tilde{S}_{\gamma_1}^{\ell}$. Here, $\tilde{S}_{\gamma_1}$ denotes the set of all polynomials in R_q whose coefficients lie in the interval $(-\gamma_1, \gamma_1]$. This computation is performed using the NTT as follows:

1. Select $y \in_R \tilde{S}_{\gamma_1}^{\ell}$ and compute $\hat{y} = \text{NTT}(y)$.
2. Compute $\hat{w} = \hat{A} \circ \hat{y}$.
3. Compute $w = \text{NTT}^{-1}(\hat{w})$.

Each of the ℓ NTT applications in step 1 and the k applications of NTT^{-1} in step 3 requires $\frac{1}{2} n \log_2 n$ $\mathbb{Z}_q$-multiplications[30]. The $k\ell$ componentwise multiplications in step 2 each require n $\mathbb{Z}_q$-multiplications. Consequently, the total cost is $\frac{1}{2}(\ell + k) n \log_2 n + k\ell n$ multiplications in $\mathbb{Z}_q$. In contrast, computing Ay using conventional polynomial multiplication would require n^2 $\mathbb{Z}_q$-multiplications for each of the $k\ell$ products, for a total of $k\ell n^2$ multiplications in $\mathbb{Z}_q$.

The ML-DSA-87 Dilithium parameter set specified in FIPS 204 uses $k = 8$ and $\ell = 7$. For these parameters (and with $n = 256$), computing w using the NTT requires approximately $2^{14.9}$ $\mathbb{Z}_q$-multiplications, whereas a naive approach using conventional polynomial multiplication would require approximately $2^{21.8}$ $\mathbb{Z}_q$-multiplications. Thus, the use of NTT yields a speedup of roughly a factor $2^{6.9} \approx 119$, representing a very substantial performance improvement.

Remark 13.5 (*Falcon NTT*) The Falcon signature scheme (FN-DSA) specifies polynomial rings $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ for two parameter sets. Both rings use the 14-bit prime $q = 12289$; the two security levels correspond to $n = 512$ and $n = 1024$. Since $2n$ divides $q - 1 = 2^{12} \times 3$, a Number-Theoretic Transform of the same type as the Dilithium-NTT can be employed for efficient polynomial multiplication in the Falcon rings.

[30] We are not counting additions, subtractions, and divisions by 2 in $\mathbb{Z}_q$, since these operations are less costly than a multiplication in $\mathbb{Z}_q$.

13.4 Kyber NTT

The Kyber NTT parameters are $n = 256$ and $q = 3329$, so the associated polynomial ring is $R_q = \mathbb{Z}_{3329}[x]/(x^{256} + 1)$. Since $q - 1 = 3328 = 2^8 \times 13$, the integer $q - 1$ is divisible by n but not by $2n$. As a result the standard NTT algorithm must be slightly modified. Specifically, the height of the NTT computation tree is reduced by one level. The modified transform is called NTT_2.

NTT_2 **definition.** Let $n = 2^k$, and let q be a prime such that $q - 1$ is divisible by n but not by $2n$. Let $\zeta \in \mathbb{Z}_q^*$ be an element of order n, so that $\zeta^{n/2} = -1$.

Lemma 13.6 For every odd integer $i \in [1, n-1]$, the polynomial $x^2 - \zeta^i$ is irreducible over $\mathbb{Z}_q$.

Proof: Suppose that $x^2 - \zeta^i$ were reducible over $\mathbb{Z}_q$. Then it would have a root c in $\mathbb{Z}_q$, implying $c^2 = \zeta^i$. Raising both sides to the n-th power gives $(c^2)^n = (\zeta^i)^n = (\zeta^n)^i = 1$, so the order of c divides $2n$. On the other hand, $c^n = (c^2)^{n/2} = (\zeta^i)^{n/2} = (\zeta^{n/2})^i = (-1)^i = -1$, which shows that the order of c does not divide n. Therefore, c has order $2n$, which is impossible since $2n$ does not divide $q-1$. This contradiction proves that $x^2 - \zeta^i$ is irreducible. □

For each odd integer $i \in [1, n-1]$, define the *quadratic ring of polynomials*

$$Q_i = \mathbb{Z}_q[x]/(x^2 - \zeta^i).$$

Elements of Q_i are polynomials of the form $c_0 + c_1 x$ with coefficients in $\mathbb{Z}_q$; such a polynomial can be represented by its coefficient pair $(c_0, c_1) \in \mathbb{Z}_q^2$. Multiplication is performed modulo $x^2 - \zeta^i$. The addition, subtraction and multiplication operations in Q_i are the following: if $a(x) = a_0 + a_1 x$, $b(x) = b_0 + b_1 x \in Q_i$, then

$$a(x) \pm b(x) = (a_0 \pm b_0) + (a_1 \pm b_1)x,$$

and

$$a(x) \cdot b(x) = (a_0 b_0 + a_1 b_1 \zeta^i) + (a_0 b_1 + a_1 b_0)x.$$

Define the ring T_q as the product of the $\frac{n}{2}$ quadratic rings $Q_1, Q_3, \ldots, Q_{n-1}$:

$$T_q = Q_1 \times Q_3 \times \cdots \times Q_{n-1}.$$

Elements of T_q can be represented as vectors in $\mathbb{Z}_q^n$.

Definition 13.7 Let $n = 2^k$, and let q be a prime such that $q - 1$ is divisible by n but not by $2n$. Let $\zeta \in \mathbb{Z}_q^*$ be an element of order n, and let $T_q = Q_1 \times Q_3 \times \cdots \times Q_{n-1}$ where $Q_i = \mathbb{Z}_q[x]/(x^2 - \zeta^i)$. The *Quadratic Number-Theoretic Transform* (NTT_2) is the map $\text{NTT}_2 : R_q \longrightarrow T_q$ defined by

$$\text{NTT}_2(a) = (a \bmod (x^2 - \zeta),\ a \bmod (x^2 - \zeta^3),\ \ldots\ , a \bmod (x^2 - \zeta^{n-1})).$$

For notational convenience, we will often write $\hat{a}$ in place of $\text{NTT}_2(a)$.

The map $\text{NTT}_2 : R_q \longrightarrow T_q$ is a ring isomorphism and is therefore invertible.

Addition and subtraction in R_q. Let $a(x), b(x) \in R_q$, and let $c(x) = a(x) \pm b(x)$. For any odd integer i, we have

$$c \bmod (x^2 - \zeta^i) = a \bmod (x^2 - \zeta^i) \pm b \bmod (x^2 - \zeta^i),$$

and so $\hat{c} = \mathrm{NTT}_2(c) = \hat{a} \pm \hat{b}$. Consequently, addition and subtraction in NTT_2 form are performed componentwise.

Fast multiplication in R_q. Let $a(x), b(x) \in R_q$, and let $c(x)$ be their product in R_q, i.e., $c(x) = a(x)b(x) \bmod (x^n + 1)$. There exists a polynomial $\ell(x) \in \mathbb{Z}_q[x]$ satisfying

$$c(x) = a(x)b(x) + \ell(x)(x^n + 1).$$

For any odd integer i, we have

$$\begin{aligned} x^n + 1 \bmod (x^2 - \zeta^i) &= (x^2)^{n/2} + 1 \bmod (x^2 - \zeta^i) \\ &= (\zeta^i)^{n/2} + 1 = (\zeta^{n/2})^i + 1 = (-1)^i + 1 = 0 \end{aligned}$$

and so

$$\begin{aligned} c(x) \bmod (x^2 - \zeta^i) &= a(x)b(x) + \ell(x)(x^n + 1) \bmod (x^2 - \zeta^i) \\ &= a(x)b(x) \bmod (x^2 - \zeta^i). \end{aligned}$$

Thus, $\mathrm{NTT}_2(c)$ is

$$\begin{aligned} \hat{c} &= (a(x)b(x) \bmod (x^2 - \zeta),\ a(x)b(x) \bmod (x^2 - \zeta^3),\ \ldots,\ a(x)b(x) \bmod (x^2 - \zeta^{n-1})) \\ &= \hat{a} \circ \hat{b}, \end{aligned}$$

where $\hat{a} \circ \hat{b}$ denotes componentwise multiplication in T_q.

As with the standard NTT, both NTT_2 and NTT_2^{-1} can be computed in time $O(n \log n)$. This yields an alternative procedure for multiplying $a(x)$ and $b(x)$ in R_q:

1. Compute $\hat{a} = \mathrm{NTT}_2(a)$ and $\hat{b} = \mathrm{NTT}_2(b)$.
2. Compute the componentwise product $\hat{c} = \hat{a} \circ \hat{b}$.
3. Recover $c = \mathrm{NTT}_2^{-1}(\hat{c})$.

Since the componentwise multiplication requires only $O(n)$ operations, the overall complexity is $O(n \log n)$, a substantial improvement over the $O(n^2)$ running time of the classical method.

We next present an explicit example of an NTT_2 computation in the case $n = 8$. Full descriptions of $O(n \log n)$-time algorithms for computing NTT_2 and its inverse NTT_2^{-1} for the general case $n = 2^k$ are left to the reader (Exercise 13.5).

NTT_2 computation for $n = 8$. To illustrate the transform, consider $n = 8$. Let q be a prime such that $q - 1$ is divisible by $n = 8$ but not by $2n = 16$, and let $\zeta \in \mathbb{Z}_q^*$ be an element of order 8. Then $\zeta^4 = -1$. The polynomial ring under consideration is $R_q = \mathbb{Z}_q[x]/(x^8+1)$, and the transform is $\mathrm{NTT}_2 : R_q \longrightarrow T_q$, where $T_q = Q_1 \times Q_3 \times Q_5 \times Q_7$ and $Q_i = \mathbb{Z}_q[x]/(x^2 - \zeta^i)$.

The NTT_2 computation is based on the identities (163) and (164) derived as follows:

$$\begin{aligned} x^8 + 1 &= x^8 - \zeta^4 \\ &= (x^4 - \zeta^2)(x^4 + \zeta^2) \qquad (163) \\ &= (x^4 - \zeta^2)(x^4 - \zeta^6) \\ &= (x^2 - \zeta)(x^2 + \zeta)(x^2 - \zeta^3)(x^2 + \zeta^3). \qquad (164) \end{aligned}$$

Let $a \in R_q$, so that $a(x) \in \mathbb{Z}_q[x]$ has degree at most 7. The computation of $\mathrm{NTT}_2(a)$ proceeds recursively down the levels of the NTT_2 *tree* shown in Figure 13.5. Specifically, the first step is to compute

$$a_0 = a \bmod (x^4 - \zeta^2) \quad \text{and} \quad a_1 = a \bmod (x^4 + \zeta^2), \qquad (165)$$

and then

$$a_{00} = a_0 \bmod (x^2 - \zeta), \quad a_{01} = a_0 \bmod (x^2 + \zeta), \qquad (166)$$

$$a_{10} = a_1 \bmod (x^2 - \zeta^3), \quad a_{11} = a_1 \bmod (x^2 + \zeta^3). \qquad (167)$$

The final output of the transform is then

$$\mathrm{NTT}_2(a) = (a_{00}, a_{10}, a_{01}, a_{11}). \qquad (168)$$

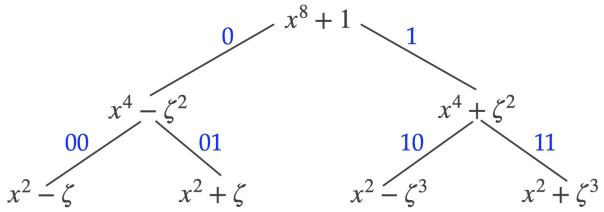


Figure 13.5: The NTT_2 tree ($n = 8$).

Example 13.8 (NTT_2 *computation*) Let $q = 233$ and $n = 8$, so the polynomial ring of interest is $R_q = \mathbb{Z}_{233}[x]/(x^8 + 1)$. Note that n divides $q - 1 = 8 \times 29$ but $2n = 16$ does not. The element $\zeta = 12 \in \mathbb{Z}_{233}^*$ has order 8. Consider the polynomials

$$a(x) = 92 + 152x + 57x^3 + 29x^5 + 133x^6 + 200x^7 \in R_q$$

and

$$b(x) = 6 + 45x^2 + 112x^3 + 4x^4 + 8x^5 + 165x^6 + 212x^7 \in R_q.$$

Our goal is to compute $c(x) = a(x)b(x) \bmod (x^8 + 1)$ using NTT_2.

We begin by computing $\hat{a} = \mathrm{NTT}_2(a)$. Write

$$a_L(x) = 92 + 152x + 57x^3 \quad \text{and} \quad a_R(x) = 29x + 133x^2 + 200x^3.$$

Then

$$a_0(x) = a_L(x) + \zeta^2 a_R(x) = 92 + 134x + 46x^2 + 198x^3$$

and

$$a_1(x) = a_L(x) - \zeta^2 a_R(x) = 92 + 170x + 187x^2 + 149x^3.$$

Reducing further yields

$$a_{00}(x) = 178 + 180x, \quad a_{01}(x) = 6 + 88x, \quad a_{10}(x) = 57 + 177x, \quad a_{11}(x) = 127 + 163x.$$

Thus,

$$\hat{a} = (178 + 180x,\ 57 + 177x,\ 6 + 88x,\ 127 + 163x).$$

Applying the same procedure to $b(x)$, we obtain

$$\hat{b} = (118 + 226x,\ 183 + 140x,\ 114 + 214x,\ 75 + 119x).$$

Hence, componentwise multiplication gives

$$\hat{c} = \hat{a} \circ \hat{b} = (59 + 189x,\ 211 + 62x,\ 11 + 132x,\ 171 + 77x).$$

Finally, applying the inverse transform (details omitted) gives

$$c(x) = \mathrm{NTT}_2^{-1}(\hat{c}) = 113 + 115x + 114x^2 + 119x^3 + 48x^4 + 205x^5 + 51x^6 + 18x^7.$$

Kyber NTT_2. The FIPS 203 standard that specifies Kyber fixes the parameters $n = 256$ and $q = 3329$. The order-256 element $\zeta = 17 \in \mathbb{Z}_{3329}^*$ is used. The corresponding Kyber ring $R_q = \mathbb{Z}_{3329}[x]/(x^{256} + 1)$ consists of polynomials of degree at most 255 with coefficients in $\mathbb{Z}_{3329}$.

The NTT_2 transform employed in Kyber, as specified in FIPS 203, differs slightly from the transform defined in Definition 13.7. In that earlier definition, the i-th component of $\hat{a} = \mathrm{NTT}_2(a)$ is $\hat{a}_i = a(x) \bmod (x^2 - \zeta^{2i-1})$ for $1 \le i \le \frac{n}{2}$. In contrast, the Kyber NTT_2 permutes these powers of ζ so that they appear in the same order as the leaves of the underlying NTT_2 computation tree. This avoids having to permute the values corresponding to the leaves of the NTT tree as was done in (168).

This ordering of leaves is concisely described using bit reversal. For an integer $v \in [0, 127]$, let $\mathrm{BitRev}(v) = \mathrm{bitrev}_7(v)$ denote the 7-bit integer obtained by reversing the 7-bit binary representation of v.

Definition 13.9 For $0 \le i \le 127$, define $\zeta_i = \zeta^{2\mathrm{BitRev}(i)+1} \bmod q$. The Kyber NTT_2 is the map $\mathrm{NTT}_2 : R_q \longrightarrow T_q$ defined by

$$\mathrm{NTT}_2(a) = (a \bmod (x^2 - \zeta_0),\ a \bmod (x^2 - \zeta_1),\ \ldots,\ a \bmod (x^2 - \zeta_{127})).$$

Kyber key generation. To generate a Kyber-KEM key pair (Algorithm 6.10), a user randomly selects $\rho \in_R \{0,1\}^{256}$ and constructs the matrix $A \in R_q^{k\times k}$ from ρ using a pseudorandom function: $A = \text{Expand}(\rho)$. The user selects $s \in_R S_{\eta_1}^k$ and $e \in_R S_{\eta_1}^k$, where S_{η_1} denotes the set of polynomials in R_q all of whose coefficients lie in the interval $[-\eta_1, \eta_1]$. The user computes the $k \times 1$ polynomial vector $t = As + e$. Their public key is (A, t), and their private key is s.

NTT_2 is used to accelerate the k^2 polynomial multiplications needed to compute the matrix-vector product As. To minimize the number of NTT_2 and NTT_2^{-1} applications, the matrix A is generated directly in NTT_2 form. Key generation proceeds as follows.

1. Compute $\hat{A} = \text{Expand}(\rho)$.
2. Select $s \in_R S_{\eta_1}^k$ and $e \in_R S_{\eta_1}^k$, and compute $\hat{s} = \text{NTT}_2(s)$ and $\hat{e} = \text{NTT}_2(e)$.
3. Compute $\hat{t} = \hat{A} \circ \hat{s} + \hat{e}$.
4. The user's public key is $(\rho, \hat{t})$, and their private key is $\hat{s}$.

The transforms NTT_2 and NTT_2^{-1} are also used during key encapsulation and key decapsulation.

13.5 Notes and further references

The Number-Theoretic Transform (NTT) was introduced independently in 1971 by Nicholson [183], Pollard [189] and Schönhage and Strassen [202]. It can be viewed as a finite-field analogue of the Discrete Fourier Transform (DFT), operating on sequences of elements from a finite field rather than on sequences of complex numbers.

The NTT variant described in §13 for polynomial multiplication in $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ is tailored for so-called negacyclic convolution. The NTT used in Dilithium was introduced by Ducas et al. [75]. A complete description, along with in-place algorithms for evaluating the Dilithium NTT and its inverse, is provided in FIPS 204 [167]. The NTT employed in Kyber first appeared in version 2.0 of the Kyber submission to NIST's post-quantum cryptography standardization project. A complete description, along with in-place algorithms for evaluating the Kyber NTT and its inverse, is provided in FIPS 203 [168]. Huang et al. [123] and Zhang, Yan, Huang and Koç [218] report on software implementations of Kyber and Dilithium on ARM and RISC processors.

13.6 Exercises

Exercise 13.1 Let $n = 2^k$. For each $0 \le i \le n-1$, prove that the leaf of the NTT computation tree indexed by i is $x - \zeta^{2\text{brv}(i)+1}$ (cf. Figure 13.1).

Exercise 13.2 Consider the NTT parameters $q = 97$, $n = 4$ and $\zeta = 64$, so the polynomial ring of interest is $R_q = \mathbb{Z}_{97}[x]/(x^4 + 1)$.
(a) Write pseudocode for evaluating NTT and NTT^{-1}.
(b) Let $a(x) = 1 + 71x + 52x^2 + 90x^3 \in R_q$, $b(x) = 83 + 31x + 13x^2 + 54x^3 \in R_q$, and let

$c(x) = a(x)b(x) \bmod (x^4 + 1)$. Verify the following computations:
(i) $\hat{a} = (4, 19, 55, 23)$.
(ii) $\hat{b} = (9, 50, 50, 29)$.
(iii) $\hat{c} = (36, 77, 34, 85)$.
(iv) $c = \text{NTT}^{-1}(\hat{c}) = 58 + 6x + 21x^2 + 68x^3$.

Exercise 13.3 Consider the NTT parameters $q = 241$, $n = 8$ and $\zeta = 115$, so the polynomial ring of interest is $R_q = \mathbb{Z}_{241}[x]/(x^8 + 1)$. Let $a(x) = 11x + 22x^2 + 33x^3 + 44x^4 + 55x^5 + 66x^6 + 77x^7 \in R_q$, $b(x) = 77x + 66x^2 + 55x^3 + 44x^4 + 33x^5 + 22x^6 + 11x^7 \in R_q$, and let $c(x) = a(x)b(x) \bmod (x^8 + 1)$. Verify the following computations:
(a) $\text{NTT}(a) = (149, 228, 183, 57, 37, 128, 176, 6)$.
(b) $\text{NTT}(b) = (235, 65, 113, 204, 184, 58, 13, 92)$.
(c) $\text{NTT}(c) = (70, 119, 194, 60, 60, 194, 119, 70)$.
(d) $c = \text{NTT}^{-1}(\text{NTT}(c)) = 171 + 185x + 202x^2 + 221x^3 + 20x^5 + 39x^6 + 56x^7$.

Exercise 13.4 Understand and then implement the Dilithium NTT and NTT^{-1} in-place algorithms described in Section 7.5 of FIPS 204 (and depicted in Figure 13.4).

Exercise 13.5 Let $n = 2^k$, and let q be a prime for which $q-1$ is divisible by n but not by $2n$. Describe $O(n \log n)$-time algorithms for evaluating NTT_2 and NTT_2^{-1} with respect to the polynomial ring $R_q = \mathbb{Z}_q[x]/(x^n + 1)$.

Exercise 13.6 Consider the NTT_2 parameters $q = 569$, $n = 8$ and $\zeta = 277$, so the polynomial ring of interest is $R_q = \mathbb{Z}_{569}[x]/(x^8 + 1)$.
(a) Write pseudocode for evaluating NTT_2 and NTT_2^{-1}.
(b) Let $a(x) = 199 + 352x^2 + 157x^3 + 2x^4 + 129x^5 + 343x^6 + 499x^7 \in R_q$,
$b(x) = 16 + 73x + 145x^2 + 332x^3 + 114x^4 + 128x^5 + 164x^6 + 312x^7 \in R_q$,
and let $c(x) = a(x)b(x) \bmod (x^8 + 1)$. Verify the following computations:
(i) $\text{NTT}_2(a) = (126 + 332x,\ 368 + 222x,\ 497 + 240x,\ 374 + 344x)$.
(ii) $\text{NTT}_2(b) = (166 + 45x,\ 264 + 402x,\ 173 + 276x,\ 30 + 138x)$.
(iii) $\text{NTT}_2(c) = (475 + 468x,\ 486 + 566x,\ 125 + 26x,\ 566 + 480x)$.
(iv) $c = \text{NTT}_2^{-1}(\text{NTT}_2(c)) = 413 + 385x + 28x^2 + 156x^3 + 524x^4 + 81x^5 + 327x^6 + 475x^7$.

Exercise 13.7 Consider the NTT_2 parameters $q = 241$, $n = 16$ and $\zeta = 111$, so the polynomial ring of interest is $R_q = \mathbb{Z}_{241}[x]/(x^{16} + 1)$.
(a) Write pseudocode for evaluating NTT_2 and NTT_2^{-1}.
(b) Let $a(x) = 92 + 238x + 121x^2 + 44x^3 + 223x^4 + 5x^5 + 225x^6 + 186x^7 + 43x^8 + 99x^9 + 37x^{10} + 196x^{11} + 26x^{12} + 95x^{13} + 16x^{14} + 231x^{15} \in R_q$,
$b(x) = 228 + 89x + 161x^2 + 17x^3 + 179x^4 + 55x^5 + 170x^6 + 82x^7 + 24x^8 + 89x^9 + 92x^{10} + 59x^{11} + 220x^{12} + 141x^{13} + 49x^{14} + 122x^{15} \in R_q$,
and let $c(x) = a(x)b(x) \bmod (x^{16} + 1)$. Verify the following computations:
(i) $\text{NTT}_2(a) = (129 + 98x,\ 181 + 36x,\ 143 + 32x,\ 61 + 64x,\ 153 + 130x,\ 31 + 94x,\ 21 + 171x,\ 17 + 74x)$.

(ii) $\text{NTT}_2(b) = (222+119x,\ 141+217x,\ 109+122x,\ 210+134x,\ 95+7x,\ 171+208x,\ 126+219x,\ 27+168x)$.
(iii) $\text{NTT}_2(c) = (30+234x,\ 154+9x,\ 186+208x,\ 166+165x,\ 44+166x,\ 158+109x,\ 201+117x,\ 89+34x)$.
(iv) $c = \text{NTT}_2^{-1}(\text{NTT}_2(c)) = 8 + 70x + 195x^2 + 211x^3 + 39x^4 + 48x^5 + 22x^6 + 145x^7 + 116x^8 + 131x^9 + 183x^{10} + 216x^{11} + 213x^{12} + 101x^{13} + 176x^{14} + 65x^{15}$.

Exercise 13.8 Understand and then implement the Kyber NTT_2 and NTT_2^{-1} in-place algorithms described in Section 4.3 of FIPS 203.

A Security definitions

Contents

A.1 Public-key encryption 247
A.2 Key encapsulation mechanisms 248
A.3 Signature schemes 249

The definitions in this section are stated informally.

A.1 Public-key encryption

Definition A.1 A *public-key encryption scheme* (PKE) consists of three algorithms:

(i) *Key Generation* (KeyGen) is a randomized algorithm that takes no input (or a security parameter) and outputs a pair (EK, DK), where EK is an encryption (public) key and DK is a decryption (private) key.

(ii) *Encryption* (Enc), which may be randomized, takes an encryption key EK and a plaintext message m, and outputs a ciphertext $c \leftarrow \mathrm{Enc}(\mathrm{EK}, m)$.

(iii) *Decryption* (Dec) takes a decryption key DK and a ciphertext c, and outputs a plaintext message $m \leftarrow \mathrm{Dec}(\mathrm{DK}, c)$.[31]

Correctness requirement: For every key pair (EK, DK) produced by KeyGen, every message m, and every ciphertext c produced by $\mathrm{Enc}(\mathrm{EK}, m)$, it must hold that

$$\mathrm{Dec}(\mathrm{DK}, c) = m.$$

Definition A.2 A public-key encryption scheme is *indistinguishable under a chosen-plaintext attack* (IND-CPA secure) if a computationally bounded adversary cannot distinguish between the encryptions of two equal-length messages of their choice.

In other words, the adversary is given the encryption key EK and chooses two equal-length messages, m_0 and m_1. The adversary is then given a challenge ciphertext $c \leftarrow \mathrm{Enc}(\mathrm{EK}, m_b)$, where $b \in \{0, 1\}$ is chosen uniformly at random. The adversary should be unable to guess b correctly with a probability significantly greater than $\frac{1}{2}$.

Definition A.3 A public-key encryption scheme is *indistinguishable under an adaptive chosen-ciphertext attack* (IND-CCA2 secure) if a computationally bounded adversary with access to a decryption oracle cannot distinguish between the encryptions of two equal-length messages of their choice, even after receiving the challenge ciphertext.

[31] More generally, the decryption algorithm can output a failure symbol $\perp$; this should occur with negligible probability for each plaintext message m.

In other words, the adversary is given the encryption key EK and access to a decryption oracle (which decrypts ciphertexts using the corresponding decryption key DK). The adversary chooses two equal-length messages, m_0 and m_1, and is then given a challenge ciphertext $c \leftarrow \mathrm{Enc}(\mathrm{EK}, m_b)$, where $b \in \{0, 1\}$ is chosen uniformly at random. The adversary may query the decryption oracle both before and after receiving c, on any ciphertext other than c itself. The adversary should be unable to guess b correctly with a probability significantly greater than $\frac{1}{2}$.

A.2 Key encapsulation mechanisms

Definition A.4 A *key encapsulation mechanism* (KEM) consists of three algorithms:

(i) *Key Generation* (KeyGen) is a randomized algorithm that takes no input (or a security parameter) and outputs a pair (EK, DK), where EK is an encapsulation (public) key and DK is a decapsulation (private) key.

(ii) *Encapsulation* (Encaps) is a randomized algorithm that takes an encapsulation key EK and outputs a ciphertext c and a key K: $(c, K) \leftarrow \mathrm{Encaps}(\mathrm{EK})$.

(iii) *Decapsulation* (Decaps) takes a decapsulation key DK and a ciphertext c, and outputs a key K: $K \leftarrow \mathrm{Decaps}(\mathrm{DK}, c)$.

Correctness requirement: For every key pair (EK, DK) produced by KeyGen, and every (c, K) produced by Encaps(EK), it must hold that

$$\mathrm{Decaps}(\mathrm{DK}, c) = K.$$[32]

Definition A.5 A key encapsulation mechanism is *indistinguishable under an adaptive chosen-ciphertext attack* (IND-CCA2 secure) if a computationally bounded adversary with access to a decapsulation oracle cannot distinguish the real key encapsulated in a challenge ciphertext from a randomly chosen key, even after receiving the challenge ciphertext.

In other words, the adversary is given the encapsulation key EK and access to a decapsulation oracle (which decapsulates ciphertexts using the corresponding decapsulation key DK). The challenger computes $(c, K_0) \leftarrow \mathrm{Encaps}(\mathrm{EK})$, chooses a key K_1 uniformly at random from the key space, selects $b \in \{0, 1\}$ uniformly at random, and gives the adversary the challenge pair (c, K_b). The adversary may query the decapsulation oracle both before and after receiving the challenge pair, on any ciphertext other than c itself. The adversary should be unable to guess b correctly with a probability significantly greater than $\frac{1}{2}$.

[32] More generally, the decapsulation algorithm can produce a key K' that is different from K; this should occur with negligible probability.

A.3 Signature schemes

Definition A.6 A *signature scheme* consists of three algorithms:

(i) *Key Generation* (KeyGen) is a randomized algorithm that takes no input (or a security parameter) and outputs a pair (VK, SK), where VK is a verification (public) key and VK is a signing (private) key.

(ii) *Signature Generation* (Sign), which may be randomized, takes a signing key SK and a message M, and outputs a signature $\sigma \leftarrow \text{Sign}(\text{SK}, M)$.

(iii) *Signature Verification* (Verify) takes a verification key VK, a message M, and a signature σ, and outputs "accept" or "reject".

Correctness requirement: For every key pair (VK, SK) produced by KeyGen, every message M, and every signature σ produced by $\text{Sign}(\text{SK}, M)$, it must hold that

$$\text{Verify}(\text{VK}, M, \sigma) = \text{“accept”}.$$

Definition A.7 A signature scheme is *secure* if it is existentially unforgeable under a chosen-message attack by a computationally bounded adversary.

In other words, given a verification key VK and access to a signing oracle (which signs messages using the corresponding signing key SK), such an adversary should be unable to produce a valid message-signature pair (M, σ) for any message M not previously submitted to the oracle.

B Answers to computational exercises

2.1 Let b_1, b_2, b_3 be the columns of B.
(a) $(7, -3, 2) = b_1 - 2b_2 - b_3 \in L$.
(b) $(-4, 4, 0) = \frac{1}{2}b_1 + \frac{1}{3}b_2 + b_3 \notin L$.

2.2 We have $B_1 = B_2 \cdot U$, where U is the unimodular matrix:

$$U = \begin{bmatrix} 2 & -1 & -1 \\ -2 & 2 & -1 \\ 1 & -1 & 1 \end{bmatrix}.$$

2.3 Let b_1, b_2, b_3 be the columns of B_1, and b'_1, b'_2, b'_3 the columns of B_2.
(a) We have $b'_1 = b_1 + b_3$, $b'_2 = b_1 + 2b_2$, $b'_3 = b_2 + b_3$, showing that $L_2 \subseteq L_1$.
(b) $b_1 = \frac{2}{3}b'_1 + \frac{1}{3}b'_2 - \frac{2}{3}b'_3$, so $b_1 \notin L_2$. Thus, $L_1 \nsubseteq L_2$.

2.11 $\lambda_1(L) = 2$.

2.12 $\lambda_1(L) = \sqrt{2}$.

2.13(a) $\pm(1, 0)$, $\pm(\frac{1}{2}, \frac{\sqrt{3}}{2})$, $\pm(\frac{1}{2}, -\frac{\sqrt{3}}{2})$.

2.15 Let e_i denote the i-th unit vector in $\mathbb{R}^n$, and let $b_i = d_i \cdot e_i$. Then $[b_1, \ldots, b_n]$ is a basis for a lattice $L \subseteq \mathbb{R}^n$ with $\lambda_i(L) = d_i$ for all $1 \le i \le n$.

2.14 We have $\lambda_1(L) = \sqrt{17}$ and $\mathrm{vol}(L) = 15$.

2.19(d) $B = [(1, -1, 0, 0), (1, 1, 0, 0), (0, 0, 1, 1), (0, 0, 1, -1)]$ is a set of linearly independent lattice vectors, but is not a basis since $\det(B) = 4$ whereas $\mathrm{vol}(L) = 2$.

3.1 $z = \pm(4, -4, 0, -2, 3, 3)$, $z = \pm(3, -1, -4, -1, -5, 2)$.

3.2 $z = (-2, 4, 2, 2, 4)$, $z = (0, -4, -5, -4, -3)$, $z = (0, -2, 5, -1, -2)$
$z = (0, 2, -4, 5, 0)$, $z = (1, 3, 3, -5, -1)$, $z = (3, 1, -3, -2, -5)$.

?? $s = (3, 14, 1)$, $e = (-1, 1, -2, 1, 1)$; $s = (9, 47, 25)$, $e = (1, 1, 0, 1, 0)$;
$s = (22, 32, 44)$, $e = (0, 1, -2, 1, 2)$; $s = (24, 11, 45)$, $e = (-1, 1, 0, 1, -2)$;
$s = (28, 12, 15)$, $e = (2, 1, 0, 1, 1)$; $s = (37, 49, 11)$, $e = (-2, 1, -2, 1, 0)$;
$s = (43, 29, 35)$, $e = (0, 1, 0, 1, -1)$; $s = (49, 9, 6)$, $e = (2, 1, 2, 1, -2)$.

4.2 $s = (-3, -2, -1, -1)$.

5.2 The three solutions up to multiplication by ± 1, $\pm x$, $\pm x^2$ are $z = 1 - 8x + 6x^2 + 4x^3$,

$z = 10 + 10x - 7x^2 - 5x^3$, and $z = -7 - 7x - 6x^2 + x^3$.

5.5 The unique solution is:

$$s = \begin{bmatrix} 36 + 35x + 26x^2 + 2x^3 \\ 17 + 18x + 20x^2 + 19x^3 \end{bmatrix}, \quad e = \begin{bmatrix} 1 - x \\ 1 - x + x^2 - x^3 \\ -1 - x + x^3 \end{bmatrix}.$$

8.14 There are multiple solutions; one set is presented below.
(a) $F(x) = -48x - 28x^2 + 27x^3$, $G(x) = -15 + 51x + 28x^2 - 47x^3$.
(b) $F(x) = 21 + 54x - 68x^2 - 31x^3$, $G(x) = -1 - 26x + 28x^2 + 4x^3$.
(c) $F(x) = 7 + x + 47x^2 - 30x^3$, $G(x) = -18 - 32x + 35x^2 + 72x^3$.

8.15 There are multiple solutions; one set is presented below.
(a) $f(x) = -33 - 22x + 11x^2$, $g(x) = -46 - 34x + 12x^2$.
(b) $f(x) = -22 - 24x - 16x^2 - 22x^3 + 5x^4$, $g(x) = 28 - 54x - x^2 + 8x^3 + 59x^4$.
(c) $f(x) = -6 - 70x + 62x^2 + 18x^3 - 8x^4 - 39x^5 + 4x^6$,
$g(x) = 14 + 45x + 21x^2 + 80x^3 - 10x^4 + 30x^5 - 22x^6$.
(d) $f(x) = 42 + 4x + 21x^2 - 4x^3 + 24x^4 + 77x^5 + 44x^6 + 8x^7 + 24x^8 - x^9$,
$g(x) = -17 + 39x + 27x^2 - x^3 - 81x^4 + 27x^5 - 65x^6 + 72x^7 - 10x^8 - 17x^9$.

8.16 There are multiple solutions; one set is presented below.
(a) $s_1(x) = -22 + 59x + 39x^2$, $s_2(x) = -49 - 21x - 27x^2$.
(b) $s_1(x) = -22 - 27x - 12x^2 + 6x^3 - 11x^4$, $s_2(x) = -51 + 27x - 33x^2 + 74x^3 + x^4$.
(c) $s_1(x) = 41 + 27x + 62x^2 + 20x^3 - 3x^4 + 29x^5 - 38x^6$,
$s_2(x) = 70 + 11x + 19x^2 - 44x^3 - 32x^4 - 67x^5 - x^6$.
(d) $s_1(x) = 20 + 80x - 32x^2 - 55x^3 + 37x^4 - 25x^5 + 11x^6 - 29x^7 - 14x^8 - 23x^9$,
$s_2(x) = -38 - 61x + 53x^2 + 80x^3 - 66x^4 - 42x^5 - 19x^6 + 16x^7 - 27x^8 - 15x^9$.

9.1 $\text{Encode}(m) = \begin{bmatrix} 640 & 192 \\ 224 & 576 \end{bmatrix}$, $\text{Decode}(c) = \begin{bmatrix} 1 & 29 \\ 26 & 29 \end{bmatrix}$.

10.7 (a) $\lambda_1(L) = 1$, $\lambda_2(L) = 2$. (b) $\lambda_1(L) = \sqrt{5}$, $\lambda_2(L) = \sqrt{17}$.
(c) $\lambda_1(L) = \sqrt{10146890}$, $\lambda_2(L) = \sqrt{19059138}$.

10.8(d) (i) $12373 = 42^2 + 103^2$. (ii) $555557 = 239^2 + 706^2$. (iii) $44444453 = 2783^2 + 6058^2$.

10.10 The following are $\frac{3}{4}$-reduced LLL bases.
(a) $(0, 0, -1, 1, 0)$, $(1, -1, 0, 1, 1)$, $(0, -1, 0, 0, -2)$, $(0, -1, -2, -1, 1)$, $(3, 1, -1, 0, 0)$.
(b) $(1, 1, 3, -3, 7)$, $(4, 8, -4, 1, 4)$, $(-6, 2, -2, -6, -4)$, $(-3, -1, -5, 5, 6)$, $(3, -5, -10, -8, 3)$.
(c) $(3, 3, -3, 2, 2)$, $(6, -1, 9, -6, 5)$, $(-8, -5, -13, -9, 10)$, $(-14, 6, 8, 19, 11)$,

$(96, -256, -31, 144, 55)$.

10.11 A basis matrix for E_8 is

$$\frac{1}{2}\begin{bmatrix} -2 & 2 & -2 & -2 & -2 & -2 & 1 & -1 \\ 2 & 2 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 2 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 2 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 2 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 2 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}.$$

10.12 Replace the first column b_1 of basis matrix B in Example 2.13 with $b_1 - b_2 - b_3$.

11.4 (a) $b_1^* = (2, 0)$, $b_2^* = (0, 2)$, $\mu_{21} = \frac{1}{2}$, and $D = 2$.
(b) The tree has three nodes at level 1 and three at level 2. The shortest nonzero vectors are $\pm(2, 0)$.

11.8 The GSA ratio is $\delta_0^{2(n-1)} = 1.0093^{998} \approx 1.03 \times 10^4$.

12.1 $s = (-154, -51, 68, -176)$, $e = (-54, 135, -257, 108, -121, -196)$.

12.2 The MLWE solution is

$$s = \begin{bmatrix} 2x + x^2 + 2x^3 \\ -1 + x - 2x^3 \\ -2 + 2x^2 - x^3 \end{bmatrix}, \quad e = \begin{bmatrix} x \\ -x - 2x^2 - x^3 \\ 2 + 2x - 2x^2 + x^3 \\ -2 + 2x \end{bmatrix}.$$

12.3 Three SIS_2 solutions are $(-6, -15, -6, 5, 10, 4, 14, -5, -3, -8, -6, 1, 15, 1)$,
$(1, -1, -23, 6, -3, 4, 4, -15, -16, 4, -7, -2, -2, 5)$, and
$(4, 9, 10, -1, -8, -21, 8, -3, 3, 5, 5, 11, -3, 16)$.

12.4 One MSIS_2 solution is

$$s = \begin{bmatrix} -5 + 5x - 7x^2 - 27x^3 + 11x^4 \\ 3 - 5x - 6x^2 + 17x^3 + 4x^4 \\ -16 - 13x - 6x^2 + 4x^3 + 10x^4 \\ -16 - 10x - 9x^2 - 5x^3 - x^4 \end{bmatrix}.$$

12.5(b) (i) $x = (-1571, 1529, -1281, 1710)$.
(ii) $x = (259, 242, 8, 194, -307)$.
(iii) $x = (-589, -1343, 711, -837, -391, -527)$.

12.6(c) (i) Kannan's embedding technique with $M = 1$ produces the same vector $v = (-1571, 1529, -1281, 1710)$ as Babai's rounding method.

(ii) Kannan's embedding techinque with $M = 3$ yields $v = (275, 191, 38, 221, -282)$, which is closer to t than the vector $x = (259, 242, 8, 194, -307)$ produced by Babai's rounding method. We have $\|v - t\|^2 = 1598$ and $\|x - t\|^2 = 1785$.
(iii) Kannan's embedding technique with $M = 1$ produces the same vector $v = (-589, -1343, 711, -837, -391, -527)$ as Babai's rounding method.

12.8 (a) $x^2 - 2$. (b) $x^4 - 10x^2 + 1$. (c) $x^8 - 40x^6 + 352x^4 - 960x^2 + 576$.
(d) $x^6 - 6x^4 - 6x^3 + 12x^2 - 36x + 1$. (e) $x^{10} - 10x^8 + 40x^6 - 6x^5 - 80x^4 - 120x^3 + 80x^2 - 120x - 23$.

12.9 (a) One solution is $\frac{48}{34}, \frac{59}{34}, \frac{76}{34}, \frac{90}{34}, \frac{113}{34}$.

(b) One solution is $\frac{929}{657}, \frac{1138}{657}, \frac{1469}{657}, \frac{1738}{657}, \frac{2179}{657}, \frac{2369}{657}, \frac{2709}{657}, \frac{2864}{657}, \frac{3151}{657}, \frac{3538}{657}$.

12.10 $K = 9550684$.

12.12 The plaintext is $M =$
22222222222222222233333333333333333333333333344444444444444444
44444444445555555555555555555555555555566666666666666666666666
77777777777777777777778888888888888888888888888.

12.13(c) (i) $s = a_2 + a_4 + a_5 + a_9$. (ii) $s = a_1 + a_3 + a_5 + a_8 + a_{10}$.

12.14(e) (i) $7919 = 5^2 + 37^2 + 42^2 + 69^2$. (ii) $59359 = 10^2 + 67^2 + 77^2 + 221^2$.
(iii) $104743 = 79^2 + 151^2 + 151^2 + 230^2$.

References

[1] D. Aggarwal and N. Stephens-Davidowitz. Just take the average! — An embarrassingly simple 2^n-time algorithm for SVP (and CVP). In *Symposium on Simplicity of Algorithms (SOSA)*, pages 12:1–12:19, 2018.

[2] D. Aharonov and O. Regev. Lattice problems in NP ∩ coNP. *Journal of the ACM*, 52:749–765, 2005.

[3] M. Ajtai. Generating hard instances of lattice problems. In *Symposium on Theory of Computing (STOC)*, pages 99–108, 1996.

[4] M. Ajtai. The shortest vector problem in L_2 is NP-hard for randomized reductions. In *Symposium on Theory of Computing (STOC)*, pages 10–19, 1998.

[5] M. Ajtai. Generating hard instances of the short basis problem. In *ICALP*, pages 1–9, 1999.

[6] M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Symposium on Theory of Computing (STOC)*, pages 284–293, 1997.

[7] M. Ajtai, R. Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *Symposium on Theory of Computing (STOC)*, pages 601–610, 2001.

[8] M. Albrecht. Lattice attacks against small-secret parameter choices in HElib and SEAL. In *EUROCRYPT*, pages 103–129, 2017.

[9] M. Albrecht, F. Göpfert, F. Virdia, and T. Wunderer. Revisiting the expected cost of solving uSVP and applications to LWE. In *ASIACRYPT*, pages 297–322, 2017.

[10] M. Albrecht, R. Player, and S. Scott. On the concrete hardness of Learning with Errors. *J. Mathematical Cryptology*, 9:169–203, 2015.

[11] A. Alfarano, E. Saxena, E. Wenger, F. Charton, and K. Lauter. Improving ML attacks on LWE with data repetition and stepwise regression. In *Forty-third International Conference on Machine Learning*, 2026. available at https://openreview.net/forum?id=Tq5dFAGeBL.

[12] E. Alkim, J. Bos, L. Ducas, P. Longa, I. Mironov, M. Naehrig, V. Nikolaenko, C. Peikert, A. Raghunathan, and D. Stebila. FrodoKEM: Learning With Errors Key Encapsulation, 2021. Round 3 submission to NIST PQC standardization project, available at https://frodokem.org/.

[13] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe. Post-quantum key exchange — A new hope. In *USENIX Security*, pages 327–343, 2016.

[14] J. Alwen, S. Krenn, K. Pietrzak, and D. Wichs. Learning with rounding, revisited: New reductions, properties and applications. In *CRYPTO*, pages 57–74, 2013.

[15] J. Alwen and C. Peikert. Generating shorter bases for hard random lattices. *Theory of Computing Systems*, 48:535–553, 2011.

[16] D. Aranha, F. Novaes, A. Takahashi, M. Tibouchi, and Y. Yarom. LadderLeak: Breaking ECDSA with less than one bit of nonce leakage. In *ACM Comp. & Comm. Security (CCS)*, pages 225–242, 2020.

[17] S. Arora and R. Ge. New algorithms for learning in presence of errors. In *ICALP*, pages 403–415, 2011.

[18] P. Azevedo-Oliveira, A. Calle Viera, B. Cogliati, and L. Goubin. Uncompressing Dilithium's public key. In *CRYPTO*, pages 417–443, 2025.

[19] L. Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6:1–13, 1986.

[20] S. Bai and S. Galbraith. An improved compression technique for signatures based on learning with errors. In *The Cryptographers' Track at RSA Conference (CT-RSA)*, pages 28–47, 2014.

[21] S. Bai, T. Lepoint, A. Roux-Langlois, A. Sakzad, D. Stehlé, and R. Steinfeld. Improved security proofs in lattice-based cryptography: Using the Rényi divergence rather than the statistical distance. *J. Cryptology*, 31:610–640, 2018.

[22] A. Banerjee, C. Peikert, and A. Rosen. Pseudorandom functions and lattices. In *EUROCRYPT*, pages 719–737, 2012.

[23] M. Barbosa, G. Barthe, C. Doczkal, J. Don, S. Fehr, B. Grégoire, Y. Huang, A. Hülsing, Y. Lee, and X. Wu. Fixing and mechanizing the security proof of Fiat-Shamir with aborts and Dilithium. In *CRYPTO*, pages 358–389, 2023.

[24] M. Barbosa, D. Connolly, J. Duarte, A. Kaiser, P. Schwabe, K. Varner, and B. Westerbaan. X-Wing: The hybrid KEM you've been looking for. *IACR Communications in Cryptology*, 1, 2024.

[25] G. Barthe, S. Belaïd, T. Espitau, P. Fouque, M. Rossi, and M. Tibouchi. GALACTICS: Gaussian sampling for lattice-based constant-time implementation of cryptographic signatures, revisited. In *ACM Comp. & Comm. Security (CCS)*, pages 2147–2164, 2019.

[26] A. Becker, L. Ducas, N. Gama, and T. Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 10–24, 2016.

[27] L. Beckwith, D. Nguyen, and K. Gaj. Hardware accelerators for digital signature algorithms Dilithium and FALCON. *IEEE Design & Test*, 41:27–35, 2024.

[28] M. Bellare and P. Rogaway. The exact security of digital signatures — How to sign with RSA and Rabin. In *EUROCRYPT*, pages 399–416, 1996.

[29] D. Bernstein. Multi-ciphertext security degradation for lattices, 2022. IACR Cryptology ePrint Archive, https://eprint.iacr.org/2022/1580.

[30] D. Bernstein, K. Bhargavan, S. Bhasin, A. Chattopadhyay, T. Chia, M. Kannwischer, F. Kiefer, T. Paiva, P. Ravi, and G. Tamvada. KyberSlash: Exploiting secret-dependent division timings in Kyber implementations. *IACR Trans. Cryptographic Hardware and Embedded Systems*, pages 209–234, 2025.

[31] A. Bienstock, L. de Castro, D. Escudero, A. Polychroniadou, and A. Takahashi. Quorus: Efficient, scalable threshold ML-DSA signatures from MPC. In *USENIX Security*, 2026.

[32] J. Blömer and J. Seifert. On the complexity of computing short linearly independent vectors and short bases in a lattice. In *Symposium on Theory of Computing (STOC)*, pages 711–720, 1999.

[33] A. Blum, A. Kalai, and H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM*, 50(4):506–519, 2003.

[34] D. Boneh, Ö. Dagdelen, M. Fischlin, A. Lehmann, C. Schaffner, and M. Zhandry. Random oracles in a quantum world. In *ASIACRYPT*, pages 41–69, 2011.

[35] D. Boneh and G. Durfee. Cryptanalysis of RSA with private key less than $N^{0.292}$. *IEEE Trans. Info. Theory*, 46:1339–1349, 2000.

[36] D. Boneh, G. Durfee, and N. Howgrave-Graham. Factoring $N = p^r q$ for large r. In *CRYPTO*, pages 326–337, 1999.

[37] D. Boneh and R. Venkatesan. Hardness of computing the most significant bit of secret keys in Diffie-Hellman and related schemes. In *CRYPTO*, pages 129–142, 1996.

[38] A. Bonnecaze and P. Solé. Quaternary constructions of formally self-dual binary codes and unimodular lattices. In *Algebraic Coding*, pages 194–205, 1994.

[39] X. Bonnetain, A. Chailloux, A. Schrottenloher, and Y. Shen. Finding many collisions via reusable quantum walks: Application to lattice sieving. In *EUROCRYPT*, pages 221–251, 2023.

[40] J. Borwein and P. Lisoněk. Applications of integer relation algorithms. *Discrete Mathematics*, 217:65–82, 2000.

[41] J. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A. Raghunathan, and D. Stebila. Frodo: Take off the ring! Practical, quantum-secure key exchange from LWE. In *ACM Comp. & Comm. Security (CCS)*, pages 1006–1018, 2016.

[42] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. Schanck, P. Schwabe, G. Seiler, and D. Stehlé. CRYSTALS-Kyber: A CCA-secure module-lattice-based KEM. In *IEEE Eur. Symp. Security & Privacy*, pages 353–367, 2018.

[43] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory*, 6(3), 2014.

[44] J. Breitner and N. Heninger. Biased nonce sense: Lattice attacks against weak ECDSA signatures in cryptocurrencies. In *Financial Crypto*, pages 3–20, 2019.

[45] A. Budroni, Q. Guo, T. Johansson, E. Mårtensson, and P. Wagner. Making the BKW algorithm practical for LWE. In *INDOCRYPT*, pages 417–439, 2020.

[46] T. Cai, J. Fan, and T. Jiang. Distributions of angles in random packing on spheres. *J. Machine Learning Research*, 14:1837–1864, 2013.

[47] K. Carrier, C. Meyer-Hilfiger, Y. Shen, and J. Tillich. Assessing the impact of a variant of MATZOV's dual attack on Kyber. In *CRYPTO*, pages 444–476, 2025.

[48] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. *J. Cryptology*, 25:601–639, 2012.

[49] J. Cassels. *An Introduction to the Geometry of Numbers*. Springer, 1997.

[50] S. Celi, R. del Pino, T. Espitau, G. Niot, and T. Prest. Efficient threshold ML-DSA. In *USENIX Security*, 2026.

[51] S. Chatterjee, N. Koblitz, A. Menezes, and P. Sarkar. Another look at tightness II: Practical issues in cryptography. In *Paradigms in Cryptography (Mycrypt)*, pages 21–55, 2016.

[52] J. Chen, Y. Shabanzadeh, E. Crnčević, T. Hoefler, and D. Alistarh. The geometry of LLM quantization: GPTQ as Babai's nearest plane algorithm. In *International Conference on Learning Representations*, 2026.

[53] Y. Chen. *Réduction de réseau et sécurité concrète du chiffrement complètement homomorphe*. PhD thesis, Université Paris Diderot (France), 2013.

[54] Y. Chen and P. Nguyen. BKZ 2.0: Better lattice security estimates. In *ASIACRYPT*, pages 1–20, 2011.

[55] J. Cheon, H. Choe, J. Seo, and H. Seong. SMAUG(-T), revisited: Timing-secure, more compact, less failure. *IEEE Access*, 12:188386–188397, 2024.

[56] J. Cheon, A. Kim, M. Kim, and Y. Song. Homomorphic encryption for arithmetic of approximate numbers. In *ASIACRYPT*, pages 409–437, 2017.

[57] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. THFE: Fast fully homomorphic encryption over the torus. *J. Cryptology*, 33:34–91, 2020.

[58] H. Cohen. *A Course in Computational Algebraic Number Theory*. Springer, 1993.

[59] H. Cohn, A. Kumar, S. Miller, D. Radchenko, and M. Viazovska. The sphere packing problem in dimension 24. *Annals of Mathematics*, 185:1017–1033, 2017.

[60] J. Conway and N. Sloane. *Sphere Packings, Lattices and Groups*. Springer, 3rd edition, 1999.

[61] J. Cooley and J. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19:297–301, 1965.

[62] D. Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *J. Cryptology*, 10:233–260, 1997.

[63] J. Coron and A. May. Deterministic polynomial-time equivalence of computing the RSA secret key and factoring. *J. Cryptology*, 20:39–50, 2007.

[64] M. Coster, A. Joux, B. LaMacchia, A. Odlyzko, C. Schnorr, and J. Stern. Improved low-density subset sum algorithms. *Computational Complexity*, 2:111–128, 1992.

[65] CrypTool-Online. Kyber (ML-KEM), 2026. https://www.cryptool.org/en/cto/kyber/.

[66] J. D'Anvers, Q. Guo, T. Johansson, A. Nilsson, F. Vercauteren, and I. Verbauwhede. Decryption failure attacks on IND-CCA secure lattice-based schemes. In *Public Key Cryptography (PKC)*, pages 565–598, 2019.

[67] J. D'Anvers, A. Karmakar, S. Roy, and F. Vercauteren. Saber: Module-LWR based key exchange, CPA-secure encryption and CCA-secure KEM. In *AFRICACRYPT*, pages 282–305, 2018.

[68] J. D'Anvers, M. Rossi, and F. Virdia. (One) failure is not an option: Bootstrapping the search for failures in lattice-based encryption schemes. In *EUROCRYPT*, pages 3–33, 2020.

[69] D. De Almeida Braga, P. Fouque, B. Lachguel, and T. Prest. Toward a secure fixed-point implementation of the Falcon signature scheme. In *CRYPTO*, pages 166–197, 2026.

[70] G. De Micheli and N. Heninger. Survey: Recovering cryptographic keys from partial information, by example. *IACR Communications in Cryptology*, 1, 2024.

[71] P. de Perthuis and F. Trenkić. Refined modelling of the primal attack, and variants against module-LWE. In *Post-Quantum Cryptography (PQCrypto)*, pages 342–376, 2026.

[72] J. Devevey, P. Fallahpour, A. Passelègue, D. Stehlé, and K. Xagawa. A detailed analysis of Fiat-Shamir with abort. In *CRYPTO*, pages 327–357, 2023.

[73] L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky. Lattice signatures and bimodal Gaussians. In *CRYPTO*, pages 40–56, 2013.

[74] L. Ducas, L. Engelberts, and P. de Perthuis. Predicting module-lattice reduction. In *ASIACRYPT*, pages 133–166, 2025.

[75] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, and G. Seiler. CRYSTALS-Dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptographic Hardware and Embedded Systems*, pages 238–268, 2018.

[76] L. Ducas, V. Lyubashevsky, and T. Prest. Efficient identity-based encryption over NTRU lattices. In *ASIACRYPT*, pages 22–41, 2014.

[77] L. Ducas and D. Micciancio. Improved short lattice signatures in the standard model. In *CRYPTO*, pages 335–352, 2014.

[78] L. Ducas and P. Nguyen. Learning a zonotope and more: Cryptanalysis of NTRUSign countermeasures. In *ASIACRYPT*, pages 433–450, 2012.

[79] L. Ducas, E. Postlethwaite, L. Pulles, and W. van Woerden. Hawk: Module LIP makes lattice signatures fast, compact and simple. In *ASIACRYPT*, pages 65–94, 2022.

[80] L. Ducas and T. Prest. Fast Fourier orthogonalization. In *International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 191–198, 2016.

[81] L. Ducas, L. Pulles, and M. Stevens. Towards a modern LLL implementation. In *ASIACRYPT*, pages 65–99, 2025.

[82] N. Dwarakanath and S. Galbraith. Sampling from discrete Gaussians for lattice-based cryptography on a constrained device. *Applicable Algebra in Engineering, Communication and Computing*, 25:159–180, 2014.

[83] U. Erez and R. Zamir. Achieving $\frac{1}{2}\log(1+\mathrm{SNR})$ on the AWGN channel with lattice encoding and decoding. *IEEE Trans. Info. Theory*, 50:2293–2314, 2004.

[84] T. Espitau, P. Fouque, B. Gérard, and M. Tibouchi. Attacks on BLISS lattice-based signatures. In *ACM Comp. & Comm. Security (CCS)*, pages 1857–1874, 2017.

[85] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986.

[86] U. Fincke and M. Pohst. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *Mathematics of Computation*, 44:463–471, 1985.

[87] P. Fouque, P. Gajland, H. de Groote, J. Janneck, and E. Kiltz. A closer look at Falcon, 2024. IACR Cryptology ePrint Archive, https://eprint.iacr.org/2024/1769.

[88] P. Fouque, P. Gajland, H. de Groote, J. Janneck, and E. Kiltz. A closer look at Falcon. In *EUROCRYPT*, pages 3–31, 2026.

[89] P. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang. Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU, 2020. v1.2 of submission to NIST PQC standardization project, available at https://falcon-sign.info/.

[90] P. Fouque, P. Kirchner, M. Tibouchi, A. Wallet, and Y. Yu. Key recovery from Gram-Schmidt norm leakage in hash-and-sign signatures over NTRU lattices. In *EUROCRYPT*, pages 34–63, 2020.

[91] P. Fouque, D. Marion, Q. Nguyen, and A. Wallet. Avengers assemble! Supervised learning meets lattice reduction: A single trace attack against CRYSTALS-Kyber key generation. *IACR Trans. Cryptographic Hardware and Embedded Systems*, pages 409–436, 2025.

[92] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO*, pages 537–554, 1999.

[93] S. Galbraith. *Mathematics of Public Key Cryptography*. Cambridge University Press, 2012.

[94] N. Gama and P. Nguyen. Predicting lattice reduction. In *EUROCRYPT*, pages 31–51, 2008.

[95] N. Gama, P. Nguyen, and O. Regev. Lattice enumeration using extreme pruning. In *EUROCRYPT*, pages 257–278, 2010.

[96] S. Gao. Bounded distance decoding for random lattices, 2025. arXiv:2506.16662.

[97] Y. Gao, Y. Feng, and H. Hu. Solving the shortest vector problem in $2^{0.7314n+o(n)}$ time via discrete Gaussian sampling on superlattices, 2026. IACR Cryptology ePrint Archive, https://eprint.iacr.org/2026/1587.

[98] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions, 2007. IACR Cryptology ePrint Archive, https://eprint.iacr.org/2007/432.

[99] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Symposium on Theory of Computing (STOC)*, pages 197–206, 2008.

[100] C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from Learning With Errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, pages 75–92, 2013.

[101] F. Giacon, F. Heuer, and B. Poettering. KEM combiners. In *Public Key Cryptography (PKC)*, pages 190–218, 2018.

[102] L. Glabush, P. Longa, M. Naehrig, C. Peikert, D. Stebila, and F. Virdia. FrodoKEM: A CCA-secure Learning With Errors key encapsulation mechanism. *IACR Communications in Cryptology*, 2, 2025.

[103] O. Goldreich, S. Goldwasser, and S. Halevi. Public-key cryptosystems from lattice reduction problems. In *CRYPTO*, pages 112–131, 1997.

[104] O. Goldreich, D. Micciancio, S. Safra, and J. Seifert. Approximating shortest lattice vectors is not harder than approximating closest lattice vectors. *Information Processing Letters*, 71:55–61, 1999.

[105] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Attribute-based encryption for circuits. *Journal of the ACM*, 62(6), 2015.

[106] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Predicate encryption for circuits from LWE. In *CRYPTO*, pages 503–523, 2015.

[107] L. Groot Bruinderink, A. Hülsing, T. Lange, and Y. Yarom. Flush, Gauss, and Reload — A cache attack on the BLISS lattice-based signature scheme. In *Cryptographic Hardware and Embedded Systems (CHES)*, pages 323–345, 2016.

[108] M. Guerreau, A. Martinelli, T. Ricosset, and M. Rossi. The hidden parallelepiped is back again: Power analysis attacks on Falcon. *IACR Trans. Cryptographic Hardware and Embedded Systems*, pages 141–164, 2022.

[109] Q. Guo, T. Johansson, E. Mårtensson, and P. Wagner. Coded-BKW with sieving. In *ASIACRYPT*, pages 323–346, 2017.

[110] C. Haidar, Q. Payet, and M. Tibouchi. Crowhammer: Full key recovery attack on Falcon with a single Rowhammer bit flip. In *CRYPTO*, pages 103–135, 2025.

[111] G. Hanrot, X. Pujol, and D. Stehlé. Terminating BKZ, 2011. IACR Cryptology ePrint Archive, https://eprint.iacr.org/2011/198.

[112] G. Hanrot and D. Stehlé. Improved analysis of Kannan's shortest lattice vector algorithm. In *CRYPTO*, pages 170–186, 2007.

[113] J. Håstad, B. Just, J. Lagarias, and C. Schnorr. Polynomial time algorithms for finding integer relations among real numbers. *SIAM J. Computing*, 18:859–881, 1989.

[114] J. Hermelink, K. Ning, R. Petri, and E. Strieder. The insecurity of masked comparisons: SCAs on ML-KEM's FO-transform. In *ACM Comp. & Comm. Security (CCS)*, pages 2430–2444, 2024.

[115] M. Hhan. Solving the shortest vector problem in $2^{0.6039n}$ time via mid-point Hessian, 2026. IACR Cryptology ePrint Archive, https://eprint.iacr.org/2026/1597.

[116] J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. Silverman, and W. Whyte. NTRUSign: Digital signatures using the NTRU lattice. In *The Cryptographers' Track at RSA Conference (CT-RSA)*, pages 122–140, 2003.

[117] J. Hoffstein, J. Pipher, and J. Silverman. NTRU: A ring-based public key cryptosystem. In *Algorithmic Number Theory (ANTS)*, pages 267–288, 1998.

[118] D. Hofheinz, K. Hövelmanns, and E. Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In *Theory of Cryptography Conference (TCC)*, pages 341–371, 2017.

[119] J. Hou, H. Jiang, and T. Ogilvie. Careful with the Ring!: On the concrete hardness gaps between LWE and MLWE. In *CRYPTO*, pages 462–497, 2026.

[120] N. Howgrave-Graham. Finding small roots of univariate modular equations revisited. In *Cryptography and Coding*, pages 131–142, 1997.

[121] N. Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In *CRYPTO*, pages 150–169, 2007.

[122] N. Howgrave-Graham and N. Smart. Lattice attacks on digital signature schemes. *Designs, Codes and Cryptography*, 23:283–290, 2001.

[123] J. Huang, A. Adomnicăi, J. Zhang, W. Dai, Y. Liu, C. Cheung, Ç. Koç, and D. Chen. Revisiting Keccak and Dilithium implementations on ARMv7-M. *IACR Trans. Cryptographic Hardware and Embedded Systems*, pages 1–24, 2024.

[124] International Organization for Standardization. Information Technology — Security Techniques — Encryption Algorithms — Part 2: Asymmetric Ciphers, 2026. ISO/IEC 18033-2 Amendment 2.

[125] P. Kampanakis, D. Stebila, and T. Hansen. RFC 10042: Post-quantum/traditional hybrid key exchange with module-lattice key-encapsulation mechanism for use in SSH, Aug. 2026. Informational. IETF.

[126] R. Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, 12:415–440, 1987.

[127] R. Kannan and L. Lovász. Covering minima and lattice-point-free convex bodies. *Annals of Mathematics*, 128:577–602, 1988.

[128] S. Khot. Hardness of approximating the shortest vector problem in lattices. *Journal of the ACM*, 52:789–808, 2005.

[129] S. Kim and A. Venkatesh. The behaviour of random reduced bases. *International Mathematics Research Notices*, 2018:6442–6480, 2018.

[130] P. Kirchner and P. Fouque. An improved BKW algorithm for LWE with applications to cryptography and lattices. In *CRYPTO*, pages 43–62, 2015.

[131] P. Klein. Finding the closest lattice vector when it's unusually close. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 937–941, 2000.

[132] N. Koblitz, S. Samajder, P. Sarkar, and S. Singha. Concrete analysis of approximate ideal-SIVP to decision ring-LWE reduction. *Advances in Mathematics of Communications*, 18:1216–1258, 2024.

[133] H. Kosuge and K. Xagawa. The security of ML-DSA against fault-injection attacks. In *ASIACRYPT*, pages 641–674, 2025.

[134] E. Krahmer, P. Pessl, G. Land, and T. Güneysu. Correction fault attacks on randomized CRYSTALS-Dilithium. *IACR Trans. Cryptographic Hardware and Embedded Systems*, pages 174–199, 2024.

[135] K. Kwiatkowski, P. Kampanakis, B. Westerbaan, and D. Stebila. RFC 10024: Post-quantum traditional (PQ/T) hybrid key agreement mechanisms for TLS 1.3, Aug. 2026. IETF.

[136] T. Laarhoven, J. van de Pol, and B. de Weger. Solving hard lattice problems and the security of lattice-based cryptosystems, 2012. IACR Cryptology ePrint Archive, https://eprint.iacr.org/2012/533.

[137] J. Lagarias. Worst-case complexity bounds for algorithms in the theory of integral quadratic forms. *J. Algorithms*, 1:142–186, 1980.

[138] J. Lagarias and A. Odlyzko. Solving low-density subset sum problems. *Journal of the ACM*, 32:229–246, 1985.

[139] A. Langlois and D. Stehlé. Worst-case to average-case reduction for module lattices. *Designs, Codes and Cryptography*, 75:565–599, 2015.

[140] A. Langlois, D. Stehlé, and R. Steinfeld. GGHLite: More efficient multilinear maps from ideal lattices. In *EUROCRYPT*, pages 239–256, 2014.

[141] C. Lee, A. Pellet-Mary, D. Stehlé, and A. Wallet. An LLL algorithm for module lattices. In *ASIACRYPT*, pages 59–90, 2019.

[142] W. Lee, R. Zhao, R. Steinfeld, A. Sakzad, and S. Hwang. High throughput lattice-based signatures on GPUs: Comparing Falcon and Mitaka. *IEEE Trans. Parallel and Distributed Systems*, 35:675–692, 2024.

[143] D. Lehmer. On arccotangent relations for π. *The American Mathematical Monthly*, 45:657–664, 1938.

[144] A. Lenstra, H. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.

[145] H. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.

[146] J. Li and P. Nguyen. A complete analysis of the BKZ lattice reduction algorithm. *J. Cryptology*, 38, 2025.

[147] W. Li, H. Wei, S. Shen, H. Yang, W. Dai, and Y. Zhao. cuFalcon: An adaptive parallel GPU implementation for high-performance Falcon acceleration. *IEEE Trans. Parallel and Distributed Systems*, 37:1153–1167, 2026.

[148] X. Lin, M. Tibouchi, Y. Yu, and S. Zhang. Do not disturb a sleeping falcon: Floating-point error sensitivity of the Falcon sampler and its consequences. In *EUROCRYPT*, pages 213–244, 2025.

[149] R. Lindner and C. Peikert. Better key sizes (and attacks) for LWE-based encryption. In *The Cryptographers' Track at RSA Conference (CT-RSA)*, pages 319–339, 2011.

[150] Z. Liu, A. Wang, C. Wei, Y. Ding, J. Zhang, A. Liu, and L. Zhu. Release the power of rejected signatures: An efficient side-channel attack on the ML-DSA cryptosystem. *IEEE Trans. Info. Forensics and Security*, 20:13356–13369, 2025.

[151] V. Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In *ASIACRYPT*, pages 598–616, 2009.

[152] V. Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT*, pages 738–755, 2012.

[153] V. Lyubashevsky. Basic lattice cryptography: The concepts behind Kyber (ML-KEM) and Dilithium (ML-DSA), 2024. IACR Cryptology ePrint Archive, https://eprint.iacr.org/2024/1287.

[154] V. Lyubashevsky and D. Micciancio. Generalized compact knapsacks are collision resistant. In *ICALP*, pages 144–155, 2006.

[155] V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. *Journal of the ACM*, 60(6), 2013.

[156] S. Marzougui, N. Wisiol, P. Gersch, J. Krämer, and J. Seifert. Machine-learning side-channel attacks on the GALACTICS constant-time implementation of BLISS. In *Int'l Conf. on Availability, Reliability and Security (ARES)*, 2022.

[157] A. May. Using LLL-reduction for solving RSA and factorization problems, 2010. Chapter 10 in [181].

[158] A. Menezes, P. van Oorschot, and B. Preneel. Post-quantum cryptography, 2026. Chapter 15 in *Textbook of Applied Cryptography*, https://sites.google.com/view/textbook-appliedcrypto.

[159] D. Micciancio. The hardness of the closest vector problem with preprocessing. *IEEE Trans. Info. Theory*, 47:1212–1215, 2001.

[160] D. Micciancio. Efficient reductions among lattice problems. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 84–93, 2008.

[161] D. Micciancio and S. Goldwasser. *Complexity of Lattice Problems: A Cryptographic Perspective*. Kluwer, 2002.

[162] D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, pages 700–718, 2012.

[163] D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM J. Computing*, 37:267–302, 2007.

[164] D. Micciancio and P. Voulgaris. Faster exponential time algorithms for the shortest vector problem. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1468–1480, 2010.

[165] T. Mukherjee and N. Stephens-Davidowitz. Lattice reduction for modules: Or how to reduce ModuleSVP to ModuleSVP. In *CRYPTO*, pages 213–242, 2020.

[166] National Institute of Standards and Technology. Recommendation for stateful hash-based signature schemes, 2020. SP 800-208.

[167] National Institute of Standards and Technology. Module-lattice-based digital signature standard, 2024. FIPS 204.

[168] National Institute of Standards and Technology. Module-lattice-based key-encapsulation mechanism standard, 2024. FIPS 203.

[169] National Institute of Standards and Technology. Stateless hash-based digital signature standard, 2024. FIPS 205.

[170] National Institute of Standards and Technology. Recommendation for key-encapsulation mechanisms, 2025. SP 800-227.

[171] P. Nguyen. Cryptanalysis of the Goldreich-Goldwasser-Halevi cryptosystem from Crypto'97. In *CRYPTO*, pages 288–304, 1999.

[172] P. Nguyen. Hermite's constant and lattice algorithms, 2010. Chapter 2 in [181].

[173] P. Nguyen and O. Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. *J. Cryptology*, 22:139–160, 2009.

[174] P. Nguyen and I. Shparlinski. The insecurity of the Digital Signature Algorithm with partially known nonces. *J. Cryptology*, 15:151–176, 2002.

[175] P. Nguyen and I. Shparlinski. The insecurity of the elliptic curve digital signature algorithm with partially known nonces. *Designs, Codes and Cryptography*, 30:201–217, 2003.

[176] P. Nguyen and D. Stehlé. LLL on the average. In *Algorithmic Number Theory (ANTS)*, pages 238–256, 2006.

[177] P. Nguyen and D. Stehlé. An LLL algorithm with quadratic complexity. *SIAM J. Computing*, 39:874–903, 2009.

[178] P. Nguyen and D. Stehlé. Low-dimensional lattice basis reduction revisited. *ACM Transactions on Algorithms*, 5, 2009.

[179] P. Nguyen and J. Stern. Cryptanalysis of the Ajtai-Dwork cryptosystem. In *CRYPTO*, pages 223–242, 1998.

[180] P. Nguyen and J. Stern. The two faces of lattices in cryptography. In *Cryptography and Lattices*, pages 146–180, 2001.

[181] P. Nguyen and B. Vallée, editors. *The LLL Algorithm: Survey and Applications*. Springer, 2010.

[182] P. Nguyen and T. Vidick. Sieve algorithms for the shortest vector problem are practical. *J. Mathematical Cryptology*, 2:181–207, 2008.

[183] P. Nicholson. Algebraic theory of Finite Fourier Transforms. *J. Computer and Systems Sciences*, 5:524–547, 1971.

[184] A. Odlyzko and H. te Riele. Disproof of the Mertens conjecture. *J. für die Reine und Angewandte Mathematik*, 357:138–160, 1985.

[185] Y. Ouyang, Y. Zhu, W. Zhu, B. Yang, Z. Zhang, H. Wang, Q. Tao, M. Zhu, S. Wei, and L. Liu. FalconSign: An efficient and high-throughput hardware architecture for Falcon signature generation. *IACR Trans. Cryptographic Hardware and Embedded Systems*, pages 203–226, 2025.

[186] C. Peikert. A decade of lattice cryptography, 2015. IACR Cryptology ePrint Archive, https://eprint.iacr.org/2015/939.

[187] M. Pohst. On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications. *ACM SIGSAM Bulletin*, 15:37–44, 1981.

[188] M. Pohst. A modification of the LLL reduction algorithm. *J. Symbolic Computation*, 4:123–127, 1987.

[189] J. Pollard. The Fast Fourier Transform in a finite field. *Mathematics of Computation*, 25:365–374, 1971.

[190] T. Pornin. A RAM-efficient implementation of Falcon, 2026. IACR Cryptology ePrint Archive, https://eprint.iacr.org/2026/1915.

[191] T. Pornin and T. Prest. More efficient algorithms for the NTRU key generation using the field norm. In *Public Key Cryptography (PKC)*, pages 504–533, 2019.

[192] A. Pouly and Y. Shen. Solving the shortest vector problem in $2^{0.63269n+o(n)}$ time on random lattices. In *EUROCRYPT*, pages 92–123, 2026.

[193] T. Prest. *Gaussian Sampling in Lattice-Based Cryptography*. PhD thesis, École Normale Supérieure (France), 2015.

[194] P. Ravi, A. Chattopadhyay, J. D'Anvers, and A. Baksi. Side-channel and fault-injection attacks over lattice-based post-quantum schemes (Kyber, Dilithium): Survey and new results. *ACM Transactions on Embedded Computing Systems*, 23(2), 2024.

[195] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM*, 56(6):1–40, 2009.

[196] V. Reis and T. Rothvoss. The subspace flatness conjecture and faster integer programming. In *Symposium on Foundations of Computer Science (FOCS)*, pages 974–988, 2023.

[197] D. Rowe, J. Breitner, and N. Heninger. The curious case of the half-half Bitcoin ECDSA nonces. In *AFRICACRYPT*, pages 273–284, 2023.

[198] C. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical Computer Science*, 53:201–224, 1987.

[199] C. Schnorr. Block reduced lattice bases and successive minima. *Combinatorics, Probability and Computing*, 3:507–522, 1994.

[200] C. Schnorr. Lattice reduction by random sampling and birthday methods. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 145–156, 2003.

[201] C. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical Programming*, 66:181–199, 1994.

[202] A. Schönhage and V. Strassen. Schnelle Multiplikation großer Zahlen. *Computing*, 7:281–292, 1971.

[203] T. Shen and X. Li. A survey on classical lattice algorithms. *Cryptography*, 10(2), 2026.

[204] V. Shoup. OAEP reconsidered. *J. Cryptology*, 15:223–249, 2002.

[205] Z. Song. Hardness of Euclidean closest vector within $n^{1/2-\varepsilon}$ and binary nearest codeword within $n^{1-\varepsilon}$, 2026. available at https://dx.doi.org/10.20944/preprints202608.0796.v2.

[206] D. Stebila, S. Fluhrer, and S. Gueron. RFC 9954: Hybrid key exchange in TLS 1.3, July 2026. Informational. IETF.

[207] D. Stehlé. Floating-point LLL: Theoretical and practical aspects, 2010. Chapter 5 in [181].

[208] D. Stehlé and R. Steinfeld. Making NTRU as secure as worst-case problems over ideal lattices. In *EUROCRYPT*, pages 27–47, 2011.

[209] N. Stephens-Davidowitz. Dimension-preserving reductions between lattice problems, 2016. unpublished manuscript.

[210] I. Stewart and D. Tall. *Algebraic Number Theory and Fermat's Last Theorem*. A K Peters, 3rd edition, 2002.

[211] Z. Straznickas and S. Weis. HAWK-n key recovery reduces to SVP in dimension $n/2 + 1$, 2026. IACR Cryptology ePrint Archive, https://eprint.iacr.org/2026/1593.

[212] V. Ulitzsch, S. Marzougui, A. Bagia, M. Tibouchi, and J. Seifert. Loop aborts strike back: Defeating fault countermeasures in lattice signatures with ILP. *IACR Trans. Cryptographic Hardware and Embedded Systems*, pages 367–392, 2023.

[213] B. Vallée. Gauss' algorithm revisited. *J. Algorithms*, 12:556–572, 1991.

[214] M. Viazovska. The sphere packing problem in dimension 8. *Annals of Mathematics*, 185:991–1015, 2017.

[215] D. Wan. NP-hardness of SVP in Euclidean space, 2026. arXiv:2603.27398.

[216] E. Wenger, E. Saxena, M. Malhou, E. Thieu, and K. Lauter. Benchmarking attacks on Learning with Errors. In *IEEE Symp. Security and Privacy*, pages 279–297, 2025.

[217] M. Yasuda. A survey of solving SVP algorithms and recent strategies for solving the SVP challenge. In *International Symposium on Mathematics, Quantum Theory, and Cryptography*, pages 189–207, 2021.

[218] J. Zhang, Y. Yan, J. Huang, and Ç. Koç. Optimized software implementation of Keccak, Kyber, and Dilithium on RV{32,64}IM{B}{V}. *IACR Trans. Cryptographic Hardware and Embedded Systems*, pages 632–655, 2025.

[219] S. Zhang, X. Lin, Y. Yu, and W. Wang. Improved power analysis attacks on Falcon. In *EUROCRYPT*, pages 565–595, 2023.

[220] Z. Zhao and J. Ding. Towards large-scale lattice attack: New lattice records by disk-based sieving. In *EUROCRYPT*, pages 154–182, 2026.

[221] Z. Zhao, J. Ding, and B. Yang. Sieving with streaming memory access. *IACR Trans. Cryptographic Hardware and Embedded Systems*, pages 362–384, 2025.

[222] Y. Zhou, W. Wang, Y. Sun, and Y. Yu. Rejected signatures' challenges pose new challenges: Key recovery of CRYSTALS-Dilithium via side-channel attacks. *IACR Trans. Cryptographic Hardware and Embedded Systems*, pages 817–847, 2025.