On the Design and Improvement of Lattice-based Cryptosystems

Vom Fachbereich Informatik der
Technischen Universität Darmstadt genehmigte

Dissertation

zur Erlangung des Grades
Doktor rerum naturalium (Dr. rer. nat.)

von

Dipl.-Wi. Inform. Rachid El Bansarkhani
geboren in Rüsselsheim.

Referenten:	Prof. Dr. Johannes Buchmann
	Prof. Dr. Ing. Tim Güneysu
Tag der Einreichung:	29.04.2015
Tag der mündlichen Prüfung:	10.06.2015
Hochschulkennziffer:	D 17

Darmstadt 2015

Wissenschaftlicher Werdegang

Oktober 2011 - heute

Wissenschaftlicher Mitarbeiter und Promotionsstudent am Lehrstuhl von Professor Johannes Buchmann, Fachbereich Informatik, Fachgebiet Theoretische Informatik - Kryptographie und Computeralgebra, Technische Universität Darmstadt

Mitarbeit an wissenschaftlichen Projekten (gefördert durch das BMBF)

- **Sinnodium** (Februar 2013 - Mai 2015)
  Absicherung der Interaktion zwischen emergenten Diensten und mobilen Endgeräten
- **Software Campus** (Januar 2013 - Dezember 2014)
  Gitterbasierte Kryptografie für die Zukunft
- **Emergent** (Oktober 2011 - Juni 2013)
  Technologien für die Umsetzung von Policies für emergente Software

Oktober 2005 - Dezember 2010

Studium der Wirtschaftsinformatik an der Technischen Universität Darmstadt

April 2008 - Juni 2011

Studium der Mathematik mit Nebenfach Informatik an der Technischen Universität Darmstadt

Publikationsliste

[P1] Rachid El Bansarkhani, Özgür Dagdelen, and Johannes Buchmann. Augmented learning with errors: The untapped potential of the error term. In *Financial Crypto 2015*, *LNCS*. Springer, 2015.

[P2] Rachid El Bansarkhani and Johannes Buchmann. High performance lattice-based CCA-secure encryption, Submitted, Cryptology ePrint Archive, Report 2015/042, 2015. http://eprint.iacr.org/.

[P3] Özgür Dagdelen, Rachid El Bansarkhani, Florian Göpfert, Tim Güneysu, Tobias Oder, Thomas Pöppelmann, Ana Helena Snchez, and Peter Schwabe. High-speed signatures from standard lattices. In Diego F. Aranha and Alfred Menezes, editors, *LATINCRYPT 2014*, volume 8895 of *LNCS*, pages 84–103. Springer, 2015.

[P4] Michael Riecker, Sebastian Biedermann, Rachid El Bansarkhani, and Matthias Hollick. Lightweight energy consumption-based intrusion detection system for wireless sensor networks, International Journal of Information Security 2014, volume 14, pages 155–167. Springer, 2014.

[P5] Rachid El Bansarkhani and Johannes Buchmann. LCPR: High performance compression algorithm for lattice-based signatures, Submitted, Cryptology ePrint Archive, Report 2014/334, 2014. http://eprint.iacr.org/.

[P6] Rachid El Bansarkhani and Johannes Buchmann. Towards lattice-based sequential aggregate signatures. In David Pointcheval and Damien Vergnaud, editors, *AFRICACRYPT 2014*, volume 8469 of *LNCS*, pages 336–355. Springer, 2014.

[P7] Michael Riecker, Dingwen Yuan, Rachid El Bansarkhani, and Matthias Hollick. Patrolling wireless sensor networks: Randomized intrusion detection. In *10th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, Q2SWinet '14, pages 61–69. ACM, 2014

[P8] Rachid El Bansarkhani, Sascha Hauke, and Johannes Buchmann. Towards security solutions for emergent business software. In Gino Brunetti, Thomas Feld, Lutz Heuser, Joachim Schnitter, and Christian Webel, editors, *Future Business Software*, Progress in IS, pages 67–80. Springer, 2014.

[P9] Rachid El Bansarkhani and Johannes Buchmann. Improvement and efficient implementation of a lattice-based signature scheme. In Lange Tanja, Kristin Lauter, and Petr Lisonek, editors, *Selected Areas in Cryptography*, volume 8282 of *LNCS*, pages 48–67. Springer, 2013.

[P10] Sidi Mohamed El Yousfi Alaoui, Pierre-Louis Cayrel, Rachid El Bansarkhani, and Gerhard Hoffmann. Code-based identification and signature schemes in software. In Alfredo Cuzzocrea, Christian Kittl, Dimitris E. Simos, Edgar Weippl, and Lida Xu, editors, *Security Engineering and Intelligence Informatics*, volume 8128 of *LNCS*, pages 122–136. Springer, 2013.

[P11] Mohammed Meziani and Rachid El Bansarkhani. An efficient and secure coding-based authenticated encryption scheme. In Roberto Di Pietro, Javier Herranz, Ernesto Damiani, and Radu State, editors, *Data Privacy Management and Autonomous Spontaneous Security*, volume 7731 of *LNCS*, pages 43–60. Springer, 2013.

[P12] Rachid El Bansarkhani and Mohammed Meziani. An efficient lattice-based secret sharing construction. In Ioannis Askoxylakis, Henrich C. Pöhls, and Joachim Posegga, editors, *Information Security Theory and Practice. Security, Privacy and Trust in Computing Systems and Ambient Intelligent Ecosystems*, volume 7322 of *LNCS*. Springer, 2012.

[P13] Rachid El Bansarkhani and Johannes Buchmann. Efficient lattice-based encryption via A-LWE in the standard model, Submitted, 2015

[P14] Rachid El Bansarkhani and Johannes Buchmann. Representation formulas for lattice problems via Cauchy integrals, Submitted, 2015

Acknowlegdement

I would like to thank god, my family and all people I met and worked with in the past years. Especially, I wish to express my deep gratitude to my supervisor Johannes Buchmann for all aspects throughout the past three years of my graduation. His great lectures and presentations inspired me and arouse my interest for cryptography in the first place. I am very grateful for his great support, experience, interesting discussions, and helpful advices in scientific and strategic matters that encouraged me to pursue my scientific goals and moreover to be involved in various challenging projects. Furthermore, I am very grateful for having my co-referee Tim Güneysu as well as Matthias Hollick, Max Mühlhäuser, and Melanie Vollkamer on my PhD committee.

At this point, I also wish to express my deep gratitude particularly to my beloved parents for everything throughout my life, which cannot be captured in words. In this regard, I also thank my whole family for encouraging and supporting me in any aspect.

Finally, I wish to express my sincere thanks to all my friends and colleagues from CDC and CASED. Special thanks to Mohammed Saied, Johannes Braun, Özgür Dagdelen, Juliane Krämer, Florian Göpfert, Mohamed El Yousfi, Sedat Akleylek, Michael Schneider, Nina Bindel, Mohammed Meziani, Sascha Hauke, Patrick Weiden, Michael Riecker, Stephan Neumann, Jurlind Budurushi and many more. I had the opportunity to work with many different and interesting people with different backgrounds on very intriguing topics in lattice-based cryptography. It was a very fascinating and motivating atmosphere at CDC with kind and humorous people. It was also a great pleasure to have many long-term guests from England, China, Taiwan and Turkey.

”Over every possessor
of knowledge is one
more knowing.”

”And of knowledge,
you have been given
only a little.”

Darmstadt,
June 2015

Rachid El Bansarkhani

Zusammenfassung

Digitale Signatur- und Verschlüsselungsalgorithmen bilden einen wesentlichen Bestandteil von kryptografischen Verfahren mit dem Ziel, die Sicherheitsbedürfnisse von gegenwärtigen und zukünftigen Privat- und Geschäftsanwendungen zu erfüllen. Jedoch sind alle in der Praxis eingesetzten asymmetrischen Verfahren aufgrund der Anfälligkeit für Quantencomputer-Angriffe infolge Shors Quantenalgorithmus gefährdet. Das Ausmaß der wirtschaftlichen und gesellschaftlichen Auswirkungen sind gewaltig, wodurch unmittelbar die Forderung nach Alternativen besteht, die klassische Systeme ersetzen, sobald Quantencomputer im großen Maßstab hergestellt werden können. Gitterbasierte Kryptografie ist als leistungsstarke Alternative hervorgetreten, die die Aufmerksamkeit der Forscher nicht nur wegen der vermuteten Resistenz gegen Quantencomputer-Angriffe auf sich zieht, sondern auch wegen ihrer einzigartigen Sicherheitsgarantie der Worst-Case-Härte von Average-Case-Instanzen. Auf diese Weise entfällt die Notwendigkeit, gesonderte Annahmen über die Average-Case Härte zu formulieren, sodass praktische Instanziierungen in der Tat Sicherheitsgarantien von Worst-Case-Instanzen genießen. Die bekanntesten Gitterangriffsalgorithmen laufen mit exponentieller Zeitkomplexität.

In dieser Arbeit tragen wir zu einem reibungslosen Übergang in eine Welt mit praktikablen gitterbasierten Verfahren bei. Dies wird durch die Entwicklung von neuen Algorithmen und kryptographischen Verfahren sowie die Verbesserung bestehender erreicht. Unsere Beiträge sind dreigeteilt.

Erstens, wir stellen neue Verschlüsselungsverfahren vor, die den Fehlerterm bei LWE-Instanzen vollständig ausschöpfen, um den Nachrichtendurchsatz signifikant zu erhöhen. Zu diesem Zweck führen wir ein neues Berechnungsproblem ein, das wir als Augmented LWE (A-LWE) bezeichnen und das sich vom ursprünglichen LWE-Problem nur in der Weise unterscheidet, wie der Fehlerterm erzeugt wird. In der Tat können beliebige Daten in den Fehlerterm eingebettet werden, ohne die Zielverteilungen zu verändern. Im Anschluss daran erfolgt ein Beweis, dass A-LWE-Instanzen ununterscheidbar von LWE-Instanzen sind und demnach auf der Schwierigkeit des LWE-Problems beruhen. Dies erlaubt es, leistungsstarke Verschlüsselungsverfahren auf Grundlage des A-LWE-Problems zu konstruieren, die einfach in der Darstellung und effizient in der Praxis sind, während gleichzeitig große Datenmengen verschlüsselt werden können, sodass Nachrichten-Expansionsfaktoren nahe 1 praktisch erreicht werden. Dies verbessert unseres Wissens nach alle bestehenden Verschlüsselungsverfahren. Aufgrund der Vielseitigkeit des Fehlerterms können weitere Zusatzeigenschaften hinzugefügt werden wie etwa CCA- bzw. RCCA-Sicherheit. Aber auch gitterbasierte Signaturen können als Teil des Fehlerterms fungieren und erweitern somit das Verschlüsselungsverfahren um einen weiteren Mechanismus, der

die Authentifikation von verschlüsselten Daten auf einfache Weise realisiert. Die Methodik zur Erzeugung des Fehlerterms bei A-LWE-Instanzen hat ebenfalls einen konzeptuell neuen und effizienten Diskret-Gauß-Sampler hervorgebracht, der die bekanntesten Verfahren, wie z.B. Knuth-Yao oder den CDT-Sampler auf Basis der Inversionsmethode, in Bezug auf die Leistungsfähigkeit übertrifft. Zur Laufzeit wird ein Wert von einer Tabelle der konstanten Größe von maximal 44 Elementen für beliebige Gauß-Parameter gesampelt. Der Gesamtspeicherbedarf beläuft sich auf die Größe der Tabelle beim bekannten CDT-Sampler. Weitere Ergebnisse beinhalten einen sehr effizienten Inversionsalgorithmus für Ringelemente in speziellen Klassen von Kreisteilungsringen. Durch die Verwendung der NTT ist es möglich, die Existenz von Inversen zu gegebenen Ringelementen effizient zu überprüfen und zu bestimmen. Eine Darstellung der entsprechenden Einheitengruppe lässt sich auf diese Weise unkompliziert und anschaulich ermitteln. Außerdem verallgemeinern wir den LWE-Inversionsalgorithmus für die Falltürkonstruktion von Micciancio und Peikert von Zweierpotenz-Moduli auf beliebig zusammengesetzte Zahlen.
Im zweiten Teil dieser Arbeit präsentieren wir eine effiziente Falltürkonstruktion für Ideal-Gitter und eine zugehörige Beschreibung des GPV-Signaturverfahrens. Durch eine verbesserte Darstellung der assoziierten Fehlermatrix kann der Signiervorgang im Vergleich zur ursprünglichen Arbeit erheblich vereinfacht werden. Dies wirkt sich unmittelbar durch eine stark optimierte Speichernutzung und Laufzeit aus. Anschließend schlagen wir einen neuartigen Kompressionsalgorithmus für GPV-Signaturen vor, die bisher als Ergebnis der Falltürkonstruktion bzw. der Anforderungen des Sicherheitsbeweises einen zu hohen Speicherverbrauch aufwiesen. Wir umgehen dieses Problem mit der Einführung des Begriffs der öffentlichen und geheimen Zufälligkeit für Signaturen. Der öffentliche Teil einer Signatur kann demnach von einer kurzen und gleichverteilten Bitfolge erzeugt werden, ohne die vorherigen Bedingungen zu verletzen. Dieses Konzept wird anschließend auf die Situation mit mehreren Teilnehmern erweitert, wodurch sich die Effizienz und der Wirkungsgrad des Kompressionsverfahrens erhöht. Schließlich schlagen wir das erste gitterbasierte und sequenzielle Aggregationsverfahren für Signaturen vor, das einer Gruppe von Teilnehmern ermöglicht, sequenziell eine aggregierte Signatur zu erzeugen, dessen Größe sich im Vergleich zur ursprünglichen Gesamtgröße aller Signaturen sehr stark reduziert hat. Der Prüfer ist jederzeit in der Lage zu verifizieren, dass jeder Teilnehmer eine Nachricht tatsächlich signiert hat. Dieser Ansatz wird mittels gitterbasierter Falltürkonstruktionen realisiert und hat viele Anwendungsbereiche.
Im letzten Teil dieser Arbeit werden theoretische Aspekte von gitterbasierten Problemen behandelt. Es werden neue Repräsentationen bzw. Relationen von interessanten Gitterproblemen vorgestellt, die auf Basis von Cauchy-Integralen hergeleitet werden. Betrachtet man Gitterpunkte als einfache Pole von komplexen Funktionen, so ist es prinzipiell möglich über Cauchy Integrale und ihren Verallgemeinerungen auf Gitterpunkte zu operieren. Beispielsweise lassen sich für den ein- und zweidimensionalen Fall, ebenfalls relevante Szenarien in kryptografischen Anwendungen, einfache Ausdrücke bzw. Formeln für die Anzahl von Gitterpunkten in einem Gebiet via trigonometrischen und elliptischen Funktionen ableiten.

Abstract

Digital signatures and encryption schemes constitute arguably an integral part of cryptographic schemes with the goal to meet the security needs of present and future private and business applications. However, almost all public key cryptosystems applied in practice are put at risk due to its vulnerability to quantum attacks as a result of Shor's quantum algorithm. The magnitude of economic and social impact is tremendous inherently asking for alternatives replacing classical schemes in case large-scale quantum computers are built. Lattice-based cryptography emerged as a powerful candidate attracting lots of attention not only due to its conjectured resistance against quantum attacks, but also because of its unique security guarantee to provide worst-case hardness of average-case instances. Hence, the requirement of imposing further assumptions on the hardness of randomly chosen instances disappears, resulting in more efficient instantiations of cryptographic schemes. The best known lattice attack algorithms run in exponential time. In this thesis we contribute to a smooth transition into a world with practically efficient lattice-based cryptographic schemes. This is indeed accomplished by designing new algorithms and cryptographic schemes as well as improving existing ones. Our contributions are threefold.

First, we construct new encryption schemes that fully exploit the error term in LWE instances. To this end, we introduce a novel computational problem that we call Augmented LWE (A-LWE), differing from the original LWE problem only in the way the error term is produced. In fact, we embed arbitrary data into the error term without changing the target distributions. Following this, we prove that A-LWE instances are indistinguishable from LWE samples. This allows to build powerful encryption schemes on top of the A-LWE problem that are simple in its representations and efficient in practice while encrypting huge amounts of data realizing message expansion factors close to 1. This improves, to our knowledge, upon all existing encryption schemes. Due to the versatility of the error term, we further add various security features such as CCA and RCCA security or even plug lattice-based signatures into parts of the error term, thus providing an additional mechanism to authenticate encrypted data. Based on the methodology to embed arbitrary data into the error term while keeping the target distributions, we realize a novel CDT-like discrete Gaussian sampler that beats the best known samplers such as Knuth-Yao or the standard CDT sampler in terms of running time. At run time the table size amounting to 44 elements is constant for every discrete Gaussian parameter and the total space requirements are exactly as large as for the standard CDT sampler. Further results include a very efficient inversion algorithm for ring elements in special classes of cyclotomic rings. In fact, by use of the NTT it is possible to efficiently

check for invertibility and deduce a representation of the corresponding unit group. Moreover, we generalize the LWE inversion algorithm for the trapdoor candidate of Micciancio and Peikert from power of two moduli to arbitrary composed integers using a different approach.
In the second part of this thesis, we present an efficient trapdoor construction for ideal lattices and an associated description of the GPV signature scheme. Furthermore, we improve the signing step using a different representation of the involved perturbation matrix leading to enhanced memory usage and running times. Subsequently, we introduce an advanced compression algorithm for GPV signatures, which previously suffered from huge signature sizes as a result of the construction or due to the requirement of the security proof. We circumvent this problem by introducing the notion of public and secret randomness for signatures. In particular, we generate the public portion of a signature from a short uniform random seed without violating the previous conditions. This concept is subsequently transferred to the multi-signer setting which increases the efficiency of the compression scheme in presence of multiple signers. Finally in this part, we propose the first lattice-based sequential aggregate signature scheme that enables a group of signers to sequentially generate an aggregate signature of reduced storage size such that the verifier is still able to check that each signer indeed signed a message. This approach is realized based on lattice-based trapdoor functions and has many application areas such as wireless sensor networks.
In the final part of this thesis, we extend the theoretical foundations of lattices and propose new representations of lattice problems by use of Cauchy integrals. Considering lattice points as simple poles of some complex functions allows to operate on lattice points via Cauchy integrals and its generalizations. For instance, we can deduce for the one-dimensional and two-dimensional case simple expressions for the number of lattice points inside a domain using trigonometric or elliptic functions.

Contents

1. Introduction **1**
1.1. Summary of Results 3

2. Preliminaries **8**
2.1. Notation 8
2.2. Lattices 8
2.3. Discrete Gaussian Distribution 9
2.4. Computational Problems 10
2.5. Cryptographic Primitives 12
2.5.1. Encryption Schemes 12
2.5.2. Digital Signature Schemes 14

I. Lattice-based Encryption **16**

3. Augmented LWE and its Hardness **19**
3.1. Main Obstacles 19
3.2. Our Contribution 20
3.3. Learning with Errors Augmented with Auxiliary Data 22
3.3.1. Message Embedding 22
3.3.2. Augmented LWE - A Generic Approach 25
3.4. Our Construction in the Random Oracle Model 27
3.4.1. A-LWE Distribution 27
3.4.2. A-LWE Hardness 28
3.5. Our Construction in the Standard Model 30
3.5.1. Tools 31
3.5.2. A-LWE Distribution 33
3.5.3. A-LWE Hardness 34

4. Building Lattice-based Encryption Schemes from A-LWE **37**
4.1. Maximum Data Size 42
4.1.1. Intersection Method 43
4.1.2. Lattices of the Form $\mathbf{p} \cdot \mathbb{Z}^{\mathbf{m}}$ 43
4.1.3. Uniform Error 45
4.2. Our Generic Construction 45
4.2.1. High Data Load Encryption (HDL Mode) 47
4.2.2. Improved Message Throughput 47

4.2.3. Optimized Generic Encryption Scheme from A-LWE 49
4.3. CCA-secure Encryption Scheme 51
4.3.1. CCA1 secure Encryption 51
4.4. pd-RCCA-secure Encryption Scheme 54
4.5. CCA2-secure Encryption Scheme 57
4.6. Asymmetric Authenticated Encryption Scheme 58
4.7. Improvement of Existing Schemes 60
4.7.1. Enhancing Existing Symmetric-Key Encryption Scheme . . . 60
4.7.2. Enhancing an Existing CCA1-secure Encryption Scheme . . . 62

5. CCA-secure Encryption Scheme from A-LWE in Practice 63
5.1. A Fast Discrete Gaussian Sampler - FastCDT 65
5.2. Techniques 71
5.2.1. Setting 71
5.2.2. Instantiation from Trapdoors for Ideal-Lattices 72
5.2.3. CCA-secure Encryption Scheme − Ring Variant 75
5.2.4. LWE Inversion for Arbitrary Modulus 75
5.2.5. Fast Tag Generation and Inversion 77
5.3. Security Analysis 78
5.3.1. Embedding Approach 79
5.3.2. Analysis of Key Recovery Attacks 80
5.4. Software Implementation and Performance Analysis 81
5.4.1. Software Implementation and Optimization 81
5.5. Implementation 83
5.5.1. Implementation Analysis 83

II. Lattice-based Signatures 91

6. Improvement of GPV Signatures 94
6.1. GPV Signature Scheme 94
6.1.1. Trapdoor Functions 94
6.1.2. Full-Domain Hash Scheme 95
6.1.3. Probabilistic Full-Domain Hash Scheme 96
6.2. Instantiation of the GPV Signature Scheme 96
6.2.1. Trapdoors for the Matrix Setting 96
6.2.2. Trapdoors for Ideal-Lattices 99
6.3. Improvements and Optimizations 102
6.3.1. Computation of the Covariance Matrix 102
6.3.2. Estimating relevant Parameters 102
6.3.3. Generation of Perturbation Vectors 102
6.3.4. Square Root Computation 103
6.3.5. Optimized Signature Scheme 105
6.4. Security and Parameters 108

6.5. Implementation 108
6.5.1. Implementation using Standard Libraries 109
6.5.2. Optimized Implementation 110
6.6. Experimental Results 112

7. Compression Scheme for Signatures 116
7.1. Methodology of Compressing Schnorr-like Signatures 116
7.2. Generic Lossless Compression of Schnorr-like Signatures 122
7.2.1. Lossless Compression Algorithm 123
7.2.2. Analysis 125
7.2.3. Security 127
7.2.4. Compression Rate of Individual Signatures 129
7.3. Compression Scheme for GPV Signatures 130
7.3.1. Tools 130
7.3.2. Conditional Rejection Sampling 134
7.3.3. Single-Signer Compression Scheme in the GPV Setting . . . 134
7.3.4. Analysis of Compressed Signatures 135
7.3.5. Entropy of Public and Secret Randomness 138
7.4. Implementation and Experimental Results 139
7.5. Generic Multi-Signer Compression Strategy 141
7.5.1. Multi-Signer Compression Scheme 142
7.5.2. Multi-Signer Compression Scheme in the GPV Setting 145
7.6. Application Scenario - Cluster-based Aggregation in Wireless Sensor Networks 147

8. Sequential Aggregate Signatures 148
8.1. Our Construction 150
8.1.1. Our Basic Signature Scheme 150
8.1.2. Informal Description 152
8.2. Security Model 153
8.2.1. Security of our Construction 155
8.3. Instantiation 158
8.3.1. Comparison with RSA-based SAS 160
8.3.2. Analysis 160
8.3.3. Proxy Signatures 161

III. Lattice Representations 162

9. Representation Formula for Lattice Problems 163
9.1. Cauchy Integrals 164
9.1.1. Complex Space 164
9.1.2. Integral Formulas 165

9.2. Representation Formulas for Lattice Problems 167
9.2.1. Number of lattice points inside a domain 168
9.2.2. One-dimensional Lattices 170
9.2.3. Two-dimensional Lattices 174
9.2.4. CVP Representation Formula for Arbitrary Lattices 176

References **180**

List of Algorithms

1. Building CDT Arrays 68
2. FastCDT 68
3. CDT Sampling 69

4. Sampling from $\Lambda_q^{\perp}(\mathbf{G})$ 98
5. Cholesky Decomposition 103

6. Compression by Signature 123
7. Compression by Seed 123
8. AS Scheme: AggSign 144
9. Verification: AggVerify 144
10. MS Compression: MCSign 146
11. Verification: MCVerify 146

12. Signing Algorithm: $\mathsf{AggSign}(\mathbf{T_i}, m_i, \Sigma_{i-1})$ 151
13. Verification Algorithm: $\mathsf{AggVerify}(\Sigma_k)$ 151

List of Figures

4.1. Basic Encryption Scheme 46
4.2. Optimized Generic Encryption Scheme 50
4.3. CCA1-secure Encryption Scheme 52
4.4. pd-RCCA-secure Encryption Scheme 55
4.5. Asymmetric Authenticated Encryption Scheme 59
4.6. Improved Symmetric Key Encryption Scheme 61

5.1. Trapdoor Generation Algorithm 73
5.2. CCA1-secure Encryption Scheme - Ring Variant 74

6.1. Basic GPV Signature Scheme 97
6.2. GPV Signature Scheme - Matrix Variant 106
6.3. GPV Signature Scheme - Ring Variant 107
6.4. Breakdown of Signing 115

7.1. Centroids Surrounded by Signatures 118
7.2. Compressed and Uncompressed Signatures 119
7.3. Lossless Compression Algorithms 123
7.4. Aggregate Signature Scheme 144
7.5. Multi-Signer Compression Scheme in the GPV Setting 146

9.1. Point in a Region 165
9.2. Number of Lattice Points in a Region 169
9.3. Number of Lattice Points in a Circle 169
9.4. Voronoi Cell of a Lattice 176

List of Tables

4.1. Parameters 40

5.1. Comparison of Discrete Gaussian Samplers 71
5.2. Parameters 84
5.3. Discrete Gaussian Error in the HDL mode with $\mathbf{A} \approx_c \mathcal{U}(\mathbb{R}_q^{l+m})$ . . . 86
5.4. Discrete Gaussian Error in the HDL mode with $\mathbf{A} \approx_s \mathcal{U}(\mathbb{R}_q^{l+m})$ . . . 87
5.5. Uniform Error in the HDL mode with $\mathbf{A} \approx_c \mathcal{U}(\mathbb{R}_q^{l+m})$ 88
5.6. Uniform Error in the HDL mode with $\mathbf{A} \approx_s \mathcal{U}(\mathbb{R}_q^{l+m})$ 89

6.1. Parameter Sets and Security Levels 108
6.2. Generic Expressions for Storage Sizes 113
6.3. Timings for GPV 114
6.4. Storage Sizes for GPV 114

7.1. Compression Rate Highlights 118
7.2. Compression Rates 141

1. Introduction

The existence of computationally hard problems represents a necessary requirement for the possibility to build cryptography on top of it. The ultimate goal is to base the security of cryptographic applications on the intractability of hard computational problems. Nowadays, cryptography emerged as an important tool in order to protect all areas of life from unauthorized access and manipulations. The economic and social importance of cryptography drastically increased as it is applied to meet the future security needs of private and business applications. In particular, as a reaction to information superiority ambitions of various entities in the world, the role of cryptography has intensified over the past few years extending its application area to virtually unknown territory which can be attributed amongst others to the effects of globalization and the associated interconnections. Digital signature schemes and encryption schemes belong to the most common cryptographic primitives used in practice with a wide range of applications such as home banking, e-government, financial services, software updates, internet, and software security solutions, just to name a few examples. To exemplify this issue more intelligibly, the number of exchanged signatures per day via the TLS/SSL internet protocol amounts to more than billions of signatures. Hence, the impact of a sudden threat to public key cryptosystems applied today is disastrous particularly for economy and thus for the stability of our highly interwoven structures. Such a threat can be induced by novel outstanding algorithms or new technologies such as quantum computers. From a strategical point of view, a preference for risk aversion inherently asks to hedge against unpredictable threats using different technologies preferably based on unrelated computational problems. This diversification strategy reduces the unsystematic risk. The seminal work of Shor in 1994 [Sho97] shows that such a threat already became reality since it is theoretically possible to break all applied public key cryptosystems using Shor's factoring algorithm. In particular, he proposed quantum algorithms that can find the order of a group in probabilistic polynomial time by means of powerful quantum computers. Consequently, all factoring and discrete log based systems are vulnerable to this type of attacks. Shor's factoring algorithm belongs to the complexity class BQP containing all problems that can be solved in quantum polynomial time with an error probability bounded by 1/3.

Quantum computers operate on so-called qubits which differ from the traditional bit representations in such a way that a function can be evaluated at the superposition of all possible values in the range. However, building large scale quantum computers with a sufficiently large number of qubits in order to fully exploit already existing algorithms is a difficult task and hence remains an ongoing research objective. This is due to the sensibility of quantum states to extraneous influence and

interaction with the environment. It is still a technical hard problem to preserve quantum states for a long time period. Despite that, for different reasons much efforts and ressources are spent in order to realize a practical quantum computer. It is believed to have in approximately two decades a first prototype. This observation induced the search for alternatives replacing classical schemes in the near future. The most popular candidates found in the literature are hash-based, multivariate, code-based and lattice-based schemes, each relying on different hardness assumptions. The latter approach has a long history and is unique in its security properties, thus, traded as a promising alternative.
Lattices are well studied mathematical objects hiding a rich combinatorial structure. Formally speaking, an n-dimenional lattice is an additive subgroup of a Euclidean vector space $\mathbb{R}^n$ that geometrically corresponds to the intersection points of an n-dimensional grid. Due to its simplicity and geometrical representation, the application areas of lattices are steadily increasing, ranging from cryptography to communication theory and combinatorial optimization. Much research has been spent on investigating the problems arising from lattices such as CVP, SVP and SIVP, just to name a few examples. Briefly speaking, the closest vector problem (CVP) asks to find a lattice point $\mathbf{x} \in \Lambda$ of a lattice Λ with minimum distance $\min \|\mathbf{x} - \mathbf{t}\|$ to a target point $\mathbf{t} \in \mathbb{R}^n$, whereas for the shortest vector problem (SVP) it is required to find a non-zero lattice point $\mathbf{x} \in \Lambda \backslash \{\mathbf{0}\}$ of minimum length $\lambda_1 = \min \|\mathbf{x}\|$. The shortest independent vector problem (SIVP), on the other hand, asks to find a basis that is as short as possible. But also from a practical point of view, these problems are of great interest since they are extensively exploited for applications such as factoring polynomials over the rationals [LLL82], integer programming [Kan87, Len83], vector quantization [CS98], construction (e.g. [AD97]) and attacking [Odl90] of cryptographic schemes and many other application areas related to computer science, communication theory and mathematics.

Interestingly, there exists an inherent relationship between these problems and they are hence subject to intense research. Many works have a focus on specifying the complexity class of lattice problems. In fact, it has been shown that the computational problems SVP, CVP and SIVP are NP-hard for exact solutions [Ajt98, BS99, vEB81] and even for approximated solutions with subpolynomial approximation factors [BS99, CN99, DKRS03, Kho04, Mic98]. This implies, however, that the time complexity for algorithms developed to find a solution to these problems is expected to be non-polynomial in the lattice dimension n. Nevertheless, many algorithms have been proposed in order to solve and analyse the corresponding lattice problems [AKS01, AKS02, Hel85, Kan87, MV10, LLL82, SE94, Bl0, Sch86, Sch87]. This led to major algorithmic improvements and novel tools used, for instance, to estimate the security of cryptographic schemes and to improve existing algorithms.
Lattice-based cryptography attracted a lot of attention in recent years as a result of a sequence of breakthrough works [Ajt96, LLL82, Reg05] yielding new cryptographic constructions. This observation is supported by various arguments such as the conjectured resistance against quantum attacks. As opposed to classical schemes

like RSA, ECDSA and DSA, lattice-based cryptography is acclaimed by its unique security guarantee to provide worst-case hardness of average-case instances. In particular, Ajtai [Ajt96] gave the first such kind of reduction from worst-case lattice problems to the average-case problem SIS. It was shown that solving SIS for certain parameters is as hard as approximating SIVP to within polynomial factors. Such a relationship between average-case and worst-case problems represents a major cornerstone in cryptography in general as it relieves cryptographers from imposing new assumptions on the hardness of average-case instances used to instantiate practical schemes. By this means, lattice-based constructions are built on top of average-case problems while enjoying worst-case hardness, hence, taking the best of both worlds. Later, Regev [Reg04] introduced a second average-case problem, called the learning with errors problem (LWE), which admits a similar worst-case to average-case relationship and is applied predominantly in lattice-based encryption schemes.

1.1. Summary of Results

The relevant background of this thesis is given in Chapter 2. In particular, it serves to introduce the theoretical foundations of lattices as well as basic notations, definitions and major concepts applied in the remainder of this thesis. The results of our research are presented in Chapter 3 - 9. Basically, it can be divided into three parts with two equally-sized blocks, where the first block consisting of Chapters 3 - 5 is focused on lattice-based encryption and the second block composed of Chapters 6 - 8 encompasses our contributions to lattice-based signatures. The last chapter is devoted to our contributions on lattice theory in general. A brief summary of each chapter is given below.

Chapter 3 - Augmented LWE and Its Hardness

The majority of lattice-based primitives from Cryptomania require the intractability of the Learning with Errors problem as a basic underlying assumption. Many new LWE variants have been proposed with a reduction from the basic LWE problem. As an advantage, this allows to instantiate new cryptographic schemes with certain properties more efficiently while enjoying hardness of the LWE problem.
Interestingly, cryptographic primitives based on LWE often do not exploit the full potential of the error term beside of its importance for security. To this end, we introduce a novel LWE-close assumption [P1, P2, P13], namely Augmented Learning with Errors (A-LWE), which allows to hide auxiliary data injected into the error term by a technique that we call message embedding. Any party knowing the secret is subsequently able to extract the embedded data. We prove in the random oracle model that the A-LWE problem is hard to solve assuming the hardness of LWE. Furthermore and more importantly, we give a standard model variant that is essentially as efficient as the previous construction.

Chapter 4 - Building Lattice-based Encryption Schemes from A-LWE

Typically, lattice-based encryption schemes follow the one-time pad approach, where the message, most often an encoded bit vector, is added to an LWE instance. As a result, a random looking ciphertext vector is obtained. However, lattice-based encryption schemes still suffer from a low message throughput per ciphertext. This is mainly due to the fact that the underlying schemes do not tap the full potentials of LWE such as the error term that remains unused except for security.
We present a novel approach towards building lattice-based encryption schemes [P1, P2, P13], which outperform existing current state-of-the-art lattice-based encryption schemes by exploiting almost the full bandwidth of the error term and the secret as further containers carrying messages. In terms of ciphertext expansion, we can embed about $\log(\alpha q/4.7)$ bits of data per coefficient of size $\log q$ bits as compared to 0.5 bits for the best known encryption schemes with $n = 512$, where n represents the main security parameter and αq denotes the discrete Gaussian parameter of the error term. Our constructions are essentially built upon the A-LWE assumption introduced in Chapter 3, which enables existing encryption schemes to strongly decrease the message expansion factor by means of additional message containers supplied by A-LWE. This inherently leads to new cryptographic applications allowing for high data load encryption and customized security properties as required, for instance, in economic environments such as stock markets and for financial transactions. To this end, we give the first lattice-based RCCA-secure encryption scheme together with constructions ensuring CCA1 and CCA2 security, respectively. Additionally, we provide these constructions also with an optional mode for high data load encryption, which allows to efficiently encrypt huge amounts of data at the expense of a minimal increase of the running time. We also show that existing encryption schemes can be improved by use of our newly developed tools such that the resulting constructions still follow the one-time pad approach while at the same time entailing further messages in the error-term at essentially no costs. Our work also comprises a novel asymmetric authenticated encryption scheme, which opens up the possibility to employ lattice-based signatures following the discrete Gaussian distribution as error vectors, hence realizing an authentication mechanism for encrypted data. The security of those constructions basically stems from the hardness to solve the A-LWE problem.

Chapter 5 - CCA-Secure Encryption Scheme from A-LWE in Practice

From A-LWE it is possible to build powerful encryption schemes that theoretically outperform existing lattice-based encryption schemes due to the possibility of hiding data in the error term. This argument is mainly supported by two observations: (1) the simplicity of encrypting data, where ciphertexts resemble basic LWE instances, and (2) a low ciphertext expansion factor as compared to the most efficient encryption scheme due to Lindner and Peikert presented at CT-RSA 2011. However, the efficiency also depends on the quality of the trapdoor, which is crucial particu-

larly for the performance in the decryption step.
We present an instantiation of the A-LWE based CCA-secure encryption scheme [P1, P2, P13] using the currently most efficient trapdoor construction for ideal lattices [P9]. To this end, we consider both the standard model and random oracle variants of A-LWE. In particular, we restrict to the ring setting $\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$ with prime modulus q satisfying $q \equiv 1 \bmod 2n$ for $n = 2^k$, and introduce different tools that allow for efficient operations in this setting. Beside of various inversion algorithms, we introduce a new CDT-like discrete Gaussian sampler outperforming current state-of-the-art samplers such as the standard CDT sampler or Knuth-Yao. Furthermore, we give a thorough security analysis as well as an efficient implementation of the scheme both in the random oracle model and the standard model. Finally, we compare the implementations of our constructions with the CPA-secure encryption scheme due to Lindner and Peikert attesting the presumed efficiency of our scheme.

Chapter 6 - Improvement of the GPV Signature Scheme

The GPV signature scheme represents a cornerstone for building provably secure lattice-based signature schemes. It is based on preimage sampleable trapdoor functions, a main building block of many lattice-based cryptosystems, allowing to solve SIS instances with the knowledge of a suitable trapdoor serving as secret key. Recently, Micciancio and Peikert proposed efficient constructions of preimage sampleable trapdoor functions. However, the practical impact of the GPV signature scheme involving any of the existing trapdoor constructions has never been investigated. This is mainly due to complex procedures making the resulting scheme less efficient.
In our work [P9] we address this research problem and provide an efficient implementation of the GPV signature scheme instantiated with the trapdoor candidate due to Micciancio and Peikert. To this end, we introduce a trapdoor variant for ideal lattices that allows to perform ring operations more efficiently as compared to a straightforward approach or the corresponding matrix variant. Furthermore, we propose several theoretical improvements enhancing the efficiency of the scheme by simplifying the key and signature generation algorithms leading to improved running times and space requirements.

Chapter 7 - Compression Scheme for Signatures

Quite recently, a sequence of new lattice-based signature schemes have been proposed. However, despite of increasing efficiency lattice-based signatures still suffer from huge signature sizes as compared to their classical counterparts. This mainly follows from the underlying constructions or implicitly from the requirements of the security proof.
This chapter is devoted to a novel and generic construction of a lossless compression algorithm for Schnorr-like signatures utilizing publicly accessible randomness [P5]. Conceptually, exploiting public randomness in order to reduce the signature size has never been considered in cryptographic applications. We illustrate the applicability of our compression algorithm using the example of the signature scheme due to Gentry et al. (GPV scheme) instantiated with the efficient trapdoor construction from Micciancio and Peikert. This compression scheme benefits from increasing the main security parameter n, which is positively correlated with the compression rate measuring the amount of storage savings. For instance, GPV signatures admit improvement factors of approximately $\lg n$ implying compression rates of about 65% at a security level of about 100 bits without suffering loss of information or decrease in security, meaning that the original signature can always be recovered from its compressed state. In the second part of this chapter, we present a multi-signer compression scheme in case more than one signer agree to share the same source of public randomness. Such a strategy of bundling compressed signatures together to an aggregate has many advantages over the single-signer approach and is even applicable in combination with lattice-based aggregate signature schemes such as the SAS scheme being introduced in Chapter 5.

Chapter 8 - Sequential Aggregate Signature Scheme

Sequential aggregate signature schemes (SAS) constitute important primitives, when it comes to save memory or the amount of traffic in the presence of multiple signers. Generally speaking, SAS schemes enable any group of signers ordered in a chain to sequentially combine their signatures such that the size of the aggregate signature is much smaller than the total size of all individual signatures.
We propose the first lattice-based sequential aggregate signature (SAS) scheme [P6] that is provably secure in the random oracle model (RO). Moreover, we instantiate our construction with trapdoor function families and describe how to generate aggregate signatures resulting in one single signature. In particular, we instantiate our construction with the provably secure NTRUSign signature scheme presented by Stehlé and Steinfeld at Eurocrypt 2011. This setting allows to generate aggregate signatures being asymptotically as large as individual ones and thus provide optimal compression rates as known from RSA-based SAS schemes.

Chapter 9 - Representation Formula for Lattices

In previous works many algorithms have been proposed in order to solve the underlying lattice problems practically. As a consequence of such a methodical approach, the exact solutions to these problems are described algorithmically with respect to the considered algorithms.
We propose representation formulas for several lattice problems [P14] such as the number of lattice points inside a domain or the solution of LWE using tools from complex analysis. By use of generalized Cauchy integrals from complex analysis, however, we can express the solution of the respective problems as a finite sum of integrals leading to a general representation formula. Generally speaking, the number of lattice-points inside a domain such as an n-dimensional ball is an important quantity required in many lattice attack algorithms in order to estimate the attack complexity. Usually, this is done via the Gauss heuristic which does not provide exact solutions. To this end, we particularly focus on the one- and two-dimensional case, where the former is indeed applied in cryptographic applications such as the hidden number problem (HNP) and the one-dimensional LWE problem. The latter is related to elliptic functions with the Weierstrass zeta function representing one of the main building blocks. This sheds a different light on the considered lattice problems and thus extends the existing theoretical framework. In the one-dimensional case, for instance, we are able to reflect the periodicity of lattices by means of trigonometric functions resulting in a closed and easy to understand expression. Subsequently, it is possible to deduce relations such as conditions for selecting parameters of interesting lattice problems.

2. Preliminaries

2.1. Notation

We denote vectors by boldface lower-case letters, e.g., $\mathbf{p}$, whereas we use for matrices boldface upper case letters $\mathbf{A}$. We will use the polynomial rings $\mathcal{R} = \mathbb{Z}[X]/\langle f(X)\rangle$ and $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ for a monic and irreducible polynomial $f(X)$ over $\mathbb{Z}$ and modulus q. Throughout this thesis we will mainly consider $q = 2^k$ for $k > \mathbb{N}$ or prime moduli. By $\mathcal{R}^{\times}$ we denote the ring of units in $\mathcal{R}$. For the ring-LWE problem, we consider cyclotomic polynomials, such as $f(X) = X^n + 1$ for n being a power of 2. The m-th cyclotomic polynomial with integer coefficients is the polynomial of degree $n = \phi(m)$, whose roots are the primitive m-th roots of unity. We also indicate ring elements by lower-case bold letters, e.g., $\mathbf{p}$, and denote vectors of ring elements by $\hat{\mathbf{p}}$. The topological closure of a domain D in a Euclidean vector space is specified by $\bar{D}$. By $\oplus$ we define the XOR operator. We let $[\ell]$ denote the set $\{1, \ldots, \ell\}$ for any $\ell \in \mathbb{N}_{\geq 1}$. By $\vec{\mathbf{v}} = v_1, \ldots, v_k$, we indicate a sequence of elements. If $\mathcal{X}$ is a set, we write $x \leftarrow_R \mathcal{X}$ to denote that x is sampled uniformly from $\mathcal{X}$. If $\mathcal{X}$ is a distribution, $x \leftarrow_R \mathcal{X}$ means that x was sampled according to $\mathcal{X}$. The statistical distance of two distributions $\mathcal{X}_1$ and $\mathcal{X}_2$ denoted by $\Delta(\mathcal{X}_1, \mathcal{X}_2)$ over a countable set $\mathcal{S}$ is defined by $\Delta(\mathcal{X}_1, \mathcal{X}_2) := \frac{1}{2}\sum_{s\in\mathcal{S}} |\mathcal{X}_1(s) - \mathcal{X}_2(s)|$.

2.2. Lattices

We start by defining Euclidean vector spaces E, which represent finite-dimensional vector spaces over the field $\mathbb{R}$ of dimension $dim_{\mathbb{R}}(E) = [E : \mathbb{R}]$. A Euclidean vector space E is equipped with the so-called inner product map $\langle \cdot, \cdot \rangle : E \times E \longrightarrow \mathbb{R}$ satisfying

- **(Linearity)** $\langle \mathbf{x}+\mathbf{y}, \mathbf{z}\rangle = \langle \mathbf{x}, \mathbf{z}\rangle + \langle \mathbf{y}, \mathbf{z}\rangle$, $\langle r\mathbf{x}, \mathbf{y}\rangle = r\,\langle \mathbf{x}, \mathbf{y}\rangle$
- **(Symmetry)** $\langle \mathbf{y}, \mathbf{x}\rangle = \langle \mathbf{x}, \mathbf{y}\rangle$
- **(Positive Definite)** $\langle \mathbf{x}, \mathbf{x}\rangle \geq 0\,.$

The topology of a Euclidean vector space is defined by its distance function $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$, which exists since E is also a metric space. For $E = \mathbb{R}^n$, for instance, we define $d(\mathbf{x}, \mathbf{y}) = \sqrt{\langle \mathbf{x}, \mathbf{y}\rangle}\,$. A subset Λ of a Euclidean vector space is said to be discrete, if there exists an $\epsilon > 0$ for each $\mathbf{x} \in \Lambda$ such that the only element $\mathbf{y}$ satisfying $d(\mathbf{x}, \mathbf{y}) < \epsilon$ is $\mathbf{y} = \mathbf{x}$, meaning that the discrete topology of Λ is defined by the distance function $d(\cdot,\cdot)$.

A discrete subset Λ of a Euclidean vector space E forms a lattice, if it is an additive subgroup of E and there exist n linearly independent vectors $\mathbf{b}_1,\ldots,\mathbf{b}_n \in E$ such that the integer linear combinations of these vectors generate Λ. More specifically, we have

$$\Lambda(\mathbf{B}) = \{\sum_{i=1}^{n} x_i\mathbf{b}_i \mid x_i \in \mathbb{Z}\} = \sum_{i=1}^{n} \mathbb{Z}\mathbf{b}_i,$$

where $\mathbf{B} = [\mathbf{b}_1,\ldots,\mathbf{b}_n]$ is called basis of $\Lambda(\mathbf{B})$. Moreover, the set $\Lambda(\mathbf{B})$ is isomorphic to $\mathbb{Z}^n$ as an abelian group. Throughout this thesis, we mainly consider integral lattices as they are typically used in cryptographic applications. In particular, we are mostly concerned with q-ary lattices $\Lambda_q^{\perp}(\mathbf{A})$ and $\Lambda_q(\mathbf{A})$, where $q = \mathit{poly}(n)$ denotes a polynomially bounded modulus and $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ is an arbitrary matrix such as a uniform random matrix. $\Lambda_q^{\perp}(\mathbf{A})$ and $\Lambda_q(\mathbf{A})$ are defined by

$$\begin{aligned}\Lambda_q^{\perp}(\mathbf{A}) &= \{\mathbf{x} \in \mathbb{Z}^m \mid \mathbf{A}\mathbf{x} \equiv \mathbf{0} \mod q\}\\ \Lambda_q(\mathbf{A}) &= \{\mathbf{x} \in \mathbb{Z}^m \mid \exists \mathbf{s} \in \mathbb{Z}^n \text{ s.t. } \mathbf{x} = \mathbf{A}^{\top}\mathbf{s} \mod q\}.\end{aligned}$$

with $q\mathbb{Z}^m \subseteq \Lambda_q^{\perp}(\mathbf{A}), \Lambda_q(\mathbf{A}) \subseteq \mathbb{Z}^m$.

Definition 2.1 (**Dual Lattice**)**.** *Let Λ be a lattice in $\mathbb{R}^n$. Its dual lattice Λ^* is defined by*

$$\Lambda^* = \{\mathbf{x} \in \mathbb{Z}^n \mid \langle \mathbf{x},\mathbf{z}\rangle \in \mathbb{Z}\ \forall \mathbf{z} \in \Lambda\}.$$

Specifically, if $\mathbf{B}$ represents a full-rank basis of Λ, then a basis of its dual lattice is given by $(\mathbf{B}^{\top})^{-1}$.

Definition 2.2 (**Determinant**)**.** *The determinante of an integral lattice Λ with basis $\mathbf{B} \in \mathbb{Z}^{n\times m}$ is defined by the map*

$$\det \Lambda = \sqrt{\det(\mathbf{B}\mathbf{B}^{\top})}.$$

The determinant of $\Lambda_q^{\perp}(\mathbf{A})$ for a uniformly sampled matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and sufficiently large $m > n$ is with very high probability q^n.

2.3. Discrete Gaussian Distribution

The discrete Gaussian distribution constitutes one of the main building blocks of lattice-based cryptographic schemes. It is often applied as a requirement of the security proof or due to its properties to sample small vectors with high probability, a desired feature in lattice-based cryptography.

Therefore, let $\mathbf{c} \in \mathbb{R}^m$ be a vector, s a positive real and $\Lambda \subset \mathbb{R}^m$ a lattice. Denote by $\mathcal{D}_{\Lambda,\mathbf{c},s}$ the n-dimensional discrete Gaussian distribution over Λ, centered at $\mathbf{c}$, with parameter s. For $\mathbf{x} \in \Lambda$, the distribution $D_{\Lambda,\mathbf{c},s}$ assigns the probability

$$\mathcal{D}_{\Lambda,\mathbf{c},s}(\mathbf{x}) := \frac{\rho_{\mathbf{c},s}(\mathbf{x})}{\sum\limits_{\mathbf{z}\in\Lambda} \rho_{\mathbf{c},s}(\mathbf{z})} \quad \text{with} \quad \rho_{\mathbf{c},s}(\mathbf{x}) = \exp\left(-\pi \left\|\mathbf{x}-\mathbf{c}\right\|^2 / s^2\right) .$$

We recall the smoothing parameter introduced by Micciancio and Regev in [MR04], which will be required in several parts throughout this thesis.

Definition 2.3. *For any n-dimensional lattice Λ and positive real $\epsilon > 0$, the smoothing parameter $\eta_\epsilon(\Lambda)$ is the smallest real $s > 0$ such that $\rho_{1/s}(\Lambda^* \backslash \{0\}) \leq \epsilon$.*

2.4. Computational Problems

In the following section, we introduce some computational problems arising in lattice theory, which also build the foundations for many of the lattice-based cryptographic schemes. We start with the definition of successive minima required to describe the related lattice problems.

Definition 2.4 (Successive Minima). *Let $B_r(\mathbf{0}) = \{\mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x}\|_2 < r\}$ be the n-dimensional open ball with radius $r \in \mathbb{R}_{>0}$ centered at $\mathbf{0}$. The k-th minimum $\lambda_k(\Lambda)$ of a lattice Λ is the radius of the smallest ball centered at $\mathbf{0}$ such that it contains k linearly independent lattice vectors or formally*

$$\lambda_k(\Lambda) = \inf\{r \mid dim(span(\Lambda \cap B_r(\mathbf{0}))) \geq k\} .$$

The closest vector problem (CVP) and its approximated version CVP_γ are recalled in the following definition.

Definition 2.5 (γ-Closest Vector Problem CVP_γ). *Let Λ be an integral lattice with basis $\mathbf{B} \in \mathbb{Z}^{n\times m}$ and $\mathbf{t} \in \mathbb{R}^n$ be a target vector. The approximate closest vector problem CVP_γ asks to find a lattice point $\mathbf{x} \in \Lambda$ satisfying*

$$\|\mathbf{x}-\mathbf{t}\|_2 \leq \gamma \cdot \min_{\substack{\mathbf{y}\in\Lambda \\ \mathbf{y}\neq\mathbf{x}}} \|\mathbf{y}-\mathbf{t}\|_2 \,, \;\; \mathbf{x} \in \Lambda$$

for a real number $\gamma \in \mathbb{R}_{\geq 1}$.

For $\gamma = 1$, the challenger is required to solve exact CVP and find a closest vector lattice vector to the target, whereas for $\gamma > 1$, a relaxed version of the former, allows the challenger to find any lattice vector close to the target but bounded to the distance of at most $\gamma \cdot \min\limits_{\substack{\mathbf{y}\in\Lambda \\ \mathbf{y}\neq\mathbf{x}}} \|\mathbf{y}-\mathbf{t}\|_2$.

Accordingly, we define the shortest vector problem as follows.

Definition 2.6 (γ-Shortest Vector Problem SVP_γ). *Let Λ be an integral lattice with basis $\mathbf{B} \in \mathbb{Z}^{n\times m}$. The approximate shortest vector problem SVP_γ asks to find a* *non-zero lattice vector* $\mathbf{x} \in \Lambda$ *with distance of at most*

$$\|\mathbf{x}\|_2 \leq \gamma \cdot \min_{\substack{\mathbf{y} \in \Lambda \\ \mathbf{y} \neq \mathbf{x}, 0}} \|\mathbf{y}\|_2 \,, \ \mathbf{x} \in \Lambda$$

to the origin for a real number $\gamma \in \mathbb{R}_{\geq 1}$.

The approximate version SVP_γ is a relaxed version of SVP in case $\gamma > 1$ analogous to the CVP case. However, for $\gamma = 1$ we have the exact version of SVP requiring to find a shortest non-zero lattice vector.

Definition 2.7 (γ**-Shortest Independent Vector Problem** SIVP_γ^k)**.** *Let* Λ *be an integral lattice with basis* $\mathbf{B} \in \mathbb{Z}^{n \times m}$. *The approximate shortest independent vector problem* SIVP_γ^k *asks to find* k *linearly independent basis vectors* $\mathbf{v}_1, \ldots, \mathbf{v}_k$ *such that*

$$\|\mathbf{v}_i\|_2 \leq \gamma \cdot \lambda_k(\Lambda)$$

for $1 \leq i \leq k$ *and a real number* $\gamma \in \mathbb{R}_{\geq 1}$.

The decision version of the problems CVP_γ and SVP_γ are in the literature referred to as GapCVP_γ and GapSVP_γ. We omit a description for these problems as they are of minor interest in this thesis.

For cryptography, we consider average-case problems that are used to instantiate lattice-based schemes more efficiently than basing the security on worst-case instances. In particular, the SIS and LWE problems represent important average-case problems boosting the construction of new lattice-based cryptographic primitives in recent years. The SIS problem is defined with respect to q-ary lattices introduced earlier in Section 2.2 and serves as an underlying computational problem for many signature schemes.

Definition 2.8 (**SIS-Problem**)**.** *Given a uniform random matrix* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, *the* $\mathsf{SIS}(n, m, q, \beta)$ *problem asks to find a short vector* $\mathbf{x} \in \mathbb{Z}^m \backslash \mathbf{0}$ *such that* $\mathbf{A}\mathbf{x} \equiv \mathbf{0} \bmod q$ *and* $\|\mathbf{x}\|_2 \leq \beta$.

Put it another way, in Definition 2.8 an instance of the shortest vector problem in the q-ary lattice $\Lambda_q^{\perp}(\mathbf{A})$ is required to be solved. In [GPV08] a tight proof for the worst-case to average-case hardness of the SIS problem has been proposed, where the first such connection was established by Ajtai in [Ajt96] and later on improved [MR04].

Theorem 2.9 (**[GPV08]**)**.** *For poly-bounded* $m, \beta = poly(n)$ *and for any prime* $q \geq \beta \cdot \omega(\sqrt{n \log n})$ *the average-case problem* $\mathsf{SIS}(n, m, q, \beta)$ *is as hard as approximating* SIVP_γ *in the worst-case to within factors* $\gamma = \beta \cdot \tilde{O}(\sqrt{n})$.

The second average-case problem, called the Learning with Errors problem or LWE, is applied in many lattice-based encryption schemes due to its features coinciding with the ones desired for encryption. Below we define the LWE distribution [Reg05]. For our purposes, we only focus on discrete Gaussian distributed error vectors. One can easily define LWE with respect to any error distribution.

Definition 2.10 (LWE Distribution). *Let n, m, q be integers and χ_e be a distribution over $\mathbb{Z}$. For $\mathbf{s} \in \mathbb{Z}_q^n$, define the LWE distribution $L^{\mathsf{LWE}}_{n,m,\alpha q}$ to be the distribution over $\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m$ obtained by first drawing $\mathbf{A} \leftarrow_R \mathbb{Z}_q^{n\times m}$ uniformly at random, sampling $\mathbf{e} \leftarrow_R \mathcal{D}_{\mathbb{Z}^m,\alpha q}$ and finally returning $(\mathbf{A}, \mathbf{b}^\top) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m$ with $\mathbf{b}^\top = \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top$.*

An overview of the computational problems arising from LWE are specified as follows.

Definition 2.11 (Learning with Error (LWE)). *Let $(\mathbf{A}, \mathbf{b})$ be a sample from $L^{\mathsf{LWE}}_{n,m,\alpha q}$ and $\mathbf{c}$ be uniformly sampled from $\mathbb{Z}_q^m$.*

The Decision Learning with Error *(*decision $\mathsf{LWE}_{n,m,\alpha q}$*) problem asks to distinguish between $(\mathbf{A}, \mathbf{b}^\top)$ and $(\mathbf{A}, \mathbf{c}^\top)$ for a uniformly sampled secret $\mathbf{s} \leftarrow_R \mathbb{Z}_q^n$.*

The Search Learning with Error *(*search $\mathsf{LWE}_{n,m,\alpha q}$*) problem asks to output the vector $\mathbf{s} \in \mathbb{Z}_q^n$ given LWE samples $(\mathbf{A}, \mathbf{b})$ for a uniformly sampled secret $\mathbf{s} \leftarrow_R \mathbb{Z}_q^n$.*

We say decision $\mathsf{LWE}_{n,m,\alpha q}$ (resp. search $\mathsf{LWE}_{n,m,\alpha q}$) is hard if all polynomial time algorithms solve decision $\mathsf{LWE}_{n,m,\alpha q}$ (resp. search $\mathsf{LWE}_{n,m,\alpha q}$) only with negligible probability.

2.5. Cryptographic Primitives

In the following section we introduce the abstract representation of certain cryptographic primitives that are of prime interest in this thesis. We restrict our considerations mainly to encryption schemes and digital signature schemes. We defer a description of trapdoor functions and its properties to the second part of this thesis concerning signature schemes.

2.5.1. Encryption Schemes

Let $\mathcal{M}$ denote the plaintext space and $\mathcal{C}$ the associated ciphertext space. The general description of a probabilistic public key encryption scheme involves three algorithms given by the triple $\mathcal{E} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$.

$\mathsf{KGen}(1^n)$ On input 1^n the algorithm $\mathsf{KGen}(1^n)$ outputs a key pair $(\mathsf{pk}, \mathsf{sk})$, where sk is the secret key and pk is the public key.

$\mathsf{Enc}(\mathsf{pk}, \mu)$ The encryption algorithm is a function $\mathsf{Enc}_{\mathsf{pk}} : \mathcal{M} \longrightarrow \mathcal{C}$ indexed by the public key, that maps a message $\mu \in \mathcal{M}$ to a ciphertext $\mathbf{c} \in \mathcal{C}$ under the public key pk.

$\mathsf{Dec}(\mathsf{sk}, \mathbf{c})$ The decryption algorithm is the inverse function $\mathsf{Dec}_{\mathsf{sk}} : \mathcal{C} \longrightarrow \mathcal{M}$ indexed by the secret key, that on input a valid ciphertext $\mathbf{c}$ computes the corresponding plaintext $\mu \in \mathcal{M}$ under secret key sk, otherwise it outputs $\perp$.

A probabilistic public key encryption scheme is said to be indistinguishable under chosen plaintext attack (IND-CPA), if an adversary with black box access to the encryption oracle ($\mathsf{OEnc}_{\mathsf{pk}}(\cdot)$) is not able to make a correct guess within a polynomial number of time steps in the following experiment.

```
**Experiment** $\mathbf{Exp}^{\mathrm{ind-CPA}}_{\mathcal{E},\mathcal{A}}(n)$
$(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^k)$
$(\mu_0,\mu_1) \leftarrow \mathcal{A}^{\mathsf{OEnc}_{\mathsf{pk}}(\cdot)}(\text{CHOOSE}, \mathsf{pk})$
$\mathbf{c}_b \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\mu_b)$ for $b \leftarrow_R \{0,1\}$
$b' \leftarrow \mathcal{A}^{\mathsf{OEnc}_{\mathsf{pk}}(\cdot)}(\text{GUESS}, \mathbf{c}_b)$
Output 1 iff
    1. $b' = b$
    2. $|\mu_0| = |\mu_1|$
```

In particular, the scheme is IND-CPA secure if the adversary $\mathcal{A}$ has negligible advantage over random guessing, that is he wins the game above with probability

$$P[\ \mathbf{Exp}^{\mathrm{ind-CPA}}_{\mathcal{E},\mathcal{A}}(n) = 1\ ] = \frac{1}{2} + \epsilon(n)$$

for a negligible function $\epsilon(n)$ in the security parameter n. In the following section we recall the definitions of (replayable) chosen ciphertext security of encryption schemes.

(Replayable) Chosen Ciphertext Security

Let $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public key encryption scheme and consider the following experiments for $\mathsf{atk} \in \{\mathsf{cca1}, \mathsf{cca2}, \mathsf{rcca}\}$:

```
**Experiment** $\mathbf{Exp}^{\mathrm{ind-atk}}_{\mathcal{E},\mathcal{A}}(n)$
$(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^k)$
$(\mu_0,\mu_1) \leftarrow \mathcal{A}^{\mathsf{Dec}(\cdot)}(\text{CHOOSE}, \mathsf{pk})$
$\mathbf{c}_b \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\mu_b)$ for $b \leftarrow_R \{0,1\}$
$b' \leftarrow \mathcal{A}^{\mathcal{O}_2(\cdot)}(\text{GUESS}, \mathbf{c}_b)$
Output 1 iff
    1. $b' = b$
    2. $|\mu_0| = |\mu_1|$
    3. $\mathbf{c}_b$ was not queried to $\mathcal{O}_2$
```

```
If $\mathcal{A}$ queries $\mathcal{O}_2(\mathbf{c})$, and
    - if $\mathsf{atk} = \mathsf{cca1}$, then return $\perp$.
    - if $\mathsf{atk} = \mathsf{cca2}$, then return $\mathsf{Dec}(\mathsf{sk}, \mathbf{c})$.
    - if $\mathsf{atk} = \mathsf{rcca}$ and $\mathsf{Dec}(\mathsf{sk}, \mathbf{c}) \notin \{\mu_0, \mu_1\}$,
          then return $\mathsf{Dec}(\mathsf{sk}, \mathbf{c})$.
    - Otherwise, return $\perp$.
```

Definition 2.12 (Chosen-ciphertext secure encryption)**.** *Let $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme.*

CCA1 security. *We say $\mathcal{E}$ is secure against non-adaptive chosen-ciphertext attacks (CCA1) if we have*

$$\Pr[\mathbf{Exp}_{\mathcal{E},\mathcal{A}}^{\mathsf{atk-cca1}}(n) = 1] \leq \mathsf{negl}(n)$$

for all polynomial-time algorithms $\mathcal{A}$.

CCA2 security. *We say $\mathcal{E}$ is secure against adaptively chosen-ciphertext attacks (CCA2) if we have*

$$\Pr[\mathbf{Exp}_{\mathcal{E},\mathcal{A}}^{\mathsf{atk-cca2}}(n) = 1] \leq \mathsf{negl}(n)$$

for all polynomial-time algorithms $\mathcal{A}$.

RCCA security. *We say $\mathcal{E}$ is secure against replayable chosen-ciphertext attacks (RCCA) if we have*

$$\Pr[\mathbf{Exp}_{\mathcal{E},\mathcal{A}}^{\mathsf{atk-rcca}}(n) = 1] \leq \mathsf{negl}(n)$$

for all polynomial-time algorithms $\mathcal{A}$.

There exists a strict hierarchy in the security notions. That is, CCA2 security implies RCCA security which in turn implies CCA1 security. All the above security notions are formulated following the indistinguishability approach. We note that alternatively one could define the security in a non-malleability approach yielding NM-CCA1, NM-CCA2, and NM-RCCA. Here, an adversary given an encryption of μ_b is essentially not able to output an encryption of μ_{1-b}. The non-malleability notions imply the indistinguishability counterparts. However, the other direction does not hold in general, for instance for CCA1 security. Moreover, the notion of non-malleable replayable CCA is strictly stronger than the indistinguishability notion of replayable CCA for polynomial message spaces [CKN03].

2.5.2. Digital Signature Schemes

Denote by $\mathcal{M}$ the message space. A digital signature scheme is composed by three algorithms $\mathcal{S} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$, which are defined as follows.

$\mathsf{KGen}(1^n)$ On input 1^n the algorithm $\mathsf{KGen}(1^n)$ outputs a key pair $(\mathsf{pk}, \mathsf{sk})$, where sk is the secret key and pk is the public key.

$\mathsf{Sign}(\mathsf{sk}, \mu)$ The signing algorithm $\mathsf{Sign}_{\mathsf{sk}}(\mu)$ is a probabilistic function indexed by the secret key, that outputs a signature σ for a message $\mu \in \mathcal{M}$ under the public key sk.

$\mathsf{Verify}(\mathsf{sk}, \sigma, \mu)$ The verification algorithm $\mathsf{Verify}_{\mathsf{sk}}(\sigma)$ is a deterministic algorithm indexed by the public key, that outputs 1 in case σ is a valid signature of the message μ under public key pk, otherwise it outputs 0.

A probabilistic public key encryption scheme is said to be existentially unforgeable under chosen message attacks (EU-CMA), if an adversary with black box access to the signing oracle ($\mathsf{OSign}_{\mathsf{sk}}(\cdot)$) is not able to produce within a polynomial number of time steps a valid signature on message of choice, for which $\mathsf{OSign}_{\mathsf{sk}}(\cdot)$ has not been queried before.

Experiment $\mathbf{Exp}_{\mathcal{S},\mathcal{A}}^{\mathsf{EU-CMA}}(n)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^k)$
$(\mu^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{OSign}_{\mathsf{sk}}(\cdot)}(\textsc{choose}, \mathsf{pk})$
Output 1 iff

1. $\mathsf{Verify}(\mathsf{sk}, \sigma^*, \mu^*) = 1$
2. μ^* has not been queried before.

The adversary wins the game if he succeeds in the experiment $\mathbf{Exp}_{\mathcal{S},\mathcal{A}}^{\mathsf{EU-CMA}}(n)$ with non-negligible advantage. Conversely, the scheme is EU-CMA secure if the adversary wins the game with probability

$$P[\ \mathbf{Exp}_{\mathcal{S},\mathcal{A}}^{\mathsf{EU-CMA}}(n) = 1\ ] \leq \epsilon(n)$$

for a negligible function $\epsilon(n)$ in the security parameter n. An even stronger security notion in this context is to provide strong unforgeability under chosen message attacks (SU-CMA), which will be considered in Chapter 8 in more detail when introducing sequential aggregate signatures based on lattices. In the corresponding game the adversary is additionally allowed to produce a valid signature on already queried messages. In this case the signature is only valid, if it has never been returned to the adversary before.

Part I.

Lattice-based Encryption

Overview

The design of provably secure encryption schemes relying on worst-case lattice problems was initiated by the seminal work of Ajtai and Dwork [AD97]. The security of this scheme is based on the worst-case hardness of approximating SVP within polynomial factors. Several other works followed with focus on improving the efficiency [GGH97a, Reg04]. Ever since the breakthrough work of Regev [Reg05], the learning with errors assumption and its ring variant [LPR10] are widely used in lattice-based cryptography to base the security of cryptographic schemes upon LWE. Indeed, since then lattice-based cryptography emerged and novel encryption schemes have been built upon this assumption, such as fully homomorphic encryption [Gen09, BV11a, GH11, Bra12, BGV12] and identity-based encryption schemes [GPV08, CHKP10, ABB10a, ABB10b] besides of CPA-secure [Reg05, PVW08, LPR10, LP11, LPR13] and CCA-secure encryption schemes [PW08, Pei09, MP12, Pei14]. Many of those encryption schemes utilize LWE in order to blind certain sensitive data following the one-time-pad approach. The LWE problem is qualified for building lattice-based encryption schemes due to two nice properties LWE instances embody. First, LWE instances are indistinguishable from uniform random samples, hence hiding its character from public viewers. This follows from the decision problem of LWE. Second, once knowing that a given sample represents indeed an LWE instance (e.g., in encryption schemes), searching for the secret vector or the error term used to build LWE samples is as hard as quantumly approximating SIVP resp. GapSVP in n-dimensional worst-case lattices for error vectors following the discrete Gaussian distribution. These properties intuitively coincide with the desirable features of encryption schemes.
More specifically, the LWE problem exists essentially in two variants, the decision and search version. Following this, the challenger is given a $poly(n)$ number of independent samples $(\mathbf{A}_i, \mathbf{b}_i^\top) \in \mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^m$, where $\mathbf{A}_i \leftarrow_R \mathbb{Z}_q^{n \times m}$, $\mathbf{e}_i \leftarrow_R \chi$, and $\mathbf{b}_i^\top = \mathbf{s}^\top \mathbf{A}_i + \mathbf{e}_i^\top \mod q$ for $\mathbf{s} \in \mathbb{Z}_q^n$ and some arbitrary distribution χ over $\mathbb{Z}^m$, typically discrete Gaussian. He is then asked to distinguish those samples from uniformly random samples from $\mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^m$. In search-LWE, however, the challenger is required to find the secret $\mathbf{s}$. Besides its presumably quantum hardness, one of the most noteworthy properties lattice-based assumptions offer is worst-case hardness of average-case instances. Starting with the works of Ajtai [Ajt96] and Micciancio and Regev [MR04], the hardness of some average-case instances of the SIS problem was shown to be hard as long as worst-case instances of the (decision version of the) shortest vector problem, known as GapSVP, are hard. The worst-case hardness of LWE was first stated by Regev [Reg05]. Regev showed that if the error vector follows the discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z}^m, \alpha q}$ with parameter $\alpha q \geq 2\sqrt{n}$,

solving search-LWE is at least as hard as quantumly solving $\tilde{O}(n/\alpha)$-SIVP and GapSVP in n-dimensional worst-case lattices. Later, Peikert [Pei09] and Brakerski et al. [BLP$^+$13] gave a classical reduction from GapSVP to LWE. In [DMQ13] Döttling and Müller-Quade proved the hardness of LWE for uniformly distributed errors. Subsequently, Micciancio and Peikert [MP13] showed that LWE remains hard even for binary errors.
This part consisting of Chapters 3-5 represents a reprint of the essential parts of [EDB15, EB15a, EB15b], where the author of this thesis was also the primary investigator and author of the publications.

3. Augmented LWE and its Hardness

The Learning with Errors (LWE) problem has gained a lot of attention in recent years leading to a series of new cryptographic applications. Specifically, it states that it is hard to distinguish random linear equations disguised by some small error from truly random ones. Interestingly, cryptographic primitives based on LWE often do not exploit the full potential of the error term beside of its importance for security. To this end, we introduce a novel LWE-close assumption, namely Augmented Learning with Errors (A-LWE), which allows to hide auxiliary data injected into the error term by a technique that we call message embedding. In particular, it enables existing cryptosystems to strongly increase the message throughput per ciphertext. We show that A-LWE is for certain instantiations at least as hard as the LWE problem. This inherently leads to new cryptographic constructions providing high data load encryption and customized security properties as required, for instance, in economic environments characterized by a large number of transactions. The security of those constructions basically stems from the hardness to solve the A-LWE problem.

3.1. Main Obstacles

Cryptographic constructions which rely on the LWE assumption usually sample an error term according to some distribution, most often Gaussian. Such a choice has many advantages over other distributions. However, many of the existing LWE-based schemes do not exploit the full potential of the error term. This observation is mainly due to three reasons, which can be summarized using the example of encryption schemes.

- First, previous LWE-based encryption schemes produce ciphertexts mainly following the idea of one-time pad encryption, where LWE samples play the role of random vectors. As a consequence, the underlying constructions heavily rely on the error term to be short in order to correctly recover the message. A major drawback of such schemes is the waste of bandwidth, i.e., all bits created for the error term are sacrificed for a few message bits.
- Second, there exist no proposals using the error term or other involved random variables as additional containers carrying auxiliary data, besides of its task to provide the required distributions. Once recognizing its feasibility, it fundamentally changes the way of building cryptosystems. For instance, in encryption schemes one may inject the message into the error term without necessarily changing the target distributions.

- Third, there is a lack of efficient trapdoor functions that recover the secret *and* the error term from an LWE instance, which is obviously a necessary condition for exploiting the error term. Only a few works such as [SSTX09, MP12] provide mechanisms to recover the error term. The most promising trapdoor candidate is proposed by Micciancio and Peikert [MP12].

We make the following conclusions. The above limitations of LWE intuitively ask for an alternative LWE definition that accounts for the modifications made to the error term, while ensuring essentially the same hardness results as the traditional LWE problem. Since such an assumption already encompasses message data within the error term, one obtains, as a consequence, a generic and practically new encryption scheme secure under the new variant of the LWE assumption, where the trapdoor function is viewed as a black box recovering the secret and the error vector from a modified LWE instance. The message is subsequently extracted from the error vector. This allows one to exploit the full bandwidth of the error vector with full access to all its entries and not just its length. Remarkably, one could even combine this approach with existing methods for encryption in order to further increase the message throughput per ciphertext. In the following sections we address this challenge and give a detailed description of how to exploit the error vector. This chapter is refers to the publications [EDB15, EB15a, EB15b], where the author of this thesis was also the primary investigator and author of the publications.

3.2. Our Contribution

Based on these observations and subsequently made conclusions, we start by giving an alternative LWE definition, called Augmented LWE (A-LWE), that extends the existing one by modifying the error term in such a way that it encapsulates additional information. We further show which instantiations yield A-LWE samples that are indistinguishable from traditional LWE samples, thereby enjoying the hardness of traditional LWE. In conjunction with the high quality trapdoor candidate from [MP12], we have full access to the error term. This result inherently yields new cryptographic applications, which ensure security in various models while simultaneously allowing for high data load encryption that is applicable, for instance, in financial environments such as stock markets operating with huge amounts of stock information. It is even possible to encrypt lattice-based signatures much more efficiently than ordinary messages, which is an interesting technique for internet protocols, where the acknowledgement of ip-packets represents an important measure for reliability. In this case, the whole entropy of the error term is supplied by lattice-based signatures.

Methodology of Message Embedding. In many lattice-based cryptographic schemes, one has to sample error terms following the discrete Gaussian distribution as a requirement for the scheme to be secure. This is often due to an LWE-based security reduction. The key concept underlying our proposal is to embed further information in the error term $\mathbf{e} \in \mathbb{Z}^m$, but in such a way that the distribution of the augmented error term is computationally close to the discrete Gaussian distribution over $\mathbb{Z}^m$. We also show that one can embed messages in uniformly distributed error vectors more efficiently, hence avoiding complex operations applied for discrete Gaussians.

The idea of our technique is the following. We employ a simple preimage sampleable full-rank matrix $\mathbf{B} \in \mathbb{Z}_p^{n' \times m}$ for integers $p, m, n' \in \mathbb{N}$ such as $\mathbf{B} = \mathbf{I}$ with $n' = m$ in order to sample vectors following the discrete Gaussian distribution $\mathcal{D}_{\Lambda_{\mathbf{v}}^{\perp}(\mathbf{B}),r}$ with $r > \eta_\epsilon(\mathbf{B})$ and a negligible parameter ϵ. Samples $\mathbf{e} \in \mathbb{Z}^m$ distributed according to $\mathcal{D}_{\Lambda_{\mathbf{v}}^{\perp}(\mathbf{I}),r}$, for instance, satisfy the equation $\mathbf{I} \cdot \mathbf{e} \equiv \mathbf{v} \bmod p$ for an arbitrary selected syndrome $\mathbf{v} \in \mathbb{Z}_p^m$. Let $F : \{0,1\}^* \to \{0,1\}^{n' \log p}$ be some random function and $(\mathsf{encode}, \mathsf{decode})$ a pair of algorithms which allows one to switch between the representations $\mathbb{Z}_p^{n'}$ and $\{0,1\}^l$ for $l = n' \log p$. We compute a random coset $\mathbf{v} = \mathsf{encode}(F(\mathsf{seed}) \oplus \mathbf{m}) \in \mathbb{Z}_p^{n'}$, where $\mathbf{m} \in \{0,1\}^l$ denotes an arbitrary message of length l bits. We first show that the distribution $\mathcal{D}_{\Lambda_{\mathbf{v}}^{\perp}(\mathbf{I}),r}$ is statistically/computationally indistinguishable from $\mathcal{D}_{\mathbb{Z}^{n'},r}$ in case the coset $\mathbf{v}$ is selected statistically/computationally close to uniform. In fact, we prove that if F is instantiated by a cryptographic hash function modeled as a random oracle, $\mathbf{v}$ is indeed indistinguishable from uniform. We only have to take care that the input to the function F, namely the seed, has sufficient (computational) min-entropy. Whoever has access to this seed can deterministically recover the message by $\mathbf{m} = \mathsf{decode}(\mathbf{B} \cdot \mathbf{e} \mod p) \oplus F(\mathsf{seed})$. This result immediately impacts all schemes that allow for error term recovery, as it enhances the compactness of the scheme. In the standard model variant, the function F is instantiated by a pseudo random generator (PRNG) resulting in distributions being computationally indistinguishable from discrete Gaussians over $\mathbb{Z}^m$.

Augmented Learning with Errors in the Random Oracle Model. The goal is to present an encryption scheme that does not suffer from high message expansion factors and simultaneously allows to fully take advantage of the aforementioned properties of LWE instances, where the representation of ciphertexts is very close to that of ordinary LWE samples. Embedding auxiliary private information into the error term raises certain new computational problems. In addition to the secret and error vector of an LWE instance, also the new embedded message is concealed. In fact, we claim that LWE samples modified as above are indistinguishable from uniform even for adversarially chosen messages. To this end, we introduce a novel problem, namely the *Augmented* LWE (A-LWE) problem, which differs from the traditional LWE problem only in the way the error term is produced. More specifically, we split the error term $\mathbf{e} \in \mathbb{Z}_q^m$ of LWE into $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2)$, where $\mathbf{e}_1 \in \mathbb{Z}^{m_1}$ and $\mathbf{e}_2 \in \mathbb{Z}^{m_2}$. An A-LWE sample is then distributed as follows. For a given $\mathbf{s} \in \mathbb{Z}_q^n$, first choose $\mathbf{A} \leftarrow_R \mathbb{Z}_q^{n \times m}$ uniformly at random. Then, sample $\mathbf{e}_1 \leftarrow_R \mathcal{D}_{\mathbb{Z}^{m_1},\alpha q}$ and $\mathbf{e}_2 \leftarrow_R \mathcal{D}_{\Lambda_{\mathbf{v}}^{\perp}(\mathbf{B}),\alpha q}$, where $\mathbf{v} = \mathsf{encode}(F(\mathbf{s}, \mathbf{e}_1) \oplus \mathbf{m})$ for some function F. The tuple

$(\mathbf{A}, \mathbf{b}^t = \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top)$ represents an A-LWE sample. We show that distinguishing A-LWE samples from traditional LWE samples is hard for properly chosen random function F. More formally, if $F = H$ is a cryptographic hash function modeled as a random oracle, the tuple $(\mathbf{s}, \mathbf{e}_1)$ has sufficient entropy in each sample and the LWE problem for parameters m, n, α, q is hard to solve, then we obtain a negligible computational distance between the LWE and A-LWE distributions. Thus, we immediately deduce the hardness of A-LWE from LWE. As an immediate consequence, the confidentiality of the message is protected as long as $\mathsf{decision\ A\text{-}LWE}$ and hence $\mathsf{decision\ LWE}$ is hard.

Augmented Learning with Errors in the Standard Model. The final step is to convert the A-LWE problem into an alternative that is hard under standard assumptions. This is indeed attained via some small modifications involving a computational extractor resp. PRNG. Due to this simple modification, we can translate all encryption schemes based on the random oracle variant of A-LWE into the standard model setting. In fact, we can show that samples $(\mathbf{A}, \mathbf{b}^\top) \in \mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^m$ constructed as above with an error term $(\mathbf{e}_1, \mathbf{e}_2) \leftarrow_R \mathcal{D}_{\mathbb{Z}^{m_1}, \alpha q} \times \mathcal{D}_{\Lambda_{\mathbf{v}}^{\perp}(\mathbf{B}), \alpha q} \in \mathbb{Z}^{m_1 + m_2}$ are indistinguishable from uniform random samples for $\mathbf{v} = \mathsf{encode}(F(\mathsf{seed}) \oplus \mathbf{m}) \in \mathbb{Z}_p^{m_2}$ and $\mathsf{seed} = \mathbf{C}\mathbf{e}_1 \bmod q$. Here $F = \mathsf{PRNG}(\cdot)$ denotes a PRNG that is fed with an input seed statistically close to uniform as per the Leftover Hash Lemma for a uniform random matrix $\mathbf{C} \in \mathbb{Z}_q^{t \times m_1}$ with $t \leq (d - 2\lambda)/\log q$ and $d = H_\infty(\mathbf{e}_1)$. Following this approach, we proof that $\mathsf{decision\ A\text{-}LWE}_{n,m_1,m_2,\alpha q}$ is hard to solve assuming the hardness of $\mathsf{decision\ LWE}_{n-t,m,\alpha q}$ (The RO case is based on the hardness of $\mathsf{decision\ LWE}_{n,m,\alpha q}$). In principal, t is chosen to be very small such as $12 - 15$ in order to produce a seed with sufficient random bits. The resulting encryption schemes are almost as efficient as in the RO model. However, we lose a small message portion of size $m_1 \cdot \log(\alpha q/4.7)$ bits due to $\mathbf{e}_1$, which is sampled according to $\mathcal{D}_{\mathbb{Z}^{m_1}, \alpha q}$.

3.3. Learning with Errors Augmented with Auxiliary Data

In this section, we show how to inject further useful information in the error vectors of LWE instances without necessarily changing its distribution. We call this technique "message embedding" and formulate a modified LWE problem definition, namely the Augmented LWE (A-LWE) problem, which involves an error term produced by use of this new approach. We show that certain instantiations of the A-LWE problem are as hard as the original LWE problem.

3.3.1. Message Embedding

We start explaining the core functionality of our work leading to conceptually new cryptographic applications such as encryption schemes. In particular, we show how to generate vectors that encapsulate an arbitrary message while simultaneously following the discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z}^m, r}$. More specifically, Lemma 3.2 and

Lemma 3.4 are used, which essentially state that a discrete Gaussian over the integers can be simulated by sampling a coset $\mathbf{b} \in \mathbb{Z}_p^{n'}$ uniformly at random for any preimage sampleable full-rank matrix $\mathbf{B} \in \mathbb{Z}_p^{n' \times m}$ and then invoking a discrete Gaussian sampler outputting a vector from $\Lambda_{\mathbf{b}}^{\perp}(\mathbf{B}) = \mathbf{c} + \Lambda_p^{\perp}(\mathbf{B})$ for $\mathbf{B} \cdot \mathbf{c} \equiv \mathbf{b} \bmod q$. However, this requires the knowledge of a suitable basis for $\Lambda_p^{\perp}(\mathbf{B})$. In fact, the random selection of the coset $\mathbf{b}$ can be made deterministical by means of a random oracle H or PRNG taking a random seed with enough entropy as input. The fact that xoring a message $\mathbf{m}$ to the output of a random function F does not change the distribution, allows to hide the message within the error vector without changing its distribution. As a result, we obtain $\mathbf{e} \leftarrow_R D_{\Lambda_{\mathbf{b}}^{\perp}(\mathbf{B}),r}$, which is indistinguishable from $D_{\mathbb{Z}^m,r}$ for $\mathbf{b} = F(\mathsf{seed}) \oplus \mathbf{m}$ using a random seed and properly chosen parameters.

More formally, let the very simple operations $\mathsf{encode} : \{0,1\}^{n' \log p} \to \mathbb{Z}_p^{n'}$ and $\mathsf{decode} : \mathbb{Z}_p^{n'} \to \{0,1\}^{n' \log p}$ allow to bijectively switch between the bit and vector representations. The embedding approach is realized by use of any preimage sampleable full-rank matrix $\mathbf{B} \in \mathbb{Z}_p^{n' \times m}$. A first idea of doing this is to sample a preimage $\mathbf{x} \leftarrow_R D_{\Lambda_{\mathbf{v}}^{\perp}(\mathbf{B}),r}$ with $\mathbf{v} = \mathsf{encode}(\mathbf{m})$ for $r > \eta_\epsilon(\mathbf{B})$ and an arbitrary message $\mathbf{m} \in \{0,1\}^{n' \log p}$ such that $\mathbf{B} \cdot \mathbf{x} \mod q = \mathsf{encode}(\mathbf{m})$ holds. Sampling from $D_{\Lambda_{\mathbf{v}}^{\perp}(\mathbf{I}),r}$ for $\mathbf{B} = \mathbf{I}$ is performed very efficiently and can be reduced to samples from $\mathcal{D}_{p\mathbb{Z}+v_i,r}$. However, since the target Gaussian distribution of many cryptographic schemes, such as the LWE encryption schemes, requires to have support $\mathbb{Z}^m$, we have to modify the coset selection to $\mathbf{m} \oplus \mathbf{r}$ for a randomly chosen vector $\mathbf{r} \leftarrow_R \{0,1\}^{n' \log p}$ prior to invoking the preimage sampler. Below in Lemma 3.2 we show that given this setup we indeed obtain a sample $\mathbf{x}$ that is distributed just as $\mathcal{D}_{\mathbb{Z}^m,r}$ with overwhelming probability. To illustrate this approach exemplarily, let $\mathbf{e} \in \mathbb{Z}^m$ denote the error term. We then split the error term $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2) \in \mathbb{Z}^{m_1+m_2}$ into two subvectors, each serving for a different purpose. The second part $\mathbf{e}_2$ is used for message embedding, whereas $\mathbf{e}_1$ provides enough entropy in order to sample a random vector $\mathbf{r}$. To this end, one has to find a proper trade-off for the choice of m_1 and m_2, since a too large value for m_2 implies low entropy of $\mathbf{e}_1$. A reasonable small lower bound is given by $m_1 \geq n$, since the discrete Gaussian vector $\mathbf{e}_1$ has min-entropy of at least $n-1$ bits as per [GPV08, Lemma 2.10].

The message embedding functionality comes at almost no cost, since it does not involve any complex procedures. One proceeds as follows. First, it is required to sample $\mathbf{e}_1 \leftarrow \mathcal{D}_{\mathbb{Z}^{m_1},r}$ using an ordinary discrete Gaussian sampler such as the novel one that we introduce in Chapter 5, then one computes $\mathbf{u} = \mathsf{encode}(F(\mathbf{e}_1))$ for some random function $F : \{0,1\}^* \to \{0,1\}^{m_2 \log p}$ and finally samples a preimage $\mathbf{e}_2 \leftarrow_R \mathcal{D}_{\Lambda_{\mathbf{v}}^{\perp}(\mathbf{B}),r}$ for the syndrome $\mathbf{v} = \mathsf{encode}(\mathbf{m} \oplus \mathbf{u}) \in \mathbb{Z}_p^{n'}$ using a preimage sampleable matrix $\mathbf{B} \in \mathbb{Z}_p^{n' \times m_2}$. Following this approach, the message is recovered by computing $\mathbf{m} = F(\mathbf{e}_1) \oplus \mathsf{decode}(\mathbf{B} \cdot \mathbf{e}_2 \mod q)$. In many cryptographic applications there are different random sources available, which can replace the role of $\mathbf{e}_1$ such that the complete bandwidth of $\mathbf{e}$ is exploited for data injection. In the following theorems we prove that it is possible to simulate the discrete Gaussian distribution

$\mathcal{D}_{\mathbb{Z}^m,r}$ (statistically or computationally) by use of a preimage sampler for any full-rank matrix $\mathbf{B}$. For uniformly distributed error vectors, for which there exist also worst-case reductions [DMQ13, MP13], the discrete Gaussian step is omitted and the error vector is simply obtained via $\mathbf{e} = \mathsf{encode}(\mathbf{m} \oplus \mathbf{u}) \in \mathbb{Z}_p^{m_2}$, where p denotes the interval width of its components.

Lemma 3.1. *([MR04, Lemma 4.4]). Let Λ be any n-dimensional lattice. Then for any $\epsilon \in (0,1)$, $s \geq \eta_\epsilon(\Lambda)$, and $\mathbf{c} \in \mathbb{R}^n$, we have*

$$\rho_{s,\mathbf{c}}(\Lambda) \in [\frac{1-\epsilon}{1+\epsilon}, 1] \cdot \rho_s(\Lambda)\,.$$

Lemma 3.2 (**Statistical**). *Let $\mathbf{B} \in \mathbb{Z}_p^{n\times m}$ be an arbitrary full-rank matrix and $\epsilon = \mathsf{negl}(n)$. The statistical distance $\Delta(\mathcal{D}_{\mathbb{Z}^m,r}, \mathcal{D}_{\Lambda_{\mathbf{v}}^\perp(\mathbf{B}),r})$ for uniform $\mathbf{v} \leftarrow_R \mathbb{Z}_p^n$ and $r \geq \eta_\epsilon(\Lambda^\perp(\mathbf{B}))$ is negligible.*

Proof. Consider the statistical distance between $\mathcal{D}_{\mathbb{Z}^m,r}$ and $\mathcal{D}_{\Lambda_{\mathbf{v}}^\perp(\mathbf{B}),r}$, where $\mathbf{v} \in \mathbb{Z}_p^n$ is chosen at random. Since $\mathbf{B}$ is a full-rank matrix, we have $\mathbb{Z}^m = \dot{\bigcup}_{\mathbf{b}\in\mathbb{Z}_p^n} \Lambda_{\mathbf{b}}^\perp(\mathbf{B})$ and $\rho_r(\mathbb{Z}^m) = \sum_{\mathbf{b}\in\mathbb{Z}_p^n} \rho_r(\Lambda_{\mathbf{b}}^\perp(\mathbf{B})) \in [\frac{1-\epsilon}{1+\epsilon}, 1] \cdot p^n \cdot \rho_r(\Lambda_p^\perp(\mathbf{B}))$. In the latter distribution $\mathcal{D}_{\Lambda_{\mathbf{v}}^\perp(\mathbf{B}),r}$ the process of sampling $\mathbf{z} \in \mathbb{Z}^m$ can be reduced to the tasks of selecting the correct partition $\Lambda_{\mathbf{v}}^\perp(\mathbf{B})$ with probability $\frac{1}{p^n}$ and subsequently sampling $\mathbf{z}$ from $\Lambda_{\mathbf{v}}^\perp(\mathbf{B})$ with probability $\frac{\rho_r(\mathbf{z})}{\rho_r(\Lambda_{\mathbf{B}\cdot\mathbf{z}}^\perp(\mathbf{B}))}$. Following this, $\mathcal{D}_{\Lambda_{\mathbf{v}}^\perp(\mathbf{B}),r}$ outputs a sample $\mathbf{z}$ with probability $P[\mathbf{X} = \mathbf{z}] = \frac{1}{p^n} \cdot \frac{\rho_r(\mathbf{z})}{\rho_r(\Lambda_{\mathbf{B}\cdot\mathbf{z}}^\perp(\mathbf{B}))}$.

$$\begin{aligned}
\Delta(\mathcal{D}_{\mathbb{Z}^m,r}, \mathcal{D}_{\Lambda_{\mathbf{v}}^\perp(\mathbf{B}),r}) \quad &= \quad \sum_{\mathbf{z}\in\mathbb{Z}^m} \left| \frac{\rho_r(\mathbf{z})}{\rho_r(\mathbb{Z}^m)} - \frac{1}{p^n} \cdot \frac{\rho_r(\mathbf{z})}{\rho_r(\Lambda_{\mathbf{Bz}}^\perp(\mathbf{B}))} \right| \\
&\overset{\textit{Lemma } 3.1}{\in} \quad \sum_{\mathbf{z}\in\mathbb{Z}^m} \left| \frac{\rho_r(\mathbf{z})}{\rho_r(\mathbb{Z}^m)} - \frac{\rho_r(\mathbf{z})}{p^n \cdot [\frac{1-\epsilon}{1+\epsilon}, 1] \cdot \rho_r(\Lambda_p^\perp(\mathbf{B}))} \right| \\
&\overset{\textit{Lemma } 3.1}{\in} \quad \sum_{\mathbf{z}\in\mathbb{Z}^m} \left| \frac{\rho_r(\mathbf{z})}{\rho_r(\mathbb{Z}^m)} - \frac{\rho_r(\mathbf{z})}{[\frac{1-\epsilon}{1+\epsilon}, \frac{1+\epsilon}{1-\epsilon}] \cdot \sum_{\mathbf{b}\in\mathbb{Z}_p^n} \rho_r(\Lambda_{\mathbf{b}}^\perp(\mathbf{B}))} \right| \\
&= \quad \sum_{\mathbf{z}\in\mathbb{Z}^m} \left| \frac{\rho_r(\mathbf{z})}{\rho_r(\mathbb{Z}^m)} - \frac{[\frac{1-\epsilon}{1+\epsilon}, \frac{1+\epsilon}{1-\epsilon}] \cdot \rho_r(\mathbf{z})}{\rho_r(\mathbb{Z}^m)} \right| \\
&\in \quad [0, \frac{2\epsilon}{1-\epsilon}] \cdot \sum_{\mathbf{z}\in\mathbb{Z}^m} \left| \frac{\rho_r(\mathbf{z})}{\rho_r(\mathbb{Z}^m)} \right| \\
&\leq \quad \frac{2\epsilon}{1-\epsilon}
\end{aligned}$$

□

Lemma 3.3. *Let $\mathcal{X}_1$ be a distribution that is indistinguishable from $\mathcal{X}_2$ and M is an efficient non-uniform PPT operation. Then, $M(\mathcal{X}_1)$ is indistinguishable from $M(\mathcal{X}_2)$.*

Lemma 3.4 (**Computational**)**.** *Let $\mathbf{B} \in \mathbb{Z}_p^{n\times m}$ be an arbitrary full-rank matrix. If the distribution of $\mathbf{v} \in \mathbb{Z}_p^n$ is computationally indistinguishable from the uniform distribution over $\mathbb{Z}_p^n$, then $\mathcal{D}_{\Lambda_{\mathbf{v}}^\perp(\mathbf{B}),r}$ is computationally indistinguishable from $\mathcal{D}_{\mathbb{Z}^m,r}$ for $r \geq \eta_\epsilon(\Lambda^\perp(\mathbf{B}))$.*

Proof. Let $\mathbf{v}' \sim \mathcal{U}(\mathbb{Z}_p^n)$ be a vector chosen at random. By contradiction, we assume that $\mathbf{e} \sim \mathcal{D}_{\Lambda_{\mathbf{v}}^\perp(\mathbf{B}),r}$ is distinguishable from $\mathbf{e}' \sim \mathcal{D}_{\Lambda_{\mathbf{v}'}^\perp(\mathbf{B}),r} \overset{Lemma\ 3.2}{\approx_s} \mathcal{D}_{\mathbb{Z}^m,r}$ in polynomial time for the given parameters and $\mathbf{v}$ chosen as above. Then, $\mathbf{v}$ is computationally distinguishable from $\mathbf{v}'$ by Lemma 3.3 with $M(\mathbf{v}_i) = \mathcal{D}_{\Lambda_{\mathbf{v}_i}^\perp(\mathbf{B}),r}$. Hence, we have a contradiction. Therefore, the distribution $\mathcal{D}_{\Lambda_{\mathbf{v}}^\perp(\mathbf{B}),r}$ is computationally indistinguishable from $\mathcal{D}_{\mathbb{Z}^m,r}$. □

3.3.2. Augmented LWE - A Generic Approach

Based on the message embedding approach as described above, we introduce an alternative LWE definition that extends the previous one in such a way that the error term is augmented with additional information. We show how the modified error distribution still coincides with $\mathcal{D}_{\mathbb{Z}^m,r}$ in order to allow for a reduction from LWE to our new assumption. We start with a generalized description of the A-LWE distribution, where F stands for a random function. Below, in Section 3.4 and Section 3.5 we give a description of how to instantiate F in order to obtain a random oracle or standard model representation of the A-LWE problem and the related hardness statements.

In the following, we introduce the A-LWE distribution and the computational problems arising from this construction similar to LWE.

Definition 3.5 (Augmented LWE Distribution)**.** *Let $n, n', m, m_1, m_2, k, q, p$ be integers with $m = m_1 + m_2$, where $\alpha q \geq \eta_\epsilon(\Lambda^\perp(\mathbf{B}))$. Let $F : \mathbb{Z}_q^n \times \mathbb{Z}^{m_1} \to \{0,1\}^{n'\cdot\log(p)}$ be a function. Let $\mathbf{B} \in \mathbb{Z}_p^{n'\times m_2}$ be a preimage sampleable full-rank matrix (such as $\mathbf{B} = \mathbf{I}$). For $\mathbf{s} \in \mathbb{Z}_q^n$, define the A-LWE distribution $L^{\mathsf{A\text{-}LWE}}_{n,m_1,m_2,\alpha q}(\mathbf{m})$ with $\mathbf{m} \in \{0,1\}^{n'\log p}$ to be the distribution over $\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m$ obtained as follows:*

- *Sample $\mathbf{A} \leftarrow_R \mathbb{Z}_q^{n\times m}$ and $\mathbf{e}_1 \leftarrow_R \mathcal{D}_{\mathbb{Z}^{m_1},\alpha q}$.*
- *Set $\mathbf{v} = \mathsf{encode}(F(\mathbf{s}, \mathbf{e}_1) \oplus \mathbf{m}) \in \mathbb{Z}_p^{n'}$.*
- *Sample $\mathbf{e}_2 \leftarrow_R \mathcal{D}_{\Lambda_{\mathbf{v}}^\perp(\mathbf{B}),\alpha q}$.*
- *Return $(\mathbf{A}, \mathbf{b}^\top)$ where $\mathbf{b}^\top = \mathbf{s}^\top\mathbf{A} + \mathbf{e}^\top$ with $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2)$.*

Accordingly, we define the augmented LWE problem(s) as follows. As opposed to traditional LWE, augmented LWE blinds, in addition to the secret vector $\mathbf{s} \in \mathbb{Z}_q^n$, also some (auxiliary) data $\mathbf{m} \in \{0,1\}^{m_2}$. Thus, we have an additional assumption that the message $\mathbf{m}$ is hard to find given A-LWE samples. Note that the decision version requires that any polynomial bounded number of samples $(\mathbf{A}, \mathbf{b}^\top)$ from the A-LWE distribution is indistinguishable from uniform random samples in $\mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^m$. Its hardness implies that no information about $\mathbf{s}$ *and* $\mathbf{m}$ is leaked through A-LWE samples. In some scenarios, e.g., in security notions of an encryption scheme, the adversary may even choose the message $\mathbf{m}$. Hence, we require in the corresponding problems that their hardness holds with respect to A-LWE distributions with adversarially chosen message(s) $\mathbf{m}$ except for the search problem of $\mathbf{m}$.

Definition 3.6 (Augmented Learning with Errors (A-LWE)).
Let n, n', m_1, m_2, p, q be integers and $\mathbf{B} \in \mathbb{Z}_p^{n' \times m_2}$ be a preimage sampleable full-rank matrix. Let P (placeholder) stand for the model underlying the respective setting, where P is replaced either by $\mathcal{RO}$ for a random oracle model instantiation or $\mathcal{S}$ in case of the standard model variant.

The Decision Augmented Learning with Errors *(*decision $\text{A-LWE}^P_{n,m_1,m_2,\alpha q}$*) problem asks upon input $\mathbf{m} \in \{0,1\}^{n' \log p}$ to distinguish in polynomial time (in n) between samples $(\mathbf{A}_i, \mathbf{b}_i^\top) \leftarrow_R L^{\text{A-LWE}}_{n,m_1,m_2,\alpha q}(\mathbf{m})$ and uniform random samples from $\mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^n$ for a secret $\mathbf{s} \leftarrow_R \mathbb{Z}_q^n$.*

The Search-Secret Augmented Learning with Errors *(*search-s $\text{A-LWE}^P_{n,m_1,m_2,\alpha q}$*) problem asks upon input $\mathbf{m} \in \{0,1\}^{n' \log p}$ to output in polynomial time (in n) the vector $\mathbf{s} \in \mathbb{Z}_q^n$ given polynomially many samples $(\mathbf{A}_i, \mathbf{b}_i) \leftarrow_R L^{\text{A-LWE}}_{n,m_1,m_2,\alpha q}(\mathbf{m})$ for secret $\mathbf{s} \leftarrow_R \mathbb{Z}_q^n$.*

The Search-Message Augmented Learning with Errors *(*search-m $\text{A-LWE}^P_{n,m_1,m_2,\alpha q}$*) problem asks to output in polynomial time (in n) the vector $\mathbf{m}$ given polynomially many A-LWE samples $(\mathbf{A}_i, \mathbf{b}_i)$ for a secret $\mathbf{s} \leftarrow_R \mathbb{Z}_q^n$ and $\mathbf{m} \in \{0,1\}^{n' \log p}$.*

We say that decision/search-s/search-m A $\text{LWE}^P_{n,m_1,m_2,\alpha q}$ *is hard if all polynomial time algorithms solve the* decision/search-s/search-m A $\text{LWE}_{n,m_1,m_2,\alpha q}$ *problem only with negligible probability.*

We note that $\mathbf{B}$ can be specified to be the identity matrix $\mathbf{I} \in \mathbb{Z}_p^{m_2 \times m_2}$ for $n' = m_2$, which has some very nice properties as we will point out in the next chapter. In the following sections, we show that if the function F is instantiated by a random oracle or a PRNG in combination with a deterministic function, the hardness of LWE is reducible to the hardness of A-LWE. To this end, we show that the LWE and A-LWE distributions are computationally indistinguishable if we assume that the former search problem is hard and the inputs to the function F have sufficient entropy in each sample given previous samples.

3.4. Our Construction in the Random Oracle Model

In this section F is specified by a cryptographic hash function modeled as random oracle H. Such a choice has three major advantages. First, the output distribution of the random oracle is the uniform distribution, hence a desirable feature with regard to Lemma 3.4 and Lemma 3.2. More specifically, the syndrome in the A-LWE distribution has to be selected uniformly at random. In fact, this requirement is ensured by use of H. One has only to take care that the input to H has sufficient entropy. Second, the random oracle behaves deterministic on the same input queries. Third, correlations among the involved random variables are destroyed such that the aimed independence of the random variables can be guaranteed. This, however, occurs with overwhelming probability for random inputs with enough entropy. In this section F is specified by a cryptographic hash function modeled as random oracle H. Such a choice has three major advantages. First, the output distribution of the random oracle is the uniform distribution, hence a desirable feature with regard to Lemma 3.4 and Lemma 3.2. More specifically, the syndrome in the A-LWE distribution has to be selected uniformly at random. In fact, this requirement is ensured by use of H. One has only to take care that the input to H has sufficient entropy. Second, the random oracle behaves deterministic on the same input queries. Third, correlations among the involved random variables are destroyed such that the aimed independence of the random variables can be guaranteed. This, however, occurs with overwhelming probability for random inputs with enough entropy.

3.4.1. A-LWE Distribution

When instantiating the A-LWE distribution from Section 3.3.2 with a cryptographic hash function H modeled as random oracle, only step 2 has to be modified replacing F by H.

Definition 3.7 (Augmented LWE Distribution)**.** *Let n, n', m, m_1, m_2, q, p be integers with $m = m_1 + m_2$, where $\alpha q \geq \eta_\epsilon(\Lambda^\perp(\mathbf{B}))$ and $\epsilon = \mathsf{negl}(\lambda) > 0$. Let $H : \mathbb{Z}_q^n \times \mathbb{Z}^{m_1} \to \{0,1\}^{n'\cdot\log p}$ be a cryptographic hash function modeled as random oracle and $\mathbf{B} \in \mathbb{Z}_p^{n'\times m_2}$ a preimage sampleable full-rank matrix. For $\mathbf{s} \in \mathbb{Z}_q^n$, define the A-LWE distribution $L^{\mathsf{A\text{-}LWE}}_{n,m_1,m_2,\alpha q}(\mathbf{m})$ with $\mathbf{m} \in \{0,1\}^{n'\log p}$ to be the distribution over $\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m$ obtained as follows:*

- *Sample $\mathbf{A} \leftarrow_R \mathbb{Z}_q^{n\times m}$ and $\mathbf{e}_1 \leftarrow_R \mathcal{D}_{\mathbb{Z}^{m_1},\alpha q}$.*
- *Set $\mathbf{v} = \mathsf{encode}(H(\mathbf{s}, \mathbf{e}_1) \oplus \mathbf{m}) \in \mathbb{Z}_p^{n'}$.*
- *Sample $\mathbf{e}_2 \leftarrow_R \mathcal{D}_{\Lambda_{\mathbf{v}}^\perp(\mathbf{B}),\alpha q}$.*
- *Return $(\mathbf{A}, \mathbf{b}^\top)$ where $\mathbf{b}^\top = \mathbf{s}^\top\mathbf{A} + \mathbf{e}^\top$ with $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2)$.*

3.4.2. A-LWE Hardness

In the following section we present the hardness statements of the different A-LWE problems. In particular, we differentiate between the decision problem and the various search problems introduced in Definition 3.6.

Theorem 3.8. *Let n, n', m, m_1, m_2, p, q be integers with $m = m_1 + m_2$. Let $H : \mathbb{Z}_q^n \times \mathbb{Z}^{m_1} \to \{0,1\}^{n' \log p}$ be a cryptographic hash function modeled as random oracle. Let $\mathbf{B} \in \mathbb{Z}_p^{n' \times m_2}$ be a preimage sampleable full-rank matrix and $\alpha q \geq \eta_\epsilon(\Lambda_q^\perp(\mathbf{B}))$ for a real $\epsilon = \mathsf{negl}(\lambda) > 0$, where λ denotes the security parameter. Furthermore, denote by χ_s and χ_{e_1} the distributions of the random vectors $\mathbf{s}$ and $\mathbf{e}_1$ involved in each A-LWE sample. If $\mathbb{H}_\infty(\mathbf{s}, \mathbf{e}_1) > \lambda$, then $L^{\mathsf{A\text{-}LWE}}_{n,m_1,m_2,\alpha q}(\mathbf{m})$ is computationally indistinguishable from $L^{\mathsf{LWE}}_{n,m,\alpha q}$ for arbitrary message $\mathbf{m} \in \{0,1\}^{n' \cdot \log p}$.*

Proof. We need to show that samples from $L^{\mathsf{LWE}}_{n,m,\alpha,q}$ are indistinguishable from $L^{\mathsf{A\text{-}LWE}}_{n,m_1,m_2,\alpha,q}(\mathbf{m})$ if we assume that the $\mathsf{search\ LWE}_{n,m,\alpha,q}$ problem is hard to solve in polynomial time and tuples $(\mathbf{s}, (\mathbf{e}_1)_i)$ for each sample i have sufficient entropy . That is, $L^{\mathsf{LWE}}_{n,m,\alpha,q} \approx_c L^{\mathsf{A\text{-}LWE}}_{n,m_1,m_2,\alpha,q}(\mathbf{m})$ for arbitrary $\mathbf{m} \in \{0,1\}^{n' \cdot \log p}$.

We consider a series of intermediate hybrid experiments. In the first hybrid, we modify the A-LWE samples in such a way that we replace $H(\mathbf{s}, \mathbf{e}_1)$ with a uniformly sampled value $\mathbf{u}$. Here, we use the fact, that $\mathbb{H}_\infty(\mathbf{s}, \mathbf{e}_1) > \lambda$ and the same input will be queried with negligible probability. Consequently, $\mathbf{v} = \mathsf{encode}(H(\mathbf{s}, \mathbf{e}_1) \oplus \mathbf{m})$ becomes uniformly distributed. The next hybrid replaces $\mathbf{e}_2$ by value $\mathbf{e}_2^*$ which is sampled according to $\mathcal{D}_{\mathbb{Z}^{m_2}, r}$. The resulting A-LWE distribution coincides with the original LWE distribution. In the following we describe the hybrids more formally.

Hybrid$_1$. In the first hybrid, in each A-LWE sample we replace the value $H(\mathbf{s}, \mathbf{e}_1)$ by a uniformly sampled value $\mathbf{u} \in \{0,1\}^{m_2}$. We argue that a (polynomial-time) distinguisher notices the difference only if it queries the random oracle on input $(\mathbf{s}, \mathbf{e}_1)$. Otherwise, if $(\mathbf{s}, \mathbf{e}_1)$ has not been queried before, the distribution of $H(\mathbf{s}, \mathbf{e}_1)$ is statistically close to the uniform distribution in $\{0,1\}^{n' \cdot \log p}$ due to the property of a random oracle drawing elements from the output range uniformly at random. Moreover, we have $\mathbb{H}_\infty(\mathbf{s}, \mathbf{e}_1) > \lambda$ such that the same input element $(\mathbf{s}, \mathbf{e}_1)$ will not be sampled again except with negligible probability. This holds, in particular, if many samples are given to the distinguisher and all $H(\mathbf{s}, (\mathbf{e}_1)_i)$ have been replaced because by assumption we have sufficient entropy such that all pairs $(\mathbf{s}, (\mathbf{e}_1)_i)$ are distinct with overwhelming probability.

We comment on a distinguisher which queries the random oracle at a certain point on $(\mathbf{s}, \mathbf{e}_1)$ below in the proof, and assume for now, that no such distinguisher exists.

Hybrid$_2$. In the next hybrid, we replace the error term $\mathbf{e}_2$ by value $\mathbf{e}_2^*$ which is sampled according to $\mathcal{D}_{\mathbb{Z}^{m_2}, r}$. Note that A-LWE samples from **Hybrid$_1$** satisfy that $\mathbf{v} = \mathsf{encode}(\mathbf{u} \oplus \mathbf{m})$ is uniformly distributed since $\mathbf{u}$ is uniformly picked (even if $\mathbf{m}$ is chosen by the distinguisher). Now, Lemma 3.2 implies that $\mathcal{D}_{\Lambda^{\perp}_{\mathbf{v}}(\mathbf{A}),r}$ is statistically indistinguishable from $\mathcal{D}_{\mathbb{Z}^{m_2},r}$ for $r \geq \eta_{\epsilon}(\Lambda^{\perp}(\mathbf{A}))$, if H has not been queried on input $(\mathbf{s}, \mathbf{e}_1)$ before. For this reason, replacing $\mathbf{e}_2$, which is distributed according to $\mathcal{D}_{\Lambda^{\perp}_{\mathbf{v}}(\mathbf{A}),r}$, by vector $\mathbf{e}_2^*$ is unnoticeable to a distinguisher.

We argue that A-LWE samples from $\mathbf{Hybrid}_2$ are indistinguishable from LWE samples. This follows from the fact that the error term in A-LWE is now identically distributed as LWE which is the only difference between A-LWE and LWE samples. We still need to argue that it is very unlikely that a distinguisher queries the random oracle H on input $(\mathbf{s}, \mathbf{e}_1)$ for some $\mathbf{e}_1$ used in an A-LWE sample.

Suppose that there exists an algorithm $\mathcal{A}$ which distinguishes in polynomial time original A-LWE samples from A-LWE samples from $\mathbf{Hybrid}_1$ with non-negligible probability. We then construct an adversary $\mathcal{A}_{LWE}$ with black-box access to algorithm $\mathcal{A}$ that solves the search $\mathsf{LWE}_{n,m,\alpha,q}$ problem in polynomial time with non-negligible probability. This contradicts the theorem assumption that search $\mathsf{LWE}_{n,m,\alpha,q}$ is hard.

Adversary $\mathcal{A}_{LWE}$ is given samples from $L^{\mathsf{LWE}}_{n,m,\alpha,q}$ and is asked to find the secret vector $\mathbf{s}$. Let us denote by q^* the query $(\mathbf{s}, \mathbf{e}_1)$ on H made by $\mathcal{A}$, where q^* is polynomially bounded by the security parameter. Whenever algorithm $\mathcal{A}$ asks for new samples, $\mathcal{A}_{LWE}$ asks for samples in her challenge and forwards them to $\mathcal{A}$. That is, $\mathcal{A}$ obtains samples from $L^{\mathsf{LWE}}_{n,m,\alpha,q}$ instead of either version of $L^{\mathsf{A\text{-}LWE}}_{n,m_1,m_2,\alpha,q}(\mathbf{m})$. We have already shown via hybrids that $L^{\mathsf{A\text{-}LWE}}_{n,m_1,m_2,\alpha,q}(\mathbf{m})$ is indistinguishable from $L^{\mathsf{LWE}}_{n,m,\alpha,q}$, if $(\mathbf{s}, \mathbf{e}_1)$ was not sent to oracle H. This means that before $\mathcal{A}$ makes query q^* to H, those samples are indistinguishable. As a result, $\mathcal{A}$ must query H on input $(\mathbf{s}, \mathbf{e}_1)$ even if given LWE samples. We stress that after returning the hash value of $(\mathbf{s}, \mathbf{e}_1)$ to $\mathcal{A}$ it may be noticed that $\mathcal{A}_{LWE}$ has tricked her. However, eavesdropping the input to oracle H suffices for $\mathcal{A}_{LWE}$ to break her search $\mathsf{LWE}_{n,m,\alpha,q}$ problem independently whether $\mathcal{A}$ aborts at this time. Hence, if $\mathcal{A}$ queries H on input $(\mathbf{s}, \mathbf{e}_1)$ with non-negligible probability, so does $\mathcal{A}_{LWE}$ solve the search $\mathsf{LWE}_{n,m,\alpha,q}$ problem with the very same probability. By assumption there does not exist such a successful algorithm.

We conclude that the step from the original A-LWE samples to $\mathbf{Hybrid}_1$ will be unnoticeable to a distinguisher if search $\mathsf{LWE}_{n,m,\alpha,q}$ is hard, and both distributions $L^{\mathsf{LWE}}_{n,m,\alpha,q}$ and $L^{\mathsf{A\text{-}LWE}}_{n,m_1,m_2,\alpha,q}(\mathbf{m})$ are computationally indistinguishable. □

Note that if the first error part $\mathbf{e}_1$ has entropy exceeding the security parameter λ, the (computational) entropy induced by $\mathbf{s}$ is not required. This is important, since a distinguisher could ask for many A-LWE samples using the same secret $\mathbf{s}$ as input to the hash function. However, as typical in encryption schemes (e.g., in [Pei09, LPR10, LP11, MP12] and in ours), if we fix a random matrix $\mathbf{A}$ and sample fresh secret vectors $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ uniformly at random for each A-LWE sample, we can indeed choose m_1 to be zero. This corresponds to the case, where an A-LWE sample is drawn once for every fresh secret $\mathbf{s}$ resulting in essentially unrelated A-

LWE instances. Hence, the secret $\mathbf{s}$ provides the sufficient randomness required as input to H. Theorem 3.8 immediately entails the following statement.

Theorem 3.9. *Let n, n', m, m_1, m_2, p, q be integers with $m = m_1 + m_2$. Let H be a cryptographic hash function modeled as random oracle as defined in Theorem 3.8. Let $\mathbf{B} \in \mathbb{Z}_p^{n' \times m_2}$ be a preimage sampleable full-rank matrix and $\alpha q \geq \eta_\epsilon(\Lambda_q^\perp(\mathbf{B}))$ for a real $\epsilon = \mathsf{negl}(\lambda) > 0$. Furthermore, denote by χ_s and χ_{e_1} the distributions of the random vectors $\mathbf{s}$ and $\mathbf{e}_1$ involved in each A-LWE sample. If $\mathbb{H}_\infty(\mathbf{s}, \mathbf{e}_1) > \lambda$, then the following statements hold.*

1. *If* $\mathsf{search}\ \mathsf{LWE}_{n,m,\alpha q}$ *is hard, then* $\mathsf{search\text{-}s}\ \mathsf{A\text{-}LWE}^{\mathcal{RO}}_{n,m_1,m_2,\alpha q}$ *is hard.*
2. *If* $\mathsf{decision}\ \mathsf{LWE}_{n,m,\alpha q}$ *is hard, then* $\mathsf{decision}\ \mathsf{A\text{-}LWE}^{\mathcal{RO}}_{n,m_1,m_2,\alpha q}$ *is hard.*
3. *If* $\mathsf{decision}\ \mathsf{LWE}_{n,m,\alpha q}$ *is hard, then* $\mathsf{search\text{-}m}\ \mathsf{A\text{-}LWE}^{\mathcal{RO}}_{n,m_1,m_2,\alpha q}$ *is hard.*

Proof. As per Theorem 3.8, $L^{\mathsf{A\text{-}LWE}}_{n,m_1,m_2,\alpha q}(\mathbf{m})$ is computationally indistinguishable from $L^{\mathsf{LWE}}_{n,m,\alpha q}$. This proves the hardness of $\mathsf{decision}\ \mathsf{A\text{-}LWE}^{\mathcal{RO}}_{n,m_1,m_2,\alpha q}$ and $\mathsf{search\text{-}m}\ \mathsf{A\text{-}LWE}^{\mathcal{RO}}_{n,m_1,m_2,\alpha q}$. And by essentially the same arguments we also deduce the hardness of $\mathsf{search\text{-}s}\ \mathsf{A\text{-}LWE}^{\mathcal{RO}}_{n,m_1,m_2,\alpha q}$, because solving the search problem implies distinguishability of A-LWE instances from uniform due to the knowledge of $(\mathbf{s}, \mathbf{e})$ and by Theorem 3.8 we obtain distinguishability of LWE instances from uniform, hence a contradiction. $\square$

We note, that these hardness results can directly be translated to the corresponding ring variants.

3.5. Our Construction in the Standard Model

In this section, we introduce an A-LWE variant that is hard under standard assumptions. Previously, the A-LWE problem was defined by use of a random oracle, that helped to destroy correlations between the secret and the output of $H(\cdot)$ and thus ensured the required distributions. Therefore, a straightforward approach to instantiate $H(\cdot)$ by a PRNG or a pseudo random function (PRF) is not obvious. In particular, by means of a random oracle the secret $\mathbf{s}$ and the output of $H(\cdot)$ are both uniformly random and hence allow the distributions of the error term and the secret behave following the basic LWE distributions such that it is not possible to distinguish between A-LWE and original LWE samples. In this section we show how to get rid of the random oracle leading to a standard model instantiation of the A-LWE problem. We formulate the A-LWE problem, where $F(\cdot, \cdot)$ is instantiated by means of a PRNG, hence, avoiding the need for a cryptographic hash function modeled as random oracle. Following this, we can prove security in the standard model for many of the A-LWE based encryption schemes which are shown to be secure in the random oracle model (see Chapter 4).

3.5.1. Tools

Prior to defining the A-LWE distribution and proving the related hardness statements, an efficient mechanism is needed that allows to replace the random oracle by tools based on standard assumptions but in such a way that the message can efficiently be recovered from the error term. Our approach is based on the idea of computational extractors following the extract-then-expand design [DSGKM12], where a statistical extractor is used to produce a seed as input to a PRNG expanding the seed to the desired output length. We will prove that we can deterministically produce a seed using the ingredients of LWE samples. In fact, we use a small part of the error term in order to derive a seed statistically close to uniform given LWE samples. That is, we sample a random matrix $\mathbf{C} \in \mathbb{Z}_q^{t\times n}$ for $t < n$ and prove that $\mathbf{C}\cdot\mathbf{e}_1 \equiv \mathbf{b} \bmod q$ is indistinguishable from a random vector in $\mathbb{Z}_q^t$ given $\mathbf{A}^\top\mathbf{s}+\mathbf{e} \bmod q$ for $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2)^\top \leftarrow \mathcal{D}_{\mathbb{Z}_q^m,\alpha q}$ and random matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$.

Lemma 3.10. ***([AGV09, Theorem 3])*** *For any integer $n > 0$, integer $q \geq 2$, an error-distribution $\chi^n = \mathcal{D}_{\mathbb{Z}^n,\alpha q}$ and any subset $S \subseteq \{0,\ldots,n\}$, the two distributions $(\mathbf{A}, \mathbf{A}^\top\mathbf{s}+\mathbf{e}, (s_i)_{i\in S})$ and $(\mathbf{A}, \mathbf{A}^\top\mathbf{s}+\mathbf{e}, \mathcal{U}(\mathbb{Z}_q^{|S|}))$ are computationally indistinguishable assuming the hardness of* decision $\mathsf{LWE}_{n-|S|,m,\alpha q}$.

As a result, it follows

$$\begin{aligned}(\mathbf{A}, \mathbf{A}^\top\mathbf{s}+\mathbf{e}, (s_i)_{i\in S}) &\approx_c (\mathbf{A}, \mathbf{A}^\top\mathbf{s}+\mathbf{e}, \mathcal{U}(\mathbb{Z}_q^{|S|}))\\ &\approx_c (\mathbf{A}, \mathcal{U}(\mathbb{Z}_q^{m}), \mathcal{U}(\mathbb{Z}_q^{|S|})),\end{aligned}$$

which proves the independence of $(s_i)_{i\in S}$ from the remaining parts of $\mathbf{s} = (s_i)_{i\in[n]}$ for $S \subseteq [n]$. As an immediate consequence, we obtain similar results, if $\mathbf{s}$ is sampled from the error distribution χ^n.

Corollary 3.11. *For any integer $n > 0$, integer $q \geq 2$, an error-distribution $\chi^n = \mathcal{D}_{\mathbb{Z}^n,\alpha q}$ and any subset $S \subseteq \{0,\ldots,n\}$, the two distributions $(\mathbf{A}, \mathbf{A}^\top\mathbf{s}+\mathbf{e}, (s_i)_{i\in S})$ and $(\mathbf{A}, \mathbf{A}^\top\mathbf{s}+\mathbf{e}, \mathcal{D}_{\mathbb{Z}_q^{|S|},\alpha q})$ are computationally indistinguishable assuming the hardness of* decision $\mathsf{LWE}_{n-|S|,m,\alpha q}$.

This is proven in a straightforward manner following the same proof steps as in Lemma 3.10. However, in this case we have

$$\begin{aligned}(\mathbf{A}, \mathbf{A}^\top\mathbf{s}+\mathbf{e}, (s_i)_{i\in S}) &\approx_c (\mathbf{A}, \mathbf{A}^\top\mathbf{s}+\mathbf{e}, \mathcal{D}_{\mathbb{Z}_q^{|S|},\alpha q})\\ &\approx_c (\mathbf{A}, \mathcal{U}(\mathbb{Z}_q^{m}), \mathcal{D}_{\mathbb{Z}_q^{|S|},\alpha q}),\end{aligned}$$

which is based on the hardness of decision $\mathsf{LWE}_{n-|S|,m,\alpha q}$. In order to account for leakage of coefficients we give an alternative definition of the LWE distribution allowing for leakage of t coefficients either of the error term or secret vector. Such a notation is indeed required to sample n-dimensional LWE instances based on the hardness of decision $\mathsf{LWE}_{n-t,m,\alpha q}$. In particular, when proving the hardness of the A-LWE problem it simplifies the representation of such instances.

Definition 3.12 (LWE Distribution with Leakage)**.** *Let n, m, q be integers and χ_e be the error distribution over $\mathbb{Z}$. By $L_{n,m,\alpha q}^{\mathsf{LWE}_{\ell(t)}}$ we denote the LWE distribution over $\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m$, which draws $\mathbf{A} \leftarrow_R \mathbb{Z}_q^{n\times m}$ uniformly at random, samples $\mathbf{e} \leftarrow_R \mathcal{D}_{\mathbb{Z}^m,\alpha q}$ and returns $(\mathbf{A}, \mathbf{b}^\top) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m$, where $\mathbf{b}^\top = \mathbf{s}^\top\mathbf{A} + \mathbf{e}^\top$ and $t > 0$ coefficients of $\mathbf{s} \in \mathbb{Z}_q^n$ are leaked.*

We use the convention that the first t coefficients are leaked if it is not explicitly specified. From Lemma 3.10 (or Corollary 3.11) it follows that it is hard to distinguish uniform random vectors from samples following the $L_{n,m,\alpha q}^{\mathsf{LWE}_{\ell(t)}}$ distribution assuming the hardness of decision $\mathsf{LWE}_{n-t,m,\alpha q}$. The next theorem provides a description of how to deterministically deduce a vector statistically close to uniform from LWE samples. This vector will serve as a seed in order to instantiate the A-LWE distribution below.

Theorem 3.13. *Let m, n be integers and $\mathbf{s} \leftarrow_R \mathcal{D}_{\mathbb{Z}^m,\alpha q}$. Furthermore, let $\mathbf{A}_2 \leftarrow_R \mathbb{Z}_q^{m\times n}$, $\mathbf{A}_1 \leftarrow_R \mathbb{Z}_q^{t\times n}$ and $\mathbf{A}_1' \leftarrow_R \mathbb{Z}_q^{t\times m}$ be uniform random matrices with $t \leq (d - 2\lambda)/\log q$, where $d = H_\infty(\mathbf{s})$ (resp. $d = H_\infty(\mathbf{e})$). Suppose that the* decision $\mathsf{LWE}_{n-t,m,\alpha q}$ *assumption holds, then*

1. $(\mathbf{A}_1\mathbf{s}, \mathbf{A}_2\mathbf{s} + \mathbf{e}) \approx_c (\mathcal{U}(\mathbb{Z}_q^t), \mathcal{U}(\mathbb{Z}_q^m))$
2. $(\mathbf{A}_1'\mathbf{e}, \mathbf{A}_2\mathbf{s} + \mathbf{e}) \approx_c (\mathcal{U}(\mathbb{Z}_q^t), \mathcal{U}(\mathbb{Z}_q^m))$

for $\mathbf{e} \leftarrow_R \mathcal{D}_{\mathbb{Z}^m,\alpha q}$. Moreover, $\mathbf{A}_1'\mathbf{e} \bmod q$ (resp. $\mathbf{A}_1\mathbf{s} \bmod q$) is a statistical extractor.

Proof. We prove the statement $(\mathbf{A}_1\mathbf{s}, \mathbf{A}_2\mathbf{s} + \mathbf{e}) \approx_c (\mathcal{U}(\mathbb{Z}_q^t), \mathcal{U}(\mathbb{Z}_q^m))$ by contradiction. Suppose there exists a PPT distinguisher D that distinguishes between the aforementioned distributions, then we construct a PPT algorithm $\mathcal{A}$ that breaks decision $\mathsf{LWE}_{n-t,m,\alpha q}$.
We note here that it suffices to prove the statements for any m as long as it is polynomially bounded. In fact, if the underlying problem is hard for $m = n$ samples, the same hardness statement must also hold for less samples. And for $m > n$, one can easily reduce the problem to $m = n$.
The input to $\mathcal{A}$ is an instance $(\mathbf{A}, \mathbf{A} \cdot \begin{bmatrix}\mathbf{0}\\ \mathbf{e}\end{bmatrix} + \mathbf{s})$ of $L_{n,n,\alpha q}^{\mathsf{LWE}_{\ell(t)}}$ due to leakage of the first t zero entries, where $\mathbf{A} = \begin{bmatrix}\mathbf{A}_1\\ \mathbf{A}_2\end{bmatrix} \in \mathbb{Z}_q^{n\times n}$ composed by the matrices $\mathbf{A}_1 \in \mathbb{Z}_q^{t\times n}$ and $\mathbf{A}_2 \in \mathbb{Z}_q^{n-t\times n}$ is a uniform random matrix that is invertible over $\mathbb{Z}_q^{n\times n}$ with non-negligible probability. For instance, the probability to sample a full rank matrix from $\mathbb{Z}_q^{l\times n}$ is equal to $\prod_{i=0}^{l-1} \frac{q^n - q^i}{q^{ln}} \geq \frac{(q^n - q^{l-1})^l}{q^{ln}} = (\frac{q^{n-l+1}-1}{q^{(n-l+1)}})^l$ for prime modulus and $l \leq n$. If $l = n$, the matrix is invertible with probability at least $(\frac{q-1}{q})^n \geq 1/e$ for $q = O(n)$. Now, $\mathcal{A}$ computes

$$\mathbf{A}^{-1} \cdot (\mathbf{A}\begin{bmatrix}\mathbf{0}\\ \mathbf{e}\end{bmatrix} + \mathbf{s}) = \mathbf{A}^{-1} \cdot \mathbf{s} + \begin{bmatrix}\mathbf{0}\\ \mathbf{e}\end{bmatrix},$$

which can be represented as $(\mathbf{Bs}, \mathbf{Cs}+\mathbf{e})$ for $\mathbf{A}^{-1} = \begin{bmatrix}\mathbf{B}\\\mathbf{C}\end{bmatrix}$. Subsequently, D is invoked which distinguishes the input $(\mathbf{Bs}, \mathbf{Cs}+\mathbf{e})$ and hence $\mathbf{A}\cdot\begin{bmatrix}\mathbf{0}\\\mathbf{e}\end{bmatrix}+\mathbf{s}$ from uniform, thus solving decision $\mathsf{LWE}_{n-t,m,\alpha q}$ as per Lemma 3.10 and Corollary 3.11. For the second statement, one observes that

$$(\mathbf{A}_1\mathbf{e}, \mathbf{A}_2\mathbf{s}+\mathbf{e}) \approx_c (\mathcal{U}(\mathbb{Z}_q^t), \mathcal{U}(\mathbb{Z}_q^m)) \iff (\mathbf{A}_1\mathbf{e}, \mathbf{A}_2^{-1}\mathbf{e}+\mathbf{s}) \approx_c (\mathcal{U}(\mathbb{Z}_q^t), \mathcal{U}(\mathbb{Z}_q^m))$$

for $m = n$ and invertible matrix $\mathbf{A}_2$, which exists with non-negligible probability as shown before. This particularly also proves the statement for $m \leq n$.
As for $m > n$, suppose the upper n rows of $\mathbf{A}_2$ are linearly independent, otherwise we can find n out of $m > n$ rows with high probability and bundle them together in the upper part. The matrix $\mathbf{A}_2$ can subsequently be extended to an invertible matrix $\mathbf{A}_e = \begin{bmatrix}\mathbf{B} & \mathbf{A}_2\end{bmatrix} \in \mathbb{Z}_q^{m\times m}$ with $\mathbf{B} = \begin{bmatrix}\mathbf{0}\\\mathbf{I}\end{bmatrix} \in \mathbb{Z}_q^{m\times m-n}$ by appending ones on the diagonal such that $\mathbf{A}_e\begin{bmatrix}\mathbf{0}\\\mathbf{s}\end{bmatrix}+\mathbf{e} = \mathbf{A}_2\mathbf{s}+\mathbf{e} \bmod q$.

$$(\mathbf{A}_1\mathbf{e}, \mathbf{A}_2\mathbf{s}+\mathbf{e}) = (\mathbf{A}_1\mathbf{e}, \mathbf{A}_e\begin{bmatrix}\mathbf{0}\\\mathbf{s}\end{bmatrix}+\mathbf{e}) \approx_c (\mathcal{U}(\mathbb{Z}_q^t), \mathcal{U}(\mathbb{Z}_q^m)) \iff (\mathbf{A}_1\mathbf{e}, \mathbf{A}_e^{-1}\mathbf{e}+\begin{bmatrix}\mathbf{0}\\\mathbf{s}\end{bmatrix}) \approx_c (\mathcal{U}(\mathbb{Z}_q^t), \mathcal{U}(\mathbb{Z}_q^m))$$

Hence, our claim follows from the first case. We note that $\mathbf{A}_1\mathbf{e} \bmod q$ is a statistical extractor by the Leftover Hash Lemma, if $t \leq (d-2\lambda)/\log q$ for $d = H_\infty(\mathbf{e})$. □

Remark. The theorem above mainly states, that it is even possible to reveal the first t entries of the error-term from LWE samples. This is equivalent to sampling a random matrix $\mathbf{A}_1 \in \mathbb{Z}_q^{t\times n}$ and outputting the vector $\mathbf{A}_1\mathbf{s} \bmod q$, which is statistically close to the uniform distribution for $t \leq (d-2\lambda)/\log q$ according to the Leftover Hash Lemma. Applying the same argument, the complete vector $\begin{bmatrix}\mathbf{A}_1\\\mathbf{A}_2\end{bmatrix}\mathbf{s}+\begin{bmatrix}\mathbf{0}\\\mathbf{e}\end{bmatrix}$ is statistically close to uniform if $m+t<n$.

3.5.2. A-LWE Distribution

The tools introduced in the previous section will allow us to deterministically derive a uniform random seed by means of the error term from LWE samples. Using this framework we start defining the A-LWE distribution $L^{\text{A-LWE}}_{n,m_1,m_2,\alpha q}(\mathbf{m})$ in the standard model. It looks very similar to the random oracle variant introduced in the previous section.

Definition 3.14 (Augmented LWE Distribution)**.** *Let $n, n', t, m, m_1, m_2, q, p$ be integers with $m = m_1 + m_2$, where $\alpha q \geq \eta_\epsilon(\Lambda^\perp(\mathbf{B}))$ for a preimage sampleable full-rank matrix $\mathbf{B} \in \mathbb{Z}_p^{n' \times m_2}$ and a real $\epsilon = \mathsf{negl}(\lambda) > 0$. Let $\mathbf{C} \in \mathbb{Z}_q^{t \times m_1}$ be a random matrix and $\mathsf{PRNG} : \mathbb{Z}_q^t \to \{0,1\}^{n' \cdot \log p}$ be a secure pseudo random generator. For $\mathbf{s} \in \mathbb{Z}_q^n$, define the A-LWE distribution $L^{\mathsf{A\text{-}LWE}}_{n,m_1,m_2,\alpha q}(\mathbf{m})$ with $\mathbf{m} \in \{0,1\}^{n' \cdot \log p}$ to be the distribution over $\mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^m$ obtained as follows:*

- *Sample $\mathbf{A} \leftarrow_R \mathbb{Z}_q^{n \times m}$ and $\mathbf{e}_1 \leftarrow_R \mathcal{D}_{\mathbb{Z}^{m_1},\alpha q}$.*
- *Set $\mathsf{seed} = \mathbf{C} \cdot \mathbf{e}_1 \bmod q$.*
- *Set $\mathbf{v} = \mathsf{encode}(\mathsf{PRNG}(\mathsf{seed}) \oplus \mathbf{m}) \in \mathbb{Z}_p^{n'}$.*
- *Sample $\mathbf{e}_2 \leftarrow_R \mathcal{D}_{\Lambda_{\mathbf{v}}^\perp(\mathbf{B}),\alpha q}$.*
- *Return $(\mathbf{A}, \mathbf{b}^\top)$ where $\mathbf{b}^\top = \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top$ with $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2)$.*

We note that we introduced many parameters in order to keep the problem as general as possible. But for applications considered in Chapter 4 and Chapter 5, it suffices to set $n' = m_2$, $p = \lfloor \alpha q / \sqrt{\ln(2(1+1/\epsilon))/\pi} \rfloor$ and $\Lambda^\perp(\mathbf{B}) = p\mathbb{Z}^{m_2}$ for $\mathbf{B} = \mathbf{I} \in \mathbb{Z}_p^{m_2}$ as we will show in the following chapter. For our constructions to work, it is possible to select small values for t such as $t = 15$ in order to ensure a seed of size $15 \cdot \log q$ bits. In fact, the seed is statistically close to uniform for $t \leq (d - 2\lambda)/\log q$ and $d = H_\infty(\mathbf{e}_1)$. In this case $\mathsf{PRNG}(\mathsf{seed})$ represents a computational extractor providing enough pseudo random bits in order to conceal the message.

3.5.3. A-LWE Hardness

The following theorem shows that LWE samples are computationally indistinguishable from A-LWE samples, when instantiated with a PRNG. Consequently, this construction inherently depends on the underlying computational problem of the PRNG.

Theorem 3.15. *Denote by λ the security parameter and let $n, n', t, m, m_1, m_2, q, p$ be integers with $m = m_1 + m_2$, where $\alpha q \geq \eta_\epsilon(\Lambda^\perp(\mathbf{B}))$ for a preimage sampleable full-rank matrix $\mathbf{B} \in \mathbb{Z}_p^{n' \times m_2}$ and a real $\epsilon = \mathsf{negl}(\lambda) > 0$. Let $\mathbf{C} \in \mathbb{Z}_q^{t \times m_1}$ be a random matrix and $\mathsf{PRNG} : \mathbb{Z}_q^t \to \{0,1\}^{n' \cdot \log p}$ be a secure pseudo random generator with $t \leq (d - 2\lambda)/\log q$ for $d = H_\infty(\mathbf{e}_1)$ and $\mathbf{e}_1 \leftarrow_R \mathcal{D}_{\mathbb{Z}^{m_1},\alpha q}$. Then, assuming the hardness of $\mathsf{decision\ LWE}_{n-t,m,\alpha q}$ the distribution $L^{\mathsf{A\text{-}LWE}}_{n,m_1,m_2,\alpha q}(\mathbf{m})$ is computationally indistinguishable from uniform for arbitrary $\mathbf{m} \in \{0,1\}^{n' \cdot \log p}$.*

Proof. Via a series of hybrids we will prove that samples from $L^{\mathsf{A\text{-}LWE}}_{n,m_1,m_2,\alpha q}(\mathbf{m})$ are indistinguishable from the uniform distribution over $\mathbb{Z}_q^{n \times m}$ assuming that the $\mathsf{decision\ LWE}_{n-t,m,\alpha,q}$ problem is hard to solve in polynomial time and the output distribution of the PRNG is indistinguishable from uniform for PPT adversaries, where $t \leq (d - 2\lambda)/\log q$ with $d = H_\infty(\mathbf{e}_1)$. This leads to

$$L^{\mathsf{A\text{-}LWE}}_{n,m_1,m_2,\alpha,q}(\mathbf{m}) \approx_c L^{\mathsf{LWE}_{\ell(t)}}_{n,m,\alpha q}$$

for arbitrary $\mathbf{m} \in \{0,1\}^{n' \cdot \log p}$. In the first hybrid, we modify the A-LWE samples in such a way that we replace $\mathsf{seed} = \mathbf{C}\mathbf{e}_1 \bmod q$ with a uniformly sampled value $\mathbf{u}_1$. This follows from the fact that $(\mathbf{C}\mathbf{e}_1, \mathbf{A}_1^\top \mathbf{s} + \mathbf{e}_1)$ is indistinguishable from uniform according to Theorem 3.13 and $\mathbf{C}\mathbf{e}_1$ is statistically close to uniform following the Leftover Hash Lemma for t chosen as above. In the next hybrid we replace the output of the PRNG with a uniform random value $\mathbf{u}_2$ as a result of the uniform random input $\mathbf{u}_1$ to the PRNG. Following this, the vector $\mathbf{v} = \mathsf{encode}(\mathbf{u}_2 \oplus \mathbf{m})$ becomes uniformly distributed. The final hybrid replaces $\mathbf{e}_2$ by a vector $\mathbf{e}_2^*$, which is sampled according to $\mathcal{D}_{\mathbb{Z}^{m_2},r}$ as per Lemma 3.4. As a result, we obtain instances being identically distributed as the original LWE distribution.

Hybrid$_1$**.** In the first hybrid, in each A-LWE sample we replace the value $\mathsf{seed} = \mathbf{C}\mathbf{e}_1 \bmod q$ by a uniformly sampled value $\mathbf{u}_1 \in \mathbb{Z}_q^t$. This mainly follows from the Leftover Hash Lemma for $t \leq (d - 2\lambda)/\log q$ with $d = H_\infty(\mathbf{e}_1)$ and the fact that $(\mathbf{C}\mathbf{e}_1, \mathbf{A}_1^\top \mathbf{s} + \mathbf{e}_1)$ is indistinguishable from uniform assuming the hardness of $\mathsf{decision\ LWE}_{n-t,m,\alpha q}$ as per Theorem 3.13. We note here, that the adversary never gets to see seed. Due to the high entropy of the seed a distinguisher will guess the correct seed only with negligible probability or at most $2^{-\lambda}$. Also by the same argument, the same seed will not be sampled except with negligible probability due to the high min-entropy of $\mathbf{e}_1$.

Hybrid$_2$**.** In the second hybrid, we replace the output $\mathsf{PRNG}(\mathbf{u}_1)$ of the PRNG by a uniform random string $\mathbf{u}_2 \in \{0,1\}^{n' \cdot \log p}$. As a result, $\mathbf{v} = \mathsf{encode}(\mathbf{u}_2 \oplus \mathbf{m})$ becomes uniformly distributed as well. A potential (polynomial-time) adversary notices the difference between the uniform distribution and the output of the PRNG only if he queries the PRNG with the correct seed $\mathbf{u}_1$ or breaks the underlying hard computational problem. But in both cases, the distinguisher succeeds only with negligible probability. Hence, this also holds, if many samples are given to the distinguisher.

We comment on a distinguisher which queries the PRNG at a certain point on $(s, \mathbf{e}_1)$ below in the proof, and assume for now, that no such distinguisher exists.

Hybrid$_3$**.** In the last hybrid, the error term $\mathbf{e}_2$ is replaced by $\mathbf{e}_2^*$ which is sampled according to $\mathcal{D}_{\mathbb{Z}^{m_2},r}$. This follows from Lemma 3.4 stating that $\mathcal{D}_{\Lambda_{\mathbf{v}}^\perp(\mathbf{B}),\alpha q}$ is computationally close to $\mathcal{D}_{\mathbb{Z}^{m_2},r}$ for a random vector $\mathbf{v} = \mathsf{encode}(\mathbf{u}_2 \oplus \mathbf{m})$, where $\mathbf{u}_2$ is computationally indistinguishable from uniform and $\alpha q \geq \eta_\epsilon(\Lambda^\perp(\mathbf{B}))$. Again, a PPT distinguisher notices a difference between both distributions only if he can distinguish $\mathbf{u}_2$ from the uniform distribution, hence solving a hard computational problem.

We stress that A-LWE samples from **Hybrid**$_3$ are indistinguishable from LWE samples. The only difference between A-LWE and LWE samples is the way the error term was constructed, which we proved via the hybrids to be identically distributed. Hence, a distinguisher can only invoke the PRNG with the correct seed either by guessing $\mathbf{Ce}_1 \bmod q$, which is $2^{-\lambda}$-hard by the Leftover Hash Lemma, or breaking A-LWE samples which is via the relation $L^{\mathsf{A\text{-}LWE}}_{n,m_1,m_2,\alpha,q}(\mathbf{m}) \approx_c L^{\mathsf{LWE}_{\ell(t)}}_{n,m,\alpha q}$ equivalent to breaking LWE samples or solving $\mathsf{search\ LWE}_{n-t,m,\alpha,q}$ and $\mathsf{decision\ LWE}_{n-t,m,\alpha q}$, respectively. By assumption such a distinguisher does not exist.

We conclude that the step from the original A-LWE samples to $\mathbf{Hybrid}_1$ will be unnoticeable to a distinguisher if $\mathsf{search\ LWE}_{n-t,m,\alpha,q}$ and $\mathsf{decision\ LWE}_{n-t,m,\alpha q}$ is hard, and both distributions $L^{\mathsf{LWE}}_{n-t,m,\alpha,q}$ and $L^{\mathsf{A\text{-}LWE}}_{n,m_1,m_2,\alpha,q}(\mathbf{m})$ are computationally indistinguishable. $\square$

This proves the decision version of the A-LWE problem. We immediately obtain the following hardness statements, which follow from the decision and search version of LWE.

Theorem 3.16. *Let n, n', m, m_1, m_2, q, p be integers with $m = m_1 + m_2$. Let $\mathsf{PRNG} : \mathbb{Z}_q^t \to \{0,1\}^{n' \cdot \log p}$ be a PRNG taking $\mathsf{seed} = \mathbf{Ce}_1 \bmod q$ as input for a random matrix $\mathbf{C} \in \mathbb{Z}_q^{t \times m_1}$ with $t \leq (d - 2\lambda)/\log q$ and $d = H_\infty(\mathbf{e}_1)$. Let $\mathbf{B} \in \mathbb{Z}_p^{n' \times m_2}$ be a preimage sampleable full-rank matrix and $\alpha q \geq \eta_\epsilon(\Lambda_q^\perp(\mathbf{B}))$ for a real $\epsilon = \mathsf{negl}(n) > 0$. Furthermore, denote by χ_{e_1} the distribution of the error vectors $(\mathbf{e}_1)_i$ involved in each A-LWE sample i. If $\mathbb{H}_\infty(\mathbf{e}_1) > \lambda$, then the following statements hold.*

1. *If* $\mathsf{search\ LWE}_{n-t,m,\alpha q}$ *is hard, then* $\mathsf{search\text{-}s\ A\text{-}LWE}^{\mathcal{S}}_{n,m_1,m_2,\alpha q}$ *is hard.*
2. *If* $\mathsf{decision\ LWE}_{n-t,m,\alpha q}$ *is hard, then* $\mathsf{decision\ A\text{-}LWE}^{\mathcal{S}}_{n,m_1,m_2,\alpha q}$ *is hard.*
3. *If* $\mathsf{decision\ LWE}_{n-t,m,\alpha q}$ *is hard, then* $\mathsf{search\text{-}m\ A\text{-}LWE}^{\mathcal{S}}_{n,m_1,m_2,\alpha q}$ *is hard.*

The proof for the second statement follows from 3.15. As for the remaining statements we refer to the proof of Theorem 3.9, which proves the validity of the first and last statements using the same argumentation line. We note here, that the seed is always kept hidden and hence believe that the hardness of A-LWE is even stronger based on $\mathsf{search\ LWE}_{n,m,\alpha,q}$ and $\mathsf{decision\ LWE}_{n,m,\alpha q}$.

4. Building Lattice-based Encryption Schemes from A-LWE

In this chapter, we propose an alternative way of encrypting data, which is even perfectly combinable with the traditional one-time pad approach. In particular, we show how to build lattice-based encryption schemes on top of the A-LWE assumption introduced in Chapter 3 by giving direct constructions that are equipped with various security properties and functionalities. In fact, based on the A-LWE assumption we can immediately derive a generic encryption scheme, where ciphertexts are represented by plain A-LWE samples. Besides of its evident security properties, that can directly be deduced from A-LWE, our construction benefits from encrypting more message bits per ciphertext and a faster encryption engine through a conceptually easier instantiation as compared to previous proposals. Furthermore, we give a detailed description of how to achieve CCA-security and publicly-detectable replayable CCA (pd-RCCA) security [CKN03], a slightly relaxed version of CCA2, but strictly stronger than CCA1. In fact, we propose the first lattice-based RCCA-secure encryption scheme. Due to the versatility of the error term, this functionality does not involve ciphertext expansion. As a third application, it is possible to replace parts of the error term by signatures that are generated according to the best known and widely used lattice-based signature schemes. Specifically, we focus on the GPV signature scheme [GPV08] in combination with the trapdoor construction [MP12] and the practical signature schemes presented in [DDLL13, Lyu12], and thus realize an asymmetric authenticated encryption scheme. As a nice byproduct, one can immediately apply the proposed concepts to the CCA-secure construction given in [MP12]. This allows us to increase the message throughput per ciphertext, while enjoying RCCA-security at almost no cost. Noteworthy, all the proposed concepts are also applicable to specific constructions such as the somewhat homomorphic symmetric key encryption scheme due to [BV11b], which does not rely on the trapdoor construction from [MP12]. This chapter to the publications [EDB15, EB15a, EB15b], where the author of this thesis was the primary investigator and author of the publications.

Overview of Contributions

Maximum Data Size. We introduce several techniques to maximize the data throughput per error vector. In particular, we show that for the choice of $\alpha q = p \cdot \sqrt{\ln(2(1+1/\epsilon))/\pi}$ the maximum data size, that can be embedded in an error term $\mathbf{e} \in \mathbb{Z}^{m_2}$ by use of a preimage sampleable full-rank matrix with

$\mathbf{e} \leftarrow \mathcal{D}_{\Lambda^{\perp}_{\mathbf{v}}(\mathbf{B}),\alpha q} = \mathcal{D}_{\mathbb{Z}^m,\alpha q}$, is bounded to $m_2 \log p$ bits for a random syndrome $\mathbf{v} = \mathsf{encode}(\mathbf{r} \oplus \mathbf{m}) \in \mathbb{Z}_p^{m_2}$ and message $\mathbf{m} \in \{0,1\}^{m_2 \log p}$. That is, we can pack the data of $\log p$ bits into an entry of the error term. It turns out that specifying $\mathbf{B} = \mathbf{I} \in \mathbb{Z}_p^{m_2 \times m_2}$ allows for fast and efficient operations such as preimage sampling and message recovery beside of realizing the maximum possible bit size for full-rank matrices. In fact, preimage sampling is performed by selecting a random vector $\mathbf{r}$ and sampling $\mathbf{e} \leftarrow \mathcal{D}_{\Lambda^{\perp}_{\mathbf{v}}(\mathbf{I}),\alpha q} = \mathcal{D}_{\mathbf{v}+p\mathbb{Z}^{m_2},\alpha q}$. The message is subsequently retrieved back via $\mathsf{decode}(\mathbf{e} \bmod p) \oplus \mathbf{r}$.

Generic Encryption Scheme. Based on the A-LWE hardness, we present a novel and generic encryption scheme, where ciphertexts are embodied by plain A-LWE samples. One merely employs an arbitrary suitable trapdoor construction for the function $g_{\mathbf{A}}(\mathbf{s},\mathbf{e}) = \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top$ that allows for error term recovery. Hence, the efficiency of encryption and decryption greatly depends on the quality of the trapdoor and the inversion algorithm. The currently most efficient candidate function is known from Micciancio and Peikert [MP12]. Note that while some encryption schemes like [MP12, SSTX09] utilize such a trapdoor function, the potential of the error term is left unrecognized. To the best of our knowledge, we provide the first lattice-based encryption scheme exploiting the error term as an (additional) data container in addition to its necessity for security.

In fact, the bit size of the message is equal to $m_2 \cdot \log p$ (in the $\mathcal{RO}$ model $m_2 = m$) resulting in a small message expansion factor, which is lower than in all of the existing schemes. Due to this relationship there is an incentive to increase the parameter m by appending l additional uniform random columns to the public key in order to efficiently encrypt large amounts of data involving less computations per ciphertext as compared to lower dimensions. We considered this case and can even show that decryption is essentially as fast as in lower dimensions. In particular, we provide an enhanced encryption scheme for high data load HDL mode, where parts of the ciphertext and thus the error term are ignored when inverting the underlying A-LWE instance. That is, one extends any initial public key $\mathbf{A}'' \in \mathbb{Z}_q^{n \times m}$ with trapdoor $\mathbf{T}$ to $\mathbf{A} = [\ \mathbf{A}' \mid \mathbf{A}''\ ] \in \mathbb{Z}^{n \times (l+m)}$ with trapdoor $\begin{bmatrix}\mathbf{0}\\ \mathbf{T}\end{bmatrix}$ and a uniform random matrix $\mathbf{A}' \in \mathbb{Z}^{n \times l}$. When inverting a ciphertext $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2) \in \mathbb{Z}^{l+m}$ – that is, an A-LWE instance – only the lower part of the ciphertext $\mathbf{c}_2$ is required to recover $\mathbf{s}$ and $\mathbf{e}$. This idea does not seem to carry over to the construction of [MP12]. In fact, their message size is fixed to nk bits for $k = \lceil \log q \rceil$ and extending the public key as above cannot be applied to their scheme. Based on the HDL mode we give some further significant optimizations that allow for an even larger message throughput realizing message expansion factors close to 1. This can be attributed to the fact that the error term related to $\mathbf{A}'$ can be as large as possible because it plays no role when inverting A-LWE instances. In particular, we select a second parameter βq that can be as large as $\beta q = d \cdot \sqrt{\ln(2(1+1/\epsilon))/\pi}$ with $d = \lfloor \frac{q}{2 \cdot 4.7^2} \rfloor$ such that the error term does not wrap around. This allows for further $l \log d$ message bits to be encrypted in the error vector related to $\mathbf{c}_1$. Going further, we omit the discrete Gaussian step and select the error term uniformly at random from $\mathbb{Z}_q^l$ such that we can exploit the full bandwidth encrypting $l \cdot \log q$ message bits in $\mathbf{c}_1$. The HDL mode is a generic approach that is applicable to all schemes being introduced in this chapter.

CCA-Secure Encryption. Based on the A-LWE hardness, we build a conceptually new and very simple CCA1-secure encryption scheme. In previous lattice-based encryption schemes such as [ABB10a, LP11, MP12, PW08], ciphertexts are computed in an one-time pad manner by adding the message to a random vector coming from the LWE distribution. Thus, an adversary succeeds in the respective security game, if he is able to distinguish LWE samples from random ones with non-negligible advantage. Our scheme, however, moves apart from this approach and focuses on the error term recovery of A-LWE samples and subsequently decoding the error term. By this means, the ciphertext represents an A-LWE instance in its purest form. This implies a direct security reduction of the scheme to A-LWE. Employing the framework proposed in [MP12], we construct a random public key $\mathbf{A}$ that is endowed with a trapdoor. In conjunction with the corresponding inversion algorithm, which is only applicable for moduli of the form $q = 2^k$ (for arbitrary moduli see construction in Chapter 5), we can efficiently recover the secret and the error term from the ciphertext $\mathbf{c}^\top = \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top$ with $\mathbf{e} \leftarrow_R \mathcal{D}_{\Lambda_{\mathbf{v}}^\perp(\mathbf{I}),\alpha q}$ for $\mathbf{v} = \mathsf{encode}(F(\mathbf{s}, \mathbf{e}_1) \oplus \mathbf{m}) \in \mathbb{Z}_p^{m_2}$ and a random function F instantiated either in the random oracle or standard model.[1] Due to $\alpha q = p \cdot \sqrt{\ln(2(1 + 1/\epsilon))/\pi} \geq \eta_\epsilon(\Lambda_q^\perp(\mathbf{I}))$, we even do not impose any further restrictions to the parameters. Such a construction is almost optimal, since we do not initiate any further transformations. As already pointed out earlier in this section the HDL mode is directly applicable to the CCA1-secure scheme and thus allows for the same benefits and properties as for the generic encryption scheme. That is, we can exploit the full bandwidth of the corresponding part of the error term, when extending the public with a uniform random matrix $\mathbf{A}'$.

We further show that message embedding can enhance the CCA-secure scheme proposed in [MP12] yielding a decrease of the message expansion factor. Put it differently, with message embedding one could choose smaller parameters for the scheme in [MP12] when encrypting the same message size. In terms of security the original proof in [MP12] gets through without any major modifications. Table 4.1 gives an overview of parameters and the corresponding sizes for various lattice-based encryption schemes where we, for simplicity, fix the ciphertext size. Fixing n, however, leads to different values and complicates a fair comparison. Note here that $c \in \mathbb{Q}_{\geq 2}$, and consequently the message throughput in our scheme is by a factor of $c \log(\alpha q/4.7)$ larger than the one from [MP12]. Here αq denotes the parameter of the discrete Gaussian distribution used to sample the error vector. This even allows to embed $c \log(\alpha q) nk$ bits of a signature into the error term, which is due to identical distributions of signatures and error vectors. Table 4.1 does not include sizes for our scheme operating in the HDL mode. In this case, the message size

[1]We show in the $\mathcal{RO}$ model that if matrix $\mathbf{A}$ is fixed and each ciphertext involves a fresh secret vector $\mathbf{s}$, the entropy of $\mathbf{s}$ is sufficient in order to sample the entire error term from $\mathcal{D}_{\Lambda_{\mathbf{v}}^\perp(\mathbf{I}),\alpha q}$.

$m = c \cdot nk$ $k = \log q$	CCA1 **[MP12]**	CCA1 **This work**	CCA1 **This work** + **[MP12]**	CPA **[LP11]**
Ciphertext size	$m \cdot k$	$m \cdot k$	$m \cdot k$	$m \cdot k$
Signature size	nk	$c \log(\alpha q) nk$	$(c \log(\alpha q) + 1) nk$	$cnk - n$
Message size	nk	$c \cdot nk \log(\alpha q/4.7)$	$(c \log(\alpha q/4.7) + 1) nk$	$cnk - n$
Message Exp.	$c \cdot k$	$\frac{k}{\log(\alpha q/4.7)}$	$\frac{k}{\log(\alpha q/4.7) + 1/c}$	$k + \frac{k}{ck-1}$
Error rate α	$\tilde{O}(1/n)$	$\tilde{O}(1/n)$	$\tilde{O}(1/n)$	$\tilde{O}(1/n)$
public key size	$n \cdot m$	$n \cdot m$	$n \cdot m$	$n \cdot (m - n)$

Table 4.1.: Parameters

raises, for instance, to $c_1 nk \log q + c_2 nk \log(\alpha q/4.7)$, where $c_1 + c_2 = c$ and the error term part related to $\mathbf{A}'$ is sampled uniformly at random containing $c_1 nk \log q$ data bits. We mainly focus on the most efficient encryption schemes including the CPA-secure encryption scheme from [LP11]. Table 4.1 does not include the less efficient schemes from [PW08, Pei09, ABB10a], which are characterized by large public keys or small LWE error-rates beside of high message expansion factors. For instance, in [PW08] the LWE error rate $\alpha = \tilde{O}(1/n^4)$ is quite small (yielding to an easier LWE instance) with public keys of size $\tilde{O}(n^2)$ bits. In [Pei09], Peikert improved the LWE error rate to $\alpha = \tilde{O}(1/n)$ but with the cost of an increased public key of size $\tilde{O}(n^3)$. The CCA-secure encryption scheme [ABB10a] provides a trade-off of the previous proposals with an LWE error rate of $\tilde{O}(1/n^2)$ and public key size of $\tilde{O}(n^2)$ bits. When comparing our approach with the CPA-secure encryption scheme from Lindner and Peikert [LP11], we attest an improvement factor of $O(\log(\alpha q))$.

Replayable Chosen-Ciphertext Secure Encryption. The notion of replayable CCA-security, which constitutes a relaxed version of CCA2-security, was introduced by Canetti et al. [CKN03] and addresses the ability of an adversary to replay ciphertexts that decrypt to the same message. An RCCA-secure encryption scheme detects modifications carried out on the ciphertext that alter the message. Valid encryptions of the same ciphertexts, however, are allowed. Canetti et al. have shown that RCCA is sufficient for most practical applications. There exists a series of RCCA-secure encryption schemes [Gro04, LV08, PR07, PSNT06, XF07]. However, to our knowledge, we are the first realizing a lattice-based RCCA-secure encryption scheme, and hence relying on the worst-case hardness of lattice problems. We show that RCCA security comes essentially through our message embedding technique with only minor modifications. Our construction resembles GPV signatures generated for the public matrix $\mathbf{I} \in \mathbb{Z}_p^{m_2 \times m_2}$. Just as for standard GPV signatures, it is required to hash all sensible (random) variables such as the tag u, the secret $\mathbf{s}$ and the lower part of the error term $\mathbf{e}_2$ involving the message to $\mathbf{v} = H(u, \mathbf{s}, \mathbf{e}_2)$ using a random oracle H. Subsequently, we sample a preimage $\mathbf{e}_1 \leftarrow \mathcal{D}_{\Lambda_{\mathbf{v}}^{\perp}(\mathbf{I})}$ that serves as the upper-part of the error term. Due to the injectivity of the trapdoor function, altering the ciphertext leads to different values for the error term or the secret such that the decryption routine outputs a failure. But modifications caused to the upper part of the error term do not result in a failure as long as short vectors from $\Lambda_p^\perp(\mathbf{I}) = p\mathbb{Z}^{m_1}$ are added.

This obviously implies a publicly-detectable RCCA-secure encryption scheme (pd-RCCA), an even stronger security guarantee than plain RCCA. In fact, we have the relation CCA2 $\Rightarrow$ pd-RCCA $\Rightarrow$ secretly-detectable RCCA $\Rightarrow$ RCCA [CKN03]. Security in the pd-RCCA model implies that a public party can check whether a modified ciphertext decrypts to the same message.

When it comes to CCA2 security, there exist many works on generic constructions [DDN00, CHJ$^+$02, HLM03, BCHK07] that ensure CCA2-security. For instance, one can use strongly unforgeable one-time signature schemes [DDN00], commitment schemes or message authentication codes (MAC) in order to transform a CPA-secure scheme into a CCA2-secure one. However, these generic constructions typically involve high complexity and overhead resulting in a less efficient encryption scheme. Our approach works differently as it uses the error term in order to provide this feature. Once having RCCA-security one can efficiently convert the scheme into a CCA2-secure encryption scheme using generic solutions as provided in [CKN03] or our individual approach at the expense of some small overhead.

Signature Embedding. There exist various approaches to provide message authentication of encrypted data. Many of them are generic and thus coupled to overhead and loss of efficiency. For instance, one can use message authentication codes (MAC) or digital signatures that are appended to the ciphertext. In our work we aim at providing this feature without suffering from the drawbacks of generic solutions through a thorough analysis of our encryption scheme.

Our goal is to replace parts of the error vector such as $\mathbf{e}_1$ completely by a lattice-based signature rather than appending it to the ciphertext or including it as a part of the message. This allows us to optimally exploit the full bandwidth of $\mathbf{e}_1$ due to some nice properties lattice-based signature schemes offer. One of the features is to let signatures be distributed following the discrete Gaussian distribution. For the underlying signature scheme itself, such a strategy has many advantages over other choices as it allows to decouple the distribution of the signature from the secret key, while sampling short signatures with higher probability. There exist many lattice-based proposals that have similar properties and perform very well in practice [DDLL13, Lyu12, MP12].

Our construction inherently relies on the capability to recover the error term from an A-LWE instance. As a result, we provide an authentication mechanism for encrypted data, since it is by construction possible to retrieve back an arbitrary discrete Gaussian vector with support $\mathbb{Z}^m$, hence also a signature, that was plugged into the error term. Therefore, we can embed signatures of size $m \cdot \log(\alpha q)$ bits into the error vector, which is far more than with the standard encryption schemes that are restricted to the message size (see Table 4.1). By αq we denote the parameter of the discrete Gaussian vector used to sample the error term. In fact, our proposal allows for a flexible selection of parameters, because we do not impose any new constraints. However, the parameters of the signature scheme should not be too large in order to correctly invert the underlying A-LWE instances.

Remarkably, when using the encryption scheme for high data load with an extended public key $\mathbf{A} = [\ \mathbf{A}' \mid \mathbf{A}''\ ] \in \mathbb{Z}^{n \times (l+m)}$ the upper part of the error term is ignored when decrypting the ciphertext. This allows us to select the parameters in such a way that A-LWE (and LWE) is hard for arbitrarily chosen parameters of the signature scheme. Therefore, one can employ the upper part of the error term for signatures. The resulting scheme has a CCA2-like behavior, where changes induced to the ciphertext are detected by the receiver. These ideas immediately help to improve the construction provided in [MP12]. In particular, we can apply the proposed techniques to the error term without changing the other ingredients. More specifically, we still build the ciphertext in an one-time pad manner, while simultaneously endowing the error vector with additional messages. The proof of security will subsequently be based on A-LWE rather than plain LWE.

Embedding Auxiliary Data in Homomorphic Encryption. As already noticed, we improve the CCA1-secure encryption scheme from [MP12], if we apply the proposed concepts from above to the error term. As a result, we have the first message being encrypted following the one-time pad approach and a second message injected into the error term. However, this encryption scheme heavily relies on a trapdoor construction. But we stress that it is also possible to improve other more specific constructions that do not require trapdoors as such. For instance, if we consider the somewhat homomorphic encryption scheme due to Brakerski and Vaikuntanathan [BV11b], we can apply essentially the same ideas without any major modifications. Indeed, it is a symmetric key encryption scheme, where a ciphertext $(\mathbf{c}_1 = \mathbf{a}, \mathbf{c}_2 = \mathbf{b} + \mathbf{m})$ is derived by adding a ring-LWE sample $\mathbf{b} = \mathbf{a}\mathbf{s} + t\mathbf{e} \in \mathcal{R}_q = \mathbb{Z}_q[X] / \langle f(X) \rangle$ to an arbitrary message $\mathbf{m} \in \mathcal{R}_t$ for t coprime to q and freshly sampled $\mathbf{c}_1 = \mathbf{a} \in \mathcal{R}_q$ with an error vector $\mathbf{e} \in \mathcal{R}$ sampled according to the discrete Gaussian distribution. The secret key is given by the secret ring element $\mathbf{s} \in \mathcal{R}_q$. After decrypting the ciphertext, we get full access to the error term via $\mathbf{e} = t^{-1}(\mathbf{c}_2 - \mathbf{c}_1\mathbf{s} - \mathbf{m})$. A quick view to this construction reveals, that the error term can be recovered very efficiently. Clearly, this has a positive impact on the performance of the different concepts, when applied to the error term.

4.1. Maximum Data Size

For the sake of generality, we used in all our statements in Chapter 3 an abstract matrix $\mathbf{B} \in \mathbb{Z}_p^{n' \times m}$ for integers p, n', and m. This is used to embed a message into the error term via $\mathbf{e}_2 \leftarrow_R \mathcal{D}_{\Lambda^{\perp}_{\mathbf{v}}(\mathbf{B}), \alpha q}$, where $\mathbf{v} = \mathsf{encode}(F(\mathsf{seed}) \oplus \mathbf{m}) \in \mathbb{Z}_p^{n'}$ is uniform random. However, we can specify concrete matrices that maximize the amount of information per entry with respect to the bound given in Lemma 4.1.

In the following section we propose several techniques in order to enhance the message throughput per discrete Gaussian vector. These techniques could also be applied to the error vector involved in the A-LWE distribution. In other words, we aim at choosing an appropriate preimage sampleable full-rank matrix $\mathbf{B} \in \mathbb{Z}_p^{n' \times m}$ such that $n' \cdot \log p$ is maximized. For now, we will focus on how to apply this technique to the different encryption schemes and omit the $\mathbf{e}_1$ term when invoking the random oracle, since the secret $\mathbf{s} \in \mathbb{Z}_q^n$ is always resampled in encryption schemes and hence provides enough entropy for each fresh encryption query. Our first approach is based on a method used to construct homomorphic signatures in [BF11]. The second approach is very simple avoiding such complex procedures in order to allow for the same message throughput.

4.1.1. Intersection Method

The intersection method as proposed in [BF11] considers two m-dimensional integer lattices Λ_1 and Λ_2 such that $\Lambda_1 + \Lambda_2 = \mathbb{Z}^m$, where addition is defined to be element wise. Therefore, let $\mathbf{m}_1$ and $\mathbf{m}_2$ be two messages, where $\mathbf{m}_1$ and $\mathbf{m}_2$ define a coset of Λ_1 and Λ_2, respectively in $\mathbb{Z}^m$. As a result, the vector $(\mathbf{m}_1, \mathbf{m}_2)$ defines a unique coset of the intersection set $\Lambda_1 \cap \Lambda_2$ in $\mathbb{Z}^m$. By the Chinese Remainder theorem one can compute a short vector $\mathbf{t}$ such that $\mathbf{t} = \mathbf{m}_1 \bmod \Lambda_1$ and $\mathbf{t} = \mathbf{m}_2 \bmod \Lambda_2$ using a short basis for $\Lambda_1 \cap \Lambda_2$. In fact, it is easy to compute any vector $\mathbf{t}$ that satisfies the congruence relations. Subsequently, by invoking a preimage sampler one obtains a short vector from $\Lambda_1 \cap \Lambda_2 + \mathbf{t}$. For instance, one can efficiently instantiate the scheme when choosing $\Lambda_1 = p\mathbb{Z}^m$ and $\Lambda_2 = \Lambda_q^{\perp}(\mathbf{A})$ for a matrix $\mathbf{A} \in \mathbb{Z}_q^{l \times m}$ with a short basis $\mathbf{T}$ and p coprime to q. Doing this, the message spaces are given by $\mathbf{m}_1 \in \mathbb{Z}^m/\Lambda_1 \cong \mathbb{Z}_p^m$ and $\mathbf{m}_2 \in \mathbb{Z}^m/\Lambda_2 \cong \mathbb{Z}_q^l$, where the isomorphisms are given by $\mathbf{x} \mapsto (\mathbf{x} \bmod p)$ and $\mathbf{x} \mapsto (\mathbf{A} \cdot \mathbf{x} \bmod p)$. Due to the simple choice of Λ_1, we obtain a short basis $\mathbf{S} = p \cdot \mathbf{T}$ for $\Lambda_1 \cap \Lambda_2 = p \cdot \Lambda_2$, where $\eta_\epsilon(\Lambda_1 \cap \Lambda_2) \leq p \cdot \eta_\epsilon(\Lambda_2)$. So, if $\mathbf{A}$ corresponds to $\mathbf{G} \in \mathbb{Z}_q^{m/k \times m}$ for $k = \log q$, we have $\eta_\epsilon(\Lambda_1 \cap \Lambda_2) \leq p \cdot 2 \cdot \omega(\sqrt{\log n})$. In our schemes, however, we have to sample a short vector $\mathbf{e}$ from $(F(\mathbf{r}) \oplus \mathbf{t}) + \Lambda_1 \cap \Lambda_2$ with parameter $\alpha q \geq \eta_\epsilon(\Lambda_1 \cap \Lambda_2)$, where $\mathbf{t}$ is computed as above and the (simplified) description $F : \{0,1\}^* \to \mathbb{Z}_q^m$ defines a random function taking a random string $\mathbf{r} \in \{0,1\}^*$ with sufficient entropy as input. The error vector is then given by $\mathbf{e} \leftarrow D_{\mathbf{b}+\Lambda_1 \cap \Lambda_2, \alpha q}$ with $\mathbf{b} = F(\mathbf{r}) \oplus \mathbf{t}$. Due to $\eta_\epsilon(\Lambda_1 \cap \Lambda_2) \leq p \cdot 2 \cdot \omega(\sqrt{\log n})$ (e.g., $\alpha q = p \cdot 2 \cdot \omega(\sqrt{\log n})$), the error vector is indistinguishable from $D_{\mathbb{Z}^m, \alpha q}$ following Lemma 3.2 and Lemma 3.4. This technique allows to embed $m \log p + m$ bits of messages into the error term.

4.1.2. Lattices of the Form $\mathbf{p} \cdot \mathbb{Z}^{\mathbf{m}}$

One realizes that for a given parameter αq for the distribution of the error vector one can be much more efficient, if one considers only the lattice $\Lambda_p^{\perp}(\mathbf{I}) = p\mathbb{Z}^m$. In this case, the message space is simply defined by the set $\mathcal{M} = \mathbb{Z}^m/\Lambda_p^{\perp}(\mathbf{I}) \cong \mathbb{Z}_p^m$. When comparing with the previous approach, for instance, it is only required to increase p by a factor of 2 in order to obtain the same message throughput $m \log 2p = m \cdot (\log p + 1)$. Furthermore the decoding and encoding phase is much faster, since encoding requires only to sample $\mathbf{e} \leftarrow D_{\mathbf{b}+p\mathbb{Z}^m,\alpha q}$ for $\mathbf{b} = F(\mathbf{r}) \oplus \mathbf{m}$ using fast discrete Gaussian samplers such as the Knuth-Yao algorithm or the more efficient FastCDT sampler that we present in Chapter 5. Decoding is performed via $F(\mathbf{r}) \oplus (\mathbf{e} \bmod p)$. Optimizing the message throughput requires to increase p such that $\eta_\epsilon(\Lambda) \leq p \cdot \mathsf{const} \leq \alpha q$ still holds for $\mathsf{const} = \sqrt{\ln(2(1+1/\epsilon))/\pi}$. Doing this, one can embed approximately $m \cdot \mathsf{const}$ bits of data, which almost coincides with the min-entropy of a discrete Gaussian with parameter αq, since $\mathsf{const} \approx 4.7$. Therefore, it is most effective to choose a parameter such that $\alpha q = p \cdot \mathsf{const}$ with $p = 2^i$ for some $i > 0$ in order to embed i bits of data into the error term.

Lemma 4.1 (**[GPV08], Theorem 3.1**). *Let $\Lambda \subset \mathbb{R}^n$ be a lattice with basis $\mathbf{S}$, and let $\epsilon > 0$. We have $\eta_\epsilon(\Lambda) \leq \| \tilde{\mathbf{S}} \| \cdot \sqrt{\ln\left(2n\left(1+\frac{1}{\epsilon}\right)\right)/\pi}$. In particular, for any function $\omega(\sqrt{\log n})$, there is a negligible $\epsilon(n)$ for which $\eta_\epsilon(\Lambda) \leq \| \tilde{\mathbf{S}} \| \cdot \omega(\sqrt{\log n})$.*

In fact, based on the bound given in Lemma 4.1, for any $\alpha q > 0$ and $\mathbf{e}_2 \leftarrow_R \mathcal{D}_{\mathbb{Z}^m,\alpha q}$ the maximum number of bits that can be embedded into a component of the error term is bounded by $\log(\alpha q/\omega(\sqrt{\log n}))$. This means that $p' = \lfloor \alpha q/\omega(\sqrt{\log n}) \rfloor$ is the largest integer such that $\mathbf{e}_2 \bmod p'$ is guaranteed with overwhelming probability to be uniform random (see Lemma 4.2). Hence, we can choose $\mathbf{B} = \mathbf{I} \in \mathbb{Z}^{m\times m}$ with $\alpha q = p \cdot \mathsf{const}$ for $p = 2^k$ allowing for k-bits of information. The data is recovered via the efficient operation $\mathbf{v} = \mathbf{e}_2 \bmod q$. For the sake of these arguments, we will use $\mathbf{B} = \mathbf{I}$ throughout this thesis.

Lemma 4.2 (**Optimal Message Bound**). *Let $r = p \cdot \omega(\sqrt{\log m})$ for integers $p, m > 0$. Furthermore, let a discrete Gaussian over $\mathbb{Z}^m$ be sampled by selecting $\mathbf{v} \in \mathbb{Z}_{p'}^{n'}$ uniformly at random and then sampling $\mathcal{D}_{\mathbf{v}+\Lambda_{p'}^{\perp}(\mathbf{B}),r}$ for a preimage sampleable matrix $\mathbf{B} \in \mathbb{Z}_{p'}^{n'\times m}$ with $p', n' > 0$. An optimal bound for the maximum number of bits, that can be injected into a discrete Gaussian (or equivalently the size of $\mathbf{v}$), is given by $m \cdot \log p$ bits.*

Proof. The proof is essentially based on the bound given by Lemma 4.1. First, in order to sample a discrete Gaussian vector over $\mathbb{Z}^m$, the condition $r \geq \eta_\epsilon(\Lambda_{p'}^{\perp}(\mathbf{B}))$ has to be satisfied for a negligible ϵ following Lemma 3.2 and Lemma 3.4. Based on Lemma 4.1 we have $p \geq \| \tilde{\mathbf{S}} \|$, where $\mathbf{S}$ denotes a basis of $\mathbf{B}$ with $\mathbf{B} \cdot \mathbf{S} \equiv \mathbf{0} \bmod q$ and $\tilde{\mathbf{S}}$ its orthogonolization. We note, that it suffices to consider $m = 1$, since each component of a discrete Gaussian vector over $\mathbb{Z}^m$ is sampled independently containing the same amount of information and randomness. Following this, $n' = 1$ and subsequently $\tilde{\mathbf{S}} = \mathbf{S} \leq p$. Hence, for $\mathbf{S} = p = p'$ we obtain the maximum bit size amounting to $\log p$ bits for v such that $\mathcal{D}_{v+\Lambda_p^{\perp}(\mathbf{B}),r} = \mathcal{D}_{v+p\mathbb{Z},r}$ is identically distributed to $\mathcal{D}_{\mathbb{Z},r}$. For $m > 1$ one takes, for instance, $\tilde{\mathbf{S}} = \mathbf{S} = p \cdot \mathbf{I}_m$ resulting in $m \cdot \log p$ bits for $\mathbf{v} \in \mathbb{Z}_p^m$. □

4.1.3. Uniform Error

For uniformly distributed errors one can directly employ the output of the random function $F(\cdot)$ as the error term. More specifically, suppose $\mathbf{e} \in ([-p,p] \cap \mathbb{Z})^m$, then let $F(\cdot) : \{0,1\}^* \to ([-p,p] \cap \mathbb{Z})^m$ be a random function such that $\mathbf{e} \leftarrow \mathsf{encode}(F(\mathbf{r}) \oplus \mathbf{m})$ for $\mathbf{m} \in \{0,1\}^{m \log_2(2p)}$. As a result, one can use the full bandwidth of the error term and inject $m \log_2(2p)$ message bits.

4.2. Our Generic Construction

Originally, in almost all previous encryption schemes ciphertexts are built in a one-time pad manner by adding the message to a random-looking vector coming from an LWE instance. By our modifications, we omit the way of encoding messages and the restrictions made to the parameters. Our aim is to let the ciphertexts resemble an ordinary A-LWE instance such that the hardness of the scheme can be directly reduced to the plain A-LWE problem. Indeed, the error term hides the message while following the required distribution. This allows for more flexibility, efficiency and larger messages per ciphertext at no cost. Even more, this greatly simplifies the security proof. As we show later, we can even lift up the security to publicly-detectable RCCA (pd-RCCA) with a simple trick ensuring non-malleability of ciphertexts. When applying these functionalities to the error term in the CCA1-secure scheme due to [MP12], the message throughput is about $O(m \cdot \log \alpha q)$ while simultaneously providing pd-RCCA security instead of CCA1as before. In addition to that, we give an intuition of how to get a CCA2-secure encryption scheme involving only minor modifications.

In what follows we provide a very basic and simple construction of an A-LWE-based encryption scheme, which will later on be the target of several optimizations. In particular, we present simple and efficient strategies to enhance the message throughput. Due to our new feature of embedding messages in the error term, we can employ any trapdoor function that allows for error term recovery.

Therefore, let $\mathsf{TDF} = (\mathsf{KeyGen}, g, g^{-1})$ be a trapdoor function with $g_{\mathbf{A}}(\mathbf{x}, \mathbf{y}) := \mathbf{x}^\top \mathbf{A} + \mathbf{y}^\top \in \mathbb{Z}^m$. The algorithm KeyGen outputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, that is close to uniform, with an associated trapdoor $\mathbf{T}$ used to invert $g_{\mathbf{A}}$. The trapdoor function satisfies $g_{\mathbf{A}}^{-1}(\mathbf{T}, \mathbf{c}) = (\mathbf{x}, \mathbf{y})$ with $\mathbf{c} = g_{\mathbf{A}}(\mathbf{x}, \mathbf{y})$ for arbitrary $\mathbf{x} \in \mathbb{Z}_q^n$ and properly chosen $\mathbf{y} \in \mathbb{Z}^m$. We note that the random function F is instantiated either by a cryptographic hash function modeled as random oracle or a PRNG in combination with a deterministic function following the standard model variant (see Chapter 3). The construction of the scheme is depicted below, where $\alpha q = p \cdot \sqrt{\ln(2(1 + 1/\epsilon))/\pi}$ is chosen to allow for data of size $\log p$ bits per entry (see Section 4.1.2).

The generic construction is mainly based on the capability of the scheme to recover the error vector. Thus, the underlying trapdoor construction acts as a black box granting full access to the secret $\mathbf{s}$ and the error term $\mathbf{e}$, when applying the secret

Basic Encryption Scheme

$\mathsf{KeyGen}(1^n)$: Generate a public matrix $\mathbf{C} \in \mathbb{Z}_q^{t \times m_1}$, public key $\mathsf{pk} := \mathbf{A} \in \mathbb{Z}_q^{n \times m}$ with trapdoor $\mathsf{sk} := \mathbf{T}$ where $(\mathbf{A}, \mathbf{T}) \leftarrow \mathsf{TDF.KeyGen}(1^n)$.

$\mathsf{Enc}(\mathsf{pk}, \mathbf{m} \in \{0,1\}^{m_2 \cdot \log p})$: Sample $\mathbf{s} \leftarrow_R \mathbb{Z}_q^n$ and subsequently compute $\mathbf{v} = \mathsf{encode}(F(\mathbf{s}, \mathbf{e}_1) \oplus \mathbf{m}) \in \mathbb{Z}_p^m$, where $\mathbf{e}_1 \leftarrow_R \mathcal{D}_{\mathbb{Z}^{m_1}, \alpha q}$.

- Random oracle model: $F(\mathbf{s}, \mathbf{e}_1) := H(\mathbf{s})$ with $m_1 = 0$ and $m_2 = m$
- Standard model: $F(\mathbf{s}, \mathbf{e}_1) := \mathsf{PRNG}(\mathbf{C}\mathbf{e}_1 \bmod q)$

Then, sample $\mathbf{e}_2 \leftarrow_R \mathcal{D}_{\mathbf{v} + p\mathbb{Z}^{m_2}, \alpha q}$. The ciphertext is given by $\mathbf{c}^\top = g_{\mathbf{A}}(\mathbf{s}, \mathbf{e})$ for $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2)$.

$\mathsf{Dec}(\mathsf{sk}, \mathbf{c})$: Compute $g_{\mathbf{A}}^{-1}(\mathbf{T}, \mathbf{c}) = (\mathbf{s}, \mathbf{e})$.
Return $\mathbf{m} = \mathsf{decode}(\mathbf{e}_2 \bmod p) \oplus F(\mathbf{s}, \mathbf{e}_1)$.

Figure 4.1.: Basic encryption scheme from A-LWE.

trapdoor to a related A-LWE instance. Once having revealed the error term, the message is recovered via the last step of the scheme involving the random function $F(\cdot)$. Improving the quality of the trapdoor and its inversion algorithm directly impacts the efficiency of the encryption scheme, since decoding of the message from $\mathbf{e}$ is performed very efficiently. We note that in case of a random oracle instantiation of $F(\cdot)$, it is sufficient to invoke H using only the secret vector $\mathbf{s}$ ($m_1 = 0$), which is always resampled for every encryption query. As a result, the whole bandwidth of the error vector $\mathbf{e} \in \mathbb{Z}^m$ can be exploited for data transmission. The security statement below will, therefore, be based on this particular setting.

Theorem 4.3. *Let F be instantiated in the random oracle model. Then, the generic encryption scheme above is CPA-secure assuming the hardness of* decision $\mathsf{A\text{-}LWE}^{\mathcal{RO}}_{n,0,m,\alpha q}$ *for* $\alpha q \geq p \cdot \sqrt{\ln(2(1+1/\epsilon))/\pi} \geq 2\sqrt{n}$.

Proof. Ciphertexts generated according to the generic encryption scheme from above correspond to plain A-LWE samples with $m_1 = 0$, where H is invoked once on the fresh input $\mathbf{s}$. By assumption decision $\mathsf{A\text{-}LWE}^{\mathcal{RO}}_{n,0,m,\alpha q}$ is hard, and consequently, an adversary is not able to distinguish a challenge ciphertext from uniformly chosen samples. □

In the standard model variant, ciphertexts exactly correspond to plain A-LWE samples as proposed in Chapter 3. Hence, the decision version of A-LWE applies.

Theorem 4.4. *Let F be instantiated in the standard model. The generic encryption scheme above is CPA-secure assuming the hardness of* decision $\mathsf{A\text{-}LWE}^{\mathcal{S}}_{n,m_1,m_2,\alpha q}$ *for* $\alpha q \geq p \cdot \sqrt{\ln(2(1+1/\epsilon))/\pi} \geq 2\sqrt{n}$.

4.2.1. High Data Load Encryption (HDL Mode)

In certain application scenarios one wishes to encrypt huge amounts of data. This is interesting within the context of financial transactions, secure backups, and high media traffic via the internet. In this case, a fast encryption and decryption engine is desired. The key idea underlying this goal is to extend the public key by an arbitrary number of random columns in order to ensure a more efficient encryption scheme at essentially the same security level. That is, in the key generation step $\mathsf{KeyGen}(1^n)$ a public key $\mathbf{A}''$ is output with a corresponding trapdoor $\mathbf{T}$. By extending the public key to $\mathbf{A} = [\mathbf{A}' \mid \mathbf{A}'']$ with a uniform random matrix $\mathbf{A}' \in \mathbb{Z}_q^{n \times l}$, the message size increases due to additional samples related to $\mathbf{A}'$. In fact, ciphertexts are generated following the basic encryption scheme

$$\begin{bmatrix} \mathbf{c}' \\ \mathbf{c}'' \end{bmatrix} = \mathbf{A}^\top \mathbf{s} + \begin{bmatrix} \mathbf{e}' \\ \mathbf{e}'' \end{bmatrix} \bmod q.$$

However, $\mathbf{c}''$ is the only ciphertext part that is by construction required to recover $\mathbf{s}$ and, hence, $(\mathbf{e}', \mathbf{e}'')$ since

$$\begin{bmatrix} \mathbf{e}' \\ \mathbf{e}'' \end{bmatrix} = \begin{bmatrix} \mathbf{c}' \\ \mathbf{c}'' \end{bmatrix} - \mathbf{A}^\top \mathbf{s} \bmod q\,.$$

As a result, one observes a performance boost as well as an increased message throughput per ciphertext. This can be attributed to the following facts. First, the additional samples $\mathbf{c}'$ are obtained using the same secret vector. Furthermore, the performance of the decryption engine benefits from the already recovered secret. That is, the error term $\mathbf{e}'$ is retrieved using the same secret as opposed to invoke the complete decryption engine in case one encrypts $\mathbf{c}'$ and $\mathbf{c}''$ separately. Finally, the error size employed to embed the data into $\mathbf{e}'$ is not restricted to any condition and can thus be chosen as large as possible. This opportunity will be discussed in the following sections.

4.2.2. Improved Message Throughput

In this section we highlight three further strategies to enhance the data throughput per ciphertext. The first and second variant require to operate the encryption scheme in the mode for high data load encryption ($\mathsf{HDL\ mode}$). The last approach, however, aims at exploiting the secret $\mathbf{s}$ as another message carrier. All approaches can easily be combined and remain secure in the standard model. The resulting schemes taking these improvements into account are presented in Section 4.2.3 and Section 4.3. The first two approaches are only applicable in the $\mathsf{HDL\ mode}$, since modifications are restricted solely to the error term related to $\mathbf{A}'$.

Maximum Discrete Gaussians. The first approach is applicable, whenever $\mathbf{A}' \in \mathbb{Z}_q^{n \times l}$ contains at least one column $l > 0$. In this case the scheme is operating in the HDL mode, which stands out due to its performance and high message throughput at essentially some small overhead. Previously, a single parameter αq was used in order to inject data into the error vector. This, however, is not optimal, since the error term associated to $\mathbf{A}'$ is not touched in order to recover $\mathbf{s}$ and hence does not represent a bottleneck. In fact, this part is ignored when decrypting ciphertexts. Once having recovered the secret $\mathbf{s}$ and $\mathbf{e}_1$, we can very efficiently recover the data injected into the error term. Hence, we can use the full bandwidth of the error term. That is, we can introduce a second parameter βq, which is used for the error vector corresponding to $\mathbf{A}'$. This parameter can be as large as $\lfloor q/(2 \cdot 4.7) \rfloor$ such that the error term does not wrap around and thus allows for errorless message recovery. Indeed, using $\beta q = p \cdot \omega(\sqrt{\log n})$ for $p = 2^k$ and $k = \lfloor \log(\frac{q}{2 \cdot 4.7 \sqrt{\ln(2(1+1/\epsilon))/\pi}}) \rfloor \approx \lfloor \log(\frac{q}{2 \cdot 4.7^2}) \rfloor$ seems to be the best choice with regard to data load. Sampling such large discrete Gaussian vectors remains almost as efficient as with small parameters, as we will point out in Section 5.1. The parameter αq associated to $\mathbf{A}''$ is, however, chosen to allow for an optimal trade-off between performance and data throughput. In this setting, the A-LWE instances produced by means of $\mathbf{A}'$ contain also statistical min-entropy beside of computational entropy when ignoring the A-LWE instances corresponding to $\mathbf{A}''$, which maintain only computational entropy. Due to this, the additional samples do not make the underlying problem easier. In fact, it is even worthwhile to ignore the respective samples when estimating the hardness of the corresponding A-LWE problem using current state-of-the-art lattice attack algorithms.

Maximum Uniform Error. Instead of letting the error vectors corresponding to $\mathbf{A}' \in \mathbb{Z}_q^{n \times l}$ be distributed following the discrete Gaussian distribution, an improved message throughput is obtained in case the error term is selected uniformly at random from $\mathbb{Z}_q^l$ via a random function such as a PRNG. At the same time the encryption engine speeds up since the discrete Gaussian step is omitted and the space complexity also improves. Formally, ciphertexts are of the following shape

$$\mathbf{A}^\top \cdot \mathbf{s} + \begin{bmatrix} \mathbf{u} \\ \mathbf{e} \end{bmatrix} \in \mathbb{Z}_q^{l+m},$$

where $\mathbf{A} = [\mathbf{A}' \mid \mathbf{A}'']$, $\mathbf{u}$ is uniform random over $\mathbb{Z}_q^l$, and $\mathbf{e}$ is sampled from the discrete Gaussian distribution such that the secret $\mathbf{s}$ can efficiently be recovered with the aid of our generalized LWE inversion algorithm described in Chapter 5 (original LWE inversion algorithm [MP12] works only for $q = 2^k$). Due to the fact that the error term is uniform in $\mathbb{Z}_q^l$ any vector could have been the secret (information-theoretically), if samples related to $\mathbf{A}''$ are not revealed. Hence, these samples do ideally not provide more information.

Message Injection into the Secret. The last approach to increase the data load involves the secret vector $\mathbf{s}$ and is applicable whenever it is sampled uniformly at random over $\mathbb{Z}_q^n$. It is exploited as a further container carrying messages. There exist two ways of doing this. First, in the random oracle model we can sample the secret vector via $\mathbf{s} = (\mathbf{r}, H(\mathbf{r}) \oplus \mathbf{m}_2)$ such that a part of the message is embedded into the secret vector using a second cryptographic hash function, where $\mathbf{r} \in \mathbb{Z}_q^l$ for $l \leq n$ denotes a uniform random seed . The second approach is more general and utilizes parts of the ciphertext as the secret vector. For instance, let the vector $\mathbf{c}$ be a standard ciphertext generated according to the CCA1-secure scheme that we present in Section 4.3. Splitting $\mathbf{c}$ into equally-sized chunks $\mathbf{c}_i$ of n elements opens up the opportunity to employ every chunk as a secret for a fresh ciphertext. Since each block $\mathbf{c}_i$ in $\mathbf{c}$ is computationally indistinguishable from uniform even given the other elements, we can generate a new ciphertext for each block $\mathbf{c}_i$ by setting $\mathbf{s}_i = \mathbf{c}_i$ and leaving this block out of $\mathbf{c}$. The remaining steps for encryption remain unchanged. Following this, we can additionally pack the data of size at most $\lfloor n \log q \rfloor$ bits into a single ciphertext. In the extreme case, we can pack all chunks $\mathbf{c} = (\mathbf{c}_1, \ldots, \mathbf{c}_{(l+m)/n})$ into $(l+m)/n$ newly generated ciphertext vectors. We decrypt via the LWE inversion algorithm invoked on all ciphertext vectors in order to recover $\mathbf{s}_i = \mathbf{c}_i$ prior to the last ciphertext part $\mathbf{c}$.

Applying the first two concepts results in an optimized and generic encryption scheme, which is detailed in the following section.

4.2.3. Optimized Generic Encryption Scheme from A-LWE

Let $l, m = m_1 + m_2, q$ be integers and $\mathsf{TDF} = (\mathsf{KeyGen}, g, g^{-1})$ be a trapdoor function with $g_{\mathbf{A}}(\mathbf{x}, \mathbf{y}) := \mathbf{x}^\top \mathbf{A} + \mathbf{y}^\top \bmod q \in \mathbb{Z}_q^{l+m}$. Via $g_{\mathbf{A}}^{-1}(\mathbf{T}, \mathbf{c}) = (\mathbf{x}, \mathbf{y})$ involving the trapdoor $\mathbf{T}$, we retrieve the input vectors $\mathbf{x}$ and $\mathbf{y}$ back for properly chosen $\mathbf{y} \in \mathbb{Z}^{l+m}$. Define by

- $\alpha q = 2^{k_1} \cdot \sqrt{\ln(2(1+1/\epsilon))/\pi}$ and $p_1 = 2^{k_1}, k_1 \in \mathbb{N}$
- $\beta q = 2^{k_2} \cdot \sqrt{\ln(2(1+1/\epsilon))/\pi}$ with $p_2 = 2^{k_2}, k_2 = \lfloor \log(\frac{q}{2 \cdot 4.7^2}) \rfloor$

the different Gaussian parameters following Section 4.2.2. If the scheme is operated in the high data load encryption mode with $l > 0$, a large error parameter βq is used to sample the discrete Gaussian vector $\mathbf{e}_3$, thus, allowing for a huge message throughput. In fact, we encrypt $k_2 = \log p_2$ bits per entry.
In order to realize the second approach with an error term distributed uniformly at random over $\mathbb{Z}_q^l$, we omit the step to sample $\mathbf{e}_3$ and directly apply $\mathbf{e} = (\mathbf{v}_2, \mathbf{e}_2, \mathbf{e}_1)$ as the error, where $\mathbf{v}_2 \in \mathbb{Z}_q^l$ with $p_2 = q$. In this case, the full bandwidth is exploited such that the complete error term encompasses data of size $l \cdot \log q + m_2 \cdot k_1$ bits. Moreover, denote by $F : \mathbb{Z}_q^n \times \mathbb{Z}^{m_1} \to \{0,1\}^c$ a random function with $c = (m_2 \cdot k_1 + l \cdot k_2)$. The scheme is applicable in the random oracle model with $F(\mathbf{s}, \mathbf{e}_1) := H(\mathbf{s})$ and $m_2 = m$ (avoiding to sample $\mathbf{C}$) or in the standard model with $F(\mathbf{s}, \mathbf{e}_1) := \mathsf{PRNG}(\mathbf{C}\mathbf{e}_1 \bmod q)$.

Generic Encryption Scheme

$\mathsf{KeyGen}(1^n)$: Generate a public matrix $\mathbf{C} \in \mathbb{Z}_q^{t \times m_1}$, public key $\mathsf{pk} := \mathbf{A} = [\mathbf{A}' \mid \mathbf{A}''] \in \mathbb{Z}_q^{n \times l+m}$ with trapdoor $\mathsf{sk} := \mathbf{T}$ where $(\mathbf{A}'', \mathbf{T}) \leftarrow \mathsf{TDF.KeyGen}(1^n)$ and a matrix $\mathbf{A}'$ sampled uniformly at random.

- Without high data load encryption $l = 0$.
- With high data load encryption $l > 0$.

$\mathsf{Enc}(\mathsf{pk}, \mathbf{m} \in \{0,1\}^c)$:

1. Sample $\mathbf{s} \leftarrow_R \mathbb{Z}_q^n, \mathbf{e}_1 \leftarrow_R \mathcal{D}_{\mathbb{Z}^{m_1}, \alpha q}$
2. Compute $(\mathbf{v}_1, \mathbf{v}_2) = \mathsf{encode}(F(\mathbf{s}, \mathbf{e}_1) \oplus \mathbf{m}) \in \mathbb{Z}_{p_1}^{m_2} \times \mathbb{Z}_{p_2}^{l}$.
   - Random oracle model: $F(\mathbf{s}, \mathbf{e}_1) := H(\mathbf{s})$
   - Standard model: $F(\mathbf{s}, \mathbf{e}_1) := \mathsf{PRNG}(\mathbf{C}\mathbf{e}_1 \bmod q)$
3. Sample $\mathbf{e}_2 \leftarrow_R \mathcal{D}_{\mathbf{v}_1 + p_1 \mathbb{Z}^{m_2}, \alpha q}$ and $\mathbf{e}_3 \leftarrow_R \mathcal{D}_{\mathbf{v}_2 + p_2 \mathbb{Z}^{l}, \beta q}$.
4. Output ciphertext $\mathbf{c}^\top = g_{\mathbf{A}}(\mathbf{s}, \mathbf{e})$ with $\mathbf{e} = (\mathbf{e}_3, \mathbf{e}_2, \mathbf{e}_1)^\top$.

$\mathsf{Dec}(\mathsf{sk}, \mathbf{c})$: Compute $g_{\mathbf{A}}^{-1}(\mathbf{T}, \mathbf{c}) = (\mathbf{s}, \mathbf{e})$.
Return $\mathbf{m} = \mathsf{decode}(\mathbf{e}_2 \bmod p_1, \mathbf{e}_3 \bmod p_2) \oplus H(\mathbf{C}\mathbf{e}_1 \bmod q)$.

Figure 4.2.: Generic encryption scheme from A-LWE involving several optimizations.

The statements below represent adapted variants of Theorem 4.3 and Theorem 4.4 restricted to the case $\beta q = \alpha q$ for the sake of simplicity. The $\mathsf{HDL\ mode}$ with a larger parameter βq does not change the security proof, since ciphertexts still remain plain A-LWE samples.

Theorem 4.5. *Let n, m, p, q be integers. Then, the generic encryption scheme above is CPA-secure assuming the hardness of* $\mathsf{decision\ A\text{-}LWE}^{\mathcal{RO}}_{n,0,m+l,\alpha q}$ *for $\alpha q = p \cdot \sqrt{\ln(2(1+1/\epsilon))/\pi} \geq 2\sqrt{n}$ and $\epsilon = \mathsf{negl}(n)$.*

Proof. A quick view to the scheme reveals that ciphertexts $\mathbf{c} \leftarrow_R L^{\mathsf{A\text{-}LWE}}_{n,0,m+l,\alpha,q}(\mathbf{m})$ are built following the A-LWE distribution, where $\mathbf{A}$ is either statistically or computationally close to uniform. Hence, Theorem 3.9 holds, where the ciphertext is still uniform random even given the message by an adversary. □

The corresponding security statement for a standard model instantiation of the scheme is given in Theorem 4.6.

Theorem 4.6. *Let $n, m = m_1 + m_2, p, q$ be integers. Then, the generic encryption scheme is CPA-secure assuming the hardness of* decision $\mathsf{A\text{-}LWE}^{\mathcal{S}}_{n,m_1,m_2+l,\alpha q}$ *for* $\alpha q = p \cdot \sqrt{\ln(2(1+1/\epsilon))/\pi} \geq 2\sqrt{n}$ *and* $\epsilon = \mathsf{negl}(n)$.

4.3. CCA-secure Encryption Scheme

Due to the new functionality of embedding messages in error vectors, we are able to propose a novel encryption scheme providing full CCA security when adopting the tagging approach presented in [Kil06, CHK04]. In fact, we get this feature for free, if we instantiate our generic construction from Section 4.2.3 with the trapdoor construction provided in [MP12]. More specifically, the authors add a tag u to the matrix $\mathbf{A}$ such that the modified matrix $\mathbf{A}_u$ keeps changing for every encryption query.

4.3.1. CCA1 secure Encryption

We start with a detailed description of the CCA1 secure encryption scheme and the involved algorithms applying the idea of tagged public keys following the approach given in [MP12]. We first consider the matrix variant and defer a description of the ring variant, which represents the basis of our implementations, to Chapter 5. Analogous to the generic encryption scheme, we let the scheme operate with two different parameters αq and βq for an improved message expansion factor, where

- $\alpha q = 2^{k_1} \cdot \sqrt{\ln(2(1+1/\epsilon))/\pi}$ and $p_1 = 2^{k_1}, k_1 \in \mathbb{N}$
- $\beta q = 2^{k_2} \cdot \sqrt{\ln(2(1+1/\epsilon))/\pi}$ with $p_2 = 2^{k_2}, k_2 = \lfloor \log(\frac{q}{2 \cdot 4.7^2}) \rfloor$

Define by $F : \mathbb{Z}_q^n \times \mathbb{Z}^{m_1} \to \{0,1\}^c$ a random function with $c = k_1 \cdot m_2 + k_2 \cdot l$ and $m = m_1 + m_2$ outputting c random bits in order to generate the error vectors related to $\mathbf{A}'$ and $\mathbf{A}''$ respectively. Furthermore, let $\mathcal{R} = \mathbb{Z}_q[x]/(f(x))$ be a ring as constructed in [MP12], where $f(x)$ denotes a monic irreducible polynomial of degree n. Furthermore, let $h : \mathcal{R} \to \mathbb{Z}_q^{n \times n}$ be an injective ring homomorphism mapping elements $a \in \mathcal{R}$ to the matrix $h(a)$. By $\mathcal{T} = \{u_1, \ldots, u_\ell\}$ we denote a large set with "unit differences" property. That is, for any two ring elements a_i and $a_j \in \mathcal{R}^\times$ with $i \neq j$ we have $a_i - a_j \in \mathcal{R}^\times$ and $h(a_i - a_j) = h(a_i) - h(a_j)$ is invertible.

CCA1-secure Encryption Scheme - Matrix Variant

$\mathsf{KeyGen}(1^n)$: Let $t \leq (d-2\lambda)/\log q$ with $d = H_\infty(\mathbf{e}_1)$ for $\mathbf{e}_1 \leftarrow \mathcal{D}_{\mathbb{Z}^n,\alpha q}$. Generate a uniform random public matrix $\mathbf{C} \in \mathbb{Z}_q^{t\times n}$, public key $\mathsf{pk} := \mathbf{A} = [\mathbf{A}' \mid \mathbf{A}''] \in \mathbb{Z}_q^{n\times l+m}$ for a statistically or computationally instantiated matrix $\mathbf{A}'' \in \mathbb{Z}_q^{n\times m}$ following [MP12] with trapdoor $\mathsf{sk} := \mathbf{R} \in \mathbb{Z}^{\bar{m}\times nk}$.

- Without high data load encryption $l = 0$.
- With high data load encryption $l > 0$.

$\mathsf{Enc}(\mathsf{pk}, \mathbf{m} \in \{0,1\}^c)$:

1. Sample a tag $u \in \mathcal{T}$ such that $h(u) = \mathbf{H} \in \mathbb{Z}_q^{n\times n}$ is invertible and generate $\mathbf{A}_u$. For instance, if $\mathbf{A}''$ is instantiated computationally, then $\mathbf{A}''_u = [\bar{\mathbf{A}} \mid \mathbf{HG} - (\bar{\mathbf{A}}\mathbf{R}_1 + \mathbf{R}_2)] \in \mathbb{Z}_q^{n\times \bar{m}+nk}$ with $k = \lceil \log q \rceil, \bar{m} = n$ and gadget matrix $\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g}^\top$. For a statistically instantiatied key we have $\bar{m} > n$ and $\mathbf{A}''_u = [\bar{\mathbf{A}} \mid \mathbf{HG} - \bar{\mathbf{A}}\mathbf{R}] \in \mathbb{Z}_q^{n\times(\bar{m}+nk)}$.
2. Sample $\mathbf{s} \leftarrow_R \mathbb{Z}_q^n$ (or $\mathbf{s} \leftarrow_R \mathcal{D}_{\mathbb{Z}^n,\alpha q}$ for $\mathbf{A}'' \approx_c \mathcal{U}(\mathbb{Z}_q^{n\times m})$) and $\mathbf{e}_1 \leftarrow \mathcal{D}_{\mathbb{Z}^{m_1},\alpha q}$.
3. Compute $(\mathbf{v}_1, \mathbf{v}_2) = \mathsf{encode}(F(\mathbf{s}, \mathbf{e}_1) \oplus \mathbf{m}) \in \mathbb{Z}_{p_1}^{m_2} \times \mathbb{Z}_{p_2}^{l}$.
   - Random oracle model: $F(\mathbf{s}, \mathbf{e}_1) := H(\mathbf{s})$ with $m_1 = 0,\ m_2 = m$.
   - Standard model: $F(\mathbf{s}, \mathbf{e}_1) := \mathsf{PRNG}(\mathbf{C}\mathbf{e}_1 \bmod q)$.
4. The ciphertext is then given by $\mathbf{c}^\top = g_{\mathbf{A}}(\mathbf{s}, \mathbf{e})$ for $\mathbf{e} = (\mathbf{e}_3, \mathbf{e}_2, \mathbf{e}_1)^\top$ with
   - $\mathbf{e}_2 \leftarrow_R \mathcal{D}_{\mathbf{v}_1 + p_1\cdot\mathbb{Z}^{m_2},\alpha q}$ for $p_1 = 2^{k_1}$.
   - $\mathbf{e}_3 \leftarrow_R \mathcal{D}_{\mathbf{v}_2 + p_2\cdot\mathbb{Z}^{l},\beta q}$ for $p_2 = 2^{k_2}$.

$\mathsf{Dec}(\mathsf{sk}, \mathbf{c})$: Compute $g_{\mathbf{A}}^{-1}(\mathbf{R}, \mathbf{c}) = (\mathbf{s}, \mathbf{e})$ for $\mathbf{A}_{\mathbf{u}} = [\mathbf{A}' \mid \mathbf{A}''_{\mathbf{u}}] \in \mathbb{Z}_q^{l+m}$ as follows. Let $\mathbf{A}'' \approx_s \mathcal{U}(\mathbb{Z}_q^{n\times m})$ or $\mathbf{A}'' \approx_c \mathcal{U}(\mathbb{Z}_q^{n\times m})$ (with $\mathbf{R} = \mathbf{R}_1$).

1. Compute $\tilde{\mathbf{c}}_1^\top = \mathbf{c}_1^\top - \mathbf{c}_2^\top \cdot \mathbf{R}$ for $\mathbf{c} = \begin{bmatrix} \mathbf{c}_3 \\ \mathbf{c}_2 \\ \mathbf{c}_1 \end{bmatrix} \in \mathbb{Z}_q^{l+\bar{m}+nk}$.
2. Invoke the LWE inversion algorithm (see Chapter 5) on $\tilde{\mathbf{c}}_1$ in order to recover $\tilde{\mathbf{s}} = \mathbf{H}\cdot\mathbf{s}$
3. Compute $\mathbf{e}^\top = \mathbf{c}^\top - \mathbf{s}^\top \cdot \mathbf{A}_u$ for $\mathbf{s} = \mathbf{H}^{-1}\tilde{\mathbf{s}}$.
4. If $\| (\mathbf{e}_1, \mathbf{e}_2) \| \geq \alpha q \cdot \sqrt{m}$ or $\| \mathbf{e}_3 \| \geq \beta q \cdot \sqrt{l}$, output $\perp$.

Return $\mathbf{m} = \mathsf{decode}(\mathbf{e}_2 \bmod p_1, \mathbf{e}_3 \bmod p_2) \oplus F(\mathbf{s}, \mathbf{e}_1)$.

Figure 4.3.: CCA1-secure encryption scheme from A-LWE.

According to [MP12] we denote by $\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g}^\top$ the gadget matrix with $\mathbf{g}^\top = (1, 2, \ldots, 2^{k-1})$. In case of uniformly distributed error vectors following the second approach from Section 4.2.2, one omits to sample $\mathbf{e}_3 \leftarrow_R \mathcal{D}_{p\mathbb{Z}^n + \mathbf{v}_2, \beta q}$ and instead employs $\mathbf{v}_2$ just as for the generic encryption scheme with uniform random vector $\mathbf{v}_2 \in \mathbb{Z}_q^l$ and $p_2 = q$. In Theorem 4.7 we prove the security of the CCA1-secure encryption scheme in the random oracle model setting. For the sake of simplicity, we restrict the proof to $l = 0$ and a statistically instantiated public key. Essentially the same proof steps are needed for a computationally instantiated public key.

Theorem 4.7. *Let $\alpha q = p \cdot \sqrt{\ln(2(1 + 1/\epsilon))/\pi}$ for an integer $p > 0$. Then, the encryption scheme above is CCA1-secure assuming the hardness of* decision $\textsf{A-LWE}^{\mathcal{RO}}_{n,0,m,\alpha q}$.

Proof. The proof is greatly simplified as compared to [MP12], since we are not required to perform any transformations to the initial A-LWE samples. In fact, we draw samples $(\mathbf{A}, \mathbf{b}^\top) \leftarrow_R L^{\textsf{A-LWE}}_{n,0,m,\alpha,q}(\mathbf{m})$ from the A-LWE distribution, where $\mathbf{b}^\top = \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top$, $\mathbf{s} \leftarrow_R \mathbb{Z}_q^n$, $\mathbf{A} \leftarrow_R \mathbb{Z}_q^{n \times m}$ and $\mathbf{e} \leftarrow_R \mathcal{D}_{\mathbf{v} + p\mathbb{Z}^m, \alpha q}$ with $\mathbf{v} = \textsf{encode}(H(\mathbf{s}) \oplus \mathbf{m})$ and $\alpha q = p \cdot \sqrt{\ln(2(1 + 1/\epsilon))/\pi} \geq \eta_\epsilon(\Lambda_p^\perp(\mathbf{I}))$. Distinguishing these samples from random ones is as hard as solving decision $\textsf{A-LWE}^{\mathcal{RO}}_{n,0,m,\alpha q}$ for the given parameters (see Theorem 4.5).
Encryption queries in our scheme are represented by ordinary A-LWE queries, thus we can give a direct reduction. Indeed, we have $\mathbf{b}_1 = \mathbf{s}^\top \bar{\mathbf{A}} + \mathbf{e}_1 \mod q$ and $\mathbf{b}_2 = \mathbf{s}^\top (h(u)\mathbf{G} - \bar{\mathbf{A}}\mathbf{R}) + \mathbf{e}_2 \mod q$, where $(\bar{\mathbf{A}}, h(u)\mathbf{G} - \bar{\mathbf{A}}\mathbf{R})$ is statistically close to uniform by the leftover hash lemma and $h(u)\mathbf{G} - \bar{\mathbf{A}}\mathbf{R}$ is $\textsf{negl}(n)$-uniform for any choice of $u \in \mathcal{T}$ following essentially the same argumentation line as in [MP12]. Hence, the advantage of the adversary in the CCA1 security game with our scheme from above is negligible. $\square$

Theorem 4.8 contains the corresponding security statement for the standard model variant of the scheme.

Theorem 4.8. *Let $\alpha q = p \cdot \sqrt{\ln(2(1 + 1/\epsilon))/\pi} \geq 2\sqrt{n}$ for an integer $p > 0$. Then, the encryption scheme above is CCA1-secure assuming the hardness of* decision $\textsf{A-LWE}^{\mathcal{S}}_{n,m_1,m_2,\alpha q}$.

Due to the simplicity of the scheme above, one can directly translate it into the ring variant based on the hardness of the corresponding ring version of A-LWE, which is based on the hardness of the related ring LWE problem.

Remark to the HDL mode. Assume, the initial public key is given by $\mathbf{A}'' = [\ \bar{\mathbf{A}}\ | \bar{\mathbf{A}}\mathbf{R} - \mathbf{H}\mathbf{G}\ ] \in \mathbb{Z}_q^{n \times m}$. Extending the public key to $\mathbf{A}$ with $\mathbf{A} = [\ \mathbf{A}' \mid \bar{\mathbf{A}} \mid h(u)\mathbf{G} - \bar{\mathbf{A}}\mathbf{R}\ ] \in \mathbb{Z}_q^{n \times (l+m)}$ allows one to encrypt $c = k_1 \cdot m_2 + k_2 \cdot l$ bits of data simultaneously by means of the message embedding approach. This is obviously not possible with the CCA1-secure encryption scheme as proposed in [MP12], since the maximum message size is solely determined by n and $k = \lceil \log q \rceil$ amounting to nk bits. The input to the inversion algorithm described in Chapter 5 is the modified trapdoor $\begin{bmatrix} \mathbf{0} \\ \mathbf{R} \\ \mathbf{I} \end{bmatrix}$ and a ciphertext $\mathbf{c}$, which then recovers $\mathbf{s}$ and $\mathbf{e} = (\mathbf{e}', \mathbf{e}'')$. Interestingly, we observe that the norm bound on the error term is only related to the part $\mathbf{e}''$ due to $\left\| \mathbf{e} \begin{bmatrix} \mathbf{0} \\ \mathbf{R} \\ \mathbf{I} \end{bmatrix} \right\| = \left\| \mathbf{e}'' \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} \right\| < q/(4 \cdot \sqrt{s_1(\mathbf{R})^2 + 1})$ in our scheme. Thus, extending the public key has no impact on how to choose $\mathbf{e}'$ in the decryption routine, except for ensuring a reasonable level of security of the underlying A-LWE instance. We now briefly explain the benefits of such a construction for a predefined amount of data.

As an advantage for encryption, we do not need to initiate so many generations of $\mathbf{s}$, u and computations of $h(u)$ as compared to the original public key, because we use the same $\mathbf{s}$ and u for a larger message size. For decryption, one notices that the inversion algorithm works almost as fast as with the original public key, since

$$(\mathbf{c}', \mathbf{c}'')^\top \begin{bmatrix} \mathbf{0} \\ \mathbf{R} \\ \mathbf{I} \end{bmatrix} = \mathbf{s}^\top \mathbf{A}_u \begin{bmatrix} \mathbf{0} \\ \mathbf{R} \\ \mathbf{I} \end{bmatrix} + \mathbf{e}^\top \begin{bmatrix} \mathbf{0} \\ \mathbf{R} \\ \mathbf{I} \end{bmatrix} = \mathbf{s}^\top \mathbf{G} + \mathbf{e}''^\top \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} .$$

This strategy is not directly applicable to the CCA1-secure encryption scheme presented in [MP12]. However, in combination with our technique to embed auxiliary data in the error term it is possible to further take advantage of this feature.

4.4. pd-RCCA-secure Encryption Scheme

The notion of CCA2 security is a desirable feature in most encryption schemes. Many of the current solutions are generic and thus coupled to overhead and loss of efficiency. Generic solutions to CCA2-secure LWE-based encryption schemes include the usage of strongly unforgeable one-time signatures [DDN00] or message authentication codes with a weak form of bit commitments [BCHK07]. Regarding the first approach the user is required to encrypt the verification key vk together with the message yielding the ciphertext $\mathbf{c}$. Next, the signature is computed over the ciphertext and is subsequently transmitted together with $\mathbf{c}$ and vk to the receiver.

In many encryption schemes we require only RCCA security, which is strictly stronger than CCA1, but slightly weaker than CCA2 security. It has been shown in [CKN03] that RCCA security is sufficient for many of the major applications of CCA-secure encryption such as authentication and key exchange etc. In particular, it ensures non-malleability in any ways that alter the message. We introduce here an elegant way to easily modify the scheme from Section 4.3.1 in order to achieve publicly detectable RCCA security (pd-RCCA) without suffering from the disadvantages of generic solutions. In fact, this is the first lattice-based construction that allows for this feature.

RCCA-secure Encryption Scheme

$\mathsf{KeyGen}(1^n)$: Generate public key $\mathsf{pk} := \mathbf{A} = [\mathbf{A}' \mid \mathbf{A}''] \in \mathbb{Z}_q^{n \times l+m}$ for a statistically or computationally instantiated matrix $\mathbf{A}'' \in \mathbb{Z}_q^{n\times m}$ following [MP12] with trapdoor $\mathsf{sk} := \mathbf{R} \in \mathbb{Z}^{\bar{m}\times nk}$.

- Without high data load encryption $l = 0$.
- With high data load encryption, then $l > 0$.

$\mathsf{Enc}(pk, \mathbf{m} \in \{0,1\}^c)$:

1. Select a nonzero $u \in \mathcal{T}$ and determine $\mathbf{A}_u = [\ \bar{\mathbf{A}}\ | h(u)\mathbf{G} - \bar{\mathbf{A}}\mathbf{R}\ ]$, with $\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g}^\top$. Then,
2. Select $\mathbf{s} \leftarrow_R \mathbb{Z}_q^n$ (or $\mathbf{s} \leftarrow_R \mathcal{D}_{\mathbb{Z}^n,\alpha q}$ for $\mathbf{A}'' \approx_c \mathcal{U}(\mathbb{Z}_q^{n\times m})$)
3. Compute $(\mathbf{v}_2, \mathbf{v}_3) = \mathsf{encode}(H_2(\mathbf{s}) \oplus \mathbf{m}) \in \mathbb{Z}_{p_1}^{m_2} \times \mathbb{Z}_{p_2}^{l}$.
4. The ciphertext is given by $\mathbf{c} = (u, \mathbf{b})$ with $\mathbf{b} = g_{\mathbf{A}}(\mathbf{s}, \mathbf{e})$ for $\mathbf{e} = (\mathbf{e}_3, \mathbf{e}_2, \mathbf{e}_1)$, where
   - $\mathbf{e}_3 \leftarrow_R \mathcal{D}_{\mathbf{v}_3 + p_2\cdot\mathbb{Z}^l,\beta q}$ for $p_2 = 2^{k_2}$
   - $\mathbf{e}_2 \leftarrow_R \mathcal{D}_{\mathbf{v}_2 + p_1\cdot\mathbb{Z}^{m_2},\alpha q}$ for $p_1 = 2^{k_1}$
   - $\mathbf{e}_1 \leftarrow \mathcal{D}_{\mathbf{v}_1 + p_1\cdot\mathbb{Z}^{m_1},\alpha q}$ with $\mathbf{v}_1 = H_1(\mathbf{s}, \mathbf{e}_2, \mathbf{e}_3, u)$.

$\mathsf{Dec}(\mathsf{sk}, \mathbf{c})$: Determine $\mathbf{A}_u = [\ \bar{\mathbf{A}}\ | \bar{\mathbf{A}}\mathbf{R} - h(u)\mathbf{G}\ ]$.

1. If parsing $\mathbf{c}$ causes an error or $u = 0$, output $\perp$. Otherwise invoke the LWE inversion algorithm from Section 5.2.4 with input parameters $(\mathbf{R}, \mathbf{A}_u, \mathbf{b})$, which outputs the values $\mathbf{s}'$ and $\mathbf{e}'$ or a failure $\perp$.
2. (Non-malleability) Check $\mathbf{e}_1' \bmod p_1 \overset{?}{=} H_1(\mathbf{s}, \mathbf{e}_2', \mathbf{e}_3', u)$ and $\|\mathbf{e}_1'\| \le \alpha q \sqrt{m_1}$.
3. (Message Recovery) If it is satisfied, compute $\mathbf{r} = H(\mathbf{s}')$ and
$$\mathbf{m} = \mathbf{r} \oplus \mathsf{decode}(\mathbf{e}_2' \bmod p_1, \mathbf{e}_3' \bmod p_2)\,.$$
4. Output $\mathbf{m}$ as the message.

Figure 4.4.: pd-RCCA-secure encryption scheme from A-LWE.

In our scheme, ciphertexts that decrypt to the same plaintext as the challenge ciphertext $(u^*, \mathbf{c}^*)$ can publicly be detected by any party. As an advantage of our construction, the transmitted data is solely restricted to the ciphertext $\mathbf{c}$ and we omit to send additional verification keys and signatures as compared to the generic

approach. We proceed by presenting the scheme with $F(\cdot)$ being instantiated in the random oracle model. This choice is based on the fact that the RCCA property involves a cryptographic hash function modeled as random oracle. However, it is also possible to apply the standard model instantiation of $F(\cdot)$ analogous to the constructions presented before in this chapter.
Accordingly, we denote by αq and βq the different parameters used to sample the error vectors with

- $\alpha q = 2^{k_1} \cdot \sqrt{\ln(2(1+1/\epsilon))/\pi}$ and $p_1 = 2^{k_1}, k_1 \in \mathbb{N}$
- $\beta q = 2^{k_2} \cdot \sqrt{\ln(2(1+1/\epsilon))/\pi}$ with $p_2 = 2^{k_2}, k_2 = \lfloor \log(\frac{q}{2 \cdot 4.7^2}) \rfloor$.

Furthermore, let $H_1 : \{0,1\}^* \to \mathbb{Z}_p^{m_1}$ and $H_2 : \mathbb{Z}_q^n \to \{0,1\}^c$ be cryptographic hash functions modeled as random oracle for $c = m_2 \cdot k_1 + l \cdot k_2$. The key concept of the RCCA-secure encryption scheme consists in replacing the lower part of the error term by $\mathbf{e}_1 \leftarrow \mathcal{D}_{\Lambda^\perp_{\mathbf{v}_1}(\mathbf{I}),\alpha q}$ with $\mathbf{v}_1 = H_1(\mathbf{s}, \mathbf{e}_2, \mathbf{e}_3, u)$. It allows the receiver to check for modifications made to the ciphertext or its ingredients $(\mathbf{s}, \mathbf{e}_2, \mathbf{e}_3, u)$ and is to be thought of a GPV signature for the public matrix $\mathbf{I}_{m_1}$. Here, m_1 can be chosen to be very small. The upper-part of the error vector, however, remains essentially unchanged. We then change the encryption and decryption routine of the CCA1-secure scheme from Section 4.3.1 as follows
For simplicity, we restrict to the case where $l = 0$ with $\alpha q = p \cdot \sqrt{\ln(2(1+1/\epsilon))/\pi} \geq \eta_\epsilon(\Lambda_p^\perp(\mathbf{I}_{m_2}))$ and integer $p > 0$.

Theorem 4.9. *The scheme in Figure 4.4 is pd-RCCA secure assuming the hardness of* decision $\mathsf{A\text{-}LWE}^{\mathcal{RO}}_{n,0,m,\alpha q}$ *for* $\alpha q = p \cdot \sqrt{\ln(2(1+1/\epsilon))/\pi} \geq 2\sqrt{n}$.

Proof. First, we have to show that the error term $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2)$ has the proper distribution. Since the coset selection for $\mathbf{e}_2$ is based on the entropy of $\mathbf{s}$ and hence the randomness of $H_2(\mathbf{s})$, the distribution of $\mathbf{e}_2$ is negligibly close to $\mathcal{D}_{\mathbb{Z}^{m_2},\alpha q}$ according to Lemma 3.2. The same argument also holds for $\mathbf{e}_1$, which uses $\mathbf{e}_2$ as the source for entropy, when computing the random coset $H_1(\mathbf{s}, \mathbf{e}_2, u)$. Thus, it follows that the distribution of the vector $\mathbf{b}$ is indistinguishable from the A-LWE distribution $L^{\mathsf{A\text{-}LWE}}_{n,0,m,\alpha,q}(\mathbf{m})$. Hence, the scheme above satisfies CCA1 security following Section 4.3.1. Suppose $(u^*, \mathbf{b}^*)$ is the challenge ciphertext. We will show that any decryption oracle query cannot further help the attacker in guessing the correct message in the security game $\mathbf{Exp}^{\mathsf{ind-rcca}}_{\mathcal{E},\mathcal{A}}(n)$ (see Section 2.5.1).
If the attacker queries $(u, \mathbf{b}^*)$ to the decryption oracle for u other than u^*, the oracle will respond with $\perp$, since $(\mathbf{e}_1^* \mod p) = H_1(\mathbf{s}^*, \mathbf{e}_2^*, u^*)$ does not hold anymore. In case $\mathbf{b} \neq \mathbf{b}^*$ for arbitrary selected u, we differentiate 3 cases:

1. $\mathbf{b}_1 = \mathbf{b}_1^*$ and $\mathbf{b}_2 \neq \mathbf{b}_2^*$, then $\mathsf{Dec}(\cdot)$ outputs $\perp$ since $(\mathbf{e}_1^* \bmod p) = H_1(\mathbf{s}^*, \mathbf{e}_2^*, u)$, but either $\mathbf{s} \neq \mathbf{s}^*$ or $\mathbf{e}_2 \neq \mathbf{e}_2^*$.
2. $\mathbf{b}_1 \neq \mathbf{b}_1^*$ and $\mathbf{b}_2 \neq \mathbf{b}_2^*$, then $\mathbf{s}$, $\mathbf{e}$ or both must have been changed. In case $\mathbf{s}$ changed and $\mathbf{e} = \mathbf{e}^*$, $\mathsf{Dec}(\cdot)$ outputs $\perp$. If $\mathbf{e}$ altered while $\mathbf{s} = \mathbf{s}^*$ is still the

same, the adversary must have known $\mathbf{s}^*$ in order to obtain $H_1(\mathbf{s}^*, \mathbf{e}_2, u)$ – and hence a preimage for an output value of the random oracle which we can use to solve the underlying A-LWE problem. Finally, if both values $\mathbf{s} \neq \mathbf{s}^*$ and $\mathbf{e} \neq \mathbf{e}^*$ and $\mathsf{Dec}(\cdot)$ outputs a message other than $\perp$, the attacker must have selected $\mathbf{s}$ and $\mathbf{e}$ or the message such that $(\mathbf{e}_1 \bmod p) = H_1(\mathbf{s}, \mathbf{e}_2, u)$ is satisfied.

3. $\mathbf{b}_1 \neq \mathbf{b}_1^*$ and $\mathbf{b}_2 = \mathbf{b}_2^*$, then the decryption oracle outputs replay in case $\mathbf{e}_2 - \mathbf{e}_2^* \in p\mathbb{Z}^{m_2}$ with $u = u^*$ and short $\mathbf{e}_2 - \mathbf{e}_2^*$ indicating that $\mathbf{b}$ and $\mathbf{b}^*$ decrypt to the same plaintext. Otherwise $\mathsf{Dec}(\cdot)$ outputs $\perp$.

In all cases, the attacker learns nothing about the message concealed in $\mathbf{b}^*$. Thus we have an IND-RCCA secure encryption scheme, which is equivalent to NM-RCCA due to the large message space. The last case implies a publicly detectable RCCA (pd-RCCA) scheme, where an arbitrary party is able to detect modified ciphertexts that decrypt to the same plaintext. Based on the relation CCA2 $\Rightarrow$ pd-RCCA $\Rightarrow$ sd-RCCA $\Rightarrow$ RCCA [CKN03], we even have a stronger security guarantee than plain RCCA. □

4.5. CCA2-secure Encryption Scheme

There exist different approaches to turn any RCCA-secure encryption scheme into a CCA2-secure scheme. A generic way of doing this, is presented in [CKN03], which does not involve any further computational assumptions. Specifically, it aims at combining a RCCA-secure public key encryption scheme with a CCA-secure symmetric key encryption scheme in order to obtain CCA2-security. It is well-known that an CCA-secure symmetric encryption scheme can be build from any secure encryption scheme. Hence, the resulting CCA2-secure scheme is said to be efficient if the underlying encryption schemes are efficient.

The second approach, that we suggest, requires to append the hash value $H_3(\mathbf{e}_1)$ to the ciphertext output by the RCCA-secure scheme presented in the previous section. We note here that H_3 is another cryptographic hash function modeled as random oracle. Following this, the new challenge ciphertext consists of the tuple $(u^*, \mathbf{b}^*, H_3(\mathbf{e}_1^*))$. We now analyze the behavior of the decryption oracle, in case the ciphertext has been modified. Assuming RCCA-security we only focus on the last case in the security game $\mathbf{Exp}_{\mathcal{E},\mathcal{A}}^{\mathsf{ind-cca2}}(n)$ (see Section 2.5.1). Thus, we need only to prove that the adversary cannot modify the error vector $\mathbf{e}_1$, since all other cases are covered due to RCCA-security. Suppose for simplicity that $l = 0$ and $\mathbf{A} = [\mathbf{A}_2, \mathbf{A}_1] \in \mathbb{Z}_q^{n \times m}$ with $\mathbf{A}_1 \in \mathbb{Z}_q^{n \times m_1}$ and $m_1 > n$

1. If $\mathbf{b}_1 \neq \mathbf{b}_1^*$,$\mathbf{b}_2 = \mathbf{b}_2^*$, and $H_3(\mathbf{e}_1) = H_3(\mathbf{e}_1^*)$, then the attacker must have found a collision.

2. If $\mathbf{b}_1 \neq \mathbf{b}_1^*$, $\mathbf{b}_2 = \mathbf{b}_2^*$, and $H_3(\mathbf{e}_1) \neq H_3(\mathbf{e}_1^*)$, then $\mathsf{Dec}(\cdot)$ outputs $\perp$, because otherwise the attacker must have known $\mathbf{e}_1 = \mathbf{b}_1 - \mathbf{s}^{*\top}\mathbf{A_1}$ and hence $\mathbf{s}^*$, which is equivalent to solving A-LWE.

This proves that the attacker is not able to derive new information about the encrypted message, even when he is given access to the decryption oracle after issuance of the challenge ciphertext.

4.6. Asymmetric Authenticated Encryption Scheme

Theoretically, it is possible to employ a digital signature as a part of the error term authenticating the encrypted message. In general, one could use any signature scheme outputting signatures satisfying the length requirements in order to be embedded into the error term. If this is not the case, one operates in the HDL mode expanding the public key to the desired length. However, many of the well-known lattice-based signature schemes produce signatures that are already distributed just like discrete Gaussians. Such a choice causes short signatures to be selected with higher probability and more importantly it allows to decouple the distribution of the signatures from the secret keys. Considering the fact that the error term is sampled from a discrete Gaussian distribution as well, we can build an authenticated encryption scheme, where parts of the error term are directly replaced by signatures. Doing this, we achieve security guarantees similar to CCA2 and moreover the receiver is able to identify the originator of the message through his embedded signature.

Lattice-based signatures due to [DDLL13, Lyu12, GPV08, MP12] are qualified for this task. It has recently been shown that these schemes are very efficient and perform very well in practice. Thus, the efficiency directly impacts our schemes allowing for fast encryption in conjunction with an authentication subroutine. Hence, there is no need for the usage of strongly unforgeable one-time signatures as before, where the verification keys have to be encrypted before transmitting the ciphertext together with the signature and the verification keys to the receiver. As an advantage, the embedded signatures optimally exploit the respective parts of the error term and further allow to identify the originator of the message. Indeed, the functionality of one-time signatures is also achieved by the more efficient non-malleability mechanisms presented in the previous sections. Notably, all of our proposals can be applied essentially without increasing the ciphertext size.

Let $q = 2^k$ and $m = m_1 + m_2 + m_3$ with $k = \log q$. As in previous sections denote by $F : \mathbb{Z}_q^n \times \mathbb{Z}^{m_2} \to \{0,1\}^c$ a random function with $c = m_3 \cdot k_1 + l \cdot k_2$ that is instantiated either in the random oracle or standard model. Furthermore, denote by αq and βq the respective parameters of the discrete Gaussian distribution:

- $\alpha q = 2^{k_1} \cdot \sqrt{\ln(2(1+1/\epsilon))/\pi}$ and $p_1 = 2^{k_1}, k_1 \in \mathbb{N}$
- $\beta q = 2^{k_2} \cdot \sqrt{\ln(2(1+1/\epsilon))/\pi}$ with $p_2 = 2^{k_2}, k_2 = \lfloor \log(\frac{q}{2 \cdot 4.7^2}) \rfloor$.

By the triple (KeyGen, Sign, Verify) we specify a signature scheme, where a part of the resulting signature is distributed following the discrete Gaussian distribution such as the practical ones [DDLL13, Lyu12, GPV08, MP12]. For instance, in [DDLL13] signatures are represented by the tuple $(\mathbf{c}, \mathbf{z})$.

Asymmetric Authenticated Encryption Scheme

$\mathsf{KeyGen}(1^n)$: Generate public key $\mathsf{pk} := \mathbf{A} = [\mathbf{A}' \mid \mathbf{A}''] \in \mathbb{Z}_q^{n \times l+m}$ for a statistically or computationally instantiated matrix $\mathbf{A}'' \in \mathbb{Z}_q^{n\times m}$ following [MP12] with trapdoor $\mathsf{sk} := \mathbf{R} \in \mathbb{Z}^{\bar{m}\times nk}$.

- Select $l > 0$ for the $\mathsf{HDL\ mode}$ and otherwise $l = 0$.

$\mathsf{Enc}(pk, \mathbf{m} \in \{0,1\}^c)$:

1. Select a nonzero $u \in \mathcal{T}$ and determine $\mathbf{A}_u = [\ \bar{\mathbf{A}}\ | h(u)\mathbf{G} - \bar{\mathbf{A}}\mathbf{R}\ ]$.
2. Select $\mathbf{s} \leftarrow_R \mathbb{Z}_q^n$. (or $\mathbf{s} \leftarrow_R \mathcal{D}_{\mathbb{Z}^n,\alpha q}$ for $\mathbf{A}'' \approx_c \mathcal{U}(\mathbb{Z}_q^{n\times m})$)
3. Sample $\mathbf{e}_2 \leftarrow_R \mathcal{D}_{\mathbb{Z}^{m_2},\alpha q}$
4. Compute $(\mathbf{v}_1, \mathbf{v}_2) = \mathsf{encode}(F(\mathbf{s}, \mathbf{e}_2) \oplus \mathbf{m}) \in \mathbb{Z}_{p_1}^{m_3} \times \mathbb{Z}_{p_2}^{l}$.
   - Random oracle model: $F(\mathbf{s}, \mathbf{e}_2) := H(\mathbf{s})$.
   - Standard model: $F(\mathbf{s}, \mathbf{e}_2) := \mathsf{PRNG}(\mathbf{C}\mathbf{e}_2 \bmod q)$ with a uniform random matrix $\mathbf{C} \in \mathbb{Z}_q^{t\times n}$ according to Section 4.3.1.
5. The ciphertext is given by $\mathbf{c} = (u, \mathbf{b})$ with $\mathbf{b}^\top = g_{\mathbf{A}}(\mathbf{s}, \mathbf{e})$ for $\mathbf{e} = (\mathbf{e}_4, \mathbf{e}_3, \mathbf{e}_2, \mathbf{e}_1)$, where
   - $\mathbf{e}_4 \leftarrow_R \mathcal{D}_{\mathbf{v}_2 + p_2 \cdot \mathbb{Z}^l, \beta q}$ for $p_2 = 2^{k_2}$.
   - $\mathbf{e}_3 \leftarrow_R \mathcal{D}_{\mathbf{v}_1 + p_1 \cdot \mathbb{Z}^{m_3}, \alpha q}$ for $p_1 = 2^{k_1}$.
   - $\mathbf{e}_1 = \mathsf{Sign}(\mathbf{s}, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_3, u)$.

$\mathsf{Dec}(\mathsf{sk}, \mathbf{c})$: Determine $\mathbf{A}_u = [\ \bar{\mathbf{A}}\ | \bar{\mathbf{A}}\mathbf{R} - h(u)\mathbf{G}\ ]$.

1. If parsing $\mathbf{c}$ causes an error or $u = 0$, output $\perp$. Otherwise invoke the LWE inversion algorithm from Section 5.2.4 with input parameters $(\mathbf{R}, \mathbf{A}_u, \mathbf{b})$, which outputs the values $\mathbf{s}'$ and $\mathbf{e}'$ or a failure $\perp$.
2. (Signature Verification) Check signature,
$$\mathsf{Verify}(\mathbf{e}_1', (\mathbf{s}, \mathbf{e}_2', \mathbf{e}_3', \mathbf{e}_4', u)) \overset{?}{=} 1\,.$$
3. (Message Recovery) Compute $\mathbf{r} = F(\mathbf{s}, \mathbf{e}_2')$ and
$$\mathbf{m} = \mathbf{r} \oplus \mathsf{decode}(\mathbf{e}_3' \bmod p_1, \mathbf{e}_4' \bmod p_2)\,.$$
4. Output $\mathbf{m}$ as the message.

Figure 4.5.: Using lattice-based signatures as a part of the error term.

Essentially, the same approach is taken for the GPV signature scheme. We note that this setting also allows for a uniformly distributed error vector $\mathbf{v}_2 \in \mathbb{Z}_q^l$ in the HDL mode without invoking the discrete Gaussian sampler as explained in Section 4.2.2. We shortly discuss how to instantiate the scheme. If the signature scheme in [MP12] is employed, one implicitly applies the probabilistic GPV signature scheme, which outputs signatures $\mathbf{z}$ for the public key $\mathbf{B}$ being distributed just as $\mathcal{D}_{\Lambda_{\mathbf{b}}^{\perp}(\mathbf{B}),s} \sim \mathcal{D}_{\mathbb{Z}^{m_1},s}$ for the hash on all malleable inputs $\mathbf{b} = H(\mathbf{s}, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4, u)$ according to Lemma 3.2. For the same message, the error term part $(\mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4)$ keeps always changing due to its high entropy. Embedding signatures in the error vector has a flavor of CCA2 security, when considering the last case in the proof of Theorem 4.9. In this particular case, building a replay is equivalent to $\mathbf{e}_1 - \mathbf{e}_1^* \in \Lambda_q^{\perp}(\mathbf{B})$ and the length of $\mathbf{e}_1 - \mathbf{e}_1^*$ is short, which is obviously a difficult task assuming the hardness of SIS. Note, that the Gaussian parameter $s = \tilde{O}(n)$ is also used for sampling $(\mathbf{e}_2, \mathbf{e}_3) \leftarrow \mathcal{D}_{\mathbb{Z}^{m_2},s} \times \mathcal{D}_{\mathbf{v}_1 + p_1 \cdot \mathbb{Z}^{m_3},s}$. Notably, the error term $\mathbf{e}_4$ related to $\mathbf{A}'$ generated in the HDL mode perfectly suits to the properties of signatures, because on this part of the error term no size restrictions are imposed with respect to the LWE inversion algorithm. One proceeds similarly in case $\mathbf{e}_1$ or $\mathbf{e}_4$ is replaced by a signature in accordance to [DDLL13, Lyu12].

4.7. Improvement of Existing Schemes

The new embedding technique offers the opportunity to improve all schemes that allow for error term recovery. In particular, if the error is sampled following the uniform or the discrete Gaussian distribution, we can apply the techniques introduced earlier in this chapter. We exemplify the applicability of our constructions at the examples of the symmetric key encryption scheme [BV11b] and the CCA1-secure encryption scheme [MP12].

4.7.1. Enhancing Existing Symmetric-Key Encryption Scheme

In this section, we show that the message embedding technique can be applied to the symmetric key encryption scheme as provided in [BV11b]. The authors propose a symmetric key encryption scheme that entails the properties of a somewhat homomorphic encryption scheme. Based on this construction they build a fully homomorphic public key encryption scheme that is KDM-secure, meaning that the scheme remains secure even when encrypting polynomial functions of the secret key. In fact, the symmetric key encryption scheme is a very efficient construction that is directly based on ring-LWE (or PLWE [BV11b]). Ciphertexts are built in a one-time pad manner. The secret key shared among the participants is the secret $\mathbf{s} \in \mathcal{R}_q = \mathbb{Z}_q[X]/\langle f(X)\rangle$ coming from ring-LWE. One typically chooses $f(X) = X^n + 1$ for $n = 2^l$ and $l \in \mathbb{N}$. We now present the encryption scheme that is enriched with our message embedding technique required to increase the message throughput. But for the sake of simplicity, we ignore the homomorphic properties of the scheme with regard to the message $\mathbf{m}_1$. Interested readers are referred to the respective descriptions in [BV11b].

Improved Symmetric Key Encryption Scheme

$\mathsf{KeyGen}(1^n)$: Let $\alpha q = p \cdot \sqrt{\ln(2(1+1/\epsilon))/\pi}$ and $n_1, n_2 > 0$ such that $n_1 + n_2 = n$. Furthermore, let t be coprime to q. Sample the coefficients of $\mathbf{s} \leftarrow_R \mathcal{R}_q$ and set the secret key $sk := \mathbf{s}$.

$\mathsf{Enc}(sk, \mathbf{m}_1 \in \mathcal{R}_t = \mathbb{Z}_t[X]/\langle f(X)\rangle, \mathbf{m}_2 \in \{0,1\}^c$, with $c = n_2 \cdot \log p)$:

1. The first message block is encoded to be a polynomial in $\mathbf{m}_1 \in \mathcal{R}_t$.
2. $\mathbf{e}_1 \leftarrow \mathcal{D}_{\mathbb{Z}^{n_1}, \alpha q}$.
3. Compute $\mathbf{v} = \mathsf{encode}(F(\mathbf{e}_1) \oplus \mathbf{m}_2) \in \mathbb{Z}_p^{n_2}$ for a second message $\mathbf{m}_2$
   - Random oracle model: $F(\mathbf{e}_1) := H(\mathbf{e}_1)$.
   - Standard model: $F(\mathbf{e}_1) := \mathsf{PRNG}(\mathbf{C}\mathbf{e}_1 \bmod q)$ with a uniform random matrix $\mathbf{C} \in \mathbb{Z}_q^{r \times n}$ according to Section 4.3.1.
4. $\mathbf{e}_2 \leftarrow \mathcal{D}_{\mathbf{v}+p\mathbb{Z}^{n_2}, \alpha q}$.
5. Sample $(\mathbf{a}, \mathbf{a}\mathbf{s} + t\mathbf{e})$, where $\mathbf{a} \leftarrow_R \mathcal{R}_q$ and $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2)$.

Output the ciphertext

$$\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2) \text{ with } \mathbf{c}_1 = \mathbf{a}, \mathbf{c}_2 = \mathbf{a}\mathbf{s} + t\mathbf{e} + \mathbf{m}_1 .$$

$\mathsf{Dec}(sk, \mathbf{c})$: Compute the first message $\mathbf{m}_1 = \mathbf{c}_2 - \mathbf{c}_1\mathbf{s} \bmod t$. We then have $t\mathbf{e} = \mathbf{c}_2 - \mathbf{c}_1\mathbf{s} - \mathbf{m}_1$. If we divide t out, we obtain $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2)$ and finally recover the second message $\mathbf{m}_2 = \mathsf{decode}\left((\mathbf{e}_2 \bmod p) \oplus F(\mathbf{e}_1)\right)$.

Figure 4.6.: Improved symmetric key encryption scheme using A-LWE based approach

A quick view to this construction shows that the error vector is efficiently obtained, because $\mathbf{c}_1\mathbf{s}$ has already been computed during decryption. And multiplication by a constant t^{-1} is performed very fast. We omit a formal proof here. Essentially, the same arguments of the proof from the original construction [BV11b] work here as well with the difference that we reduce the security of the scheme to the ring variant of the A-LWE problem. After this modification, we can additionally embed messages of size $n_2 \log p$ bits into the error term. However, we stress that the homomorphic property does not carry over to this additional slot of message. We basically obtain an encryption scheme working on message pairs $(\mathbf{m}_1, \mathbf{m}_2)$ which is homomorphic only with respect to $\mathbf{m}_1$. This scheme may be of independent interest for novel applications, since it can also be invoked in the HDL mode.

4.7.2. Enhancing an Existing CCA1-secure Encryption Scheme

All the features provided in the previous sections can be utilized in order to optimize the encryption scheme [MP12]. We particularly focus on the error term $(\bar{\mathbf{e}}, \mathbf{e}_1)$ that is recovered by the LWE inversion algorithm. The other scheme ingredients remain unchanged. One observes that the error term is unused and offers additional space for messages of size approximately $m \cdot \log p$ bits as compared to nk bits ensured in the original scheme. Hence, the total message size sums up to $m \cdot \log p_1 + l \cdot \log p_2 + nk$ bits, where $l > 0$ only in case the scheme is operated in the HDL mode described in Section 4.2.1. The scheme in [MP12] requires two parameters $\alpha q, \alpha' q$ for the error term, which have to satisfy $\alpha q, \alpha' q \geq p_1 \cdot \sqrt{\ln(2(1+1/\epsilon))/\pi}$ in order to apply the message embedding approach. In order to ensure the different security notions one has to apply the approaches introduced in the sections before. All these properties can be included at essentially the same efficiency. But one has to take care about the parameters in case the error term is augmented with authentication data as described in Section 4.6.

5. CCA-secure Encryption Scheme from A-LWE in Practice

This chapter is devoted to scrutinize the practical impact of the CCA-secure encryption scheme introduced in Chapter 4. Basically, it is instantiated by means of a suitable trapdoor function viewed as a black box such as the candidate specified in [MP12, P9]. As a consequence, the owner of the trapdoor is empowered to recover the secret and error term from an A-LWE instance and hence reveal the injected data. Following this approach, one can realize RCCA-secure and CCA2-secure encryption algorithms that are not build upon the classical one-time pad approach as before. Beside of its presumed efficiency due to the resemblance of ciphertexts to ordinary LWE samples, the scheme further allows to be combined with the one-time pad concept, hence, taking the best of both worlds. From a theoretical point of view our CCA1-secure scheme by construction admits to draw conclusions about its encryption performance and message throughput per ciphertext. In fact, its simplicity and resemblance to ordinary LWE samples suggests an efficient encryption routine. However, the decryption engine greatly depends on the quality of the trapdoor. This chapter refers to the publications [EDB15, EB15a, EB15b]. The author of this thesis was the primary investigator and author of these publications.

Our Contribution

We adopt the trapdoor candidate from Micciancio and Peikert in the considered setting allowing for simplified instantiations and practically efficient algorithms. In particular, we apply an efficient ring variant of the trapdoor construction that we introduce in Chapter 6 in order to allow for fast computations and low storage consumption for secret and public keys.

LWE Inversion for Arbitrary Moduli. We introduce a new LWE inversion algorithm for arbitrary moduli and hence generalize the algorithm from Micciancio and Peikert [MP12] using a different approach. There exist many application scenarios, where it is preferred to have a modulus of a specific shape such as a prime modulus satisfying $q \equiv 1 \bmod 2n$ in the ring setting offering the possibility to apply the fast NTT transformation. The inversion algorithm given in [MP12] is only suitable for moduli of the form $q = 2^k$. For moduli different to this shape, the inversion algorithm fails to recover the secret in LWE instances. Our algorithm, however, can efficiently be applied to LWE instances employing arbitrary moduli.

Inversion of Ring Elements. We propose, to our knowledge, the fastest inversion algorithm for ring elements in special classes of rings $\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$, where ring elements correspond to polynomials with $n = 2^l$ coefficients and $q \equiv 1 \bmod 2n$. This theoretical result has many other application areas in algebraic number theory. The inversion algorithm can check arbitrary ring elements with regard to invertibility by one NTT transformation. In fact, we can directly describe the structure of the unit group. Inverting elements in $\mathcal{R}_q^\times$ is reduced to inversion in $\mathbb{Z}_q^\times$, which is cyclic due to prime modulus. When working in the NTT representation, we are even not required to apply the NTT forward and backward transformations. This will be particularly important for the tagging approach, where a tag is chosen uniformly at random from the ring of units and is subsequently applied to the public key when encrypting messages.

Scheme Instantiation and Security. We instantiated the CCA1-secure encryption scheme in the ring setting for public keys being both statistically and computationally indistinguishable from uniform. This is obtained by use of the ring variant of the most recent trapdoor candidate [P9, MP12] ensuring CCA1-security almost for free. Moreover, we considered different parameter sets (e.g., error and secret key parameter) for $n = 512$ varying the message throughput per ciphertext. The proposed high data load encryption mode additionally allows to encrypt large blocks of data using the same secret and at a minimal increase of the running time. The system under scrutiny will, moreover, be analyzed in terms of security using state-of-the-art tools from cryptanalysis such as the embedding approach from [AFG13] and [BG14] suitable for a bounded number of LWE samples. We differentiate key recovery attacks from attacks against the ciphertext.

Novel and Efficient Discrete Gaussian Sampler (FastCDT). Current state-of-the-art discrete Gaussian samplers get less efficient once the error size is increased. This is very crucial since all of the proposed encryption schemes from Chapter 3 by construction rely on a large error size in order to allow for an optimal message throughput. In fact, a new approach towards building discrete Gaussian samplers is desired such that sampling of discrete Gaussians with large parameters is essentially as efficient as with small ones. We therefore designed a new and powerful discrete Gaussian sampler, called FastCDT, that is more efficient than all previous samplers. The sampling procedure is almost independent from the parameter and uses at runtime a CDT table of constant size containing at most 44 entries with overwhelming probability for all $\beta q = p \cdot \omega(\sqrt{\log n})$ and integers $p > 0$. We will show that almost the whole probability mass is concentrated on the 10 to 11 mid elements of the table such that we need to consider only 5 to 6 entries most of the time. At the same time FastCDT is capable of sampling from any desired partition $\Lambda_i^\perp = i + p\mathbb{Z}$ without any modifications and even faster than from $\mathbb{Z}$.

Implementation and Analysis. In order to attest the conjectured efficiency of the scheme, we implemented the scheme in software for $n = 512$. This implementation is optimized with respect to the underlying architecture using the AVX and AVX2 instruction set. To this end, we applied adapted variants of the techniques introduced in [GOPS13] to our setting. In particular, we adopted several optimizations for the polynomial representation and polynomial multiplication by use of efficient NTT operations. Using the same platform and optimizations, we implemented the scheme in the standard and random oracle model comparing it with the efficient encryption scheme due to Lindner and Peikert [LP11]. First, we notice that applying the FastCDT sampler to all considered schemes leads to a significant performance boost. For the random oracle variant, for instance, we observe improvement factors ranging between 1.3 and 2.2 as compared to the usage of current state-of-the-art samplers such as the standard CDT sampler for the same set of parameters. Our implementation results further show that the standard model variant performs, as expected, almost as efficient as its counterpart that is secure in the random oracle model. Depending on how the distribution of the error polynomials related to $\mathbf{A}'$ (in the HDL mode) is selected, we attest running times of 10 cycles per message bit for encryption and about $6 - 8$ cycles per bit for decryption in case the error is distributed according to the discrete Gaussian distribution. This represents an improvement factor of about $10 - 24$ for encryption and $3 - 4$ for decryption, when comparing with [LP11] and even faster in comparison to the Open-SSL implementation of RSA. For uniformly distributed error polynomials in the HDL mode, we testify improvement factors up to 92 and 7 for encryption and decryption,respectively, which opens up the possibility, also due to the low message expansion factors close to 1, to be applied in high data load scenarios.

Therefore, we start by introducing several tools that serve to instantiate the scheme appropriately and may be of independent interest for other applications. Subsequently, we present a software implementation of the scheme testifying its presumed efficiency.

5.1. A Fast Discrete Gaussian Sampler - FastCDT

In this section we present a new discrete Gaussian sampler that improves upon the existing discrete Gaussian samplers. It is highly efficient and outperforms even the currently most efficient standard inversion CDT based discrete Gaussian samplers with respect to running time and working memory. The respective tables of our new sampler can also be generated on the fly at the cost of being slightly slower due to the generation of at most 44 elements (or 10 elements in most cases). It is therefore efficiently applicable in constrained devices characterized by limited resources.

In fact, the basic tool required to realize such a sampler is given by Lemma 3.4 in Chapter 3 instantiated with a lattice of the form $\Lambda = p \cdot \mathbb{Z}^m$, for an integer $p > 0$. In order to sample a vector $\mathbf{e} \in \mathbb{Z}^m$ statistically close to $\mathcal{D}_{\mathbb{Z}^m,\alpha q}$ for $\alpha q = p \cdot 4.7$ with $\eta_\epsilon(\Lambda) \leq p \cdot \sqrt{\ln(2(1+1/\epsilon))/\pi} \approx 4.7 \cdot p = \alpha q$, one samples a vector $\mathbf{b} \leftarrow_R \mathbb{Z}_p^m$

uniformly at random and subsequently a discrete Gaussian from $\mathcal{D}_{\mathbf{b}+p\mathbb{Z}^m,\alpha q}$ following Lemma 3.4. This seems to be more expensive due to two sampling steps, but a closer look reveals that almost the whole entropy of $\mathbf{e}$ is coming from a uniform random source, which is far more efficient than sampling discrete Gaussians. This is particularly interesting for constrained devices preferring uniformly sampled vectors over other distributions. The remaining entropy is obtained via the sampling step $\mathcal{D}_{\mathbf{b}+p\mathbb{Z}^m,\alpha q}$. The support $\mathsf{supp} = \{\mathbf{b} + p\mathbb{Z}^m\}$ of this distribution consists of at most 44 elements $\mathsf{supp} \cap [-4.7^2 p, 4.7^2 p]$ with overwhelming probability, which allows to be generated dynamically rather than in the key generation step since the number of required CDT table entries amounts to 44 elements at most. But almost the whole probability mass is concentrated on the 10 mid elements (one case with 11 elements) following Lemma 5.1, since two neighboring elements of $\mathbf{b} + p\mathbb{Z}^m$ have a large relative distance of p decreasing the corresponding probabilities rapidly. This makes the sampler very flexible allowing to be instantiated following one of the three approaches presented below providing different trade-offs between flexibility, storage requirements and running time. This is not possible when using Knuth-Yao or the standard CDT technique, which depends on p with table size of $4.7 \cdot \alpha q \approx \lceil 4.7^2 \cdot p \rceil$ entries. Our algorithm requires a constant table size of 44 elements for arbitrary p. As another performance advantage, one observes based on the shape of the support that almost the whole probability mass is concentrated on the set $S_i = [-5p, 5p] \cap \{b_i + p\mathbb{Z}\}$ (see Lemma 5.1), making binary search less efficient than testing the CDT table entries of S_i via linear search starting with b_i or $p - b_i$. Only with small probability, one looks into the remaining table entries. In the following, we highlight the various trade-offs between flexibility, storage requirements and running time.

1. Generate the whole table at the beginning. This will occupy as much memory as the standard CDT approach. The algorithm will start to operate on the dedicated $10 - 11$ elements, out of which only 5 (either left or right) are considered via linear search once having sampled the partition $\mathbf{b} \in \mathbb{Z}_p^m$. This set of elements encompasses almost the whole probability mass. In case CDT does not find the correct value within this range, the standard binary search algorithm is invoked on the remaining elements out of 22. The expected value for the number of table lookups is approximately 2.

2. Generate everything on the fly. Whenever a partition has been sampled, the algorithm starts generating 44 elements, from which one samples an element while focusing on the $10-11$ mid elements via linear search first. This approach is interesting for high performance processors maintaining a low working memory. But also for constrained devices with low memory capacities it can be useful if the parameter is huge.

3. Generate only the $10 - 11$ elements per partition $\mathbf{b} \in \mathbb{Z}_p^m$. This is done at the start of the algorithm (or on the fly). The remaining ones out of 22 (only half of the elements are needed) are generated in case the algorithm gets out of the

range (see Lemma 5.1).

In summary, our approach looks equivalent to the standard CDT approach in case $p = 1$ or equivalently $\alpha q = \omega(\sqrt{\log n}) \approx 4.7$. However, if $p > 0$ our approach requires only to sample an additional uniform vector $\mathbf{b} \in \mathbb{Z}_p^m$, whereas the other steps remain exactly as efficient as for $p = 1$. In case of the standard CDT sampler the table size increases and hence the number of table lookups. For instance, FastCDT requires in average to sample two uniformly random elements and about two table lookups (at most $\log(22)$ lookups) as compared to the standard CDT sampler with one uniformly random element and about $\log(4.7p) \approx 2+\log(p)$ table look ups in the worst-case. Furthermore, our approach can generate the table elements (only 44 elements) dynamically, if required, as opposed to generating the complete table containing $4.7 \cdot \alpha q$ elements in the key generation phase. The most flexible approach is particularly favorable if αq is large. It is also noteworthy to mention that the FastCDT sampler combines many samplers, because it can also be used to sample a discrete Gaussian vector from any given partition modulo p. It is noteworthy that both the standard CDT and FastCDT samplers claim the same total storage size for the respective tables.

Lemma 5.1. *Let $p \in \mathbb{Z}$ and $s = p \cdot \sqrt{\frac{\ln(2(1+1/\epsilon))}{\pi}} \approx p \cdot 4.7$. Then, $|S_i| \leq 11$ and $\rho(S_i)/\rho(\Lambda_p^\perp) \approx 0.99$ for $\Lambda_i^\perp = i + p\mathbb{Z}$ and $S_i = \{i + p\mathbb{Z}\} \cap [-5p, 5p]$.*

Proof. First, we obtain $\rho(\Lambda_i^\perp) = \rho(\Lambda_p^\perp) \pm \mathsf{negl}(n)$ as per Lemma 3.1, since $\rho_{s,-i}(\Lambda_p^\perp) = \rho_s(\Lambda_p^\perp + i) = \rho_s(\Lambda_i^\perp)$ and $\eta_\epsilon(\Lambda_p^\perp) \leq p \cdot \sqrt{\ln(2(1+1/\epsilon))/\pi}$ for a negligible parameter $\epsilon \leq 2^{-100}$ according to Lemma 4.1. Furthermore, it suffices to consider the restricted set $\Lambda_p^\perp \cap [-4.7s, 4.7s]$ with at most 45 elements, which are assigned the overwhelming portion of the probability mass. One easily obtains the following interesting result

$$\begin{aligned}
\rho(\Lambda_p^\perp) = \sum_{j=-\infty}^{\infty} e^{-\pi \cdot \frac{(j \cdot p)^2}{(4.7 \cdot p)^2}} &= -1 + 2\sum_{j=0}^{\infty} e^{-\pi \cdot \frac{(j \cdot p)^2}{(4.7 \cdot p)^2}} \\
&= -1 + 2\sum_{i=0}^{\infty} (e^{-\pi/4.7^2})^{j^2} \\
&= 4.7
\end{aligned}$$

with high probability for all p. This series represents a theta function independent from p and can hence be computed using only a small number of representatives. Furthermore, one easily verifies the inequality $|S_i| \leq 11$ such that we have only to show that $\rho(S_i)/\rho(\Lambda_p^\perp) \approx 0.99$. There are only two cases to be considered, $i = 0$ and $i > 0$. As for $i = 0$, we obtain a finite series independent from p:

$$\rho(S_i)/\rho(\Lambda_p^\perp) = \frac{1}{\rho(\Lambda_p^\perp)} \sum_{j=-5}^{5} e^{-\pi \cdot \frac{(j \cdot p)^2}{(4.7 \cdot p)^2}} = \frac{1}{\rho(\Lambda_p^\perp)} \sum_{j=-5}^{5} e^{-\pi \cdot \frac{j^2}{4.7^2}} \approx 0.997\,.$$

For $i > 0$, we have $|S_i| = 10$ and

$$\begin{aligned}\rho(S_i)/\rho(\Lambda_p^\perp) &= \frac{1}{\rho(\Lambda_p^\perp)}\sum_{j=-5}^{4} e^{-\pi\cdot\frac{(j\cdot p+i)^2}{(4.7\cdot p)^2}} = \frac{1}{\rho(\Lambda_p^\perp)}\sum_{j=-5}^{4} e^{-\pi\cdot(\frac{j+i/p}{4.7})^2} \\ &\geq 1-2\cdot\frac{1}{\rho(\Lambda_p^\perp)}\sum_{j=5}^{22} e^{-\pi\cdot\frac{j^2}{4.7^2}} \approx 0.985\,.\end{aligned}$$

□

We now illustrate the three algorithms of FastCDT. Algorithm 1 initializes the respective tables, Algorithm 2 samples a coset $\mathbf{b} \in \mathbb{Z}_p^m$ uniformly at random and Algorithm 3 selects an element from the induced table via linear search starting with the mid element.

Algorithm 1: Building CDT Arrays

```
  Data: Integer $p$, $\alpha q = p \cdot \omega(\sqrt{\log n}) \approx p \cdot 4.7$
1 $\mathsf{supp}_i = \{i + p\mathbb{Z}\} \cap (-4.7\alpha q, 4.7\alpha q)$
2 $c = \max_{0 \leq i \leq p-1} |\mathsf{supp}_i| \leq 45$
3 for $i = 0 \to p-1$ do
4     for $j = -22 \to 22$ do
5         $\mathsf{CDT}_{b_i}(j) = \sum_{l=-22}^{j} \rho(i + l \cdot p)/\rho(\mathsf{supp}_i)$
6     end
7 end
```

Algorithm 2: FastCDT

```
  Data: Integer $p$, $\alpha q = p \cdot \omega(\sqrt{\log n}) \approx p \cdot 4.7$
1 Sampling from $\mathcal{D}_{\mathbb{Z}^n,\alpha q}$ :
2     $\mathbf{b} \leftarrow_R \mathbb{Z}_p^n$
3     $\mathbf{z} \leftarrow_R \mathcal{D}_{\mathbf{b}+p\mathbb{Z}^n,\alpha q}$ via Algorithm 3 for $\mathbf{b}, p, n$
4 Output $\mathbf{z}$
```

Algorithm 3: CDT Sampling

```
Data: $p \in \mathbb{Z}, n \in \mathbb{N}, \mathbf{b} \in \mathbb{Z}_p^n$
Given $\mathsf{CDT}_{b_i} : [-22, 22] \cap \mathbb{Z} \to [0,1]$ for $1 \le i \le n$
for $i = 1 \to n$ do
    $r \leftarrow_R [0,1]$
    for $j = 0 \to \pm 5$ do
        $z_i \leftarrow \mathsf{linSearch}(r, \mathsf{CDT}_{b_i}(j))$
        if $z_i = \perp$
        for $j = \pm 6 \to \pm 22$ do
            $z_i \leftarrow \mathsf{binSearch}(r, \mathsf{CDT}_{b_i}(j))$
        end
    end
end
```

Previous work relied on the entropy of normally distributed variables. In Lemma 5.2 we deduce a closed expression for the entropy of discrete Gaussian distributed vectors.

Lemma 5.2. *Let $p \in \mathbb{Z}$ and $s = p \cdot \sqrt{\frac{\ln(2(1+1/\epsilon))}{\pi}} \approx p \cdot 4.7$. Then, the entropy of a discrete Gaussian $r \leftarrow_R \mathcal{D}_{\mathbb{Z},s}$ is given by $H_\infty(r) = \log(e^{1/2}s)$ (with overwhelming probability).*

Proof. We can restrict the support of the sampler to $[-4.7s, 4.7s]$ from which a discrete Gaussian is sampled with overwhelming probability following [Ban95, Lemma 2.4]. From Lemma 3.2 and the algorithms for $\mathsf{FastCDT}$, we have to sample an integer b uniformly at random from the range $[0, p-1]$. Subsequently, we sample a discrete Gaussian via the distribution $\mathcal{D}_{b+p\mathbb{Z},s}$. We note that $\rho(\Lambda_i^\perp) = \rho(\Lambda_p^\perp) \pm \mathsf{negl}(n)$ as per Lemma 3.1 and Lemma 5.1. The entropy is then given by

$$
\begin{aligned}
H_\infty(r) &= -\sum_{i=-\infty}^{\infty} P(i) \cdot \log P(i) \;=\; -\sum_{i=0}^{p-1} \sum_{j=-\infty}^{\infty} \frac{\rho(i + j\cdot p)}{p \cdot \rho(\Lambda_i^\perp)} \cdot \log\left(\frac{\rho(i + j\cdot p)}{p \cdot \rho(\Lambda_i^\perp)}\right) \\
&= -\frac{1}{p \cdot \rho(\Lambda_p^\perp)} \sum_{i=0}^{p-1} \sum_{j=-\infty}^{\infty} \rho(i + j\cdot p) \cdot \log(\rho(i + j \cdot p)) + \log(p \cdot \rho(\Lambda_p^\perp))\,.
\end{aligned}
$$

For the left term of the last equation, we can deduce a simple series with parameter 4.7 for all p. In fact, we have

$$
\begin{aligned}
-\sum_{j=-\infty}^{\infty} \rho(x+j\cdot p)\cdot\log(\rho(x+j\cdot p)) &= \sum_{j=-\infty}^{\infty} e^{-\pi\frac{(x+j\cdot p)^2}{s^2}}\cdot\pi\frac{(x+j\cdot p)^2}{s^2}\log e \\
&= \sum_{j=-\infty}^{\infty} e^{-\pi\frac{(x/p+j)^2}{4.7^2}}\cdot\pi\frac{(x/p+j)^2}{4.7^2}\log e
\end{aligned}
$$

An easy computation shows that the maximum of $\rho(x)\cdot\log(\rho(x))$ is at $x=\frac{s}{\sqrt{\pi}}$ with $\rho(x)\cdot\log(\rho(x))=1/e$. Furthermore, we see from the last equation that we need only to consider $x\in[0,p]\cap\mathbb{Z}$ or equivalently $y=x/p\in[0,1]\cap\mathbb{Q}$ for arbitrary p. Thus, we have to compute

$$\sum_{j=-\infty}^{\infty} e^{-\pi\frac{(y+j)^2}{4.7^2}}\cdot\pi\frac{(y+j)^2}{4.7^2}\log e = 2.35\cdot\log e,$$

which is derived from the expectation value via $\frac{\pi\cdot\rho(\Lambda_p^\perp)\cdot\log e}{4.7^2}E[x^2]$, since

$$E[(x+c)^2]=E[x^2]=\sum_{j=-\infty}^{\infty}\frac{e^{-\pi\frac{x^2}{4.7^2}}}{\rho(\Lambda_p^\perp)}\cdot x^2=\frac{4.7^2}{2\cdot\pi},$$

where the first equation follows from [MR04] for $c\in\mathbb{R}$. In fact, it has been shown that $E[x^2]=\frac{4.7^2}{2\pi}$ and $E[(x-c)^2]\leq 4.7^2\cdot\left(\frac{1}{2\pi}+\frac{\epsilon'}{1-\epsilon'}\right)$ for $4.7\geq 2\eta_{\epsilon'}(\mathbb{Z})$. Following this, we need to solve the equation $4.7=2\cdot\sqrt{\frac{\ln(2(1+1/\epsilon'))}{\pi}}$ with respect to ϵ'. Since ϵ' is negligible, we still obtain an expression very close to $E[x^2]$.

Using this, we deduce the following expression

$$\frac{1}{p\cdot\rho(\Lambda_p^\perp)}\sum_{i=0}^{p-1}\sum_{j=-\infty}^{\infty}\rho(i+j\cdot p)\cdot\log(\rho(i+j\cdot p))=\frac{p\cdot 2.35\cdot\log e}{p\cdot\rho(\Lambda_p^\perp)}=0.5\log e$$

Hence, we have $H_\infty(r)=\log(p\cdot\rho(\Lambda_p^\perp))+0.5\log e=\log(e^{1/2}s)$ where $\rho(\Lambda_p^\perp)=4.7$ with high probability following Lemma 5.1. $\square$

Table 5.1 compares different state-of-the-art discrete Gaussian samplers such as the standard CDT sampler and Knuth-Yao with FastCDT. In fact, by use of FastCDT we realize improvement factors of about $1.7-2.0$ with respect to Knuth-Yao and even $1.5-2.6$ as compared to the standard CDT sampler. Theoretically, one would expect the FastCDT sampler to be for a large p essentially as fast as with small parameters for a given efficient uniform random source. This observation is due to the constant table size at runtime. However, the small cache memory of today's architectures causes delays in case a requested table is moved into the cache. This occurs when p becomes large and hence the number of tables increases.

Parameter **p**	**Knuth-Yao** Timings in sec.	**CDT** Timings in sec.	**FastCDT** Timings in sec.
16	5.6	5.1	3.3
128	6.6	7.3	3.4
1024	7.7	9.6	3.7
4096	8.4	11.3	4.3
Improvement Factor	$\times\mathbf{1.7}-\ \times\mathbf{2.0}$	$\times\mathbf{1.5}-\ \times\mathbf{2.6}$	-

Table 5.1.: Comparison of different samplers for 10^8 samples.

Using FastCDT, the timings for the encryption engine in [P2] considerably improved. In average, we achieve an improvement factor of about 1.5 for the proposed parameters.

5.2. Techniques

Prior to starting with a description of our implementation in Section 5.4, we define the setting and introduce several tools required to operate the scheme efficiently. In particular, we propose algorithms that make use of the features accompanying the scheme in consideration.

5.2.1. Setting

We operate in the ring setting, where lattices correspond to ideals in the associated rings. This allows for more efficient algorithms as compared to the unstructured counterparts in $\mathbb{Z}_q^n$. More specifically, we will focus on cyclotomic rings of the form $\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^n+1\rangle$ for integers $q > 0$ and n being a power of two, where $\Phi_{2n}(X) = X^n + 1$ is a cyclotomic polynomial that is irreducible over $\mathbb{Z}[X]$. Cyclotomic rings have very nice structures that allow for efficient and specialized algorithms [Pol71] and furthermore provide similar worst-case to average-case hardness results [LPR10]. Though it seems to be preferable to operate with a modulus of the form $q = 2^l$, when considering the trapdoor construction and the corresponding LWE inversion algorithm from [MP12], it might be more advantageous to select a prime q such that $q = 1 \bmod 2n$. In this case $\Phi_{2n}(X) = X^n + 1$ splits into n linear factors over $\mathbb{Z}_q[X]$ such that $\mathcal{R}_q \cong \mathbb{Z}_q[X]/\langle g_1(X)\rangle \times \ldots \times \mathbb{Z}_q[X]/\langle g_n(X)\rangle$, where $g_i(X)$ denotes a linear polynomial. Due to this fact, there exists an element $\omega \in \mathbb{Z}_q$ of order $2n$ that satisfies $\Phi_{2n}(\omega) = 0 \bmod q$ since $2n|q-1$ and $\mathbb{Z}_q^\times = \mathbb{Z}_q\backslash\{0\}$ is a cyclic group. Therefore, we can write $g_i(X) = X - \xi_i$ for some element $\xi_i \in \mathbb{Z}_q$ and use the NTT [Win96] in order to efficiently perform polynomial multiplication as already observed in [GOPS13]. Let ξ be an element of order n and $\psi^2 = \xi \bmod q$. Then two polynomials $\mathbf{r}, \mathbf{u} \in \mathcal{R}_q$ are multiplied by first transforming $\mathbf{r} = (r_0, \ldots, r_{n-1})$ and $\mathbf{u}$ to $T(\mathbf{r}) = (r_0, \psi r_1, \ldots, \psi^{n-1} r_{n-1})$ and $T(\mathbf{u})$ via the bijective map $T : \mathcal{R}_q \to \mathcal{R}_q$ and subsequently computing $T(\mathbf{c}) = \mathsf{NTT}_\xi^{-1}[\mathsf{NTT}_\xi(T(\mathbf{r})) \circ \mathsf{NTT}_\xi(T(\mathbf{u}))]$. Following this approach, it is not required to double the input length to the NTT [Win96] and there is no need to use the less efficient FFT on the complex numbers. Moreover, one deduces from the representation of $\mathcal{R}_q$ that the number of invertible elements in $\mathcal{R}_q$ is given by $(q-1)^n = q^n(1-1/q)^n$ or simply by the ratio $(1-1/q)^n$, which is non-negligible for large enough values of q. In fact, when choosing $q = 8383489$ and $n = 512$ this ratio is approximately 1. This fact helps us to choose better parameters in the trapdoor generation algorithm. Beyond that, we propose a fast technique to generate ring elements and the corresponding inverses required for tagging the public key in order to ensure CCA security. This method is mainly possible due to the existence of the NTT. Following this, we are able to deduce a representation of the structure of the group of units $\mathcal{R}_q^\times$. In addition, we present in Section 5.2.4 an efficient LWE inversion algorithm for arbitrarily composed moduli q including prime numbers, since the algorithm given in [MP12] is only successfully applicable if q is a power of two. Generic approaches such as Babai's algorithm are less efficient and hence not appropriate for practical applications.

5.2.2. Instantiation from Trapdoors for Ideal-Lattices

In this paragraph we shortly recap the trapdoor generation algorithm [P9, MP12], which is used in order to construct a public key that is indistinguishable from uniform and allows to invert LWE instances. We will restrict to the ring variant following the construction from Chapter 6 with polynomials in $\mathcal{R}_q$. Thus, let $k = \lceil \log q \rceil$ and $\bar{m} > 0$ be an integer. The trapdoor generation algorithm gets as input an additional flag that is set either to $\mathsf{t} := \mathsf{statistical}$ or $\mathsf{t} := \mathsf{computational}$ invoking the respective trapdoor generation algorithms for public keys being either statistically or computationally close to uniform.

Trapdoor Generation

- $\mathsf{TrapGen}(1^n, \mathsf{t} := \mathsf{computational})$ Sample a single polynomial $\mathbf{a}_1 \in \mathcal{R}_q$ uniformly at random ($\bar{m} = 1$). Let $f_{\mathbf{a}_1}(\mathbf{x}, \mathbf{y}) = \mathbf{a}_1 \cdot \mathbf{x} + \mathbf{y} \in \mathcal{R}_q$ be a function. Sample $2k$ random polynomials $\mathbf{r}_{i,j}$ according to $\mathcal{D}_{\mathbb{Z}^n, \alpha q}$ viewing polynomials as coefficient vectors with parameter αq (e.g., $\alpha q \geq 2\sqrt{n}$) for $1 \leq i \leq k$ and $j \in \{1, 2\}$. The secret key is given by $\mathsf{sk} = [\mathbf{r}_{1,1}, \ldots, \mathbf{r}_{k,1}]$ with the corresponding public key $\mathsf{pk} := \mathbf{A}$

$$\mathbf{A} = [\mathbf{a}_1, \mathbf{g}_1 - f_{\mathbf{a}_1}(\mathbf{r}_{1,1}, \mathbf{r}_{1,2}), \ \ldots \ , \mathbf{g}_k - f_{\mathbf{a}_1}(\mathbf{r}_{k,1}, \mathbf{r}_{k,2})] \ .$$

  Analogously, if a tag $\mathbf{t}_u$ is used, we obtain $\mathbf{A}_u$ by multiplication of $\mathbf{t}_u$ with $\mathbf{g}_i$ for $1 \leq i \leq k$.

- $\mathsf{TrapGen}(1^n, \mathsf{t} := \mathsf{statistical})$ Sample $\bar{m}$ polynomials $\hat{\mathbf{a}} = [\mathbf{a}_1, \ldots, \mathbf{a}_{\bar{m}}] \in R_q^{\bar{m}}$ uniformly at random. By $h_{\hat{\mathbf{a}}}(\hat{\mathbf{x}}) = \sum_{i=1}^{\bar{m}} \mathbf{a}_i \mathbf{x}_i$ we define a generalized compact knapsack parametrized by the elements in $\hat{\mathbf{a}}$. Sample k vectors of polynomials $\hat{\mathbf{r}}_i = [\mathbf{r}_{i,1}, \ldots, \mathbf{r}_{i,\bar{m}}] \in \mathcal{R}^{\bar{m}}$ according to some distribution $\mathcal{D}$ and define $\mathbf{a}_{\bar{m}+i} = h_{\hat{\mathbf{a}}}(\hat{\mathbf{r}}_i)$ for $1 \leq i \leq k$. The public key is then given by $\mathsf{pk} := \mathbf{A}$ with

$$\mathbf{A} = [\mathbf{a}_1, \ \ldots \ , \mathbf{a}_{\bar{m}}, \mathbf{g}_1 - \mathbf{a}_{\bar{m}+1}, \ \ldots \ , \mathbf{g}_k - \mathbf{a}_{\bar{m}+k}] \, ,$$

  where $\mathbf{g}_i$ denotes the constant polynomial consisting only of zero coefficients except for the constant term 2^{i-1}. The trapdoor polynomials define the secret key $\mathsf{sk} = [\hat{\mathbf{r}}_1, \ldots, \hat{\mathbf{r}}_{\bar{m}}]$. If a tag $\mathbf{t}_u$ is taken into account, we have

$$\mathbf{A}_u = [\mathbf{a}_1, \ \ldots \ , \mathbf{a}_{\bar{m}}, \mathbf{t}_u \cdot \mathbf{g}_1 - \mathbf{a}_{\bar{m}+1}, \ \ldots \ , \mathbf{t}_u \cdot \mathbf{g}_k - \mathbf{a}_{\bar{m}+k}] \, .$$

Figure 5.1.: Trapdoor generation for CCA1-secure encryption schemes

There exist many choices of how to select the parameter $\bar{m}$ and the distribution $\mathcal{D}$ such that the polynomials $\mathbf{a}_{\bar{m}+i}$ are statistically close to uniform over $\mathcal{R}_q$. In general, there exists an inherent relationship, where a large number $\bar{m}$ of random polynomials allows to select the entries of the trapdoor polynomials $\mathbf{r}_{i,j}$ to be of small size. Conversely, a small number of random polynomials leads to larger values. In fact, one can apply the regularity bound from [SSTX09, Lemma 6], which essentially states that the statistical distance of $\mathbf{a}_{\bar{m}+i} = h_{\hat{\mathbf{a}}}(\hat{\mathbf{r}}_i)$ from the uniform distribution over $\mathcal{R}_q$ is upper bounded by

$$\epsilon \leq \frac{1}{2}\sqrt{\left(1 + \frac{q}{B^{\bar{m}}}\right)^n - 1},$$

where $\mathcal{D}$ corresponds to the uniform distribution over a set $[-b, b]^n \cap \mathbb{Z}^n$ with $B = 2b + 1$. For instance, if one reconsiders the parameters $q = 8383489$ and $n = 512$ from above, it is possible to set $\bar{m} = 46$ and $b = 2^4$ in order to ensure a statistical distance of about 2^{-100}. This bound is impractical for a low value for $\bar{m}$. A better regularity bound is given by [SS11, Lemma 3.1] and [LPR13, Corollary 7.5] for this case. For inverting LWE instances it is also important that the parameters are chosen to be small enough in order to efficiently recover the secret. The computational instantiation requires only one single polynomial $\mathbf{a}_1$ sampled uniformly at random. The other polynomials are sampled from the LWE distribution using $\mathbf{a}_1$. Applying this approach requires to sample the secret following the discrete Gaussian distribution in order to correctly recover the error.

CCA1-secure Encryption Scheme - Ring Variant

$\mathsf{KeyGen}(1^n)$: Let $t \leq (d - 2\lambda(n))/\log q$ with $d = H_\infty(\mathbf{e}_1)$ for $\mathbf{e}_1 \leftarrow \mathcal{D}_{\mathbb{Z}^n,\alpha q}$. Generate a uniform random public matrix $\mathbf{C} \in \mathbb{Z}_q^{t \times n}$, public key $\mathsf{pk} := \mathbf{A} = [\mathbf{A}' \mid \mathbf{A}''] \in \mathcal{R}_q^{l+m}$ for a statistically or computationally instantiated vector of polynomials $\mathbf{A}'' \in \mathcal{R}_q^m$ with trapdoor $\mathsf{sk} := \hat{\mathbf{r}}$.

- Select $l > 0$ for the $\mathsf{HDL\ mode}$ and otherwise $l = 0$.

$\mathsf{Enc}(\mathsf{pk}, \mathbf{m} \in \{0,1\}^c)$:

1. Sample a tag $\mathbf{t}_u$ from a large tag space ($\mathcal{R}_q$) and generate $\mathbf{A}_u$ with $\mathbf{A}''_u = [\bar{\mathbf{a}}_1, \ldots, \bar{\mathbf{a}}_m, \mathbf{t}_u\mathbf{g}_1 - \mathbf{a}_1, \ldots, \mathbf{t}_u\mathbf{g}_k - \mathbf{a}_k]$, where $k = \lceil \log q \rceil$ and $\bar{m} = 1$ for a computational instantiated public key and $\bar{m} > 1$ in case of a statistical instantiation.
2. Sample $\mathbf{s} \leftarrow_R \mathcal{R}_q$ (or $\mathbf{s} \leftarrow \mathcal{D}_{\mathcal{R},\alpha q}$ for $\mathbf{A} \approx_c \mathcal{U}(\mathcal{R}_q^{l+m})$). If $m_1 = 1$, sample $\mathbf{e}_{m+l} \leftarrow_R \mathcal{D}_{\mathcal{R},\alpha q}$.
3. $\hat{\mathbf{v}} = (\mathbf{v}_1, \ldots, \mathbf{v}_{m+l-m_1}) = \mathsf{encode}(F(\mathbf{s}, \mathbf{e}_{m+l}) \oplus \mathbf{m}) \in \mathcal{R}_{p_2}^l \times \mathcal{R}_{p_1}^{m_2}$
   - Random oracle model: $F(\mathbf{s}, \mathbf{e}_{m+l}) := H(\mathbf{s})$ with $m_1 = 0$.
   - Standard model: $F(\mathbf{s}, \mathbf{e}_{m+l}) := \mathsf{PRNG}(\mathbf{C} \cdot \mathbf{e}_{m+l} \bmod q)$ with $m_1 = 1$.
4. Sample $\mathbf{e}_i \leftarrow_R \mathcal{D}_{p_1\mathcal{R}+\mathbf{v}_i,\beta q}$ for $1 \leq i \leq l$ and $\mathbf{e}_i \leftarrow_R \mathcal{D}_{p_2\mathcal{R}+\mathbf{v}_i,\alpha q}$ for $l+1 \leq i \leq m+l-m_1$, where $\mathbf{v}_i + p\mathcal{R}$ is viewed as the set of vectors $\mathbf{v}_i + p\mathbb{Z}^n$.

The ciphertext is then given by $\hat{\mathbf{c}} = g_{\mathbf{A}_u}(\mathbf{s}, \hat{\mathbf{e}})$ for $\hat{\mathbf{e}} = [\mathbf{e}_1, \ldots, \mathbf{e}_{m+l}]$.

$\mathsf{Dec}(\mathsf{sk}, \hat{\mathbf{c}}, u)$: Compute $g^{-1}_{\mathbf{A}_u}(\hat{\mathbf{r}}, \hat{\mathbf{c}}) = (\mathbf{s}, \hat{\mathbf{e}})$ for $\mathbf{A}_u = [\mathbf{A}' \mid \mathbf{A}''_u] \in \mathcal{R}_q^{l+m}$:

1. **Statistical instantiation** Let $\mathbf{c}' = (\mathbf{c}_{l+1}, \ldots, \mathbf{c}_{l+\bar{m}})$. Then compute $\tilde{\mathbf{c}}_{l+\bar{m}+i} = \mathbf{c}_{l+\bar{m}+i} - h_{\mathbf{c}'}(\hat{\mathbf{r}}_i)$ for $1 \leq i \leq k$ and $m = \bar{m} + k$. Now, invoke the LWE inversion algorithm from Section 5.2.4 on $\tilde{\mathbf{c}}_{l+\bar{m}+1}, \ldots \tilde{\mathbf{c}}_{l+\bar{m}+k}$ in order to recover $\tilde{\mathbf{s}} = \mathbf{t}_u \cdot \mathbf{s}$. Subsequently, compute $\hat{\mathbf{e}} = \hat{\mathbf{c}} - \mathbf{A}_u\mathbf{s}$ for $\mathbf{s} = \mathbf{t}_u^{-1}\tilde{\mathbf{s}}$.
2. **Computational instantiation** Determine the polynomials $\tilde{\mathbf{c}}_{l+i+1} = \mathbf{c}_{l+i+1} - \mathbf{c}_{l+1} \cdot \mathbf{r}_{i,1}$ for $1 \leq i \leq k$. The remaining steps are equal to the statistical instantiation.

If $\| \mathbf{e}_i \| \geq \beta q \cdot \sqrt{n}$ for $1 \leq i \leq l$ or $\| \mathbf{e}_j \| \geq \alpha q \cdot \sqrt{n}$ for $l+1 \leq i \leq m+l$, output $\perp$. Else return $\mathbf{m} = \mathsf{decode}(\hat{\mathbf{e}}_2 \bmod p_2, \hat{\mathbf{e}}_1 \bmod p_1) \oplus F(\mathbf{s}, \mathbf{e}_{m+l})$ for $\hat{\mathbf{e}}_2 = [\mathbf{e}_1, \ldots, \mathbf{e}_l]$ and $\hat{\mathbf{e}}_1 = [\mathbf{e}_{l+1}, \ldots, \mathbf{e}_{l+m-m_1}]$.

Figure 5.2.: Ring variant of the CCA1-secure encryption scheme.

5.2.3. CCA-secure Encryption Scheme – Ring Variant

In Figure 5.2 we present the ring variant of the CCA1-secure scheme from Section 4.3. To this end, the scheme is operated with two different parameters αq and βq for an improved message expansion factor as described in Section 4.2.2. Furthermore, define $\mathcal{R}_q = \mathbb{Z}_q[X]/\left<X^n+1\right>$ for n a power of two and

- $\alpha q = 2^{k_1} \cdot \sqrt{\ln(2(1+1/\epsilon))/\pi}$ with $p_1 = 2^{k_1}$
- $\beta q = 2^{k_2} \cdot \sqrt{\ln(2(1+1/\epsilon))/\pi}$ with $k_2 = \lfloor \log(\frac{q}{2 \cdot 4.7^2}) \rfloor$ and $p_2 = 2^{k_2}$.

Abusing notation, we denote by m_1 and m_2 the number of polynomials rather than the number coefficients as before. Moreover, denote by $F : \mathcal{R}_q \times \mathbb{Z}_q^t \to \{0,1\}^c$ a random function for $c = n \cdot k_1(m-1) + n \cdot k_2 \cdot l$ with $m_1 = 1$ and $m_2 = m-1$ outputting c random bits in order to generate the error polynomials with respect to $\mathbf{A}'$ and $\mathbf{A}''$, respectively.

For uniformly distributed error polynomials in the HDL mode, one omits to sample $\mathbf{e}_i \leftarrow_R \mathcal{D}_{p\mathbb{Z}^n + \mathbf{v}_i, \beta q}$ for $1 \leq i \leq l$ and takes $\mathbf{v}_i$ instead similar to the generic encryption scheme in Section 4.2.3, where $\mathbf{v}_i$ is uniform random over $\mathcal{R}_q$ with $p_2 = q$.

We note that sampling polynomials according to the distribution $\mathcal{D}_{p\mathcal{R}+\mathbf{v}_i,\alpha q}$ is efficiently performed by sampling vectors according to $\mathcal{D}_{p\mathbb{Z}^n+\mathbf{v}_i,\alpha q}$ and identifying the obtained vectors as polynomials via the coefficient embedding. Furthermore, we point out that $\mathcal{R}_q^\times$ is suitable to be considered as a large tag space for certain parameters. Especially, in view of our implementation we present theoretical and practical arguments in Section 5.2.5 supporting this choice.

5.2.4. LWE Inversion for Arbitrary Modulus

In [MP12] an efficient LWE inversion algorithm has been proposed that works only for moduli q being a power of two. The second part of the LWE inversion algorithm fails whenever the modulus is of different type. Roughly speaking, the algorithm recovers the components of the secret $\mathbf{s}$ bitwise starting from the last polynomial. Shifting $\mathbf{s}$ to the left, i.e. multiplication by powers of two, deletes the most significant bits $\bmod\ q$. However, if the modulus is not of this form, the scaled secret is wrapped around and this approach does not work anymore. In the following we give a description of our approach. The first step remains essentially the same. For the sake of simplicity, let $l = 0$ such that $\mathbf{A} = \mathbf{A}''$ and $\hat{\mathbf{c}} = [\mathbf{c}_1, \ldots, \mathbf{c}_{\bar{m}+k}] = \mathbf{A}\mathbf{s} + [\mathbf{e}_1, \ldots, \mathbf{e}_{\bar{m}+k}]$ be a ciphertext.

LWE Inversion Algorithm For Arbitrary Modulus

- Step 1: Set $\tilde{\mathbf{c}}_{\bar{m}+i} = \mathbf{c}_{\bar{m}+i} - \sum_{j=1}^{\bar{m}} \mathbf{c}_j \mathbf{r}_{i,j} = \mathbf{g}_i \mathbf{s} + (\mathbf{e}_i + \sum_{j=1}^{\bar{m}} \mathbf{e}_j \mathbf{r}_{i,j}) = \mathbf{g}_i \mathbf{s} + \mathbf{p}_i$.
- Step 2: Let $\|\hat{\mathbf{p}}\|_\infty \leq b$ with overwhelming probability. When implementing the scheme, one can decrease the bound b, since the length of $\mathbf{r}_{i,j}$ needs not to be upper bounded due to direct access to the secret key. The algorithm is independently applied on each entry of $\mathbf{s} = (s_1, \ldots, s_n)$. Therefore, we start by recovering the bits of $s_1 = \sum_{i=0}^{k-1} a_i 2^i$. One proceeds successively and recovers the most significant bits at the beginning as opposed to the least significant bits following [MP12].
  1. For $i = 0,\ i \leq k$: Let $\mathbf{t} = \mathbf{g}_i \mathbf{s} + \mathbf{p}_i$ and $c = t_1 - 2^i(a_{k-1}2^{k-1} + \ldots + a_l 2^l) \bmod q$, where $a_{k-1}, \ldots, a_l \in \{0,1\}$ represent all bits of s_1 recovered up to the i-th iteration.
  2. Check the first bits of $c - b$ and $c + b$ in terms of equality, since $2^i s_1 \in [t_1 - b, t_1 + b]$. For instance, if the bit representation of $c - b$ and $c + b$ is $10100\ldots$ and $10110\ldots$, then s_1 (for $i = 0$) must have most significant bits 101 with $a_{k-1}a_{k-2}a_{k-3} = 101$.

     Case a: If the number of recovered bits in 2. is non-zero, then jump to 1. with $i = i + 1$ and proceed with $c = t_1 - 2^i(a_{k-1}2^{k-1} + \ldots + a_l 2^l) \bmod q$, where $a_{k-1}, \ldots, a_l \in \{0,1\}$ represent all bits recovered up to the i-th iteration.

     Case b: If in Step 2 no bits could be recovered due to differing bit representations, then $c \pm b$ has bit representations $10000\ldots$ and 01111. Theoretically, both representations are possible due to the perturbation vector, which is upper bounded by b (e.g. $\log q - \log b \geq 6$). Therefore, one creates two instances each for a different representation and continues the algorithm with $a_{l+1}a_{l+2}a_{l+3}a_{l+4}a_{l+5} = 10000$ and $a_{l+1}a_{l+2}a_{l+3}a_{l+4}a_{l+5} = 01111$.

  If the second case occurs at least once, the correct secret $\mathbf{s}$ is attained by checking whether the polynomials $\mathbf{e}'_i = \mathbf{c}_i - \mathbf{g}_i \mathbf{s} \bmod q$ lie in $[-4.7 \cdot \alpha q, 4.7 \cdot \alpha q]^n$ for all $1 \leq i \leq k$ after normalization of the entries of $\mathbf{e}'_i$ to the range $[-q/2, q/2]^n$, because otherwise there exists an i such that the normalized $\mathbf{e}'_i$ are not all contained in $[-4.7 \cdot \alpha q, 4.7 \cdot \alpha q]^n$ due to injectivity of $g_{\mathbf{A}}(\cdot,\cdot)$ for the chosen parameters.

The choice of a parameter for the error term can be derived with the help of the following lemma.

Lemma 5.3. *Let the secret key be sampled according to the discrete Gaussian distribution or uniformly at random with parameter r_{sec}. In order to correctly invert the LWE instance $\hat{\mathbf{c}}$, parameters for the error term are given by*

$$\alpha q \leq \frac{q}{4(1 + r_{sec}\sqrt{\bar{m}n})} \frac{1}{\sqrt{n}}.$$

Proof. Since $\mathbf{p}_i = (\mathbf{e}_i + \sum_{j=1}^{\bar{m}} \mathbf{e}_j \mathbf{r}_{ij})$, we have $\|\mathbf{p}_i\| \leq \|\mathbf{e}_i\| + \sqrt{\bar{m}}\,\|\mathbf{e}_j\|\,\|\mathbf{r}_{ij}\| \leq q/4$. For instance, if $\mathbf{r}_{ij}$ is chosen from a discrete Gaussian distribution with parameter r_{sec} (or alternatively components uniformly at random from $[-r_{sec}, r_{sec}] \cap \mathbb{Z}$). It follows $\|\mathbf{r}_{ij}\| \leq r_{sec}\sqrt{n}$ and subsequently $\|\mathbf{e}_j\| \leq \frac{q}{4(1+r_{sec}\sqrt{\bar{m}n})}$ by rearrangement of the terms. This, however, implies that the parameter αq of the error term is bounded by $\frac{q}{4(1+r_{sec}\sqrt{\bar{m}n})}\frac{1}{\sqrt{n}}$, since $\|\mathbf{e}_j\| \leq \alpha q \sqrt{n}$. $\square$

In order to estimate the bound b which affects the running time, we can compute $\|\mathbf{r}_{ij}\|$ practically, since we have direct access to the key material. In any case, b is upper bounded by $b \leq \|\mathbf{p}_i\|_\infty = 4.7 \cdot \alpha q \,\|\hat{\mathbf{r}}_i\|_2 \leq 4.7 \cdot \alpha q \cdot r_{sec}\sqrt{\bar{m}n + 1}$. This mainly follows from [Ban95, Lemma 2.4] with $\hat{\mathbf{r}}_i = (\mathbf{r}_{i,1}, \ldots, \mathbf{r}_{i,\bar{m}})$ viewed as a vector of $\mathbb{Z}_q^{n\bar{m}}$ rather than an element of $\mathcal{R}_q^{\bar{m}}$.

5.2.5. Fast Tag Generation and Inversion

Ensuring CCA and RCCA security requires to apply the tagging approach as realized in [MP12]. It is needed to generate a tag from a large set $\mathcal{U}$. We propose a technique which allows to generate tags and its corresponding inverses much faster than other proposals such as polynomial division in combination with the extended Euclidean algorithm. Furthermore, it allows to efficiently check whether an element belongs to the ring of invertible elements $\mathcal{R}_q^\times$. Conceptually, it is based on the NTT transform that acts as an isomorphism mapping polynomials $\mathbf{f}$ from $\mathcal{R}_q$ to $\tilde{\mathbf{f}}$ from $\mathbb{Z}_q[X]/\langle g_1(X)\rangle \times \ldots \times \mathbb{Z}_q[X]/\langle g_n(X)\rangle$ via $\tilde{\mathbf{f}} = \mathbf{A} \cdot T(\mathbf{f}) \bmod q$, i.e., $\tilde{f}_i = \sum_{k=0}^{n-1} f_k \psi^k \xi^{ik}$, for $\mathbf{A} = (\xi^{ij})_{1 \leq i,j \leq n}$ and an element $\xi \in \mathbb{Z}_q$ of order n [Win96]. Multiplication of two polynomials is performed componentwise after applying the NTT transform on each polynomial.

Theorem 5.4. *Let q be a prime and n be a power of 2 s.t. $q \equiv 1 \bmod 2n$. Moreover, let $\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^n+1\rangle$ and $\tilde{\mathbf{f}} = \mathbf{A} \cdot T(\mathbf{f}) \bmod q$ for $\mathbf{f} \in \mathcal{R}_q^\times$ and $T(\mathbf{f}) = (f_0, \psi f_1, \ldots, \psi^{n-1} f_{n-1})$ with NTT transformation matrix $\mathbf{A} = (\xi^{ij})_{1\leq i,j \leq n}$, where $f_0, \ldots, f_{n-1}$ denote the coefficients of the polynomial $\mathbf{f}$ (coefficient embedding). The inverse of $\mathbf{f}$ is then given by $\mathbf{f}^{-1} = \mathbf{g}$, where*

$$T(\mathbf{g}) = n^{-1}\mathbf{A}^{-1}\tilde{\mathbf{g}}$$

and $\tilde{g}_i \cdot \tilde{f}_i = 1 \bmod q$ with $1 \leq i \leq n$. Moreover, an element $\mathbf{f}$ possesses an inverse if only if $\tilde{f}_i \neq 0 \bmod q \ \forall i$.

Proof. One can easily check that the polynomial $\mathbf{c}$ with constant 1 and zero coefficients elsewhere has NTT transform $\tilde{\mathbf{c}} = (1, \ldots, 1)$. Hence, two elements $\mathbf{f}$ and $\mathbf{g}$ are inverses of each other in case $\tilde{g}_i \cdot \tilde{f}_i = 1 \bmod q$ for $1 \leq i \leq n$ due to componentwise multiplication $T(\mathbf{c}) = \mathsf{NTT}_\xi^{-1}(\mathsf{NTT}_\xi(T(\mathbf{f})) \circ \mathsf{NTT}_\xi(T(\mathbf{g}))) = \mathsf{NTT}_\xi^{-1}(\tilde{\mathbf{c}})$. This can be attributed to the fact that the NTT maps polynomials to the ring $\mathbb{Z}_q[X]/\langle g_1(X)\rangle \times \ldots \times \mathbb{Z}_q[X]/\langle g_n(X)\rangle$, where inversion is performed componentwise over $\mathbb{Z}_q$. As a result, one can easily check that an element is invertible whenever all $\tilde{f}_i \neq 0 \bmod q \ \forall i$. $\square$

From Theorem 5.4 it follows that the unit group $\mathcal{R}_q^\times = \mathsf{NTT}_\xi^{-1}((\mathbb{Z}_q \backslash \{0\})^n)$ contains $(q-1)^n$ elements such that for two random polynomials $\mathbf{g}, \mathbf{f} \in \mathcal{R}_q^\times$ the difference $\mathbf{g} - \mathbf{f}$ lies in $\mathcal{R}_q^\times$ with probability $(1 - \frac{1}{q-1})^n$, since both $\tilde{\mathbf{g}}$ and $\tilde{\mathbf{f}}$ are units and further have distinct components. More precisely, the difference $\mathbf{g} - \mathbf{f}$ is invertible, if all components of $\mathbf{A} \cdot T(\mathbf{f} - \mathbf{g}) \bmod q$ are non-zero. For large enough q the unit difference property, used only in the security proof, holds with overwhelming property. For practice, however, it suffices to set $q = 8383489$. Since multiplication always involves the NTT transform and tagging requires to multiply the tag or its inverse, we generate from a seed the components of $\tilde{\mathbf{f}}$ rather than the coefficients of $\mathbf{f}$. This approach has two major advantages over the standard way, since one NTT transformation is saved and it is, furthermore, very easy to generate invertible elements as it is only required to generate n random elements from $\mathbb{Z}_q \backslash \{0\}$. This does apparently not hold in case one desires to generate an invertible element directly from $\mathcal{R}_q$. Following this, inversion of $\tilde{\mathbf{f}}$ is also performed in the NTT state by generating the corresponding $\tilde{\mathbf{g}}$ according to Theorem 5.4. Such a strategy allows to compute inverses over $\mathbb{Z}_q$ rather than $\mathcal{R}_q^\times$, which is much more efficient.

5.3. Security Analysis

In order to estimate the security of the scheme, we mainly adopt the embedding approach, which is more appropriate for a bounded number of samples as observed in [BG14]. The distinguishing attack, however, provides better results if the number of available LWE samples is large. In principal, the embedding approach proceeds by reducing the LWE problem to the unique shortest vector problem (u-SVP). One mainly differentiates the standard embedding approach [AFG13] with the variant that has recently been shown to be more efficient especially for a small number of samples [BG14]. In the following, we give a description of the main ingredients of the embedding approach for the matrix variant.

5.3.1. Embedding Approach

Let $(\mathbf{A}^\top, \mathbf{b})$ be an LWE sample with $\mathbf{b} = \mathbf{A}^\top \mathbf{s} + \mathbf{e} \bmod q$, where $\mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z}^m,s}$ follows the discrete Gaussian distribution. The idea of this attack is to turn the problem of finding a closest lattice point (CVP) to the target vector $\mathbf{b}$ into a unique-SVP problem. Therefore, the authours start with a carefully crafted matrix $\mathbf{A}_e = \begin{bmatrix} \mathbf{A}^\top & \mathbf{b} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}$ and the corresponding q-ary embedding lattice

$$\Lambda_q(\mathbf{A}_e) = \{\mathbf{u} \in \mathbb{Z}^{m+1} \mid \exists \mathbf{x} \in \mathbb{Z}^{n+1} \text{ s.th. } \mathbf{A}_e \mathbf{x} \equiv \mathbf{u} \bmod q\}\,.$$

A short lattice point is given by $\mathbf{u} = \begin{pmatrix} \mathbf{e} \\ 1 \end{pmatrix} = \mathbf{A}_e \begin{pmatrix} -\mathbf{s} \\ 1 \end{pmatrix}$. Its length is upper bounded by $s\sqrt{m}$. In [GN08] it was conjectured that a lattice basis reduction algorithm will find a shortest vector with high probability if

$$\lambda_2(\Lambda)/\lambda_1(\Lambda) \geq \delta^{dim(\Lambda)} \cdot \tau \tag{5.1}$$

is satisfied for an algorithm-characteristical Hermite-factor δ and a lattice- resp. algorithm-specific constant $\tau \approx 0.4$. One observes by this relationship that the gap between the first and second successive minimum of the lattice Λ plays an important role for the success probability of the underlying algorithm. In order to estimate the successive minima we need the determinant of the lattice, which is given by $\det(\Lambda_q(\mathbf{A}_e)) = q^{m-n}$ with overwhelming probability for a random lattice. Subsequently, by use of the Gaussian heuristic one can deduce estimations for the lengths of the successive minima

$$\lambda_i(\Lambda) \approx \frac{\Gamma(1 + dim(\Lambda)/2)^{1/dim(\Lambda)}}{\sqrt{\pi}} \det(\Lambda)^{1/dim(\Lambda)}. \tag{5.2}$$

Substitution of λ_1 and λ_2 in Equation (5.1) by Equation (5.2) and rearrangement of the terms provides

$$\delta \approx \left(\frac{\Gamma(1 + \frac{m+1}{2})^{1/(m+1)}}{\sqrt{\pi \cdot m} \cdot \tau \cdot s} q^{\frac{m+1-n}{m+1}} \right)^{1/(m+1)}$$

for the required Hermite factor in order to break LWE samples by means of the embedding approach, where $dim(\Lambda_q(\mathbf{A}_e)) = m + 1$. Now, it is possible to estimate the time required to successfully mount an attack on LWE and subsequently derive the bit security of the underlying LWE instances. In particular, it is needed to preprocess the lattice basis by a strong basis reduction algorithm such as BKZ or the more advanced BKZ 2.0 in order to achieve the required Hermite factor. For instance, the authours of [LP11] proposed a model that is deduced by a limited set of experiments and subsequent extrapolations

$$\log_2(T(\delta)) = 1.8/\log_2(\delta) - 110\,. \tag{5.3}$$

These experiments were performed on a computer allowing for $2.3 \cdot 10^9$ operations per second.

The standard embedding approach from above is not so efficient in case one is given only a few LWE samples. As a result, the optimal attack dimension is never reached. To circumvent this situation, one changes the embedding lattice as follows

$$\Lambda_q^{\perp}(\mathbf{A}_o) = \{\mathbf{v} \in \mathbb{Z}^{m+n+1} \mid \mathbf{A}_o \cdot \mathbf{v} = \mathbf{0} \mod q\}$$

and hence allowing for a finer analysis, where $\mathbf{A}_o = \left[\mathbf{A}^\top \mid \mathbf{I} \mid \mathbf{b}\right]$. Following this approach, one increases the dimension from $m+1$ to $m+n+1$. By a trivial computation one verifies that $\mathbf{u} = \left(\mathbf{s}, \mathbf{e}, -1\right)^T \in \Lambda_q^{\perp}(\mathbf{A}_o)$. The length of this vector is bounded by $s\sqrt{m+n+1}$. Using the framework from above with $\det(\Lambda_q^{\perp}(\mathbf{A}_o)) = q^m$ one obtains the estimated security level. The ring variant requires to multiply the number of polynomials by n, i.e., $m = t \cdot n$ for $\mathbf{A} \in \mathcal{R}_q^t$.

5.3.2. Analysis of Key Recovery Attacks

Above we analyzed how to solve A-LWE instances in order to find the message or secret, but it is also possible to recover the key and decrypt the ciphertext. Therefore, we have to differentiate between the statistical and computational instantiation of the public key. We, therefore, restrict to the ring variant.

- **Statistical Instantiation:** In order to recover the key, we have to solve the ring-ISIS problem for k instances $\mathbf{a}_{\bar{m}+i} = h_{\hat{\mathbf{a}}}(\hat{\mathbf{r}}_i)$ with $1 \leq i \leq k$ and uniform random vector of polynomials $\hat{\mathbf{a}} = [\mathbf{a}_1, \ldots, \mathbf{a}_{\bar{m}}] \in \mathcal{R}_q^{\bar{m}}$. In this case, we have to find preimages $\hat{\mathbf{x}}_i$ such that $\mathbf{a}_{\bar{m}+i} = h_{\hat{\mathbf{a}}}(\hat{\mathbf{x}}_i)$ and the inequality $\|\mathbf{p}_i\| = \| \mathbf{e}_i + \sum_{j=1}^{\bar{m}} \mathbf{e}_j \mathbf{x}_{ij} \| \leq \|\mathbf{e}_i\| + \|\mathbf{e}_j\| \, \|\mathbf{x}_{ij}\| \sqrt{\bar{m}} \leq q/4$ is satisfied for $1 \leq i \leq k$ (see Section 5.2.4). Lemma 5.3 gives a reasonable upper bound on the length of $\|\hat{\mathbf{x}}_i\|_2$ for $1 \leq i \leq k$. Then, we have to find a short vector in the lattice $\Lambda^{\perp}(\hat{\mathbf{c}}_i) = \{\hat{\mathbf{x}} \in \mathcal{R}^{\bar{m}+1} \mid \sum_{i=1}^{\bar{m}+1} \mathbf{c}_i \mathbf{x}_i \equiv \mathbf{0} \bmod q\}$, where $\hat{\mathbf{c}}_i = [\hat{\mathbf{a}}, -h_{\hat{\mathbf{a}}}(\hat{\mathbf{x}}_i)]$. In fact, the vector $[\hat{\mathbf{r}}_i, 1]$ is contained in the lattice, which is by construction a solution to the ring-SIS problem. By means of the embedding approach explained in the previous section one derives the bit security.

- **Computational Instantiation:** Here, the public key is composed by k LWE instances and a uniform random polynomial $\mathbf{a}_1$ used to generate the LWE samples. For each LWE sample in the public key a new secret and error has been generated. Therefore, we are required to consider each sample independently within the security analysis. To this end, we can use the embedding approach from above or alternatively the approach from [MR09] in order to estimate its security.

5.4. Software Implementation and Performance Analysis

At the inplementation front we considered several optimizations. As for the polynomial representation and optimizations with respect to the NTT we refer to the work [GOPS13], which provides a description of an optimized implementation exploiting the single-instruction multiple-data (SIMD) instructions of the AVX instruction set extension in modern chipsets. We present a description of the main ingredients of our optimizations in Section 5.4.1 and continue in the subsequent section with our performance analysis and implementation results.

5.4.1. Software Implementation and Optimization

AVX and AVX2 Intel equipped the Sandy Bridge microarchitecture with the AVX instruction set that extends the previously 16 128-bit xmm vector registers of the Streaming SIMD Extensions (SSE) to 256-bit ymm registers. This offers the possibility to operate within these registers with either vectors of 4 double-precision or 8 single-precision floating-point numbers in arithmetic computations. As a result, implementations are enabled to perform one addition instruction and one multiplication instruction on those vectors within each cycle. This powerful tool has been utilized in the implementations [GLP12, GOPS13] leading to remarkable speedups. Beside of the floating-point multiply-accumulate instructions, the AVX2 extension set further allows to exploit the registers for integer operations. In our implementations, we are using AVX and AVX2 whenever possible.

Polynomial Representation

The polynomial representation mainly follows the work [GOPS13], which is optimized for $n = 512$. In particular, polynomials are stored in an array of 512 double-precision floating point values. Using the single instruction multiple-data (SIMD) instructions of AVX allows to operate on vectors of 4 double-precision floating points in the 256-bit ymm registers such that 4 double-precision multiplications and 4 additions of polynomial coefficients are performed in each cycle via the instructions vmultpd and vaddpd, respectively. In fact, only 64 polynomial coefficients fit into the available 16 ymm registers.

Polynomial Multiplication and NTT

For polynomial multiplication, we use the NTT transformation, which exists due to the choice of $q \equiv 1 \bmod 2n$ for $n = 512$ as already discussed in Section 5.2.1. The NTT benefits from the fact, that the root of unity is an element of $\mathbb{Z}_q$ and hence avoids to operate with complex roots needed for the standard FFT. We also adopt the optimizations from [GOPS13] made to the NTT.

Tag Generation

As for generating the tag, we use a seed in order to generate the coefficients of the NTT transformation of the tag $\mathsf{NTT}_\xi(T(\mathbf{u}))$ as described in Section 5.2.5. As a consequence, the corresponding polynomial will never be transformed back to $\mathbf{u}$, hence, saving one transformation. Inversion of $\mathsf{NTT}_\xi(T(\mathbf{u}))$ in the decryption step is performed component-wise over $\mathbb{Z}_q$, allowing to be much faster than over $\mathcal{R}_q$. Furthermore, the polynomials $\mathbf{g}_i$ have a constant coefficient 2^i and zero coefficients elsewhere. Hence, multiplication of $\mathbf{u}$ with $\mathbf{g}_i$ requires only to multiply the components of $\mathsf{NTT}_\xi(T(\mathbf{u}))$ with 2^i rather than transforming $\mathbf{g}_i$ to its NTT representation.

Storing in the NTT Representation

Due to the existence of the NTT with $\xi \in \mathbb{Z}_q$, we can store the whole public key $\mathbf{A}$ in its NTT representation without increasing the storage requirements. This even leads to a faster encryption and decryption engine, since the step of applying the NTT on the public key prior to polynomial multiplication is not required anymore. This saves one transformation in each step. The way of generating tags as shown above is perfectly adapted to this case.

Sampling Discrete Gaussians

The error term and potentially also the secret of A-LWE and LWE instances are sampled according to the discrete Gaussian distribution. In our implementation we use our FastCDT discrete Gaussian sampler introduced in Section 5.1, since it outperforms the traditional and currently most efficient counterparts CDT and Knuth-Yao even for small parameters. However, we are required to initialize the tables of our FastCDT sampler in order to sample according to $\mathcal{D}_{b+p\mathbb{Z}}$ for all $b \in \mathbb{Z}_p$. That is, we have to construct p tables, where each of them is filled with at most 44 elements (see Section 5.1).

High Data Load Encryption

In order to allow for high data load encryption, the number of polynomials $l > 0$ in $\mathbf{A}' \in \mathcal{R}_q^l$ must be non-zero. These polynomials are completely random and can hence be generated from a random seed. Therefore, it suffices to only store $\mathbf{A}'' \in \mathcal{R}_q^m$ and a seed for $\mathbf{A}'$ in its NTT representation allowing for faster operations while saving storage by use of a seed. Also with respect to security, one observes that increasing l does not decrease the bit security, since the optimal dimension has already been exceeded by $\mathbf{A}''$.

Generation of Random Polynomials

Seeds are produced by means of the Linux kernel random-generator /dev/urandom. We instantiate the random oracle $H(\cdot)$ or the PRNG, when encrypting messages, by a pseudo-random generator using Salsa20 stream cipher in order to stretch the input seed to the desired number of uniform random bits. This allows to produce as many random bits as required.

5.5. Implementation

This section is devoted to an overview of our software implementation and analysis. The implementation is realized based on the ring variant of the CCA1-secure scheme presented in Section 5.2.3. To this end, we considered both a standard and random oracle instantiation of the scheme. And for the sake of comparison, we also included the current state-of-the-art CPA-secure scheme due to Lindner and Peikert [LP11]. The optimizations introduced in the previous sections such as the new sampler FastCDT are applied whenever it is possible. For instance, the timings for the setting considered in [P2] improve by factors ranging between 1.3 and 2.2 in case the FastCDT sampler is used in place of the Knuth-Yao sampler. We also applied the new sampler FastCDT to each scheme that consumes discrete Gaussian distributed vectors. The different schemes are implemented on an

- Intel Core i7-4500U processor operating at 1.8GHz and 4GB of RAM. We used a gcc-4.8.2 compiler with compilation flags Ofast, mavx2, msse2avx, march=corei7-avx and march=core-avx-2.

5.5.1. Implementation Analysis

We implemented the scheme in two different ways by use of the different message embedding techniques from Section 4.2.2 in the HDL mode. The following two subsections consider these two strategies in more detail.

1. The error term associated to $\mathbf{A}' \in \mathcal{R}_q^l$ in the HDL mode is distributed following the discrete Gaussian distribution with a parameter βq much larger than the parameter αq used for the remaining polynomials. In principal, one could choose the largest possible parameter $\beta q = p_2 \cdot \sqrt{\ln(2(1+1/\epsilon))/\pi}$ for $p_2 = 2^{k_2}$ and $k_2 = \lfloor \log(\frac{q}{2 \cdot 4.7\sqrt{\ln(2(1+1/\epsilon))/\pi}} \rfloor \approx \lfloor \log(\frac{q}{2 \cdot 4.7^2}) \rfloor$ such that the coefficients of the error polynomials do not wrap around. Theoretically, the performance of FastCDT does almost not depend on the choice of k as already pointed out in Section 5.1. However, when it comes to implementing the scheme, the sampler

Parameter	**Description**	**Used for**
n	Dimension	$n = 512$
q	Modulus	$q \equiv 1 \bmod 2n$
$\mathcal{R}_q$	Cyclotomic ring	$\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$
p_1, p_2	Message range	$p_i = 2^{x_i}$, x_i bits/coeff.
αq	Error distribution $\mathcal{D}_{\mathbb{Z}^n,\alpha q}$ associated to $\mathbf{A}''$	$\alpha q = 4.7 \cdot p_1$
βq	Error distribution $\mathcal{D}_{\mathbb{Z}^n,\beta q}$ associated to $\mathbf{A}'$	$\beta q = 4.7 \cdot p_2$
m	Number of polynomials in $\mathbf{A}''$	$\mathbf{A}'' \in \mathcal{R}_q^m$
l	High data load encryption mode: $l > 0$	$\mathbf{A}' \in \mathcal{R}_q^l$
μ	Seed to generate $\mathbf{A}' \in \mathcal{R}_q^l$	
r_{sec}	Parameter of the secret key distribution	$\mathcal{D}_{\mathbb{Z}^n,r_{sec}}$ or $\mathcal{U}([-r_{sec}, r_{sec}]^n)$
$\bar{m}$	Number of random polynomials generating $\mathbf{A}'' \in \mathcal{R}_q^m$	$m = \bar{m} + k$
Message size	$m \cdot n \log_2 p_1 + l \cdot n \log_2 p_2$	
Ciphertext size	$(m + l)n \log_2(q)$	
Public key size	$\mu + mn \log_2(q)$	
Secret key size	$\bar{m}nk \log_2(r_{sec})$	

Table 5.2.: Parameters

must load different tables depending on the random vector $\mathbf{b}$ into the cache and hence causes delays for a large number of tables. Therefore, to account for these hardware-based effects the parameter βq is chosen such that the cycles per message size ratio is minimized. Furthermore, it is possible to partially resolve this problem by sampling more than one coefficient from a certain table, whenever the number of coefficients $l \cdot n$ related to $\mathbf{A}'$ is larger than the possible range $[0, 2^{k_2}]$ of the random entries b_i. If l is large enough, one requires in average $l \cdot n/2^{k_2}$ samples per partition $i + 2^{k_2}\mathbb{Z}$ for $0 \leq i < 2^{k_2}$.

2. The error polynomials are sampled uniformly at random from the largest possible range $\mathbb{Z}_q$. One avoids to sample discrete Gaussians and the related problems arising from cache delays. Furthermore, one can exploit the full possible range for the message. A larger value for the number of polynomials l in $\mathbf{A}'$ leads to message expansion factors converging towards 1 for the overall scheme.

The most time-critical operations are polynomial multiplications, which take about 14922 cycles including three NTT transformations, which require the major block of the running time. However, by means of the optimizations above, we can store the public key in the NTT representation and hence save one NTT transformation. Our performance results for a discrete Gaussian distributed error term in the HDL mode are given in Table 5.3 and Table 5.4, each for a different instantiation of the public key $\mathbf{A}$ according to Section 5.2.2. In particular, Table 5.3 contains the implementation results for a computationally instantiated public key, whereas Table 5.4 depicts the relevant data for a statistically instantiated public key. On the other hand Table 5.5 and Table 5.6 contain the corresponding implementation results for a uniformly distributed error term for samples related to $\mathbf{A}'$ in the HDL mode.

We provide timings (cycles per message bit), bit security, message sizes, and message expansion factors (ciphertext size/message size ratio) depending on different parameters defined in Table 5.2. At the first glance, one observes in all tables for the standard and random oracle variant with $l = 0$ that the ciphertext size of [LP11] is larger than in our setting by a factor of $2 \cdot \log p$ for the same message size. More specifically, [LP11] generates ciphertexts of size $2n \log q$ bits for n message bits, meaning that in average half a bit is encrypted per entry, whereas in our case we can encrypt $\log p = \log(\alpha q/4.7)$ bits per entry. This leads to message expansion factors between 5.8 and 3 in case we encrypt 4 and 8 bits per entry (of size 23 bits) as compared to a factor of 46 for [LP11]. As an immediate consequence, our encryption engine must be much faster than [LP11], since encryptions represent A-LWE instances resembling ordinary LWE samples in its purest form. In fact, when comparing the timings of both schemes, we observe that our scheme (for $l = 0$) outperforms [LP11] by factors between 10 and approximately 20 for the same message size and conservatively chosen parameters in our setting. That is, the timings of [LP11] would be even worse in case we choose the same discrete Gaussian parameter. Huge improvement factors are achieved in the high data load encryption mode for $l > 0$, because we can extend the public key by random polynomials and encrypt messages using the same secret. The overhead is solely restricted to generating new error polynomials and multiplying the secret with the random polynomials from $\mathbf{A}'$, which can in turn be generated (specifically the NTT representation) from a random seed.

In fact, we get much better improvement factors if l is increased and the error polynomials corresponding to $\mathbf{A}'$ are sampled from a wider discrete Gaussian distribution, hence encrypt more bits per entry while being still secure due to large βq. For decryption no length conditions are imposed on the error polynomials related to $\mathbf{A}'$. From a security point of view, our scheme has a bounded number of samples exceeding the optimal dimension with respect to the embedding approaches from [AFG13] and [BG14]. As a result, the bit security of the scheme lies independently from the number of samples $l + m$ between 279 and 395 for a computationally instantiated public key and about 202 for a statistical instantiation, in case the ciphertext is attacked. The higher security level for computationally instantiated public keys stems from the possibility to select larger parameters for the error size and hence a smaller message expansion factor. Due to the bound from Lemma 5.3, there exists a relationship among the lengths resp. parameters of the error polynomials associated to $\mathbf{A}''$ and the secret keys, which should not exceed a certain threshold in order to correctly recover the secret. Therefore, one is recommended to select a proper tradeoff between these parameters. A lower parameter r_{sec} of the secret key allows to select a higher parameter αq. Furthermore, the tables indicate that the decryption routine of [LP11] is faster (factors between 1.6 and 4.2) than ours for $l = 0$. This is mainly due to the trapdoor and the corresponding LWE inversion algorithm from Section 5.2.4, whose efficiency depends on the number of polynomial multiplications and the bound b, where a high bound allows only to recover a few bits per step. But for increasing l, the decryption engine of our scheme gets much faster. Once having recovered $\mathbf{s}$ the ciphertext part $\hat{\mathbf{c}}'$ related to $\mathbf{A}'$ is decrypted by

$\hat{\mathbf{e}}' = \hat{\mathbf{c}}' - \mathbf{A}' \cdot \mathbf{s} \bmod q$, which is similar to $\mathbf{c}_1 \cdot \mathbf{r}_2 + \mathbf{c}_2$ from [LP11] in terms of operations, except that we can decrypt more bits per entry in our scheme as compared to [LP11]. As a consequence, for large enough l decryption is performed faster than [LP11] as depicted in the tables below. In the following two sections we elaborate the impact of different distributions applied to the error polynomials related to $\mathbf{A}'$. In fact, we differentiate between the discrete Gaussian distribution and the uniform distribution in the HDL mode.

Discrete Gaussian Error in the HDL Mode

For discrete Gaussian distributed error polynomials, the cycles per message bit ratio is minimized for $\beta q = 2^{12} \cdot 4.7$. That is, we can embed 12 bits of message per entry. Hence, $p_2 = 2^{12}$. As a result, we obtain the following table which compares the standard model variant with the random oracle variant and the CPA-secure scheme from Lindner and Peikert [LP11].

	Parameters			**Sizes** (in bits)		**Timings** (in cycles/bit)		**Security** (in bits)		
	$\mathbf{r_{sec}}$	$\mathbf{p_1}$	$\mathbf{p_2}$	**Msg**	**Msg Exp.**	**Enc**	**Dec**	**[AFG13]**	**[BG14]**	**KeyRec.**
$\mathcal{S}-$**Model**										
$\mathbf{l = 0}$										
	46	2^6	-	70656	4	15.7	90.2	279	355	272
	20	2^7	-	82432	3.4	14.2	79.9	331	436	207
	8	2^8	-	94208	3	12.5	43.5	395	554	213
$\mathbf{l = 2} \cdot m$										
	46	2^6	2^{12}	365568	2.3	10.4	19.5	279	355	272
	20	2^7	2^{12}	377344	2.2	10.3	19.4	331	436	207
	8	2^8	2^{12}	389120	2.2	9.9	12.5	395	554	213
$\mathbf{l = 10} \cdot m$										
	46	2^6	2^{12}	1545216	2	9.8	7.1	279	355	272
	20	2^7	2^{12}	1556992	2	9.8	7.7	331	436	207
	8	2^8	2^{12}	1568768	2	9.9	6.2	395	554	213
$\mathcal{RO}-$**Model**										
$\mathbf{l = 0}$										
	46	2^6	-	73728	3.8	14.9	88.6	279	355	272
	20	2^7	-	86016	3.3	13.5	76.1	331	436	207
	8	2^8	-	98304	2.9	11.8	42.1	395	554	213
$\mathbf{l = 2} \cdot m$										
	46	2^6	2^{12}	614400	1.3	10.7	19.6	279	355	272
	20	2^7	2^{12}	626688	1.3	10.2	19.1	331	436	207
	8	2^8	2^{12}	638976	1.3	9.9	12.6	395	554	213
$\mathbf{l = 10} \cdot m$										
	46	2^6	2^{12}	2777088	1.1	9.4	7.1	279	355	272
	20	2^7	2^{12}	2789376	1.1	9.7	7.8	331	436	207
	8	2^8	2^{12}	2801664	1.1	10.5	6.3	395	554	213
LP11 **[LP11]**										
	-	$\alpha q = 75$	-	512	46	230.4	26.8	-	250	250
RSA 1024 (Open-SSL, 80 bit)										
	-	-	-	1024	1	2137	3330	-	-	-

Table 5.3.: Comparison of CPA-secure scheme [LP11] with the CCA-secure scheme from Section 5.2.3 for $\mathbf{A} \approx_c \mathcal{U}(\mathbb{R}_q^{l+m})$

	Parameters				**Sizes** (in bits)		**Timings** (in cycles/bit)		**Security** (in bits)		
	$\mathbf{r_{sec}}$	$\bar{\mathbf{m}}$	$\mathbf{p_1}$	$\mathbf{p_2}$	**Msg**	**Msg Exp.**	**Enc.**	**Dec.**	**[AFG13]**	**[BG14]**	**KeyRec.**
$\mathcal{S}$**−Model**											
$\mathbf{l = 0}$											
	16	46	2^4	-	139264	5.8	22.7	127.5	202	250	>> 250
	16	23	2^4	-	92160	5.9	23.4	117.6	202	250	>> 250
	8	46	2^4	-	139264	5.8	22.7	119.7	202	250	>> 250
	8	23	2^4	-	92160	5.9	23.6	110.3	202	250	>> 250
$\mathbf{l = 2 \cdot} m$											
	16	46	2^4	2^{12}	987136	2.5	11.8	21.3	202	250	>> 250
	16	23	2^4	2^{12}	657408	2.5	12.1	20.2	202	250	>> 250
	8	46	2^4	2^{12}	987136	2.5	12.1	20.3	202	250	>> 250
	8	23	2^4	2^{12}	657408	2.5	12.1	18.8	202	250	>> 250
$\mathbf{l = 10 \cdot} m$											
	16	46	2^4	2^{12}	4378624	2	9.8	7.6	202	250	>> 250
	16	23	2^4	2^{12}	2918400	2	9.7	7.5	202	250	>> 250
	8	46	2^4	2^{12}	4378624	2	9.6	7.5	202	250	>> 250
	8	23	2^4	2^{12}	2918400	2	9.9	7.2	202	250	>> 250
$\mathcal{RO}$**−Model**											
$\mathbf{l = 0}$											
	16	46	2^4	-	141312	5.8	22	125	202	250	>> 250
	16	23	2^4	-	94208	5.8	22.4	118.6	202	250	>> 250
	8	46	2^4	-	141312	5.8	22.1	118.5	202	250	>> 250
	8	23	2^4	-	94208	5.8	22.1	108.2	202	250	>> 250
$\mathbf{l = 2 \cdot} m$											
	16	46	2^4	2^{12}	989184	2.5	11.7	21.3	202	250	>> 250
	16	23	2^4	2^{12}	659456	2.5	11.9	20.1	202	250	>> 250
	8	46	2^4	2^{12}	989184	2.5	11.6	20.3	202	250	>> 250
	8	23	2^4	2^{12}	659456	2.5	12.5	18.7	202	250	>> 250
$\mathbf{l = 10 \cdot} m$											
	16	46	2^4	2^{12}	4380672	2	9.8	7.6	202	250	>> 250
	16	23	2^4	2^{12}	2920448	2	10	7.4	202	250	>> 250
	8	46	2^4	2^{12}	4380672	2	9.8	7.3	202	250	>> 250
	8	23	2^4	2^{12}	2920448	2	10	7.2	202	250	>> 250
LP11 **[LP11]**											
	-	-	$\alpha q = 75$	-	512	46	230	26.8	-	250	250
RSA 1024 (Opens-SSL, 80 bit)											
	-	-	-	-	1024	1	2137	3330	-	-	-

Table 5.4.: Comparison of CPA-secure scheme [LP11] with the CCA-secure scheme from Section 5.2.3 for $\mathbf{A} \approx_s \mathcal{U}(\mathbb{R}_q^{l+m})$

Table 5.3 and Table 5.4 contain different sizes such as the message size denoting the number of bits encrypted per ciphertext. One obtains the approximate ciphertext size by multiplication of the message size with the message expansion factor. The timings for encryption and decryption are given by the average number of cycles required to encrypt/decrypt one message bit. Table 5.3 contains the implementation results for a computationally instantiated public key, which allows for a more efficient scheme in terms of performance due to the choice of better parameters with a higher message size per coefficient. This can particularly be seen for $l = 0$ leading to message expansion factors of approx. 3 as compared to 6 for a statistically instantiated public key. Moreover, the performance of decryption is faster for a computationally instantiated public key. This follows from the low number of polynomial multiplications required to invert the corresponding A-LWE instances. However, we see that the message expansion factor decreases as soon as l raises in the HDL mode. For $l = 10 \cdot m$, for instance, we have a factor of 2 and it is principally possible to achieve even factors approximating 1 for large enough l. We note that the security of the schemes using either of the embedding approaches from [AFG13, BG14] do not exploit the ingredients of the standard model variant and hence lead to the same security level as in the random oracle model. This is due to the fact, that the random seed $\mathbf{Ce}_1 \bmod q$ has never been published and can thus not be exploited for an attack. As a result, we do not loose security when switching to the standard model using current state-of-the-art attack algorithms. If one compares the CCA1-secure scheme presented in this thesis with the CPA-secure scheme due to Lindner and Peikert in both tables, one observes improvement factors for encryption ranging between 10 and 25 for the considered parameters. However, when it comes to decryption, the CCA1-secure scheme outperforms LP11 only in the HDL mode for $l \geq 2$.

Uniform Error in the HDL Mode

In the second run of experiments we changed the error distribution of error polynomials related to $\mathbf{A}'$. We used the uniform distribution over the full range $\mathcal{R}_q$.

	Parameters		**Sizes** (in bits)		**Timings** (in cycles)		**Security** (in bits)		
	$\mathbf{r_{sec}}$	$\mathbf{p_1}$	**Msg**	**Msg Exp.**	**Enc.**	**Dec.**	**[AFG13]**	**[BG14]**	**Key Recov.**
$\mathcal{S}$**−Model**									
$\mathbf{l = 0}$									
	46	2^6	70656	4	15.7	90.2	279	355	272
	20	2^7	82432	3.4	14.2	79.9	331	436	207
	8	2^8	94208	3	12.5	43.5	395	554	213
$\mathbf{l = 2 \cdot} m$									
	46	2^6	611328	1.4	3.5	12	279	355	272
	20	2^7	623104	1.4	3.2	11.9	331	436	207
	8	2^8	634880	1.3	3.7	9	395	554	213
$\mathbf{l = 10 \cdot} m$									
	46	2^6	2774016	1.1	2.3	4.1	279	355	272
	20	2^7	2785792	1.1	2.5	4.3	331	436	207
	8	2^8	2797568	1.1	2.5	3.6	395	554	213
$\mathcal{RO}$**−Model**									
$\mathbf{l = 0}$									
	46	2^6	73728	3.8	14.9	88.6	279	355	272
	20	2^7	86016	3.3	13.5	76.1	331	436	207
	8	2^8	98304	2.9	11.8	42.1	395	554	213
$\mathbf{l = 2 \cdot} m$									
	46	2^6	368640	1.4	3.3	12.3	279	355	272
	20	2^7	380928	1.4	3.4	12.1	331	436	207
	8	2^8	393216	1.3	3.7	9.1	395	554	213
$\mathbf{l = 10 \cdot} m$									
	46	2^6	1548288	1.1	2.3	4.1	279	355	272
	20	2^7	1560576	1.1	2.5	4.4	331	436	207
	8	2^8	1572864	1.1	3	4.1	395	554	213
LP11 **[LP11]**									
	-	$\alpha q = 75$	512	46	230.4	26.8	-	250	250
RSA 1024 (Opens-SSL, 80 bit)									
	-	-	1024	1	2137	3330	-	-	-

Table 5.5.: Comparison of CPA-secure scheme [LP11] with the CCA-secure scheme from Section 5.2.3 for $\mathbf{A} \approx_c \mathcal{U}(\mathbb{R}_q^{l+m})$

	Parameters			**Sizes** (in bits)		**Timings** (in cycles/bit)		**Security** (in bits)		
	$\mathrm{r}_{\mathrm{sec}}$	$\bar{\mathrm{m}}$	p_1	**Msg**	**Msg Exp.**	**Enc.**	**Dec.**	**[AFG13]**	**[BG14]**	**KeyRec.**
$\mathcal{S}$**−Model**										
$\mathbf{l=0}$										
	16	46	2^4	139264	5.8	22.7	127.5	202	250	>> 250
	16	23	2^4	92160	5.9	23.4	117.6	202	250	>> 250
	8	46	2^4	139264	5.8	22.7	119.7	202	250	>> 250
	8	23	2^4	92160	5.9	23.6	110.3	202	250	>> 250
$\mathbf{l=2}\cdot m$										
	16	46	2^4	1693696	1.4	4	12.4	202	250	>> 250
	16	23	2^4	1128448	1.4	4	11.6	202	250	>> 250
	8	46	2^4	1693696	1.4	3.9	11.9	202	250	>> 250
	8	23	2^4	1128448	1.4	3.9	11	202	250	>> 250
$\mathbf{l=10}\cdot m$										
	16	46	2^4	7911424	1.1	2.6	4.3	202	250	>> 250
	16	23	2^4	5273600	1.1	2.6	4.2	202	250	>> 250
	8	46	2^4	7911424	1.1	2.6	4.2	202	250	>> 250
	8	23	2^4	5273600	1.1	2.6	4	202	250	>> 250
$\mathcal{RO}$**−Model**										
$\mathbf{l=0}$										
	16	46	2^4	141312	5.8	22	125	202	250	>> 250
	16	23	2^4	94208	5.8	22.4	118.6	202	250	>> 250
	8	46	2^4	141312	5.8	22.1	118.5	202	250	>> 250
	8	23	2^4	94208	5.8	22.1	108.2	202	250	>> 250
$\mathbf{l=2}\cdot m$										
	16	46	2^4	1695744	1.4	4	12.3	202	250	>> 250
	16	23	2^4	1130496	1.4	3.8	11.5	202	250	>> 250
	8	46	2^4	1695744	1.4	3.9	11.9	202	250	>> 250
	8	23	2^4	1130496	1.4	3.8	10.9	202	250	>> 250
$\mathbf{l=10}\cdot m$										
	16	46	2^4	7913472	1.1	2.5	4.2	202	250	>> 250
	16	23	2^4	5275648	1.1	2.4	4.2	202	250	>> 250
	8	46	2^4	7913472	1.1	2.6	4.1	202	250	>> 250
	8	23	2^4	5275648	1.1	2.4	4.1	202	250	>> 250
LP11 **[LP11]**										
	-	-	$\alpha q = 75$	512	46	230	26.8	-	250	250
RSA 1024 (Opens-SSL, 80 bit)										
	-	-	-	1024	1	2137	3330	-	-	-

Table 5.6.: Comparison of CPA-secure scheme [LP11] with the CCA-secure scheme from Section 5.2.3 for $\mathbf{A} \approx_s \mathcal{U}(\mathbb{R}_q^{l+m})$

Such a strategy of using the uniform distribution following Section 4.2.2 has two desirable impacts. First, we can encode messages of size $\log q$ bits per entry, hence allowing for the largest possible bit size. Second, the last discrete Gaussian step is omitted such that the resulting scheme is even more efficient in terms of performance. As for estimating the security level of the scheme the samples corresponding to $\mathbf{A}'$ are ignored due to the huge error size. These samples do not provide additional information and are, thus, not useful for current attack algorithms.
The implementation results are given in Table 5.5 and Table 5.6. One observes, that for $l = 0$ the figures must coincide with the ones given in Table 5.3 and Table 5.4, since the scheme is not operated in the HDL mode for this parameter setting. But for $l = 10 \cdot m$, the encryption and decryption engine is approximately 100 times faster than LP11. This can be attributed to the fact that we can pack much more data into the same ciphertext size leading to message expansion factors of about

1.1 as compared to 46 in the CPA-secure setting LP11. Avoiding to sample discrete Gaussians for error polynomials related to $\mathbf{A}'$ makes the scheme very efficient. Also the bit security of our scheme is very high, thus being applicable in environments characterized by high security standards. From Table 5.5, we observe that for $l = 10 \cdot m$ encryption and decryption take only 2.4 resp. 4.1 cycles per bit resulting in an improvement factor of approximately 92 resp. 6.5 as compared to LP11 for a similar level of security. For the computational variant in Table 5.5, the improvement factor is with 100 and 6.5 even higher.

Part II.

Lattice-based Signatures

Overview

Digital signature schemes belong arguably to the most commonly used cryptographic primitives in practice with a wide range of applications. Hence, digital signatures are subject to intense research. The construction of lattice based signature schemes appeared to be a big challenge up to the last couples of years. This is due to the absence of practical constructions enjoying provable security. First attempts to build lattice-based signature schemes such as GGH [GGH97b] and NTRU Sign [HHGP$^+$03] failed due to the vulnerability to statistical attacks as shown, for instance, in [DN12, GJSS01, NR06]. This fundamentally changed in 2008 with the GPV signature scheme introduced by Gentry et al. [GPV08] and the one-time signature scheme (LM-OTS) due to Lyubashevsky and Micciancio[LM08]. The latter one operates in ideal lattices which allows for faster computations and smaller key sizes while providing provable security. When using Merkle Trees one can transform LM-OTS into a full signature scheme. Subsequent works [Lyu08, Lyu09] build upon the one time signature scheme using the Fiat-Shamir transform [FS87]. It mainly takes its inspiration from Schnorr's signature scheme, since it essentially follows the same methodology of constructing signatures except for the final rejection sampling step. Recently, Lyubashevsky proposed an efficient construction [Lyu09, Lyu12] that performs very well on hardware [GLP12] and software [GOPS13]. The most recent construction [DDLL13] presented at Crypto 2013 has been proven to be practically efficient taking advantage of an improved discrete Gaussian sampler and an NTRU-like key generation procedure.
The hash-and-sign approach in turn was reconsidered in [GPV08] leading to constructions that admit security based on the hardness to solve the SIS Problem. The GPV signature scheme is a representative of this approach, which initiated the design of improving preimage sampleable trapdoor functions (PSTF) [AP09, GPV08, MP12, Pei10], the main building block of the GPV signature scheme, which has recently become practical [MP12]. In [BZ13] it has been shown that the scheme is secure even in the quantum random oracle model (EUF-qCMA secure). The ultimate goal is to construct a uniform random matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ endowed with a trapdoor $\mathbf{S} \in \mathbb{Z}^{m\times m}$, but in such a way that $\mathbf{S}$ has small entries and $\mathbf{A} \cdot \mathbf{S} \equiv 0 \mod q$ is satisfied. By means of the secret matrix $\mathbf{S}$ a signer can produce short preimages $\mathbf{x}$ for the hash value $H(\mu)$ of a message μ to be signed such that $\mathbf{Ax} \equiv H(\mu) \bmod q$. The quality of the secret matrix immediately transfers to the quality of the signatures and hence plays a major role for assessing the security. Therefore, improving the algorithms for key generation is an ongoing research objective. Such constructions were considered for the first time in [Ajt99] and later on improved by [AP09, Pei10], but unfortunately they are inefficient and

thus not suitable for practice. This is because the involved algorithms are complex and expensive in terms of space and running time. However, Micciancio and Peikert recently proposed in [MP12] an elegant trapdoor construction which allows for fast signature generation while providing an improved output quality. The second part of this thesis, consisting of Chapters 6 to 8, represents a reprint of the essential parts of [EB13, EB14a, EB14b], where the author of this thesis was the primary investigator and author of the publications.

6. Improvement of GPV Signatures

In this chapter we present the first software implementation of the GPV signature [GPV08] scheme combined with the trapdoor construction from Micciancio and Peikert [MP12] as it admits strong security proofs and is believed to be very efficient in practice. Moreover, we introduce an efficient trapdoor variant for ideal lattices, that is based on the ring-LWE problem, and propose optimizations that improve the scheme in terms of space and running time. For instance, the memory size of the perturbation matrix is lowered by a factor of about 240 as compared to the original representation from [MP12]. The perturbation matrix is a key determinant of the running time in the signing step and is particularly required in order to sample integer vectors with a given covariance. In both variants, the matrix and ring variant, we considerably improved the running times of key and signature generation due to a simplified representation of the perturbation matrix. For the considered parameters, for example, the running times of key and signature generation are lowered by a factor of 30-190 and 2-6 in the ring variant. The scheme under scrutiny will be analyzed with respect to storage sizes, running times, and security levels for different parameter sets. Furthermore, we will show that the ring variant has a 5-8 times faster signature generation engine with a verification procedure that is faster by a factor of approximately 20-40 as compared to the matrix variant.

6.1. GPV Signature Scheme

This section is devoted to a description of the GPV signature scheme relying on preimage sampleable trapdoor functions. In fact, the GPV signature scheme has been introduced in [GPV08] inspired by a classical counterpart due to [BR93, BR96, Cor00]. It is often also referred to as (probabilistic) full domain hash schemes. We therefore start by a brief overview of trapdoor functions and the main building blocks prior to presenting a description of the GPV signature scheme and the related algorithms.

6.1.1. Trapdoor Functions

In the following, we recall some basic definitions and properties of trapdoor functions [GPV08], that are required in several parts of this work. Later, we will particularly focus on collision-resistant preimage sampleable trapdoor functions that allow any signer knowing the trapdoor to create signatures in full domain hash schemes such as the GPV signature scheme. According to [AP09, GPV08, MP12, Pei10] there exists a polynomial-time algorithm TrapGen that on input the security parameter

1^n outputs public key $\mathbf{A}$ and the corresponding trapdoor $\mathbf{T}$ such that the trapdoor function $f_{\mathbf{A}} : B_n \longrightarrow R_n$ can efficiently be evaluated and satisfies the following properties:

1. The output distribution of $f_{\mathbf{A}}(x)$ is uniform at random over R_n given x is sampled from the domain B_n according to $\mathsf{SampleDom}(1^n)$, e.g., $\mathcal{D}_{\mathbb{Z}^n,s}$ with $s = \omega(\sqrt{\log n})$ following [GPV08, Lemma 5.2].

2. Anyone knowing the trapdoor can efficiently sample preimages $x \longleftarrow \mathsf{SamplePre}(\mathbf{T}, y)$ for a given syndrome $y \in R_n$ such that $f_{\mathbf{A}}(x) = y$, where x is distributed as $\mathsf{SampleDom}(1^n)$. By the one-way property the probability to find a preimage $x \in f_{\mathbf{A}}^{-1}(y) \subseteq B_n$ of a uniform syndrome $y \in R_n$ without the knowledge of the trapdoor is negligible.

3. The conditional min-entropy property of $\mathsf{SampleDom}(1^n)$ for a given syndrome $y \in R_n$ implies that two preimages x', x distributed as $\mathsf{SampleDom}(1^n)$ differ with overwhelming probability. This is due to the large conditional min-entropy of at least $\omega(\log n)$.

4. The preimage sampleable trapdoor functions are collision resistant, meaning that it is infeasible to find $x_1, x_2 \in B_n$ such that $f_{\mathbf{A}}(x_1) = f_{\mathbf{A}}(x_2)$ and $x_1 \neq x_2$.

Due to the results of [Ajt99, GPV08] a lot of research has been made on the construction of preimage sampleable trapdoor functions in recent years resulting in a sequence of improving works [AP09, GPV08, MP12, Pei10]. These constructions often satisfy these properties only statistically, meaning that the statistical distance between the claimed distributions and the provided ones are negligible. As a result the security proofs of cryptographic schemes involving concrete constructions hold only statistically, which is sufficient for practice.

6.1.2. Full-Domain Hash Scheme

The GPV signature scheme is a representative of the full domain hash scheme, which is secure in the random oracle model and exploits the properties of collision-resistant trapdoor functions. Furthermore, it is stateful, meaning that it does not generate new signatures for messages already signed. This can be attributed to the fact that a potential attacker could otherwise use two signatures of the same message in order to construct an element of the kernel, which solves $SIS_{q,n,\beta}$ and hence allows the attacker to provide a second preimage of any message. One can remove the need for storing message signature pairs by employing the probabilistic approach [GPV08], where the signer samples an extra random seed, which is appended to the message when calling $H(\cdot)$.

The GPV signature scheme consists mainly of sampling a preimage from a hash function endowed with a trapdoor. It is composed by 3 algorithms $\mathcal{S} = (\mathsf{KeyGenGPV}, \mathsf{SignGPV}, \mathsf{VerifyGPV})$ described as follows:

$\mathsf{KeyGenGPV}(1^n)$ On input 1^n the algorithm $\mathsf{TrapGen}(1^n)$ outputs a key pair $(\mathbf{A}, \mathbf{T})$, where $\mathsf{sk} := \mathbf{T}$ denotes the secret key (or trapdoor) and $\mathsf{pk} := \mathbf{A}$ represents the public key describing the preimage sampleable trapdoor function $f_{\mathbf{A}}$.

$\mathsf{SignGPV}(\mathbf{T}, \mathsf{msg})$ The signing algorithm computes the hash value $H(\mathsf{msg})$ of the message msg and looks up $H(\mathsf{msg})$ in its table, where $H(\cdot)$ is modeled as a random oracle. If it finds an entry, it outputs σ_m. Otherwise, it samples a preimage $\mathbf{z} \leftarrow \mathsf{SamplePre}(\mathbf{T}, H(\mathsf{msg}))$ of $H(\mathsf{msg})$ and outputs $\mathbf{z}$ as the signature.

$\mathsf{VerifyGPV}(\mathbf{z}, \mathsf{msg})$ The verification algorithm checks the satisfaction of $H(\mathsf{msg}) = f_{\mathbf{A}}(\mathbf{z})$ and $\mathbf{z} \in B_m$. If both conditions are valid, it outputs 1, otherwise 0.

6.1.3. Probabilistic Full-Domain Hash Scheme

The probabilistic approach additionally requires the signer to generate a random seed r (e.g., $r \in \{0,1\}^n$) which is appended to the message msg. Doing this, we can sign the same message msg several times, since the r-part always differs except with negligible probability. Thus, we can consider $\mathsf{msg}||r$ as the extended message to be signed.

6.2. Instantiation of the GPV Signature Scheme

In the following section we give a description of the GPV signature scheme instantiated with a new trapdoor notion due to Micciancio and Peikert [MP12], which has been shown to improve all relevant bounds of the previous proposals [Ajt99, AP09]. We, therefore, start with the matrix variant following [MP12, GPV08]. The security of this construction is based on the hardness of ℓ_2-SIS.

6.2.1. Trapdoors for the Matrix Setting

Similar to the constructions of [Ajt99, AP09], the authors of [MP12] start with a uniform random matrix $\bar{\mathbf{A}} \in \mathbb{Z}^{n \times \bar{m}}$ and extend it to a matrix $\mathbf{A} = [\bar{\mathbf{A}} | \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}] \in \mathbb{Z}^{n \times m}$ via deterministic transformations. The main idea behind this proposal is to use a primitive matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times \omega}$ for $\omega = n \cdot k$ and $k = \lceil \log q \rceil$, which has the property of generating $\mathbb{Z}_q^n$ and for which one can easily sample preimages. Due to the nice structure of this matrix one can find a basis $\mathbf{S} \in \mathbb{Z}^{\omega \times \omega}$ satisfying the congruence relation $\mathbf{G} \cdot \mathbf{S} \equiv \mathbf{0} \mod q$.

Below we provide the main algorithms of the GPV signature scheme in conjunction with the trapdoor construction from [MP12].

Basic GPV Signature Scheme

$\mathsf{KeyGenGPV}(1^n) \to (\mathbf{A}, \mathbf{R})$:

Sample $\bar{\mathbf{A}} \leftarrow_R \mathbb{Z}_q^{n \times \bar{m}}$ and $\mathbf{R} \leftarrow_R \mathcal{D}$ such that $\mathbf{R} \in \mathbb{Z}^{\bar{m} \times \lceil \log_2(q) \rceil \cdot n}$ and D is a distribution which depends on the instantiation (typically $\mathcal{D}_{\mathbb{Z}^{\bar{m} \times \lceil \log_2(q) \rceil \cdot n}, \alpha q}$) [MP12]. Output the signing key $\mathbf{R}$ and the verification key $\mathbf{A} = [\bar{\mathbf{A}} | \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}] \in \mathbb{Z}_q^{n \times m}$ where $\mathbf{G}$ is a primitive matrix.

$\mathsf{SignGPV}(\mathsf{msg}, \mathbf{R}) \to \mathbf{z} \in \mathbb{Z}^m$:

Compute the syndrome $\mathbf{u} = H(\mathsf{msg})$, sample $\mathbf{p} \leftarrow_R \mathcal{D}_{\mathbb{Z}^m, \sqrt{\mathbf{\Sigma_p}}}$, and determine the perturbed syndrome $\mathbf{v} = \mathbf{u} - \mathbf{A} \cdot \mathbf{p}$. Then sample $\mathbf{x} \leftarrow_R \mathcal{D}_{\Lambda^\perp_{\mathbf{v}}(\mathbf{G}), r}$ with $r \geq 2 \cdot \sqrt{\ln(2n(1 + \frac{1}{\epsilon}))/\pi}$. Compute $\mathbf{z} = \mathbf{p} + \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} \mathbf{x}$ and output the signature $\mathbf{z}$.

$\mathsf{VerifyGPV}(\mathsf{msg}, \mathbf{z}, (H, \mathbf{A})) \to \{0, 1\}$:

Check whether $\mathbf{A} \cdot \mathbf{z} \equiv H(\mathsf{msg})$ and $\|\mathbf{z}\|_2 \leq s\sqrt{m}$. If so, output 1 (accept), otherwise 0 (reject).

Figure 6.1.: Basic GPV signature scheme

Throughout this chapter fix the modulus q to be 2^k for some $k \in \mathbb{N}$ and use the primitive matrix $\mathbf{G}$ as defined in [MP12]. To this end, we start defining the primitive vector $\mathbf{g}^T := (1, 2, 4, \ldots, 2^{k-1}) \in \mathbb{Z}_q^k$ since $\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g}^T$. Due to its simple structure one can find an associated basis $\mathbf{S}_k$ for the lattice $\Lambda_q^\perp(\mathbf{g}^T)$ which has the following shape

$$\mathbf{S_k} = \begin{bmatrix} 2 & & & 0 \\ -1 & 2 & & \\ & \ddots & \ddots & \\ 0 & & -1 & 2 \end{bmatrix} \in \mathbb{Z}_q^{k \times k}.$$

By means of the vector $\mathbf{g}^T$ and the associated basis $\mathbf{S_k}$ one can easily create $\mathbf{S} \in \mathbb{Z}_q^{nk \times nk}$ and the primitive matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times nk}$, respectively:

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}^T & & 0 \\ & \ddots & \\ 0 & & \mathbf{g}^T \end{bmatrix}, \qquad \mathbf{S} = \begin{bmatrix} \mathbf{S_k} & & 0 \\ & \ddots & \\ 0 & & \mathbf{S_k} \end{bmatrix}.$$

An optimal bound for the smoothing parameter of the lattice $\Lambda_q^\perp(\mathbf{g}^T)$ is easily obtained using the orthogonalized basis $\tilde{\mathbf{S}}_\mathbf{k} = 2 \cdot \mathbf{I}_k$. Since $\|\tilde{\mathbf{S}}\| = \|\tilde{\mathbf{S}}_\mathbf{k}\| = 2$, we have $\eta_\epsilon(\Lambda_q^\perp(\mathbf{G})) \leq r = 2 \cdot \sqrt{\ln\left(2n\left(1 + \frac{1}{\epsilon}\right)\right)/\pi}$ according to [GPV08, Theorem 3.1].

Using this parameter we can bound the preimage length. Due to the bound [Ban93, Lemma 1.5] the probability of the preimage to be greater or equal to $r \cdot \sqrt{n \cdot k}$ is bounded by $2^{-n \cdot k} \cdot \frac{1+\epsilon}{1-\epsilon}$.

Sampling Algorithms for Preimages and Perturbations

In what follows we describe the preimage sampling algorithm for a syndrome $t \in \mathbb{Z}_q$ from the coset $\Lambda_t^{\perp}(\mathbf{g}^{\top}) = \{\mathbf{x} \mid \mathbf{g}^{\top} \cdot \mathbf{x} \equiv t \mod q\}$ using the randomized nearest plane algorithm [MP12]. Due to the nice properties of the orthogonalized basis, the algorithm reduces to a few steps with $a_0 = t$.

Algorithm 4: Sampling from $\Lambda_q^{\perp}(\mathbf{G})$

```
  Data: $a_0 \in \mathbb{Z}_q, k > 0, q, r$
1 for $i = 0 \to k-1$ do
2     $v_i \leftarrow \mathcal{D}_{2\mathbb{Z}+a_i,r}$
3     $a_{i+1} = (a_i - v_i)/2$
4 end
5 Output: $\mathbf{v} = (v_0, \ldots, v_{k-1})^T$
```

The resulting vector $\mathbf{v} \in \Lambda_t^{\perp}(\mathbf{g}^{\top})$ is distributed as $\mathcal{D}_{\Lambda_t^{\perp}(\mathbf{g}^{\top}),r}$. Similarly one can sample preimages from $\Lambda_{\mathbf{u}}^{\perp}(\mathbf{G})$ for a syndrome vector $\mathbf{u} \in \mathbb{Z}_q^n$ by independently running n instances of the algorithm on each component of $\mathbf{u}$.
The authors provide two different types of instantiations for the trapdoor generation algorithm, namely the statistical and computational instantiation. We prefer to focus on the latter one since it allows for a public key with a smaller number of columns, hence, leading to a more efficient implementation as compared to a statistical instantiation. Therefore, we will always refer to the computational instantiation in the rest of this chapter. Such a representation can easily be achieved by generating a uniform random matrix $\bar{\mathbf{A}} \in \mathbb{Z}^{n \times n}$ and sampling a trapdoor $\mathbf{R} = \begin{bmatrix} \mathbf{R_1} \\ \mathbf{R_2} \end{bmatrix}$ from the discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z}^{2n \times nk},\alpha q}$ where $\alpha \in \mathbb{R}_{>0}$ satisfies $\alpha q > \sqrt{n}$. The resulting matrix $\mathbf{A} = [\mathbf{I}_n \mid \bar{\mathbf{A}} \mid \mathbf{G} - (\tilde{\mathbf{A}}\mathbf{R_2} + \mathbf{R_1})]$ is an instance of decision $\mathsf{LWE}_{n,m,\alpha q}$ and hence pseudorandom when ignoring the identity submatrix.
Applying the framework of [GPV08] requires to sample a spherically distributed preimage for a given syndrome $\mathbf{u} \in \mathbb{Z}_q^n$ using Gaussian sampling algorithms and the trapdoor $\mathbf{R}$. In fact, the spherical distribution is a common tool to make the distribution of the signature independent from the secret key. The Gaussian sampling algorithm mainly consists of two parts. The first part involves the trapdoor $\mathbf{R}$ which is used to transform a sample $\mathbf{x}$ from the set $\Lambda_{\mathbf{u}}^{\perp}(\mathbf{G})$ with parameter $r \geq \|\tilde{\mathbf{S}}\| \cdot \sqrt{\ln(2n(1+\frac{1}{\epsilon}))/\pi}$ to a sample $\mathbf{y} = \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} \cdot \mathbf{x}$ of the set $\Lambda_{\mathbf{u}}^{\perp}(\mathbf{A})$. Due to the fact that $\begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix}$ is not a square matrix and the non-spherical covariance

$\mathbf{COV} = r^2 \begin{bmatrix}\mathbf{R}\\ \mathbf{I}\end{bmatrix} [\mathbf{R}^T \ \mathbf{I}]$ is not of full-rank, the distribution of $\mathbf{y}$ is skewed and hence leaks information about the trapdoor. An attacker could collect samples and reconstruct the covariance matrix. Therefore, we need the second part to correct this flaw. This can be done by adding perturbations from a properly chosen distribution. Using the convolution technique from [Pei10], we can choose a parameter s in such a way that s^2 is slightly larger than the largest absolute eigenvalue of the covariance $\mathbf{COV}$ and generate Gaussian pertubations $\mathbf{p} \in \mathbb{Z}^m$ having covariance $\mathbf{\Sigma_p} = s^2\mathbf{I} - \mathbf{COV}$. In order to obtain a vector $\mathbf{z}$ that is from a spherical Gaussian distribution with parameter s, one samples a preimage $\mathbf{y}$ for an adjusted syndrome $\mathbf{v} = \mathbf{u} - \mathbf{Ap}$ from $\Lambda_{\mathbf{v}}^{\perp}(\mathbf{A})$. The vector $\mathbf{z} = \mathbf{p} + \mathbf{y}$ provides then a spherical distributed sample satisfying $\mathbf{Az} \equiv \mathbf{u} \mod q$.

6.2.2. Trapdoors for Ideal-Lattices

Based on [MP12] we present a new trapdoor construction for ideal lattices [P9] such that the elements of the primitive vector $\mathbf{g}^\top$ are considered ring elements of $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ for $\mathcal{R} = \mathbb{Z}[X]/\phi_m(X)$ (rather than vectors of $\mathbb{Z}_q^k$), where $\phi_m(X)$ denotes the m-th cyclotomic polynomial. Our construction exploits different properties of the underlying ring such as avoiding the need for a polynomial matrix $\hat{\mathbf{G}}$ as required in the matrix setting. This results in a more efficient instantiation.

Trapdoors for Computationally Instantiated Public Key

The public key is generated by drawing k samples $(\bar{\mathbf{a}}_i, \bar{\mathbf{a}}_i\mathbf{r}_i + \mathbf{e}_i)$ from the ring-LWE distribution. By this, we obtain a public key that is pseudorandom and enjoys the hardness of ring-LWE. Following [ACPS09] one can use the error distribution in order to sample the trapdoor polynomials $\hat{\mathbf{r}} \in \mathcal{R}^k$ and $\hat{\mathbf{e}} \in \mathcal{R}^k$. This does not incur any security flaws. Indeed, this property is essential for the signature scheme to work due to the need for smaller secret keys. As in the matrix variant one can use only one uniformly distributed sample $\bar{\mathbf{a}}_1$ rather than a set in $\mathbf{A}$. By a standard hybrid argument the hardness of distinguishing $\bar{\mathbf{a}}_1\mathbf{r_i} + \mathbf{e_i}$ from uniformly distributed samples can be reduced to decision ring-LWE [BPR12]. Thus, we obtain a public key of the following shape:

$$\mathbf{A} = [\mathbf{1},\ \bar{\mathbf{a}}_1,\ \mathbf{g}_1 - (\bar{\mathbf{a}}_1\mathbf{r}_1 + \mathbf{e}_1),\ \ldots\ ,\ \mathbf{g}_k - (\bar{\mathbf{a}}_1\mathbf{r}_k + \mathbf{e}_k)]$$

Similar to the matrix version $\hat{\mathbf{g}}^\top = [1, \cdots, 2^{k-1}]$ defines the primitive vector of polynomials where each component is considered as a constant polynomial. Sampling from $\Lambda_{\mathbf{u}}^{\perp}(\hat{\mathbf{g}}^\top) = \{\hat{\mathbf{x}} \in R_q^k \mid \mathbf{g}_1\mathbf{x}_1 + \cdots + \mathbf{g}_k\mathbf{x}_k = \mathbf{u}\ \}$ is performed as in the matrix case with $\mathbf{y}^\top = [\mathbf{x}_1^{(0)}, \ldots, \mathbf{x}_k^{(0)}, \ldots, \mathbf{x}_1^{(n)}, \ldots, \mathbf{x}_k^{(n)}]$ satisfying $\mathbf{Gy} \equiv \mathbf{u} \mod q$, where $\mathbf{x}_j^{(i)}$ is the i-th coefficient of the j-th polynomial. The resulting vector $\mathbf{y}$ is from a spherical Gaussian distribution having covariance matrix $r^2\mathbf{I}$. Sampling a preimage for a syndrome $\mathbf{u} \in \mathcal{R}_q$ requires to sample polynomials $\hat{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_k)$ from $\Lambda_{\mathbf{u}}^{\perp}(\hat{\mathbf{g}}^\top)$. These are then used to construct the preimage $\hat{\mathbf{z}} = [\sum_{i=1}^{k} \mathbf{e}_i\mathbf{x}_i,\ \sum_{i=1}^{k} \mathbf{r}_i\mathbf{x}_i,\ \mathbf{x}_1,\ \ldots\ ,\ \mathbf{x}_k] \in \mathcal{R}_q^{k+2}$. It can easily be verified, that $\mathbf{A}\hat{\mathbf{z}} \equiv \mathbf{u}$ holds. With the same arguments as in the matrix case we need to add some perturbation to transform the skewed distribution into a spherical one. Since we mainly operate on rings modulo $X^n + 1$ with n a power of two, multiplication of polynomials $\mathbf{r}_i\mathbf{x}_i$ corresponds to matrix multiplication $\mathbf{Rot}(\mathbf{r}_i)\mathbf{x}_i$. The matrix $\mathbf{Rot}(\mathbf{r}_i)$ consists of n columns $[\mathbf{r}_i, \mathrm{rot}(\mathbf{r}_i), \cdots, \mathrm{rot}^{n-1}(\mathbf{r}_i)]$ with $\mathrm{rot}(\mathbf{y}) = [-y_{n-1}, y_0,\ \ldots\ , y_{n-2}]$ defining the rotation in anti-cyclic integer lattices. Formally, the ring specific rotations are obtained via $\mathrm{rot}^j(\mathbf{y}) = \mathbf{y} \cdot X^j$. The usage of other irreducible polynomials is also possible, but have the drawback of larger expansion factors which imply increased preimage lengths. The covariance matrix of the preimage has the following shape:

$$\mathbf{COV} = r^2 \begin{bmatrix} \mathbf{R_1} \\ \mathbf{R_2} \\ \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{R_1}^\top & \mathbf{R_2}^\top & \mathbf{I} \end{bmatrix}$$

with $\mathbf{R_1} = [\mathbf{Rot}(\mathbf{e}_1) \mid \ldots \mid \mathbf{Rot}(\mathbf{e}_k)]$ and $\mathbf{R_2} = [\mathbf{Rot}(\mathbf{r}_1) \mid \ldots \mid \mathbf{Rot}(\mathbf{r}_k)]$, respectively. One observes, that the computation of this matrix is very simple since matrix multiplication corresponds to polynomial multiplication with $\beta(\mathbf{x}) = [x_1, -x_n, -x_{n-1}, \ldots, -x_2]$ which is the first row of $\mathbf{Rot}(\mathbf{x})$:

$$\mathbf{COV} = r^2 \begin{bmatrix} \mathbf{Rot}(\sum_{i=1}^{k} \mathbf{e}_i\beta(\mathbf{e}_i)) & \mathbf{Rot}(\sum_{i=1}^{k} \mathbf{e}_i\beta(\mathbf{r}_i)) & \mathbf{R_1} \\ \mathbf{Rot}(\sum_{i=1}^{k} \mathbf{r}_i\beta(\mathbf{e}_i)) & \mathbf{Rot}(\sum_{i=1}^{k} \mathbf{r}_i\beta(\mathbf{r}_i)) & \mathbf{R_2} \\ \mathbf{R_1}^\top & \mathbf{R_2}^\top & \mathbf{I} \end{bmatrix}.$$

Now one can use the techniques from the previous section in order to generate perturbations that are viewed as vectors in $\mathbb{Z}^{n(k+2)}$. A perturbation vector $\mathbf{p} \in \mathbb{Z}^{n(k+2)}$ is then split into $k+2$ parts of length n. Each part corresponds to a perturbation polynomial $\mathbf{p}_i \in \mathcal{R}$. In order to provide a preimage for a syndrome polynomial $\mathbf{u}$ one samples perturbations $\mathbf{p}_1, \ldots, \mathbf{p}_{k+2} \in \mathcal{R}$ as shown before. Then we create sample polynomials $\hat{\mathbf{x}}$ from $\Lambda^\perp_{\mathbf{u}-\mathbf{Ap}}(\hat{\mathbf{g}}^\top)$. The resulting preimage $\hat{\mathbf{z}}$ is then spherically distributed:

$$\hat{\mathbf{z}} = \begin{bmatrix} \mathbf{p}_1 + \hat{\mathbf{e}} \cdot \hat{\mathbf{x}}, & \mathbf{p}_2 + \hat{\mathbf{r}} \cdot \hat{\mathbf{x}}, & \mathbf{p}_3 + \mathbf{x}_1, & \ldots\ , & \mathbf{p}_{k+2} + \mathbf{x}_k \end{bmatrix}.$$

Now, we give a short description of how to instantiate the ring-LWE problem and how to sample the secret keys $\mathbf{r}_i$ and $\mathbf{e}_i$ for $1 \le i \le k$ according to [DD12]. The authors provide different from the work [LPR10] a relatively simple ring-LWE setting avoiding the work in the dual ring $\mathcal{R}_q^\vee$ or the H-Space [LPR10] which turns out to be more convenient in certain applications. Following the paper of [BLP+13] we can take q to be a power of two as in the matrix variant. Such choices are more suitable for practice since the nice sampling algorithms introduced in the previous section are applicable. A prime number would involve costly sampling procedures which lead to a slower signature generation engine. As stated in [ACPS09] it is possible to generate both the secret key $\mathbf{r}_i$ and $\mathbf{e}_i$ from the same error distribution without affecting the security. Indeed, this property is important in order to make the trapdoor construction work based on the ring-LWE assumption. Specifically, we need small keys to provide short preimages. If one operates in the ring $\mathbb{Z}_q[X]/\langle X^n+1\rangle$ with n a power of two, the coefficients of both $\mathbf{r}_i$ and $\mathbf{e}_i$ are chosen from the Gaussian distribution on the rationals and then rounded to the nearest integers. In particular, the polynomials $\mathbf{r}_i$ and $\mathbf{e}_i$ are distributed as $\lceil \mathcal{D}_{\mathbb{Q}^n,c} \rfloor$ for $c = \sqrt{n}\alpha q(\frac{nl}{2\log(nl/2)})^{1/4}$ where l is the number of samples and $\alpha q > \omega(\sqrt{\log 2n})$. In practical applications, however, we sample the polynomials from the discrete Gaussian distribution and omit the last term [DD12] or set $l = 1$ due to the fact that the polynomials in the public key are computed from different secrets $\mathbf{r}_i$. For other choice of cyclotomic polynomials Φ_m it is possible to sample the trapdoor polynomials in extension rings according to [DD12, Theorem 2]. But a better approach is to use the framework presented in [LPR13] since it allows to work in arbitrary cyclotomic rings without incurring any ring-dependent expansion factor.

Trapdoors for Statistically Instantiated Public Key

We briefly explain another ring construction that is derived from [Mic07] and leads to a statistical instantiation of the public key. Take $k = \lceil \log_2 q \rceil$ and $\bar{m} = O(\log n)$. Then select $\bar{m}$ uniformly random polynomials $\hat{\mathbf{a}} = [\mathbf{a}_1, \ldots, \mathbf{a}_{\bar{m}}] \in \mathcal{R}_q^{\bar{m}}$. Define by $h_{\hat{\mathbf{a}}}(\hat{\mathbf{x}}) = \sum_{i=1}^{\bar{m}} \mathbf{a}_i \mathbf{x}_i$ a generalized compact knapsack. Furthermore choose k vectors $\hat{\mathbf{r}}_i$ for $1 \leq i \leq k$, each consisting of $\bar{m}$ random polynomials $\mathbf{r}_{i1}, \ \ldots \ , \ \mathbf{r}_{i\bar{m}}$ of degree $n-1$ with small coefficients. By [SSTX09, Lemma 6], which is an adapted variant of the regularity lemma of [Mic07], the function values $\mathbf{a}_{\bar{m}+i} = h_a(\hat{\mathbf{r}}_i)$ with $1 \leq i \leq k$ are essentially uniformly distributed. For instance, if Φ_m splits into n linear factors over some finite field, then the statistical distance from uniformity is bounded by

$$\epsilon \leq \frac{1}{2}\sqrt{\left(1 + \frac{q}{B^{\bar{m}}}\right)^n - 1},$$

where $\mathbf{r}_{ij}$ are uniformly distributed over a set $[-b, b]^n \cap \mathbb{Z}^n$ with $B = 2b + 1$. Thus, we can create uniform random vector of polynomials $\mathbf{A}$ endowed with the trapdoor $\hat{\mathbf{r}}_i \in \mathcal{R}^{\bar{m}}$ for $1 \leq i \leq k$:

$$\mathbf{A} = [\mathbf{a}_1, \ \ldots \ , \ \mathbf{a}_{\bar{m}}, \ \mathbf{g}_1 - \mathbf{a}_{\bar{m}+1}, \ \ldots \ , \ \mathbf{g}_k - \mathbf{a}_{\bar{m}+k}] \, .$$

To generate a preimage of a given syndrome polynomial $\mathbf{u} \in \mathcal{R}_q$, one has to sample a vector $\hat{\mathbf{x}} \in \Lambda_{\mathbf{u}}^{\perp}(\hat{\mathbf{g}}^{\top})$ using the methods from above. As one can easily verify, the vector $\hat{\mathbf{y}} = [\hat{\mathbf{r}}_1\mathbf{x}_1, \ \ldots \ , \ \hat{\mathbf{r}}_k\mathbf{x}_k \ , \ \mathbf{x}_1, \ \ldots \ , \mathbf{x}_k]$ is a preimage of the syndrome $\mathbf{u}$ for $\mathbf{A}$. Using the techniques from the descriptions before, one can produce spherically distributed samples.

6.3. Improvements and Optimizations

In our implementation we have to face several challenges that affect the performance of the signature scheme both in the matrix and ring variant. In the following sections address this research question and provide different improvement opportunities.

6.3.1. Computation of the Covariance Matrix

First, we observed that the computation of the covariance matrix **COV** is too expensive in terms of running time. Since the basis matrix **COV** is sparse, we were able to significantly reduce the computational efforts. It can be split into four parts as below. The only block to be computed is the symmetric matrix $\mathbf{RR}^T$.

$$\mathbf{COV} = r^2 \begin{bmatrix} \mathbf{RR}^T & \mathbf{R} \\ \mathbf{R}^T & \mathbf{I} \end{bmatrix}$$

In the ring variant the computation of the covariance matrix is much faster because multiplication is performed in polynomial rings as explained in Section 6.2.2. Running these parts in parallel offers another source of optimization.

6.3.2. Estimating relevant Parameters

Following [MP12], it is required to set the parameter s large enough such that it is independent from a specific trapdoor. In particular, s is chosen to be not smaller than the term $\sqrt{s_1(\mathbf{R})^2+1}\cdot\sqrt{6}\cdot a$, where $s_1(\mathbf{R})$ denotes the largest singular value of the secret key $\mathbf{R}$ and $a = \sqrt{\ln(2n(1+\frac{1}{\epsilon}))/\pi}$. The perturbation covariance matrix $\mathbf{\Sigma_p} = s^2\mathbf{I}_m - \mathbf{COV}$ is well-defined, if one selects s such that $s > s_1(\begin{bmatrix}\mathbf{R}\\ \mathbf{I}\end{bmatrix})\cdot r$ is satisfied. Since $\mathbf{R}$ is a subgaussian random variable, the matrix $\mathbf{R}$ satisfies $s_1(\mathbf{R}) \leq C\cdot(\sqrt{2n}+\sqrt{n\cdot k}+4.7)\cdot\alpha q$ except with probability $\approx 2^{-100}$ according to [MP12, Lemma 2.9]. The universal constant C is very close to $1/\sqrt{2\pi}$.

6.3.3. Generation of Perturbation Vectors

One of the main ingredients of the signature scheme is the idea of creating perturbations [MP12] in order to get spherically distributed preimages that do not carry any information about the secret key. A perturbation vector is generated by means of the distribution $\mathcal{D}_{\mathbb{Z}^m,\sqrt{\mathbf{\Sigma_p}}}$ which outputs random vectors from $\mathbb{Z}^m$ with covariance matrix $\mathbf{\Sigma_p}$. By [Pei10] this can be achieved by sampling a vector $\mathbf{p}$ according to $\lceil\sqrt{\mathbf{\Sigma_p}-a^2\mathbf{I}}\cdot\mathcal{D}_1^m\rfloor_a$, where $\mathcal{D}_1^m$ denotes the m-dimensional Gaussian distribution. Each vector sampled from $\mathcal{D}_1^m$ has entries coming from the standard continuous Gaussian distribution with parameter 1. $\lceil\cdot\rfloor_a$ denotes the randomized rounding operation from [Pei10] with parameter $a = r/2 \geq \sqrt{\ln(2n(1+\frac{1}{\epsilon}))/\pi}$, which rounds each coordinate of the vector independently to a nearby integer using the discrete

Gaussian distribution. The generation of perturbation vectors requires the square root computation $\sqrt{\mathbf{\Sigma_p} - a^2\mathbf{I}}$. Below we discuss one method of doing this and provide an improved algorithm through a better analysis.

6.3.4. Square Root Computation

The Cholesky decomposition splits any positive definite matrix $\mathbf{M}$ into the product of a lower triangular matrix and its conjugate transpose, i.e., $\mathbf{M} = \mathbf{L} \cdot \mathbf{L}^T$, and runs in time $O(m^3) = O((k+2)^3 n^3)$. If one selects $k = 19$, then the constant factor grows to 9261, which is very high compared to $n = 256$. The Cholesky decomposition is needed to generate perturbations that have covariance matrix $\mathbf{\Sigma_p}$, where $\sqrt{\mathbf{\Sigma_p}}$ is the Cholesky matrix. An algorithm for the Cholesky decomposition is given in Algorithm 5.

Algorithm 5: Cholesky Decomposition Algorithm

```
   Data: Matrix $\mathbf{L} \in \mathbb{Z}^{m \times m}$
   Result: Lower triangular part of $\mathbf{L}$
 1 for $k = 1 \to m$ do
 2     $l_{kk} = \sqrt{l_{kk}}$;
 3     for $i = k+1 \to m$ do
 4         $l_{ik} = l_{ik}/l_{kk}$;
 5     end
 6     for $j = k+1 \to m$ do
 7         for $i = j \to m$ do
 8             $l_{ij} = l_{ij} - l_{ik} l_{jk}$;
 9         end
10     end
11 end
```

When decomposing the matrix $\mathbf{\Sigma_p} - a^2\mathbf{I}$ into its roots, one can improve the running time by our modified Cholesky decomposition taking into account the $n^2k^2 - n \cdot k$ zero entries, meaning that one can skip line 8 in Algorithm 5 whenever l_{ik} or l_{jk} is known to be zero. Due to the sparsity of $\mathbf{\Sigma_p} - a^2\mathbf{I}$ this occurs very often. We call this optimized algorithm variant 1.

Although this optimization in variant 1 noticeably improves the timings of key generation, the algorithm is still inefficient and is the main source of slow key generation. Moreover, the resulting perturbation matrix is dense and has no structure, which leads to high memory claims in order to store the matrix of floating point entries and to worse running times for signature generation. This is due to the fact that each generation of a perturbation vector requires to multiply a huge triangular matrix consisting of multi-precision floating point entries with a floating point vector. To circumvent this problem we applied a pivoting strategy followed by the Block

Cholesky decomposition, meaning that we permute the covariance matrix such that $\mathbf{P\Sigma_p P}^\top = \mathbf{\Sigma'_p}$.

This corresponds to left multiplication of the permutation matrix $\mathbf{P} = \begin{bmatrix} 0 & \mathbf{I}_{nk} \\ \mathbf{I}_{2n} & 0 \end{bmatrix}$ to the public key $\mathbf{A}$. It is obvious that this transformation does not cause any security flaws because it is a simple reordering. The advantage of using $\mathbf{P}$ is a perturbation covariance matrix $\mathbf{\Sigma'_p}$ with a nice structure which enables us to work with Schur complements [Zha10] in a very efficient way:

$$\mathbf{\Sigma'_p} = s^2\mathbf{I}_m - r^2 \begin{bmatrix} \mathbf{I}_{nk} & \mathbf{R}^\top \\ \mathbf{R} & \mathbf{RR}^\top \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{I}_{nk} \\ \mathbf{I}_{2n} & 0 \end{bmatrix} \mathbf{\Sigma_p} \begin{bmatrix} 0 & \mathbf{I}_{nk} \\ \mathbf{I}_{2n} & 0 \end{bmatrix}^\top .$$

Therefore we get an algorithm which outperforms the optimized Cholesky decomposition applied on the non-permuted matrix by a factor of 30-190. Furthermore, we obtain a signature generation engine which yields a factor improvement of 2-6 in the ring variant. This is due to the sparse matrix and its nice structure. In both the key and signature generation steps the factor grows as n increases. In general, the Schur complement is defined as follows.

Lemma 6.1. *Let the matrix* $\mathbf{S_i} = \left[\begin{array}{c|c} b_i & \mathbf{h}_i^\top \\ \hline \mathbf{h}_i & \mathbf{C}_i \end{array}\right] \in \mathbb{R}^{m-i \times m-i}$ *be symmetric positive definite with* $b_i > 0$*. Then, the Schur complement given by*

$$\mathbf{S}_{i+1} := \mathbf{S}_i - \frac{1}{b_i}\mathbf{h}_i\mathbf{h}_i^\top \in \mathbb{R}^{m-i-1 \times m-i-1}$$

is well-defined and also symmetric positive definite.

This decomposition is successively applied on the submatrices $\mathbf{S_i} \in \mathbb{R}^{m-i \times m-i}$. Doing this, one obtains an efficient method to construct the columns of the matrix $\sqrt{\mathbf{\Sigma'_p} - a^2\mathbf{I}}$. The first nk colums $\frac{1}{\sqrt{b}} \cdot \begin{bmatrix} b \cdot \mathbf{I} \\ r^2\mathbf{R} \end{bmatrix} \in \mathbb{R}^{m \times nk}$ for $b = s^2 - r^2 - a^2 = s^2 - 5a^2$ involve only a simple scaling operation. Therefore, we need no additional memory in order to store these columns. Due to the sparse columns multiplication involves only the non-zero columns $(\mathbf{R})_i$ of the matrix $\mathbf{R} = \begin{bmatrix} \mathbf{R_1} \\ \mathbf{R_2} \end{bmatrix}$. Thus, transformations are focused only on the $(2n \times 2n)$ matrix:

$$\begin{aligned} \mathbf{S}_{nk} &= (s^2 - a^2)\mathbf{I} - r^2\mathbf{RR}^\top - \frac{r^4}{b}\sum_{i=1}^{nk}(\mathbf{R})_i(\mathbf{R})_i^\top && (6.1) \\ &= (s^2 - a^2)\mathbf{I} - (r^2 + \frac{r^4}{b})\mathbf{RR}^\top \in \mathbb{R}^{2n \times 2n} . && (6.2) \end{aligned}$$

The last sum of vector products reduces to the simple scaling operation $\frac{r^4}{b}\mathbf{RR}^\top$. Thus, one can save the costly vector product computations. When continuing the decomposition on the remaining matrix $\mathbf{S}_{nk}$ one obtains the decomposition matrix.

One can easily verify that

$$\mathbf{X}\mathbf{X}^\top = \mathbf{\Sigma}'_{\mathbf{p}} - a^2\mathbf{I}, \;\; \mathbf{X} = \begin{bmatrix} \sqrt{b}\mathbf{I}_{\mathbf{nk}} & 0 \\ \frac{r^2\mathbf{R}}{\sqrt{b}} & \mathbf{L} \end{bmatrix}$$

holds. Consequently one needs only to store $n(2n+1)$ floating point entries of the last part $\mathbf{L} = \mathsf{decomp}(\mathbf{S}_{\mathbf{nk}})$ instead of $m(m+1)/2$ in the case without any modifications. This induces an improvement factor that is $m(m+1)/2n(2n+1) \approx 240$ for $n = 512$ and $k = 29$. A nice side effect of this transformation is a much faster algorithm for generating perturbations since the number of operations drastically decreases as the factor grows. In the matrix version, one makes use of the sparse decomposition matrix. In particular $\sqrt{\mathbf{\Sigma}_{\mathbf{p}} - a^2\mathbf{I}} \cdot \mathcal{D}_1^m$ is reduced to the simple scaling operation of $\sqrt{b} \cdot \mathbf{d}_1$ and the matrix vector multiplication $\begin{bmatrix} \frac{r^2}{\sqrt{b}}\mathbf{R} & \big| & \mathbf{L} \end{bmatrix} \cdot \mathbf{d}$ for $\mathbf{d} = (\mathbf{d}_1, \mathbf{d}_2) \in \mathbb{R}^{\bar{m}+nk}$ and $\mathbf{d} \leftarrow_R \mathcal{D}_1^m$. Especially, in the ring version we preserve the nice properties of polynomial multiplication and therefore use only the scaled set of trapdoor polynomials $\frac{r^2}{\sqrt{b}}\mathbf{e}_i, \frac{r^2}{\sqrt{b}}\mathbf{r}_i$, and the lower triangular matrix $\mathbf{L} = \begin{bmatrix} \mathbf{L_1} \\ \mathbf{L_2} \end{bmatrix}$ in order to generate perturbations. Specifically, one obtains the perturbation vector $\tilde{\mathbf{p}} = [\tilde{\mathbf{p}}_\mathbf{1}, \ldots, \tilde{\mathbf{p}}_{k+2}] \in \mathcal{T}^{k+2}$ for $\mathcal{T} = \mathbb{R}[X]/\langle X^n + 1\rangle$, where

$$\tilde{\mathbf{p}}_1 = \frac{r^2}{\sqrt{b}} \sum_{i=1}^{k} \mathbf{e}_i \cdot \mathbf{d}_i + \mathbf{L_1} \cdot \begin{bmatrix} \mathbf{d}_{k+1} \\ \mathbf{0} \end{bmatrix} \tag{6.3}$$

$$\tilde{\mathbf{p}}_2 = \frac{r^2}{\sqrt{b}} \sum_{i=1}^{k} \mathbf{r}_i \cdot \mathbf{d}_i + \mathbf{L_2} \cdot \begin{bmatrix} \mathbf{d}_{k+1} \\ \mathbf{d}_{k+2} \end{bmatrix} \tag{6.4}$$

$$\tilde{\mathbf{p}}_i = \sqrt{b} \cdot \mathbf{d}_{i-2} \text{ for } 3 \le i \le k+2\,. \tag{6.5}$$

The polynomials in $\tilde{\mathbf{p}}$ are ordered in such a way that we do not need to change the order of the polynomials in the public key. In fact, the matrix $\mathbf{X}$ is only permuted back after decomposition by changing the rows. The elements $\mathbf{d}_i \leftarrow \mathcal{D}_1^n$ are sampled from the continuous Gaussian distribution with parameter 1 or equivalently normally distributed with standard deviation $1/\sqrt{2\pi}$. Thus, we get a fast signature generation algorithm which is about five times faster than its matrix analogue. It is also worth to mention that these operations can also be executed in parallel.

6.3.5. Optimized Signature Scheme

In this section we give a description of the optimized signature scheme when instantiated computationally, meaning that the matrix $\mathbf{A}$ is an instance of the LWE distribution and therefore pseudorandom when ignoring the identity submatrix. Considering the GPV-signature scheme it is eminently suitable to instantiate the trapdoor construction computationally such that the public key $\mathbf{A}$ is educible by fewer columns as compared to a statistically instantiated public key. We first start with the matrix variant and subsequently present the corresponding ring variant.

Matrix Variant

The following figure contains a description of the GPV signature scheme in case we consider unstructured matrices. In fact, it represents an adapted variant of the basic signature scheme from Section 6.2.1, where the signing step is modified for the optimizations described in Section 6.3.

GPV Signature Scheme - Matrix Variant

$\mathsf{KeyGenGPV}(1^n)$: Sample $\bar{\mathbf{A}} \leftarrow_R \mathbb{Z}_q^{n \times n}$, and each entry of the secret key $\mathbf{R} \in \mathbb{Z}^{2n \times n \cdot k}$ from $\mathcal{D}_{\mathbb{Z},\alpha q}$, where $q = 2^k$ and $\alpha q \geq 2\sqrt{n}$. Output the signing key $\mathbf{R}$, the verification key $\mathbf{A} = [\ \mathbf{I}_n \mid \bar{\mathbf{A}} \mid \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}]$ and parameter s such that $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ with $m = 2n + n \cdot k$ and $\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g}^\top \in \mathbb{Z}^{n \times n \cdot k}$ is the primitive matrix consisting of n copies of the vector $\mathbf{g}^\top = [1, 2, \ldots, 2^{k-1}]$.

$\mathsf{SignGPV}(\mathsf{msg}, \mathbf{R}) \rightarrow \mathbf{z} \in \mathbb{Z}^m$:

1. ***Syndrome***. Compute the syndrome $\mathbf{u} = H(\mathsf{msg})$, sample $\mathbf{p} \leftarrow_R \mathcal{D}_{\mathbb{Z}^m, \sqrt{\mathbf{\Sigma_p}}}$ with $\sqrt{\mathbf{\Sigma_p} - a^2\mathbf{I}} = \begin{bmatrix} \frac{r^2}{\sqrt{b}}\mathbf{R} & \mathbf{L} \\ \sqrt{b}\mathbf{I_{nk}} & 0 \end{bmatrix}$ (see Section 6.3.3).
2. ***Perturbation***. Sample $\mathbf{d}_1 \leftarrow_R \mathcal{D}_1^{nk}$ and $\mathbf{d}_2 \leftarrow_R \mathcal{D}_1^{2n}$, where $\mathcal{D}_1$ is the continuous Gaussian distribution with parameter 1. Compute $\tilde{\mathbf{p}} = (\tilde{\mathbf{p}}_\mathbf{1}, \tilde{\mathbf{p}}_2)$, where $\tilde{\mathbf{p}}_1 = \frac{r^2}{\sqrt{b}}\mathbf{R}\mathbf{d}_1 + \mathbf{L}\mathbf{d}_2$ and $\tilde{\mathbf{p}}_2 = \sqrt{b} \cdot \mathbf{d}_1$. Sample perturbation $\mathbf{p} = (\mathbf{p_1}, \mathbf{p_2}) = \tilde{\mathbf{p}} + \mathcal{D}_{\mathbb{Z}^{n \cdot (2+k)} - \tilde{\mathbf{p}}, a}$ with $\mathbf{p}_1 = \tilde{\mathbf{p}}_\mathbf{1} + \mathcal{D}_{\mathbb{Z}^{2n} - \tilde{\mathbf{p}}_\mathbf{1}, a}$, $\mathbf{p}_2 = \tilde{\mathbf{p}}_\mathbf{2} + \mathcal{D}_{\mathbb{Z}^{n \cdot k} - \tilde{\mathbf{p}}_\mathbf{2}, a}$ for $\eta_\epsilon(\mathbb{Z}) \leq a$ and $a = \sqrt{\ln\left(2n\left(1 + \frac{1}{\epsilon}\right)\right)/\pi}$.
3. ***Signing***. Determine adjusted syndrome $\mathbf{v} = \mathbf{u} - \mathbf{A}\mathbf{p} \in \mathbb{Z}_q^n$. Sample vector $\mathbf{x} \leftarrow \mathcal{D}_{\Lambda_\mathbf{v}^\perp(\mathbf{G}), r}$ with $r = 2a$. Output signature $\mathbf{z} \leftarrow \mathbf{p} + \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} \mathbf{x}$.

$\mathsf{VerifyGPV}(\mathsf{msg}, \mathbf{z}, (H, \mathbf{A})) \rightarrow \{0, 1\}$:
Check whether $\mathbf{A} \cdot \mathbf{z} \equiv H(\mathsf{msg})$ and $\|\mathbf{z}\|_2 \leq s\sqrt{m}$ is satisfied. If so, output 1 (accept), otherwise 0 (reject).

Figure 6.2.: Matrix variant of the GPV signature scheme.

Ring Variant

As for the ring variant, we let, for instance, $\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$ be a cyclotomic ring with $q = 2^k$ and n a power of 2. The values for r, k and a are exactly as in the matrix case. Here, the public key is built following the trapdoor construction presented in Section 6.2.2. Similar to the matrix variant, the signing step is subject to the optimizatios given in Section 6.3.

GPV Signature Scheme - Ring Variant

$\mathsf{KeyGenGPV}(1^n)$: Sample $\bar{\mathbf{a}} \leftarrow_R \mathcal{R}_q$, and each entry of the secret key $\mathbf{r}_i, \mathbf{e}_i \in \mathcal{R} = \mathbb{Z}[X]/\langle X^n+1\rangle$ from $\mathcal{D}_{\mathbb{Z},\alpha q}$ viewing ring elements as vectors in $\mathbb{Z}^n$ for $\alpha q \geq 2\sqrt{n}$. Output the signing key $\hat{\mathbf{r}} = [\mathbf{r}_1 \ldots, \mathbf{r}_k]$, the verification key $\mathbf{A} = [\mathbf{1},\ \bar{\mathbf{a}}_1,\ \mathbf{g}_1 - (\bar{\mathbf{a}}_1\mathbf{r}_1 + \mathbf{e}_1),\ \ldots\ ,\ \mathbf{g}_k - (\bar{\mathbf{a}}_1\mathbf{r}_k + \mathbf{e}_k)]$, and parameter s such that $\mathbf{A} \in \mathcal{R}_q^{k+2}$ with $\mathbf{g}_i = 2^{i-1}$ being primitive polynomials consisting of the constant coefficient 2^{i-1} and zeros else.

$\mathsf{SignGPV}(\mathsf{msg}, \mathbf{R}) \rightarrow \hat{\mathbf{z}} \in \mathcal{R}^m$:

1. ***Syndrome***. Compute the syndrome $\mathbf{u} = H(\mathsf{msg}) \in \mathcal{R}_q$, sample $\mathbf{p} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sqrt{\mathbf{\Sigma_p}}}$ following Section 6.3.3.
2. ***Perturbation***. Sample continuous Gaussians $\mathbf{d}_1, \ldots, \mathbf{d}_{k+2} \leftarrow_R \mathcal{D}_1^n$. Compute $\tilde{\mathbf{p}} = [\tilde{\mathbf{p}}_\mathbf{1}, \ldots, \tilde{\mathbf{p}}_{k+2}]$, where $\tilde{\mathbf{p}}_1 = \frac{r^2}{\sqrt{b}} \sum_{i=1}^{k} \mathbf{e}_i\mathbf{d}_i + \mathbf{L_1} \cdot \begin{bmatrix} \mathbf{d}_{k+1} \\ \mathbf{0} \end{bmatrix}$, $\tilde{\mathbf{p}}_2 = \frac{r^2}{\sqrt{b}} \sum_{i=1}^{k} \mathbf{r}_i\mathbf{d}_i + \mathbf{L_2} \cdot \begin{bmatrix} \mathbf{d}_{k+1} \\ \mathbf{d}_{k+2} \end{bmatrix}$ and $\tilde{\mathbf{p}}_i = \sqrt{b} \cdot \mathbf{d}_{i-2}$ for $3 \leq i \leq k+2$. Sample perturbation $\mathbf{p} = [\mathbf{p}_1, \ldots, \mathbf{p}_{k+2}]$ with $\mathbf{p}_i = \tilde{\mathbf{p}}_i + \mathcal{D}_{\mathbb{Z}^n - \tilde{\mathbf{p}}_i, a}$.
3. ***Signing***. Determine adjusted syndrome $\mathbf{v} = \mathbf{u} - \mathbf{Ap} \in \mathcal{R}_q$. Sample vector of polynomials $\hat{\mathbf{x}} \leftarrow \mathcal{D}_{\Lambda_\mathbf{v}^\perp(\hat{\mathbf{g}}^\top), r}$ following Section 6.2.2 with $r = 2a$. Output signature
$$\hat{\mathbf{z}} = \left[\mathbf{p}_1 + \hat{\mathbf{e}} \cdot \hat{\mathbf{x}},\quad \mathbf{p}_2 + \hat{\mathbf{r}} \cdot \hat{\mathbf{x}},\quad \mathbf{p}_3 + \mathbf{x}_1,\quad \ldots\ ,\quad \mathbf{p}_{k+2} + \mathbf{x}_k\right]\ .$$

$\mathsf{VerifyGPV}(\mathsf{msg}, \hat{\mathbf{z}}, (H, \mathbf{A})) \rightarrow \{0,1\}$:
Check whether $\mathbf{A} \cdot \hat{\mathbf{z}} \equiv H(\mathsf{msg})$ and $\|\hat{\mathbf{z}}\|_2 \leq s\sqrt{n(k+2)}$ is satisfied. If so, output 1 (accept), otherwise 0 (reject).

Figure 6.3.: Ring variant of the GPV signature scheme.

6.4. Security and Parameters

We considered two models in order to estimate the security level of the scheme. First, we estimate δ by means of the framework provided in [RS10]. This approach is considered to result in conservative estimations for the security level of the scheme. For this reason, we also chose to include the more common approach by Micciancio and Regev [MR08], which is often used and offers a more appropriate basis for comparison. Such a strategy of using two different models simultaneously also induces a security range of the scheme following different attack scenarios.

When applying the framework of [RS10] we get Table 6.1 which contains different parameter sets with their corresponding estimated sub-lattice attack dimension d and the more relevant estimated Hermite factor δ. The SIS norm bound is denoted by ν. Columns marked with $\star$ provide according to [GPV08, Proposition 5.7] additional worst-case to average-case hardness such that $q \geq \nu \cdot \omega(\sqrt{n \log n})$ is satisfied. The parameters of the scheme should be set in such a way that $\delta \approx 1.0064$ in order to ensure about 100 bits of security [RS10].

n	$128^\star$	$128^\star$	256	$256^\star$	284	$284^\star$	384	$384^\star$	484	$484^\star$	512	$512^\star$	1024	1024
k	24	27	24	27	24	28	24	29	24	29	24	30	27	30
m	3328	3712	6656	7424	7384	8520	9984	11136	12584	15004	13312	16384	29696	32768
q	2^{24}	2^{27}	2^{24}	2^{27}	2^{24}	2^{29}	2^{24}	2^{29}	2^{24}	2^{29}	2^{24}	2^{30}	2^{27}	2^{30}
d	327	365	606	677	665	768	871	1041	1073	1282	1128	1393	2362	2610
ν	4.6e5	5.2e5	1.3e6	1.4e6	1.5e6	1.7e6	2.4e6	2.8e6	3.3e6	3.9e6	3.6e6	4.4e6	1.1e7	1.2e7
δ	1.0201	1.0181	1.0117	1.0105	1.0107	1.0094	1.0085	1.0071	1.0070	1.0059	1.0067	1.0055	1.0034	1.0031
λ bits	< 75	< 75	75	78	78	82	86	94	95	103	97	108	148	158

Table 6.1.: Parameter sets with the corresponding estimated sub-lattice attack dimensions d and Hermite factors δ according to [RS10].

Different to [MR08] the approach taken in [RS10] requires to determine the optimal sub-dimension $d = \{x \in \mathbb{Z} \mid q^{2n/x} \leq \nu\}$ of the matrix $\mathbf{A}$ consisting of m columns and n rows. The lattice $\Lambda_q^\perp(\mathbf{A}')$ generated by $\mathbf{A}'$ when leaving out $m - d$ columns from $\mathbf{A}$ has still determinant q^n with very high probability. This means that a solution $\mathbf{v} \in \Lambda_q^\perp(\mathbf{A}')$ with $\|\mathbf{v}\| \leq \nu$ can easily be tranformed to the vector $(\mathbf{v}, \mathbf{0})$ such that $\mathbf{A} \cdot (\mathbf{v}, \mathbf{0}) \equiv 0 \mod q$ holds. For a given d we obtain the Hermite factor $\delta = 2^{n \cdot \log_2(q)/d^2}$ implying that a sufficiently good HSVP solver can find vectors $\mathbf{v} \in \Lambda_q^\perp(\mathbf{A}')$ bounded by $q^{2n/d}$. From the Hermite factor one can compute the effort $T(\delta)$ required to solve $\delta - \mathsf{HSVP}$ according to [RS10, Conjecture 3]. Subsequently, one maps the result to the corresponding security levels (e.g. see [RS10, Table 2]).

6.5. Implementation

We implemented the scheme following two approaches. The first approach uses only standard libraries without optimizing the implementation with respect to the underlying architecture. The second approach, however, does not rely on standard libraries. In particular, it employs (amongst others) new implementations for polynomial representation and multiplication using enhanced algorithms such as self-made FFT subroutines involving the AVX and AVX2 instruction sets. Our optimizations also capture sampling algorithms such as an improved perturbation generation algorithm and the usage of the FastCDT sampler. We considered both the matrix and ring variant of the scheme presented in Section 6.3.5.

6.5.1. Implementation using Standard Libraries

We implemented the GPV signature scheme, the trapdoor generation, and sampling algorithms in C using the Fast Library for Number Theory (FLINT 2.3) and the GNU Scientific Library (GSL 1.15). FLINT comprises different data types for matrices and vectors operating in rings such as $\mathbb{Z}_q$ and $\mathbb{Z}_q[X]$ whereas the GSL library provides a huge variety of mathematical tools from linear algebra, that can be applied on different primitive data types. We also included the Automatically Tuned Linear Algebra Software Library (ATLAS) which is an empirical tuning system that creates an individual BLAS (Basic Linear Algebra Subprograms) library on the target platform on which the library is installed on. Specifically, this library provides optimized BLAS routines which have a significant impact on the running times of the used mathematical operations in the key and signature generation steps. Hence, it is always recommended to include this library whenever one has to work with GSL. For the representation of matrices in $\mathbb{Z}_q^{n\times m}$ FLINT provides the data structure `nmod_mat_t` which comes into use in our implementation of the matrix version. Regarding the ring version, working with polynomials is performed by using the data structure `nmod_poly_t`. FLINT makes use of a highly optimised Fast Fourier Transform routine for polynomial multiplication and some integer multiplication operations.

The experiments were performed on a Sun XFire 4400 server with 16 Quad-Core AMD Opteron(tm) Processor 8356 CPUs running at 2.3GHz, having 64GB of memory and running 64bit Debian 6.0.6. We used only one core in our experiments. The experimental results for this implementation are given in [P9].

Sampling

For sampling discrete Gaussian distributed integers in the key generation step we used the inversion transform method rather than rejection sampling because the number of stored entries is small and can be deleted afterwards. This improves the running times of the sampling step significantly. In particular, suppose the underlying parameter is denoted by s. We precompute a table of cumulative probabilties p_t from the discrete Gaussian distribution with $t \in \mathbb{Z}$ in the range $[-\omega(\sqrt{\log n})\cdot s, \omega(\sqrt{\log n})\cdot s]$. We then choose a uniformly random $x \in [0,1)$ and find t such that $x \in [p_{t-1}, p_t]$. This can be done using binary search. The same method is applied when sampling preimages from the set $\Lambda_{\mathbf{u}}^{\perp}(\mathbf{G})$ with parameter r. This parameter is always fixed and relatively small. Storing this table takes about

150 bytes of memory. In this case signature generation is much faster than with simple rejection sampling. But, unfortunately, this does not apply in the randomized rounding step because the center always changes and thus involves a costly recomputation of tables after each sample. Therefore we used rejection sampling from [GPV08] instead. As for sampling continuous Gaussians with parameter $t = 1$, we used the Ziggurat algorithm [MT84] which is one of the fastest algorithms to produce continuous Gaussians. It belongs to the class of rejection sampling algorithms and uses precomputed tables. When operating with multiprecision vectors such as sampling continuous random vectors, one should use at least λ bits of precision for a cryptographic scheme ensuring a security level of λ (e.g., 16 bytes floating points for $\lambda = 100$).

Random Oracle Instantiation

For the GPV signature scheme a random oracle $H(\cdot)$ is required which on an input message msg outputs a uniform random response $H(\mathsf{msg})$ from its image space. In most practical applications this is achieved by a cryptographic hash function together with a pseudorandom generator which provides additional random strings in order to extend the output length. In our implementation we used SHA256 together with the GMSS-PRNG [BDK+07] because strings of arbitrary size are mapped to vectors from $\mathbb{Z}_q^n$. Each component of the vector has at most $\lfloor \log q \rfloor$ bits.

$$\begin{aligned} \mathit{Rand} &\leftarrow H(\mathit{Seed}_{\mathit{in}}) \\ \mathit{Seed}_{\mathit{out}} &\leftarrow (1 + \mathit{Seed}_{\mathit{in}} + \mathit{Rand}) \mod 2^n. \end{aligned}$$

The first $\mathit{Seed}_{\mathit{in}}$ is the input message, and the function is repeated until enough random output Rand is generated.

6.5.2. Optimized Implementation

In the following section we present an implementation that is based on self-made subroutines such as polynomial and matrix multiplication optimized for different parameter sets. Furthermore, we applied enhanced sampling algorithms that come into use in the signing step and represent a key determinant for the running time. The respective algorithms make also use of the AVX instruction sets utilized to run similar operations in parallel realizing remarkable speed-ups. These properties were also observed in several works [GOPS13]. We therefore adopt this approach in order to enhance the performance of the scheme from Section 6.3.5.

Due to lack of the AVX resp. AVX2 instruction sets on the platform used to run experiments based on the implementation from Section 6.5, the following implementation and the corresponding experiments were run on a Notebook that is specified by an

- Intel Core i7-4500U processor operating at 1.8GHz and 4GB of RAM. We used a gcc-4.8.2 compiler with compilation flags Ofast, mavx2, msse2avx, march=corei7-avx, and march=core-avx-2.

Discrete Gaussian Sampling

In order to sample discrete Gaussian distributed vectors $\mathbf{x} \leftarrow \mathcal{D}_{\Lambda^{\perp}_{\mathbf{v}}(\mathbf{G}),r}$, which can be reduced to have entries sampled from $\mathcal{D}_{2\mathbb{Z},r}$ or $\mathcal{D}_{1+2\mathbb{Z},r}$, we apply the improved discrete Gaussian sampler FastCDT introduced in Section 5.1, that perfectly matches to this kind of distributions. Furthermore, we sampled the entries of the private key both in the matrix and ring variant using FastCDT with parameter $\alpha q = p \cdot 4.7$ for $p = \lceil \sqrt{n}/4.7 \rceil$ such that $\alpha q > \sqrt{n}$. However, for the randomized rounding operation, which follows the discrete Gaussian distribution, we apply the rejection sampling algorithm. In particular, we need to sample $\lceil \mathbf{c} \rfloor_a$, which is equivalent to $\mathbf{c} + \mathcal{D}_{\mathbb{Z}^m - \mathbf{c},a}$. Due to the real vector $\mathbf{c} \in \mathbb{R}^m$ the support always changes such that generating the corresponding tables is quite inefficient. Since $\rho_{a,c_i}(\mathbb{Z}) = \rho_a(\mathbb{Z} - c_i) \in \rho_a(\mathbb{Z}) \cdot [\frac{1-\epsilon}{1+\epsilon}, 1]$ for $a \geq \eta_\epsilon(\mathbb{Z})$ as per Lemma 3.1, we need to compute $\rho_a(\mathbb{Z})$ only once for all $\mathbf{c} \in \mathbb{R}^m$, hence saving unnecessary computations. Furthermore, it is useful to sample $\lceil \bar{c} \rfloor_a$ for $\bar{c} = \lceil c \rceil - c \in (0,1)$, since $\lceil c \rfloor_a = \lceil c \rceil - \lceil \bar{c} \rfloor_a$ and the center of the distribution is always within the range $\bar{c} \in (0,1)$.

AVX and AVX2

We already explained the significance of the AVX and AVX2 instruction sets in Section 5.4, when implementing our A-LWE based encryption scheme. In our implementations, we are using AVX and AVX2 whenever possible. For instance, the FFT for polynomial multiplication is optimized by use of AVX due to computations with double precision complex numbers. Furthermore, it is exploited for scaling operations such as $\tilde{\mathbf{p}}_\mathbf{2} = \sqrt{b} \cdot \mathbf{d}_2$ and the multiplication of the decomposition matrix $\mathbf{L}$ with continuous Gaussians in the signature generation step (see Figure 6.3 and Figure 6.2). In fact, one observes remarkable speed ups.

Polynomial Representation and Multiplication

Following the efficient implementation [GOPS13] of the NTT [Win96], we implemented the FFT for polynomial multiplication by use of AVX and AVX2. Due to non-prime modulus $q = 2^k$, it is not possible to apply the NTT. We are considering cyclotomic rings of the special form $\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^n + 1 \rangle$ for n a power of 2. Therefore, the FFT is instantiated with the (complex) n-th root of unity. Similar to [GOPS13], we precomputed tables of the relevant constants prior to invoking the signing and verification algorithm. As a result, we achieve fast signing and verification engines.

Matrix-Vector Multiplication

Matrix-vector operations accomplished via additions and multiplications over the integers were performed by use of the AVX2 instruction set. In fact, our implementation of the matrix variant is built upon the implementation specified in [P3], which has been optimized with respect to matrix-vector operations.

Random Oracle Instantiation

For the random oracle instantiation, we applied the Salsa20 stream cipher as in Section 5.4. It stretches a uniform random input seed to a uniform random output of arbitrary length. Its evident performance has been observed in several works such as [GOPS13, P3]. We refer to Section 5.4 specifying how to generate uniform random elements such as polynomials or vectors.

6.6. Experimental Results

In this section we present our experimental results with respect to the optimized implementation from Section 6.5.2 and compare the matrix version with the ring variant as described in Section 6.3.5. In most works private keys and signature sizes are estimated based on the underlying distributions ignoring the norm bound of the sampled vectors and thus lead to overestimations of signature sizes. By Lemma 6.2 we show that we can ignore the underlying distributions and focus solely on the norm bound. This allows us to give tighter bounds as compared to previous proposals. For instance, in [Lyu12] signatures $\mathbf{y} \in \mathbb{Z}^m$ are distributed as discrete Gaussians with standard deviation σ. The estimated signature size is $m \cdot \lceil \log_2(12 \cdot \sigma) \rceil$ bits (ignoring the norm bound). In our case signatures are distributed as discrete Gaussians with parameter s such that $\|\mathbf{y}\|_2 < s \cdot \sqrt{m}$. Using Lemma 6.2 the bit size needed to represent $\mathbf{y}$ is bounded by $m \cdot (1 + \lceil \log_2(s) \rceil)$ bits. The private key $\mathbf{R} \in \mathbb{Z}^{2n \times n \cdot k}$ from Section 6.2.1 can be viewed as a vector $\mathbf{r}$ with $2n^2k$ entries such that $\|\mathbf{r}\|_2 < \alpha q \cdot \sqrt{2n^2k}$ by [Ban93, Lemma 1.5].

Lemma 6.2. *Let $\mathbf{v} \in \mathbb{Z}^n$ be a vector with $\|\mathbf{v}\|_2 < b \cdot \sqrt{n}$. Then, the maximum number of bits required to store this vector is bounded by $n \cdot (1 + \lceil \log_2(b) \rceil)$.*

Proof. We determine the maximum number of bits needed to store a vector $\mathbf{v}$ bounded by $\|\mathbf{v}\|_2 < b \cdot \sqrt{n}$ by means of Lagrange multipliers [Lar12]. The general form of Lagrange multipliers is defined by $L(v_1, \ldots, v_n) = f(v_1, \ldots, v_n) + \lambda \cdot g(v_1, \ldots, v_n)$, where $g(\cdot)$ takes into account the constraints and $f(\cdot)$ is the function to be maximized. Obviously, the maximum number of bits grows with increasing norm bound. Therefore, let $\mathbf{v} \in \mathbb{N}^n$ (ignoring the signs) be a vector such that $\|\mathbf{v}\|_2^2 = \sum_{i=1}^{n} v_i^2 = nb^2$. Now, consider the log entries of the vector $\mathbf{v}$, which are needed to determine the bit size of any vector. Applying simple logarithm rules we have $\sum_{i=1}^{n} \log_2(v_i) = \log_2(\prod_{i=1}^{n} v_i)$. Since log is monotone increasing, maximizing log is equivalent to maximizing the product. The function giving the constraint is $g(v_1, \ldots, v_n) = nb^2 - \sum_{i=1}^{n} v_i^2$. We then maximize the function $L(v_1, \ldots, v_n, \lambda) = f(v_1, \ldots, v_n) + \lambda \cdot g(v_1, \ldots, v_n)$, where $f(v_1, \ldots, v_n) = \prod_{i=1}^{n} v_i$. Taking the partial derivatives we get $n+1$ equations:

$$\frac{\Delta L}{\Delta v_i} = \frac{\Delta f}{\Delta v_i} + \frac{\lambda \cdot \Delta g}{\Delta v_i} = \prod_{j=1, j\neq i}^{n} v_j - 2\lambda v_i = 0, \qquad \forall 1 \leq i \leq n$$

$$\frac{\Delta L}{\Delta \lambda} = nb^2 - \sum_{i=1}^{n} v_i^2 = 0\,.$$

By reordering the first n equations, we get $\lambda = \frac{v_1 \cdot \ldots \cdot v_{i-1} \cdot v_{i+1} \cdot \ldots \cdot v_n}{2v_i}$ for $1 \leq i \leq n$. It is easy to see that the only solution is $v_i = b$, that satisfies all equations for $\forall 1 \leq i \leq n$, because from any two out of the first n equations it follows $v_i = v_j,\ i \neq j$. By the last equation we then obtain $v_i = b$. The only extremum we obtain is $\mathbf{v} = (v_1, \ldots, v_n) = (b, \ldots, b)$ with $f(\mathbf{v}) = b^n$. Since we have $0 = f(\mathbf{v}') < b^n$ for the boundary points $v_i' = b \cdot \sqrt{n}$ with $v_j' = 0$ and $j \neq i$, the extremum $\mathbf{v}$ is a maximum. Therefore the maximum possible bit size required to store such a vector is bounded by $n \cdot \lceil \log_2(b) \rceil$. We need an additional bit for the sign of each entry. This concludes the proof. The proof can be extended to any p-norm $1 \leq p < \infty$. $\square$

Based on Lemma 6.2 we deduce the following table containing expressions for various sizes such as the private key and public key size.

	Public Key (bits)	Private Key (bits)	Signature (bits)
Trapdoor [GPV08, MP12]	nmk	$2n^2k(1 + \lceil \log_2 \alpha q \rceil)$	$m \cdot (1 + \lceil \log_2 s \rceil)$

Table 6.2.: GPV-Trapdoor storage requirements

Below we provide two tables comparing the ring variant with the matrix variant. They contain the filesizes of the private key, public key, perturbation matrix, and the signature (see Table 6.4) as well as the running times for signature generation and verification (see Table 6.3). For the sake of comparison, we restrict the parameter set to $n = 512$ and $q = 2^{24}, 2^{27}, 2^{29}$ (see Section 7.4 for $n = 1024$). The experimental results for our implementation from Section 6.5.1 using standard libraries are given in [P9]. For this setting, we realized improvement factors of $30 - 90$ for key generation and approximately $2 - 6$ for signing, where the security parameter n ranges between 128 and 1024. This is due to an improved perturbation matrix involving less complex operations as compared to the original work [MP12] not scrutinizing this time consuming issues (see Section 6.3.4). However, by the subsequent optimizations from Section 6.5.2, we achieve even better results as depicted in Table 6.3. In the last column, we indicate the improvement factors of signing and verification due to our optimized implementation in comparison to the implementation from Section 6.5.1 using standard libraries. For $n = 512$ and $q = 2^{29}$, for instance, we improved the signing and verification engine by a factor of 5 and 21, respectively.

GPV Signature	**Parameters**	**Timings (cycles)**		**Security (bits)**		**Improvement**
Scheme	**q**	**Sign**	**Verify**	**[MR09]**	**[RS10]**	**Factor**
Ring Variant						
n = 512	2^{24}	13395600	464400	> 300	97	4.6 | 14.9
	2^{27}	14810400	514800	> 300	103	4.8 | 17.3
	2^{29}	15796800	558000	> 300	107	5.1 | 20.6
Matrix Variant						
n = 512	2^{24}	59862600	9558000	> 300	97	4 | 4.3
	2^{27}	67384800	10733400	> 300	103	4.3 | 4.5
	2^{29}	73746000	11930400	> 300	107	4.2 | 4.2

Table 6.3.: Timings for the GPV Signature Scheme

GPV Signature	**Parameters**	**Sizes (in kB)**				**Improvement**
Scheme	**q**	**PubKey**	**SecKey**	**Signature**	**Perturb. Mat.**	**Factor**
Ring Variant						
n = 512	2^{24}	37.5	21.3	24.5	4100	169
	2^{27}	47.3	23.9	27.4	4100	211
	2^{29}	54.4	25.7	29.4	4100	241
Matrix Variant						
n = 512	2^{24}	19,200	9984	24.5	4100	169
	2^{27}	24,192	11232	27.4	4100	211
	2^{29}	27,840	12064	29.4	4100	241

Table 6.4.: Sizes of the GPV Signature Scheme

In Table 6.4, we see that the relevant sizes in the ring variant are significantly smaller than in the matrix variant of the scheme. The last column reflects the improvement caused by the optimized decomposition matrix exploiting the sparsity and structure of $\mathbf{\Sigma_p}$. The improvement factor is related to the space requirements of the perturbation matrix in the original work [MP12]. In fact, the space requirement of our scheme is smaller by a factor of $(k + 2)^2/4$, which mainly stems from the decomposition matrix $\mathbf{L} \in \mathbb{R}^{2n\times 2n}$.

It is also worth mentioning that the authors of [MP12] explain the possibility of splitting the signing algorithm into an offline and online phase. The task of generating perturbations is independent from the message to be signed, hence it is possible to generate them in advance or create many samples and store them. This obviously requires to periodically create the perturbation matrix or storing it. From a practical point of view we do not consider such a breakdown in our implementations. But indeed, generating perturbations amounts after the optimizations from Section 6.5.2 to more than 80 percent (see Figure 6.4) of the running time in the ring variant. In Figure 6.4 we present a breakdown of the running time for signing into four major parts which are the most time consuming. In particular, we differentiate the generation of perturbations $\hat{\mathbf{p}}$, sampling of $\hat{\mathbf{x}}$, computation of the syndrome polynomial $\mathbf{v} = \mathbf{A}\hat{\mathbf{p}}$, polynomial multiplications $\hat{\mathbf{e}} \cdot \hat{\mathbf{x}}$ and $\hat{\mathbf{r}} \cdot \hat{\mathbf{x}}$. By our experiments we obtain Figure 6.4 illustrating the average measurements for different parameter sets .

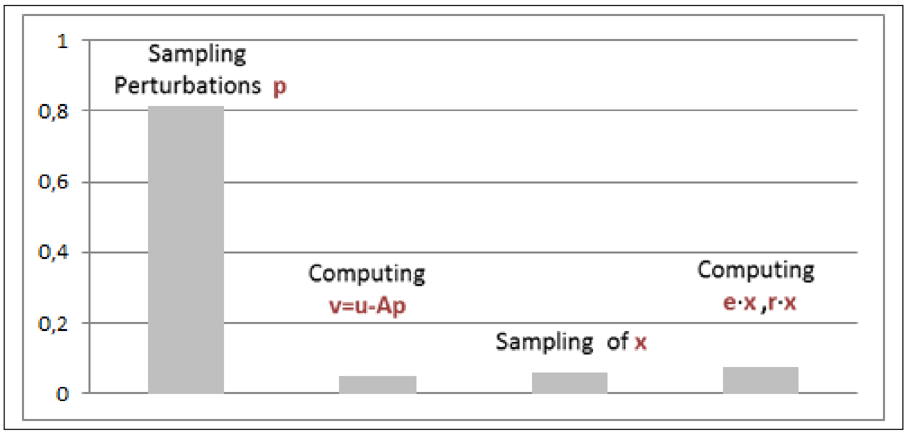

Figure 6.4.: Breakdown of signing running time into the major parts

7. Compression Scheme for Signatures

In this chapter, we introduce a generic and novel high performance compression algorithm for Schnorr-like signature schemes moving current state-of-the-art signature schemes towards practicality. This concept is realized based on the existence of publicly accessible randomness. The notion of public randomness was firstly introduced in [HL93]. We revisit this feature in the context of lattice-based cryptography and show how it can be extended to other distributions such as the discrete Gaussian distribution. In particular, we provide mechanisms that make use of public randomness in order to decrease the signature size. More specifically, we differentiate the randomness used to generate signatures into its public and secret portion. Randomness that is publicly accessible [HL93] can be read by all parties. Consequently, it is not required to hide this portion of randomness and hence can be generated publicly, for instance, by means of a random seed. The key idea underlying our lossless compression algorithm is to reduce the public share of a signature to a short uniform random seed. We exemplify the applicability of our new compression algorithm using the example of the GPV signature scheme employing the most recent PSTF construction [MP12]. Furthermore, it can be applied to the sequential aggregate signature scheme that we present in Chapter 8. To the best of our knowledge, such a compression strategy has never been used for cryptographic applications. This chapter refers to the publication [EB14a], where the author of this thesis was the primary investigator and author of the paper.

7.1. Methodology of Compressing Schnorr-like Signatures

Conceptually, Schnorr signatures $\mathbf{z} = f_{\mathbf{s}}(\mathbf{c}) + \mathbf{y}$ are constitued of two main building blocks, namely $f_{\mathbf{s}}(\mathbf{c}) = \mathbf{s} \cdot \mathbf{c}$, which involves the secret key $\mathbf{s}$, and $\mathbf{y}$ sampled from a proper distribution $\mathcal{Y}(\mathbf{x})$ acting as masking term in order to conceal the secret. In many lattice-based signature schemes the magnitude of the entries in $\mathbf{y}$, when considering $\mathbf{y}$ as a vector with n independent entries, are very huge as compared with the entries in $f_{\mathbf{s}}(\mathbf{c})$ and thus leak information about $\mathbf{y}$. More specifically speaking, if the maximum bound h on the entries of $f_{\mathbf{s}}(\mathbf{c})$ is relatively small as compared to the entries of $\mathbf{y}$, it is possible to specify a narrow range $C = \mathbf{z} + [-h, h]^n$, from which the masking term $\mathbf{y} \in C$ was sampled. Here, $h = \max\limits_{\mathbf{s},\mathbf{c}} \|f_{\mathbf{s}}(\mathbf{c})\|_\infty$ denotes the maximum bound on the entries for any choice of $\mathbf{s}$ and $\mathbf{c}$. This range is publicly accessible and can be revealed by any party viewing the signature. This shows that one part of $\mathbf{y}$ can be learned publicly and the other part remains secret. Our goal is to exploit the public part (or public randomness) supplied by $\mathbf{y}$ and hence the set C. To illustrate the way our compression algorithm works more precisely, suppose we have a fresh vector $\mathbf{z}$ (e.g. signature) distributed as above. We will show that arbitrary many other signers can exploit public randomness by secretly sampling their masking term $\mathbf{y}'$ from C (more precisely from any set $B \supseteq C$) according to the conditional probability distribution $\mathcal{Y}(\mathbf{x})/P_{\mathbf{y}\sim\mathcal{Y}}[\ \mathbf{y} \in C\ ]$ for $\mathbf{x} \in C$. Since $\mathbf{y}$ is independently sampled, we have $\mathbf{y} \in C$ with probability $P_{\mathbf{y}\sim\mathcal{Y}}[\ \mathbf{y} \in C\ ]$ (or shortly $P[\ C\ ]$) for arbitrary fixed $\mathbf{z}$. Hence, exploiting public and secret randomness leads to a vector $\mathbf{y}'$ that is distributed as $P[\ \mathbf{y} \in\ C\ ] \cdot P[\ \mathbf{y}' = \mathbf{x} \mid \mathbf{y}' \in C] = P[\ C\ ] \cdot \mathcal{Y}(\mathbf{x})/P[\ C\ ] = \mathcal{Y}(\mathbf{x})$, which exactly coincides with the required distribution using conditional probability rules. We note that this is, however, only possible, if the true probability distribution of $\mathbf{y}$ is publicly known. Some signature schemes with a rejection sampling step at the end do not meet these conditions and are thus not covered by our framework. Following this approach, we derive an upper bound for the maximum distance of two signatures $\|\mathbf{z} - \mathbf{z}'\|_\infty = \|\mathbf{z} - \mathbf{s}' \cdot \mathbf{c}' - \mathbf{y}'\|_\infty \leq 2h$. A necessary condition for compression is given by $2h < \|\mathbf{z}\|_\infty$, which is typically satisfied for current state-of-the-art signature schemes. A compressed signature is identified by the tupel $(\mathbf{z}, \mathbf{z} - \mathbf{z}')$, where $\mathbf{z}$ is called the centroid and serves to recover $\mathbf{z}'$. To prove security, we simulate our compression algorithm using an oracle for uncompressed signatures from the underlying signature schemes. Subsequently, we show that uncompressed signatures can publicly be transformed into compressed ones. An immediate consequence from this implies that the same centroid can be utilized by different other signers with different keys such that only one single centroid is required to uncompress the signatures of the respective signers. This obviously induces a conceptually new multi-signer compression scheme, where a set of users participate in producing a bundle of compressed signatures using the same source of public randomness. Such a strategy constitutes a simple way of aggregating signatures. Going further, since the distribution $\mathcal{Z}$ of signatures $\mathbf{z}$ can always be simulated by a cryptographic hash function modeled as random oracle in combination with a rejection sampling algorithm, we forbear to store the centroid $\mathbf{z} \in \mathbb{Z}^m$ and store a short seed $\mathbf{r} \in \{0,1\}^\mu$ instead that serves as input to a sampler for $\mathcal{Z}$ (typically discrete Gaussian or uniform distribution). This strongly reduces the signature size, since the share of the signature associated to public randomness can deterministically be recovered by use of $\mathbf{r}$. Doing this, it is even possible to compress individual signatures to $(\mathbf{r}, \mathbf{z} - \mathbf{z}')$ of size $\mu + m \log 2h$ bits involving a fresh seed for every newly generated signature. A bundle of compressed signatures in the multi-signer compression scheme is subsequently represented by the tupel $(\mathbf{r}, \mathbf{z} - \mathbf{z}_1, \mathbf{z} - \mathbf{z}_2, \ldots, \mathbf{z} - \mathbf{z}_l)$, where $\mathbf{z}_i$ denotes the signature of the i-th signer.

Geometrically speaking, the proposed compression algorithm is akin to vector quantization techniques [GG91, Gra84] applied for lossy video and audio compression (e.g. MPEG-4). But from an algorithmic point of view our scheme works differently as it requires the signers to sample signatures $\mathbf{z}'$ within short distance to a vector called centroid $\mathbf{z}$ (see Figure 7.1), which could be a signature or a vector sampled from the discrete Gaussian distribution using a short random seed as

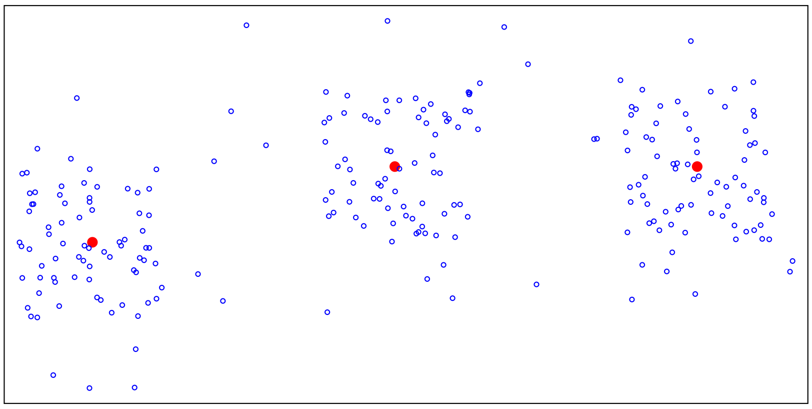

Figure 7.1.: Centroids (red circles) are surrounded by signatures from different signers (blue circles). Each signature belongs to one cluster defined by its centroid.

input. This allows for high compression rates without loss of quality because it is always possible to recover the signatures after compression. As a result, it suffices to store only the seed and the difference $\mathbf{z} - \mathbf{z}'$ of the signature to the centroid. This apparently avoids the need to store complete signatures (see Figure 7.2 *left* and *right*). When employing GPV signatures, for instance, the implied storage savings amount to approximately 65 % for practical parameters (see Table 7.1) yielding a factor improvement of approximately $\lg n$ with n being the main security parameter. Based on this compression strategy we derive a multi-signer compression scheme (see Figure 7.1) that allows an arbitrary number of signers sharing the same source of public randomness to combine their signatures to an aggregate resp. bundle of reduced storage size.

		Signature size in [kB] before/after comp.		Compression rate [%]		Factor improvement	
n	k	Ring	Mat	Ring	Mat	Ring	Mat
384	24	22 / 8	20 / 8	65	61	2.8	2.6
512	29	37 / 13	33 / 12	66	62	2.9	2.7
1024	30	81 / 26	72 / 26	68	64	3.1	2.8

Table 7.1.: Compression rates in the ring and matrix variant for different parameter sets.

Compression of GPV Signatures.

Ever since the seminal work [GPV08] the hash-and-sign approach for building signatures becomes more and more attractive for use in cryptographic applications. However, the construction of new and more efficient preimage sampleable trap-

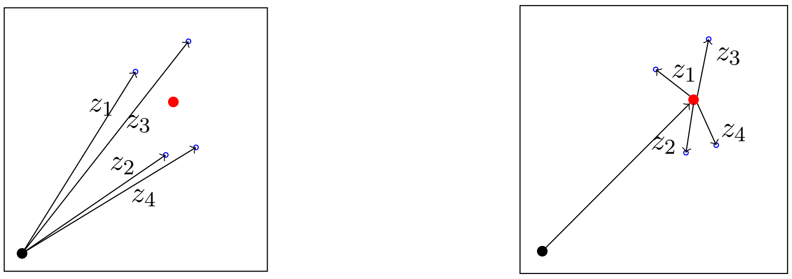

Figure 7.2.: Complete signatures of different signers are stored (*left*). Compressed signatures from different signers are stored in relation to the centroid (*right*).

door functions entailing tighter bounds and simpler instantiations appears to be a challenging task in lattice-based cryptography. One of the main goals of those constructions is to reduce the signature size while preserving security. Decreasing the parameter s governing the signature size is often not readily possible without affecting the security, since the security proof [GPV08] requires $s \geq \eta_\epsilon(\Lambda_q^\perp(\mathbf{A}))$ to be satisfied for a random matrix $\mathbf{A}$. Usually, the quality s is almost tight due to the construction of the public key $\mathbf{A}$. Thus, enhancing the quality always involves the construction of new trapdoor families. In our work, we provide a very different approach to reduce the signature size by exploiting large amounts of public randomness accessible to any party viewing the signature.

To get an impression of how our compression algorithm works, we believe it is reasonable to first sketch the GPV signature scheme instantiated with the efficient trapdoor construction from [MP12]. The GPV signature scheme was a big move towards provably secure lattice-based signatures. Similar to the full-domain hash schemes and its variants in [BR93, BR96, Cor00], it is based on collision-resistant preimage sampleable (trapdoor) functions (PSTF) $\mathsf{f}_{\mathbf{A}} : B_n \to R_n$, which enable a dedicated signer to sample preimages $\mathbf{z} \in B_n$ for arbitrary given target vectors $\mathbf{y} \in R_n$ such that $\mathsf{f}_{\mathbf{A}}(\mathbf{z}) = \mathbf{y}$ holds, but other than that signer none is capable of producing preimages. The security of this scheme consists in reducing the problem of finding collisions for $\mathsf{f}_{\mathbf{A}}(\cdot)$ to the hardness of forging signatures (see Chapter 6). In the course of years, several constructions of PSTF families appeared [GPV08, AP09, Pei10], where the collision-resistance stems from the hardness of SIS, which is in turn believed to withstand quantum attacks for properly chosen parameters. The main drawback of all those schemes is the lack of efficiency due to complex procedures. Recently, Micciancio and Peikert [MP12] proposed an elegant trapdoor construction, that is characterized by efficient operations providing tighter bounds for all relevant quantities and thus improving upon previous constructions. But also in practice they appear to be efficient, which can also be attributed to the corresponding ring variant introduced in Section 6.2.2 and Section 6.3.5. We now describe one instantiation of the digital signature scheme that is most suitable for GPV: The signer

generates a random matrix $\bar{\mathbf{A}} \in \mathbb{Z}^{n\times n}$ and a secret matrix $\mathbf{R} \in \mathbb{Z}^{2n\times nk}$ with entries sampled from the discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z},\alpha q}$, where $\alpha q > \sqrt{n}$ and $q = 2^k$. The public key is given by $\mathbf{A} = [\mathbf{I}_n \mid \bar{\mathbf{A}} \mid \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}]$, where $\mathbf{G} \in \mathbb{Z}^{n\times nk}$ is called the gadget, a matrix of special structure $\mathbf{I}_n \otimes \mathbf{g}^\top$ with $\mathbf{g}^\top = (1, 2, \ldots, 2^{k-1})$, which allows to sample preimages more efficiently. In the signing step the signer computes $\mathbf{u} = H(\mathsf{msg})$ for a message msg of choice, samples a perturbation vector $\mathbf{p}$ and a preimage $\mathbf{x} \leftarrow_R \mathcal{D}_{\Lambda^\perp_{\mathbf{v}}(\mathbf{G}),r}$ for $\mathbf{v} = \mathbf{u} - \mathbf{A}\cdot\mathbf{p} \bmod q$ and $r > \eta_\epsilon(\Lambda^\perp_q(\mathbf{G}))$. The resulting signature $\mathbf{z} = (\mathbf{z}^{(1)}, \mathbf{z}^{(2)}) = \begin{bmatrix}\mathbf{R}\\ \mathbf{I}\end{bmatrix}\cdot\mathbf{x} + \mathbf{p} = (\mathbf{R}\mathbf{x} + \mathbf{p}_1, \mathbf{x} + \mathbf{p}_2) \in \mathbb{Z}^{2n} \times \mathbb{Z}^{nk}$ is spherically distributed. Similar to the signature schemes [Lyu08, Lyu09, GLP12], the perturbation vector is used in order to keep the distribution of the signature independent from the secret key and thus not leaking information about its structure. Hence, it is no longer feasible to successfully mount an attack similar to [NR09, DN12]. Verification of signatures is performed by checking the validity of $\mathbf{A}\mathbf{z} \equiv H(\mathsf{msg}) \bmod q$ and $\|\mathbf{z}\| \leq s\sqrt{2n+nk}$.
We now turn our focus on the way $\mathbf{p} = (\mathbf{p}_1, \mathbf{p}_2)$ is generated, since it plays an important role for our compression algorithm. Specifically, we sample perturbations $\mathbf{p} = \lceil\sqrt{\mathbf{\Sigma}_p - a^2\mathbf{I}}\cdot\mathbf{d}\rfloor_a$, where $\lceil\cdot\rfloor_a$ denotes the randomized rounding operation from [Pei10] and $\mathbf{d}$ is sampled from the continuous Gaussian distribution $\mathcal{D}_1^{2n+nk}$ with parameter 1. Following Section 6.3.3, the perturbation matrix can be represented by $\sqrt{\mathbf{\Sigma}_\mathbf{p} - a^2\mathbf{I}} = \begin{bmatrix}\frac{\mathbf{R}}{\sqrt{b}} & \mathbf{L}\\ \sqrt{b}\mathbf{I}_\mathbf{nk} & 0\end{bmatrix}$ with $b = s^2 - 5a^2$, $a = r/2 = \sqrt{\ln\left(2n\left(1+\frac{1}{\epsilon}\right)\right)/\pi}$ and $\mathbf{L}$ denoting the decomposition matrix. This immediately leads to the simplified representation of the perturbation vector $\mathbf{p}$ with $\mathbf{p}_2 = \sqrt{b}\mathbf{d}_2 + \mathcal{D}_{\mathbb{Z}^{n\cdot k}-\sqrt{b}\mathbf{d}_2,a}$ and $\mathbf{d}_2 \leftarrow_R \mathcal{D}_1^{nk}$. Following the abstract form of a Schnorr-like signature as above, the lower part of the signature is represented by $\mathbf{z}^{(2)} = f_\mathbf{I}(\mathbf{x}, \mathbf{y}^{(2)}) + \mathbf{y}^{(2)} = \mathbf{I}\cdot\mathbf{x} + \mathcal{D}_{\mathbb{Z}^{n\cdot k}-\mathbf{y}^{(2)},r} + \mathbf{y}^{(2)}$ with $\mathbf{y}^{(2)} = \sqrt{b}\mathbf{d}_2$ and $h = \max\left\|f_\mathbf{I}(\mathbf{x}, \mathbf{y}^{(2)})\right\|_\infty \leq 4.7\cdot\sqrt{5}a$. By scaling the lower part of any signature to $\mathbf{z}^{(2)}/\sqrt{b}$ we extract large amounts of information about the continuous Gaussian $\mathbf{d}_2$ used for sampling the perturbation vector. This randomness is publicly accessible [HL93] and can be read by all parties. Indeed, the security level of cryptographic schemes should not be based on public random inputs according to [HL93], because any adversary can analyze public random strings and exploit them for potential attacks. In particular, we have $\mathbf{d}_2 \in C = \frac{\mathbf{z}^{(2)}}{\sqrt{b}} + [-\frac{h}{\sqrt{b}}, \frac{h}{\sqrt{b}}]^{nk}$ except with negligible probability. Due to the huge value of $\sqrt{b}$ as compared to h the set C is of small width containing little entropy. By use of rejection sampling, it is possible to sample a random variable $\mathbf{d}_2' \in C$ according to the probability density function $f(\,\mathbf{x} \mid \mathbf{x} \in C\,) = e^{-\pi\|\mathbf{x}\|_2^2}/P[\,C\,]$ in order to get a full realization of a continuous Gaussian. More specifically, the first signer samples a continuous Gaussian $\mathbf{d}_2$, which lies in any set C with probability $P[\,C\,]$ following the basic signature scheme and outputs the signature subvector $\mathbf{z}^{(2)}$. The second signer extracts the public randomness, namely the target range C of $\mathbf{d}_2$, and samples secretly $\mathbf{d}_2'$ according to $f(\,\mathbf{x} \mid \mathbf{x} \in C\,)$. Employing public and secret randomness results in a random vector $\mathbf{d}_2'$ following the probability density function $f(\,\mathbf{x} \mid \mathbf{x} \in C\,)\cdot P[\,C\,] = f(\,\mathbf{x}\,) = e^{-\pi\|\mathbf{x}\|_2^2}$, which is

distributed just as $\mathcal{D}_1^{nk}$ applying conditional probability rules. As a consequence, by one of our main statements (Theorem 7.7) the difference $\| \mathbf{z}^{(2)} - \mathbf{z}_1^{(2)} \|_\infty \leq 2h$ requires at most 7 bits per entry with $\mathbf{z}_1 = (\mathbf{z}_1^{(1)}, \mathbf{z}_1^{(2)})$ being the signature of the second signer. Any number of parties with different secret keys can utilize the same source of public randomness analogously. Hence, an abitrary signature $\mathbf{z}_1$ can be represented by the centroid $\mathbf{z}^{(2)}$ in combination with the compressed signature $(\mathbf{z}_1^{(1)}, \mathbf{z}^{(2)} - \mathbf{z}_1^{(2)})$ (see Figure 7.2). This, however, requires $\mathbf{z}^{(2)}$ always to be fresh such that the stream of signatures $\mathbf{z}_1$ generated by a certain signer are uncorrelated. We highlight that even higher compression rates can be realized if the centroid $\mathbf{z}^{(2)}$ is generated differently. Due to the dual role of $\mathbf{z}^{(2)}$ to serve as centroid for compression and source of public randomness, any signer can instead sample a fresh and short random seed $\mathbf{r} \in \{0,1\}^\mu$ as input to a discrete Gaussian sampler producing vectors being distributed just like signatures. By doing this, our construction even allows to compress individual signatures because the large centroid is now replaced by a short seed. Furthermore, the dependency to a signature from a signer with different public key is removed. From $\mathbf{r}$ one can deterministically recover the centroid and uncompress signatures. As a consequence, the verification costs increase due to an additional call to the discrete Gaussian sampler before checking the validity. Employing this strategy leads to storage improvement factors of $\lg n$.

Multi-Signer Compression Scheme (MCS).

The above-described compression algorithm represents the heart of a conceptually new multi-signer compression scheme, where a set of signers participate to construct an aggregate signature (bundle of compressed signatures including the seed) on messages of choice such that its size is much smaller than the total size of all individual signatures. An intuitive way of aggregating signatures is to let the signers independently compress their signatures before forwarding them to the verifier. As a disadvantage, such a strategy places a burden on the verifier as it is required to invoke the Gaussian sampler for each transmitted seed, which leads to unsatisfactory running times due to costly computations. To overcome this obstacle, the signers agree on a random seed prior to the actual scheme execution. As a result, the verifier calls the Gaussian sampler once, whose output is used as centroid for each compressed signature (see Figure 7.2 *left* and *right*). This drastically reduces the verification costs as well as the number of seeds to be transmitted. The security of this scheme trivially stems from the unforgeability of each individual (un)compressed signature since each of them is verified independently from the remaining ones. Note that as with individual signatures, random seeds have to be fresh for any newly computed aggregate signature.
Furthermore, one notices as an additional benefit of the multi-signer compression scheme that the aggregate signature is not completely rejected, if a compressed signature out of the bundle fails to verify, which is apparently different from classical aggregate signature schemes. In fact, only the signature, that failed the checks, is considered not valid. A potentially interesting modification requires the seed to be

made a shared secret among the participants, which are subsequently the only ones being capable of uncompressing and verifying signatures. We exemplify the applicability of our construction within wireless sensor networks. We particularly show that it is conceivable to use a predistributed seed together with a counter maintained by each senor node. For any compression request, the actual counter is incremented and subsequently appended to the seed, which in turn serve as input to a cryptographic hash function modeled as random oracle outputting random strings to launch the discrete Gaussian sampler.

7.2. Generic Lossless Compression of Schnorr-like Signatures

In this section we introduce a novel compression algorithm for signatures following a Schnorr-like construction $\mathbf{z} = f_{\mathbf{s}}(\mathbf{c}) + \mathbf{y}$. Conceptually, such signature schemes are characterized by simple representations and efficient operations. After establishing a framework for lossless compression, we show how to derive a customized compression algorithm for the optimized GPV signature scheme from Section 6.3.5. In fact, the algorithm exploits the representation of perturbations according to Section 6.3.3.
In general, lossless compression of data aims at reducing the bits needed to identify a data unit by removing statistical redundancy without loss of quality. Vector quantization [GG91, Gra84] is a technique from signal processing that belongs to the class of lossy data compression algorithms. It divides a large set of data viewed as vectors into clusters. For each cluster, the algorithms heuristically determine a centroid such that the distance between any vector in the cluster and its centroid is minimized. The whole set of data points is then represented by the centroids. Such algorithms are employed, for instance, for audio and video compressions like the Twin vector quantization (VQF) for MPEG-4. In order to achieve lossless compression, it is essentially required to store the direction vectors, which preferably should have small entries. But in practice, lossless compression strategies based on vector quantization techniques are rather rare due to low compression rates as compared to other alternatives. However, the approach we propose makes use of the fact that the centroids are known just before sampling the signatures, which is different to current vector quantization techniques. In particular, we exploit the structure and properties of signature constituents in order to reduce the amount of information needed to recover signatures. Conceptually, one defines the centroids in advance and each signer samples its signature around the centroids (see Figure 7.1). Doing this, one has only to store the direction vectors rather than all individual signatures as depicted in Figure 7.2. Notably, we can even show that the large centroid needs not to be stored due to the existence of simulators for signatures such as random vectors or discrete Gaussians providing the required public randomness. By means of a short random seed, which serves as input a discrete Gaussian sampler (or alternatively RO) acting as simulator for signatures, one can deterministically recover the centroid. Following this strategy, we achieve storage improvement factors of

about $2.5 - 3.8$ for the GPV signature scheme applying practical parameters and approximately $\lg n$ for the general case. The compression factor is asymptotically optimal in the main security parameter.

7.2.1. Lossless Compression Algorithm

In the following, we introduce our generic compression algorithm. We call it the LCPR algorithm (Lattice-based Compression from Public Randomness). We consider two approaches. The first approach compresses signatures with respect to a given signature serving as a centroid. Therefore, we shortly write $\mathbf{v}$ is compressed w.r.t. $\mathbf{w}$, when $\mathbf{w}$ is used as the source of public randomness and acts as the centroid for compression. The second approach requires to generate the centroid, that acts as a supplier of public randomness, from a short random seed. Specifically, the seed serves as input to a sampler that produces vectors being distributed just like signatures.

```
Algorithm 6: Compression by Signature
  Data: Fresh signature $\mathbf{z}_1 = f_{\mathbf{s}_1}(\mathbf{c}_1) + \mathbf{y}_1 \in \mathbb{Z}^m$ of Signer
        1 with $\mathbf{y}_1 \sim \mathcal{Y}$ and $\mathbf{z}_1 \sim \mathcal{Z}$

1 Set $h := \max_{\mathbf{s},\mathbf{c}} \|f_{\mathbf{s}}(\mathbf{c})\|_\infty$
2 Set $C := \mathbf{z}_1 + [-h,h]^m,\ P[\ C\ ] := P_{\mathbf{y}\sim\mathcal{Y}}[\mathbf{y} \in C]$
3 Sample $\mathbf{y}_2 \leftarrow \mathcal{Y}(\mathbf{x})/P[\ C\ ],\ \mathbf{x} \in C$
4 $\mathbf{z}_2 = f_{\mathbf{s}_2}(\mathbf{c}_2) + \mathbf{y}_2$

5 Output $\mathbf{z} = (\mathbf{z}_1,\ \mathbf{z}_1 - \mathbf{z}_2)$
```

```
Algorithm 7: Compression by Seed
  Data: Distribution of signatures $\mathcal{Z}$

1 Sample $\mathbf{r} \leftarrow \mathcal{U}(\{0,1\}^l)$
2 Sample $\mathbf{z}_1 \leftarrow \mathcal{Z}$ using input seed $\mathbf{r}$
3 Set $h = \max_{\mathbf{s},\mathbf{c}} \|f_{\mathbf{s}}(\mathbf{c})\|_\infty$
4 Set $C = \mathbf{z}_1 + [-h,h]^m,\ P[\ C\ ] := P_{\mathbf{y}\sim\mathcal{Y}}[\mathbf{y} \in C]$
5 Sample $\mathbf{y}_2 \leftarrow \mathcal{Y}(\mathbf{x})/P[\ C\ ],\ \mathbf{x} \in C$
6 $\mathbf{z}_2 = f_{\mathbf{s}_2}(\mathbf{c}_2) + \mathbf{y}_2$

7 Output $\mathbf{z} = (\mathbf{r},\ \mathbf{z}_1 - \mathbf{z}_2)$
```

Figure 7.3.: Lossless Compression Algorithms

Informal Description

The main idea of our compression algorithm is the fact, that one portion of randomness used to generate a signature can publicly be read out. Thus, it is possible to either exploit public randomness (having the same distribution) from other signers or to generate public randomness from a short seed with enough entropy such that a verifier can reconstruct the public portion of randomness with the aid of this seed. This concept, however, requires to preserve the distribution of public randomness, meaning that public randomness should always follow the correct distribution. As a result, if one applies for every newly generated signature fresh public randomness, it directly follows that the sequence of produced signatures via the compression scheme are independent and identically distributed according to the required distribution $\mathcal{Z}$. This means in particular that there exist no correlations among the signatures.

In Figure 7.3 we present two generic compression algorithms. We briefly describe the main steps required to compress a signature with respect to a given fresh signature (Algorithm 1) or using a simulator for signatures (Algorithm 2) with a short input seed. First, we note that the (conditional) probability distribution of $\mathbf{y}_1$ must publicly be known, otherwise it is not possible to apply the compression scheme. Signature schemes applying rejection sampling on the signature are not covered. This issue will be explained below in this section. Each time the signer wants to compress its signature he asks for fresh public randomness (fresh seed or $\mathbf{z}_1$). Therefore, we consider signatures following a Schnorr-like construction in a more abstract representation form $\mathbf{z} = f_{\mathbf{s}}(\mathbf{c}) + \mathbf{y}$, where $f_{\mathbf{s}}(\mathbf{c})$ describes a function of the secret key and is, hence, kept secret within the process of signature generation. However, $\mathbf{y}$ is called the masking term required to conceal the secret key and to obtain the desired target distribution of the signature. In many schemes the magnitude of the entries in $\mathbf{y}$ are huge as compared to $f_{\mathbf{s}}(\mathbf{c})$. This offers the opportunity to read and exploit public randomness. Let $C = \mathbf{z}_1 + [-h, h]^m$. In Algorithm 1 a fresh signature $\mathbf{z}_1$ of an arbitrary signer is given. By using only public parameters a second signer, that is different from the first signer, extracts public randomness identified by a (narrow) set C from which $\mathbf{y}_1 \in C$ was sampled with overwhelming probability. Subsequently, he samples its own masking term $\mathbf{y}_2$ secretly from the set C, particularly also from any set $B \supseteq C$ such as $\mathbf{z}_1 + [-\mathbf{c}_1 \cdot h, \mathbf{c}_2 \cdot h]$ for randomly chosen vectors $\mathbf{c}_i \in \mathbb{R}^m_{\geq 1}$ and $i = 1, 2$, using the conditional probability distribution $\mathcal{Y}(\mathbf{x})/P[\ C\ ]$, where $P[\ C\ ]$ denotes the probability of the event $\mathbf{y} \in C$ under the distribution $\mathcal{Y}$.

Finally, the signer outputs a compressed signature $(\mathbf{z}_1, \mathbf{z}_1 - \mathbf{z}_2)$ with $\mathbf{z}_2 = f_{\mathbf{s}_2}(\mathbf{c}_2) + \mathbf{y}_2$. Algorithm 2 allows to compress individual signatures without involving any other party providing a fresh signature. In fact, the distribution $\mathcal{Z}$ of a signature can be simulated by use of a random oracle $H : \{0,1\}^\mu \to \{0,1\}^t$ with $\mu < t$ in combination with a rejection sampling algorithm. Therefore, we replace a real signature by a sample $\mathbf{z}_1 \leftarrow \mathcal{Z}$ generated by means of a random seed $\mathbf{r} \leftarrow_R \{0,1\}^\mu$. The remaining steps are identical to those in Algorithm 1. In the last step, however, the signer outputs the compression $(\mathbf{r}, \mathbf{z}_1 - \mathbf{z}_2)$ which includes a short seed rather than a huge signature $\mathbf{z}_1$. We note, that arbitrary many other signers can exploit the same public randomness using either of the algorithms. But the same signer is not allowed to reuse the same randomness twice in order to keep the distribution of own signatures independent from previous samples. Consequently, each newly generated signature involves a fresh seed such that the chain of signatures $\mathbf{z}_2^i$ are independent and identically distributed according to $\mathcal{Z}$. The procedure of uncompressing signatures is very efficient, since it mainly requires to recover $\mathbf{z}_1$ using the seed $\mathbf{r}$ (Algorithm 2).

7.2.2. Analysis

The authors of [HL93] were the first classifying the notion of randomness into its public and secret portion. Publicly accessible randomness is the part that can be read by all parties and particularly also by an adversary. The secret portion of randomness, on the other hand, is only known to the party enacting the cryptographic primitive. This distinction is essential because a potential attacker can exploit public randomness in order to mount an attack on the respective cryptographic primitive. As a consequence, the security of any scheme should mainly depend on the secret portion of randomness. However, the authors made such a distinction only for uniform random strings. In our work, we extend this notion also to other distributions such as Gaussians-like distributions and show how this allows to build a strong compression algorithm. The key idea underlying this construction is to reuse public randomness in order to sample signatures within short distance to the centroids.

We begin with a formal definition of public randomness and some technical results explaining how to exploit public randomness.

Theorem 7.1 (**Public Randomness**). *Let $\mathcal{Y}$ be a distribution and $y \leftarrow \mathcal{Y}$ with $y \in C = z + [-h, h]$ for $h > 0$ and $z \in \mathbb{R}$. Then, there exists a bijective transformation $\phi : \{0,1\}^* \times [b_1, b_2) \to \mathbb{R}$ for $b_2, b_1 \in \mathbb{R}$ with $b_2 - b_1 = 1$ such that $\phi^{-1}(\frac{z}{2h} + [-0.5, 0.5]) = (a_0, \ldots, a_m) \times [b_1, b_2)$ for $(a_0, \ldots, a_m) \in \{0,1\}^m$ and $m \in \mathbb{N}$. Moreover, we have $\phi^{-1}(\frac{y}{2h}) \in (a_0, \ldots, a_m) \times [b_1, b_2)$, where $(a_0, \ldots, a_m)$ is called public randomness, and the probability of $\mathbf{a} = (a_0, \ldots, a_m)$ to occur is $P_{y \sim \mathcal{Y}}[\ y \in C\ ]$.*

Proof. It is always possible to write a real number r as $r = x + t$ with $x \in \mathbb{Z}$ and $t \in [b_1, b_2)$ such that $b_2 - b_1 = 1$ and r can bijectively be mapped back to x and t. Intuitively, we fill the gap between two consecutive integers with reals modulo 1. Any integer x can now be transformed into its binary representation $\mathbf{a} = (a_0, \ldots, a_m)$. Let $b_1 = -0.5 + c$ and $b_2 = 0.5 + c$, where $c = \frac{z}{2h} - \lceil \frac{z}{2h} \rfloor \in (-0.5, 0.5)$, then any element $r \in \frac{z}{2h} + [-0.5, 0.5]$ satisfies $\phi^{-1}(r) \in \{\mathbf{a}\} \times [b_1, b_2)$ with $\mathbf{a}$ being the binary representation of $\lceil \frac{z}{2h} \rfloor$, since $r \in \sum_{i=1}^{m} a_i 2^i + [b_1, b_2) = \lceil \frac{z}{2h} \rfloor + [c - 0.5, c + 0.5) = \frac{z}{2h} + [-0.5, 0.5]$. But indeed, we have also $\phi^{-1}(\frac{y}{2h}) \in \{\mathbf{a}\} \times [b_1, b_2)$. As a result, $\mathbf{a}$ is the same for all elements in that range. Therefore, the bit string $\mathbf{a} = (a_0, \ldots, a_m)$ is called the public randomness induced by C and can be extracted by any party viewing C. Let $\mathcal{X}$ denote the distribution $\phi^{-1}(\mathcal{Y}/2h)$, where a vector $\phi^{-1}(\frac{y}{2h})$ sampled according to this distribution involves $y \leftarrow \mathcal{Y}$. We know that the probability is invariant with respect to bijective transformations and hence obtain $\mathbf{a}$ with probability

$$\begin{aligned} P_{(\mathbf{x},t)\sim\mathcal{X}}\,[\ (\mathbf{x},t)\in\{\mathbf{a}\}\times[b_1,b_2)\ ] &= P_{\phi(\mathbf{x},t)\sim\mathcal{Y}/2h}\left[\ \phi(\mathbf{x},t)\in\frac{z}{2h}+[-0.5,0.5]\ \right] \\ &= P_{y\sim\mathcal{Y}}\,[\ y\in C\ ] \text{ with } y=\phi(\mathbf{x},t)\cdot 2h\,. \end{aligned}$$

Note, that the support of $\mathcal{Y}$ can differ from $\mathbb{R}$. In fact, the proof works for any distribution over a subset of $\mathbb{R}$ and by association $\mathbb{Z}$. □

As already indicated above the m-bit string $(a_0,\ldots,a_m)$ is called public randomness, that can be accessed by any party viewing the signature. Basically, the knowledge of h and the signature z suffice to determine C. As an immediate consequence of Theorem 7.1, we obtain less number of public random bits, in case the range of C gets wider due to increasing values for h. The following result states that it is possible to exploit $(a_0,\ldots,a_m)$ or less bits of it in order to get a full realization from the target distribution.

Theorem 7.2 **(Exploiting Public Randomness).** *Let $y_1 \leftarrow \mathcal{Y}$ with $y_1 \in C = z+[-h,h]$ for $h>0$ and $z\in\mathbb{R}$. And let $\phi:\{0,1\}^*\times[b_1,b_2)\to\mathbb{R}$ be a bijective transformation as defined in Theorem 7.1. Then, we obtain a full realization y from $\mathcal{Y}$ by sampling $y\in C$ according to the probability distribution $P_{y\sim\mathcal{Y}}\,[\ y=y_2\mid y\in C\ ]$.*

Proof. From Lemma 7.1, we deduce that $\phi^{-1}(\frac{z}{2h}+[-0.5,0.5])=\{\mathbf{a}\}\times[b_1,b_2)$ for $\mathbf{a}=(a_0,\ldots,a_m)$. Hence, the event $\mathbf{x}=(a_0,\ldots,a_m)$ occurs with probability $P_{y_1\sim\mathcal{Y}}\,[\ y_1\in C\ ]$. Suppose first, that $\mathcal{Y}$ is a discrete distribution and $\mathcal{X}$ denotes the distribution $\phi^{-1}(\mathcal{Y}/2h)$, where $(\mathbf{x},t)\leftarrow\mathcal{X}$ is equivalent to sampling $y\leftarrow\mathcal{Y}$ and outputting $\phi^{-1}(\frac{y}{2h})$. Then, the term $t\in[b_1,b_2)$ is sampled according to the probability distribution

$$\begin{aligned} P_{(\mathbf{x},t)\sim\mathcal{X}}\,[\ t=t_1\mid\mathbf{x}=\mathbf{a}\ ] &= P_{(\mathbf{x},t)\sim\mathcal{X}}\,[\ (\mathbf{x},t)=(\mathbf{x},t_1)\mid\mathbf{x}=\mathbf{a}\ ] \\ &= P_{y\sim\mathcal{Y}}\,[\ y=y_2\mid y\in C\ ] \text{ with } y_2=\phi(\mathbf{x},t)\cdot 2h \end{aligned}$$

Once having sampled t according to this probability distribution, we obtain a full realization $(\mathbf{x},t)$ that is distributed as

$$\begin{aligned} P_{(\mathbf{x},t)\sim\mathcal{X}}\,[\ \mathbf{x}=\mathbf{a}\ ]\cdot P_{(\mathbf{x},t)\sim\mathcal{X}}\,[\ t=t_1\mid\mathbf{x}=\mathbf{a}\ ] &= P_{(\mathbf{x},t)\sim\mathcal{X}}\,[\ (\mathbf{x},t)=(\mathbf{a},t_1)\ ] \\ &= P_{y\sim\mathcal{Y}}\,[\ y=y_2\ ]\,. \end{aligned}$$

□

Analogously, one obtains similar results for the continuous case. The main difference here is to consider the probability density function instead. Concerning the algorithms in Figure 7.3 the following theorem mainly states that exploiting public randomness indeed does not change the distribution of signatures. Moreover, it indicates a necessary condition for compression.

Theorem 7.3. *The compression algorithm provided in Figure 7.3 outputs signatures* $\mathbf{z}_2 \in \mathbb{Z}^m$ *distributed according to* $\mathcal{Z}$ *with* $\max \|\mathbf{z}_1 - \mathbf{z}_2\|_\infty \leq 2h$, *for* $h = \max_{\mathbf{s}} \|f(\mathbf{s})\|_\infty$. *Hence, the size of a compressed signature* $(\mathbf{r}, \mathbf{z}_1 - \mathbf{z}_2)$ *is bounded by*

$$\lceil m \cdot \log 2h \rceil + \mu \text{ bits},$$

where $\mathbf{r}$ *occupies* μ *bits of memory.*

Proof. For simplicity, assume $m = 1$ and we are given a signature $\mathbf{z}_1 = f_{\mathbf{s}_1}(\mathbf{c}_1) + \mathbf{y}_1$ as in Figure 7.3, where $\mathbf{y}_1$ is independently sampled according to the distribution $\mathcal{Y}$. Then, we have $\mathbf{y}_1 \in C = \mathbf{z}_1 + [-c_1 \cdot h, c_2 \cdot h]$ for all $c_1, c_2 \geq 1$ (see Theorem 7.1). Thus, let $c_1, c_2 = 1$. The probability of $\mathbf{y}_1 \in C = \mathbf{z} + [-h, h]$ for any fixed choice of $\mathbf{z}$ is $P[\ C\ ]$ under the distribution $\mathcal{Y}$, since $\mathbf{y}_1$ is independently sampled. Subsequently, the term $\mathbf{y}_2$ is secretly sampled from C according to the distribution $\mathcal{Y}/P[\ C\ ]$ by reusing the publicly accessible randomness C induced by $\mathbf{y}_1$. We now analyze the distribution of $\mathbf{y}_2$, when exploiting public and secret randomness. Indeed, the probability of the event $\mathbf{y}_2 = \mathbf{x}$ for $\mathbf{x} \in C$ is given by $P[\mathbf{y}_1 \in C \wedge \mathbf{y}_2 = \mathbf{x} \mid \mathbf{y}_2 \in C] = P[\,C\,] \cdot \mathcal{Y}(\mathbf{x})/P[\,C\,] = \mathcal{Y}(\mathbf{x})$ according to Theorem 7.2, which exactly coincides with the required distribution. The continuous case works similar and requires to consider the probability density function. Thus, we obtain $\max \|\mathbf{z}_1 - \mathbf{z}_2\|_\infty = \max \|\mathbf{z}_1 - f_{\mathbf{s}_2}(\mathbf{c}_2) + \mathbf{y}_2\|_\infty \leq (c_2 + c_1)h$. We observe that $\mathbf{z}_1$ is identified to be the source for public randomness and is subsequently required as a centroid for compression. With focus on compressing individual signatures, we can provide both features by a simulator for the distribution of signatures $\mathcal{Z}$ using a short random seed $\mathbf{r} \in \{0,1\}^\mu$ as input to a cryptographic hash function modeled as random oracle in combination with a rejection sampler. Following this approach, $\mathbf{z}_1$ is replaced by $\mathbf{r}$ and can deterministically be recovered at any time by use of the simulator. Thus, the signature size is bounded by $\lceil m \cdot \log 2h \rceil + \mu$ bits, (in general $\lceil m \cdot \log(c_2 + c_1)h \rceil + \mu$), where μ denotes the bit size of $\mathbf{r}$. Remarkably, it is even possible that arbitrary many other signers can exploit the same source of public randomness in exactly the same way. $\square$

7.2.3. Security

The following theorem essentially states that compressed signatures are as secure as uncompressed ones.

Theorem 7.4. *If there exists a (polynomial-time) adversary* $\mathcal{A}$ *that can break compressed signatures, there exists a (polynomial-time) algorithm* $\mathcal{B}^{\mathcal{A}}$ *that uses* $\mathcal{A}$ *in order to break the original signature scheme with uncompressed signatures.*

Proof. In order to prove that compressed signatures are as secure as standard uncompressed ones (e.g. standard GPV signatures), we proceed via a sequence of games. In fact, we use Algorithm 1 as an oracle whose output vectors are distributed like signatures and finally serve as a centroid. The challenge compressed signature is given by $(\mathbf{z}_1^*, \mathbf{z}_1^* - \mathbf{z}_2^*)$, where $\mathbf{z}_1^*$ denotes the centroid for compression.

Game 0

The game $\mathbf{G_0}$ represents the interaction of the challenger with the original compression scheme. The challenger is allowed to make polynomially many queries to a signing oracle producing compressed signatures $(\mathbf{z}_1, \mathbf{z}_1 - \mathbf{z}_2)$ in combination with the corresponding centroids $\mathbf{z}_1$ for compression. The centroids follow the same distribution $\mathcal{Z}$ as signatures. In addition, the challenger is given access to a random oracle H and an oracle OComp, where H is queried on messages of choice producing uniform random vectors. For a vector $\mathbf{c}$ distributed as $\mathcal{Z}$ as input, OComp produces in accordance to the generic construction in Figure 7.3 a compressed vector $(\mathbf{c}, \mathbf{c} - \mathbf{x})$, where $\mathbf{x}$ is distributed as $\mathcal{Z}$ and the centroid is given by $\mathbf{c}$.

Game 1

In game $\mathbf{G_1}$, we change the way the signing oracle responds to signature requests and the challenge compressed signature $(\mathbf{z}_1^*, \mathbf{z}_1^* - \mathbf{z}_2^*)$ is produced, but in a way that it introduces only a $negl(n)$ statistical distance to $\mathbf{G_0}$. The signing oracle now outputs only uncompressed signatures (standard signatures). The signing oracle from $\mathbf{G_0}$, which generates compressed signatures together with the corresponding centroids, is now simulated as follows. The signing oracle is queried in order to obtain an uncompressed signature $\mathbf{z}_2$. Subsequently, OComp is called on input $\mathbf{z}_2$, which then returns a compressed vector $(\mathbf{z}_2, \mathbf{z}_2 - \mathbf{z}_1)$ with $\mathbf{z}_2$ being its centroid. Finally, the compressed signature $(\mathbf{z}_1, \mathbf{z}_1 - \mathbf{z}_2)$ is output, where $\mathbf{z}_1$ acts as centroid. Since $\mathbf{z}_1$ and $\mathbf{z}_2$ are distributed according to $\mathcal{Z}$, the attacker cannot distinguish between the games $\mathbf{G_0}$ and $\mathbf{G_1}$.
The security proof shows that an attacker cannot distinguish between the games $\mathbf{G_0}$ and $\mathbf{G_1}$. In fact, we showed that an attacker, that can break signatures in $\mathbf{G_0}$, can also be used to attack uncompressed signatures in $\mathbf{G_1}$. And this concludes the proof. □

The theorem above mainly states that it is hard to break compressed signatures provided the hardness of the original signature scheme.

Note to the Compression Algorithm

We note that signature schemes due to [Lyu09, Lyu12, GOPS13, DDLL13] are not covered by our framework presented in Section 7.2.1. This is mainly due to the final rejection sampling step hiding the true (conditional) probability distribution of $\mathbf{y}$. In fact, rejection sampling is one of the Monte Carlo methods that allows to sample from arbitrary target distributions using an initial proposal distribution, which is used to envelop the target distribution and to generate samples efficiently. Hence, if the target distribution for signatures $\mathbf{z} = f_{\mathbf{s}}(\mathbf{c}) + \mathbf{y}$ is the uniform distribution $\mathcal{U}(B)$ over some range B, any distribution can be selected for $\mathbf{y}$ as long as the distribution of the sum $\mathbf{z}$ lies above the target distribution such that rejection sampling is applicable. The target distributions are always chosen to be independent from the secret key,

meaning that despite of different secret keys the signatures of different signers are identically distributed within a certain setting (identical parameters etc.). However, the real distribution of $\mathbf{y}$ is not the uniform distribution $\mathcal{U}(B)$, if one considers only samples that resulted in valid signatures $\mathbf{z}$, since some of them have been rejected. This can be attributed to the role of $f_{\mathbf{s}}(\mathbf{c})$ whose distribution adds together with the true distribution of $\mathbf{y}$ to the uniform distribution. As a result, we never get to see the (conditional) probability distribution of $\mathbf{y}$ that resulted in valid signatures, since this would leak information about the secret key. As a result, the algorithm from Section 7.2.1 is not applicable. If signatures would be generated in one run by use of uniformly sampled $\mathbf{y}$, we could collect many signature samples and apply the law of large numbers in order to gather further useful information about the secret key. Therefore, signature schemes with a rejection sampling step at the end are excluded from the presented compression scheme. Prior to applying the framework in Section 7.2.1, it has to be ensured that the signature scheme follows the abstract construction $\mathbf{z} = f_{\mathbf{s}}(\mathbf{c}) + \mathbf{y}$, where the scheme is secure even with public knowledge of the (conditional) probability distribution of $\mathbf{y}$.
In general, it is possible under some conditions to apply the convolution technique in order to sample sums of random variables in one run. Therefore, it is required to know the covariance matrix of at least one random variable in the sum. And the covariance matrix of the other random variable is determined based on the known covariance matrix and the target distribution. The framework above aims at signature schemes that generate signatures in one run. That is, we sample exactly one $\mathbf{y}$ for every signature $\mathbf{z}$. Then, we are guaranteed to have the correct conditional probability distribution, if it is publicly available. Exactly this case happens to occur for the lower part of a GPV signature. This part does not involve the secret key and the respective distributions are also known to a certain extent in advance. The lower part of a signature can be simplified to $\mathbf{z} = \mathbf{Ix} + \mathbf{c} + \sqrt{b}\mathbf{d}$, where $\mathbf{I}$ is the identity matrix, $\sqrt{b}\mathbf{d}$ is a scaled continuous Gaussian with known parameter and center, $\mathbf{x}$ and $\mathbf{c}$ are discrete Gaussians with small parameters and unknown centers. Setting $\mathbf{y} = \sqrt{b}\mathbf{d}$ allows to apply the compression scheme developed in the previous sections. The GPV signature scheme constructs signatures in one run by use of the convolution technique. The perturbation vector is independently sampled from the remaining part.

7.2.4. Compression Rate of Individual Signatures

Let $h = \max\limits_{\mathbf{s},\mathbf{c}} \|f_{\mathbf{s}}(\mathbf{c})\|_\infty$ and $\mathbf{z}$ be the centroid generated by use of the seed $\mathbf{r}$ of size μ bits serving as input to a simulator for signatures. The compression rate of an individual signature $\mathbf{z}_1$:

$$\theta(1) = 1 - \frac{\mathsf{size}(\mathbf{z}_{CS})}{\mathsf{size}(\mathbf{z}_1)} = 1 - \frac{\lceil m \cdot \log 2h \rceil + \mu}{\lceil m \cdot \log \max \|\mathbf{z}_1\|_\infty \rceil},$$

where the denominator indicates the maximum bit size of an uncompressed signature. In many state-of-the-art signature schemes, we have $\max \|\mathbf{z}\|_\infty = \tilde{O}(n)$ or $\tilde{O}(n^{1/2})$ dependend on the scheme and its instantiation with $\max \|\mathbf{z} - \mathbf{z}_1\|_\infty = o(n)$, when applying the compression algorithm from Section 7.2.1. Following this, we achieve compression rates of roughly

$$\tau(1) = 1 - \frac{o(\log n)}{\tilde{O}(\log n)}$$

implying asymptotically an improvement factor of $O(\log n)$.

7.3. Compression Scheme for GPV Signatures

In the following section, we provide a detailed description of how to apply the framework from Section 7.2 on GPV signatures that are produced by means of the trapdoor construction [MP12]. We refer to Section 6.3.5 for a description of the optimized signature scheme.

7.3.1. Tools

Signatures generated within this framework essentially resemble Schnorr signatures $(\mathbf{z}^{(1)}, \mathbf{z}^{(2)}) = \begin{bmatrix}\mathbf{R}\\ \mathbf{I}\end{bmatrix} \cdot \mathbf{x} + \mathbf{p}$ with $\mathbf{z}^{(1)} \in \mathbb{Z}^{2n}$ and $\mathbf{z}^{(2)} \in \mathbb{Z}^{nk}$. Hence, a signature is of the form $\mathbf{z} = f_{\mathbf{s}}(\mathbf{c}) + \mathbf{y}$ in accordance to the abstract representation from Section 7.2 with $\mathbf{s} = \begin{bmatrix}\mathbf{R}\\ \mathbf{I}\end{bmatrix}$. It is even possible to split the signature into $\mathbf{z} = (f_{\mathbf{R}}(\mathbf{c}) + \mathbf{y}^{(1)}, f_{\mathbf{I}}(\mathbf{c}) + \mathbf{y}^{(2)})$. The exact specifications of $f(\cdot)$, $\mathbf{y}$ and $\mathbf{c}$ will be given below. However, we are primarily concerned with the lower part of the signature $\mathbf{z}^{(2)} = f_{\mathbf{I}}(\mathbf{c}) + \mathbf{y}^{(2)} = \mathbf{I} \cdot \mathbf{x} + \mathbf{p}^{(2)}$ due to the large difference of magnitudes of $\mathbf{x}$ and $\mathbf{p}^{(2)}$.

Suppose we have l parties $S_1, \ldots, S_l$ that want to sign individual messages $\mathsf{msg}_1, \ldots, \mathsf{msg}_l$. For the sake of simplicity, we restrict to the case, where $l = 2$ and both parties use in accordance to the optimized signature scheme in Section 6.3.5 the same signing parameter s, security parameter n and modulus $q = 2^k$, meaning that the trapdoor functions $f_{\mathbf{A}_1}$ and $f_{\mathbf{A}_2}$ of the signers have the same domain $B = \{\ \mathbf{z} \in \mathbb{Z}^{n(2+k)} \mid \|\mathbf{z}\| \leq s\sqrt{n(2+k)}\ \}$ and range $R = \mathbb{Z}_q^n$. Our compression strategy focuses on the signature subvector $\mathbf{z}^{(2)} = \mathbf{x} + \mathbf{p}^{(2)} \in \mathbb{Z}^{n \cdot k}$, where $\mathbf{p}^{(2)}$ is distributed as $\tilde{\mathbf{p}}^{(2)} + \mathcal{D}_{\mathbb{Z}^{n\cdot k} - \tilde{\mathbf{p}}^{(2)}, r}$ with $\tilde{\mathbf{p}}^{(2)} \leftarrow \sqrt{b} \cdot \mathcal{D}_1^{n \cdot k}$ and $\mathbf{x}$ is sampled from $\mathcal{D}_{\Lambda^\perp_{\mathbf{v}_i}(\mathbf{G}), r}$ with $\mathbf{v}_i = H(\mathsf{msg}_i) - \mathbf{A}_i \mathbf{p}_i$ for $i = 1, 2$ and $\mathbf{p} = (\mathbf{p}^{(1)}, \mathbf{p}^{(2)})$. This leads to the following representation of $\mathbf{z}^{(2)} = f_{\mathbf{I}}(\mathbf{c}) + \mathbf{y}^{(2)}$, where

- $\mathbf{y}^{(2)} \leftarrow \tilde{\mathbf{p}}^{(2)} = \sqrt{b} \cdot \mathbf{d}$ with $\mathbf{d} \leftarrow \mathcal{D}_1^{n \cdot k}$
- $f_{\mathbf{I}}(\mathbf{x}, \mathbf{y}) := \mathbf{I} \cdot \mathbf{x} + \mathcal{D}_{\mathbb{Z}^{n \cdot k} - \mathbf{y}^{(2)}, r}$ with $\mathbf{c} = (\mathbf{x}, \mathbf{y})$.

Based on this representation we can apply the tools developed in Section 7.2. In fact, we have

$$\mathbf{y}^{(2)} \in \mathbf{z}^{(2)} + [-h, h]^{nk} \iff \mathbf{d} \in C = \frac{\mathbf{z}^{(2)}}{\sqrt{b}} + \left[-\frac{h}{\sqrt{b}}, \frac{h}{\sqrt{b}}\right]^{nk}, \;\; h = \max \|f_{\mathbf{I}}(\mathbf{c})\|_\infty \,.$$

Prior to stating the main theorem of this section, which indicates an upper-bound for the size of a compressed signature, we prove some useful statements. For instance, in Lemma 7.5 we essentially show that we can sample any continuous Gaussian $d \leftarrow \mathcal{D}_1$ by first sampling a set B_i with probability $P[\ B_i\ ]$ and then selecting a continuous Gaussian from B_i according to the probability densitity function $f(x \mid x \in B_i)$. In Lemma 7.6 we provide a more general result than [Ban95, Lemma 2.4]. It is a very helpful instrument in order to bound sums of discrete Gaussians having different supports Λ_i, parameters s_i and centers $\mathbf{c}_i$. It trivially subsumes Lemma [Ban95, Lemma 2.4]. By use of this result we give an upper-bound for h and hence for the compressed signature. In Theorem 7.7 we prove that an arbitrary signer, that is different from the first one, can reuse public randomness following essentially the same arguments as in Theorem 7.2 by sampling its own continuous Gaussian from C such that the difference of the lower part of its signature to the centroid $\mathbf{z}^{(2)}$ is sufficiently small.

Lemma 7.5. *Let X be distributed according to the countinous Gaussian distribution $\mathcal{D}_1$ with parameter $s = 1$ and center $\mu = 0$. Directly sampling $d \leftarrow \mathcal{D}_1$ is equivalent to first sampling a set B_i with probability $P[B_i] = \int_{B_i} e^{-\pi x^2} dx$ and then sampling a continuous Gaussian from B_i according to the probability density function $f(x \mid x \in B_i) = \frac{1}{P[B_i]} e^{-\pi x^2}$ for $x \in B_i$, where B_i depicts a partition of $\mathbb{R}$ for $1 \le i \le n$.*

Proof. The probability densitity function of a sample distributed according to $\mathcal{D}_1$ is $f(x) = e^{-\pi x^2}$. Using conditional probability rules we have

$$\begin{aligned} P[B_i] \cdot f(x \mid x \in B_i) &= P[B_i] \cdot \frac{1}{P[B_i]} e^{-\pi x^2} \\ &= e^{-\pi x^2} \text{ for } x \in B_i, \; P[B_i] = \int_{B_i} e^{-\pi x^2} dx, \end{aligned}$$

which exactly coincides with the probability density function of a continuous Gaussian with parameter 1. □

Lemma 7.6. *Let $(\Lambda_i)_{1 \le i \le n} \in \mathbb{R}^{n_i}$ be a sequence of n_i-dimensionial lattices. Then for any reals $s_i \ne s_j > 0$ such that $1 \le i, j \le n$ and $T > 0$, and $x_i \in \mathbb{R}^{n_i}$, we have*

$$\Pr_{d_i \sim \mathcal{D}_{\Lambda_i, c_i, s_i}} [|\sum_{i=1}^{k} \langle \mathbf{x}_i, \mathbf{d}_i - \mathbf{c}_i \rangle| \ge T \cdot \|\ (s_1\mathbf{x}_1, \ldots, s_k\mathbf{x}_k)\ \|] < 2e^{-\pi T^2}$$

Proof. One can easily verify that $\mathcal{D}_{\Lambda_i,\mathbf{c}_i,s_i}$ and $s_i \cdot \mathcal{D}_{\Lambda_i',\mathbf{c}_i',1}$ define the same distribution, where Λ_i' and $\mathbf{c}_i'$ denote the scaled lattice Λ_i/s_i and center $\mathbf{c}_i/s_i$ respectively. In the rest of the proof, we will use this equivalence when considering the distribution on the lattice Λ_i. The cartesian product $\mathcal{L} = \Lambda_1/s_1 \times \cdots \times \Lambda_k/s_k$ of lattices is again a $(\sum_i n_i)$-dimensional lattice since we can always construct basis vectors for $\mathcal{L}$ using the basis vectors of Λ_i. For any countable set $A = A_1 \times \cdots \times A_k \subset \mathcal{L}$ the probability measure on it is defined by $\rho_{(\mathbf{c}_1',\ldots,\mathbf{c}_k')}(A) = \prod_i \rho_{c_i'}(A_i)$. Let $\mathbf{x} = (\mathbf{x_1}, \ldots, \mathbf{x_k})$ define the vector composed by k subvectors $\mathbf{x_i} \in \mathbb{R}^{n_i}$ and $\mathbf{c}' = (\mathbf{c}_1', \ldots, \mathbf{c}_k')$ respectively. Then we obtain the following equalities:

$$\begin{aligned}
& \langle \mathbf{x}, (\mathcal{D}_{\Lambda_1,\mathbf{c}_1,s_1} - \mathbf{c}_1, \ldots, \mathcal{D}_{\Lambda_k,\mathbf{c}_k,s_k} - \mathbf{c}_k) \rangle && (7.1)\\
= \; & \langle \mathbf{x_1}, \mathcal{D}_{\Lambda_1,c_1,s_1} - \mathbf{c}_1 \rangle + \cdots + \langle \mathbf{x_k}, \mathcal{D}_{\Lambda_k,\mathbf{c}_k,s_k} - \mathbf{c}_k \rangle && (7.2)\\
= \; & \langle \mathbf{x_1}, s_1 \cdot (\mathcal{D}_{\Lambda_1',\mathbf{c}_1',1} - \mathbf{c}_1') \rangle + \cdots + \langle \mathbf{x_k}, s_k \cdot (\mathcal{D}_{\Lambda_k',\mathbf{c}_k',1} - \mathbf{c}_k') \rangle && (7.3)\\
= \; & \langle s_1 \cdot \mathbf{x_1}, \mathcal{D}_{\Lambda_1',\mathbf{c}_1',1} - \mathbf{c}_1' \rangle + \cdots + \langle s_k \cdot \mathbf{x_k}, \mathcal{D}_{\Lambda_k',\mathbf{c}_k',1} - \mathbf{c}_k' \rangle && (7.4)\\
= \; & \langle (s_1 \cdot \mathbf{x_1}, \ldots, s_k \cdot \mathbf{x_k}), (\mathcal{D}_{\Lambda_1',\mathbf{c}_1',1} - \mathbf{c}_1', \ldots, \mathcal{D}_{\Lambda_k',\mathbf{c}_k',1} - \mathbf{c}_k') \rangle && (7.5)\\
= \; & \langle (s_1 \cdot \mathbf{x_1}, \ldots, s_k \cdot \mathbf{x_k}), \mathcal{D}_{\mathcal{L},\mathbf{c}',1} - \mathbf{c}' \rangle \,. && (7.6)
\end{aligned}$$

The claim now follows from equation 7.6 and [Pei07, Lemma 5.1] with unit vector $\frac{(s_1 \cdot \mathbf{x_1}, \ldots, s_k \cdot \mathbf{x_k})}{\|(s_1 \cdot \mathbf{x_1}, \ldots, s_k \cdot \mathbf{x_k})\|}$ □

If we set $T \approx 4.69$ the probability of that inequality to hold is less than 2^{-100}. In the following, we state our main theorem of this section, which enables an arbitrary group of signers to compress signatures.

Theorem 7.7. *Assume a signer is given access to an oracle (e.g. a signing or discrete Gaussian oracle) providing spherically distributed signatures $\mathbf{z} = (\mathbf{z}^{(1)}, \mathbf{z}^{(2)})$ with $\mathbf{z}^{(1)} \in \mathbb{Z}_q^{2n}$, $\mathbf{z}^{(2)} \in \mathbb{Z}_q^{nk}$ and parameter s according to the signing algorithm from Section 6.3.5. He is then able to produce spherically distributed signatures $\mathbf{z}_1 = (\mathbf{z}_1^{(1)}, \mathbf{z}_1^{(2)})$ such that the following bound on $\mathbf{z}^{(2)} - \mathbf{z}_1^{(2)}$ holds with overwhelming probability*

$$\log \| \mathbf{z}^{(2)} - \mathbf{z}_1^{(2)} \|_\infty \leq 7 \,.$$

Proof. Consider the subvector $\mathbf{z}^{(2)} = f_{\mathbf{I}}(\mathbf{x}, \mathbf{y}) + \mathbf{y}^{(2)} \in \mathbb{Z}^{n \cdot k}$ consisting of the last $n \cdot k$ entries of $\mathbf{z} \in \mathbb{Z}^{n(k+2)}$, that is generated according to the optimized signature scheme from Section 6.3.5. As stated in [Pei10] the subvector $\mathbf{z}^{(2)}$ can be written as $\mathbf{z}^{(2)} = \mathbf{x} + \lceil \mathbf{c} \rfloor_a = \mathbf{x} + \mathbf{c} + \mathcal{D}_{\mathbb{Z}^{n \cdot k} - \mathbf{c}, a}$, where $\lceil \cdot \rfloor_a$ denotes the randomized rounding operation and $\mathbf{c} = \sqrt{s^2 - 5a^2} \cdot \mathbf{d}$, $\mathbf{d} \leftarrow_R \mathcal{D}_1^{n \cdot k}$ with a as above. By [MP12] the parameter s can be as small as $\sqrt{\mathbf{s}_1(R)^2 + 1} \cdot \sqrt{6} \cdot a \gtrsim \mathbf{s}_1(R) \cdot \sqrt{6} \cdot a$ (see Section 6.3.2) and when applying [MP12, Lemma 2.9], one sets $\mathbf{s}_1(R)$ to at least $1/\sqrt{2\pi} \cdot (\sqrt{2n} + \sqrt{nk}) \cdot \alpha q$.

Using $b = s^2 - 5a^2$, one can deduce a range C for the continuous Gaussian vector used to generate $\mathbf{z}^{(2)}$. Thereto, we have to compute $h = \max \|f_{\mathbf{I}}(\mathbf{c})\|_\infty$ providing a bound to the sum $x_i + \mathcal{D}_{\mathbb{Z}-c_i,a}$. One notices that $\mathcal{D}_{\mathbb{Z}-c_i,a}$ and $-c_i + \mathcal{D}_{\mathbb{Z},c_i,a}$ are identically distributed. As per Lemma 7.6 the sum is within the range $[-4.7 \cdot \sqrt{r^2+a^2}, 4.7 \cdot \sqrt{r^2+a^2}]$, except with negligible probability. As a result, it is possible to determine a concrete range for the continuous Gaussian vector $\mathbf{d}$ by employing only public data following Section 7.2.1. In fact, we have

$$\mathbf{d} \in C = \frac{\mathbf{z}^{(2)}}{\sqrt{b}} + \left[-\frac{h}{\sqrt{b}}, \frac{h}{\sqrt{b}}\right]^{nk} \text{ with } h = 4.7 \cdot \sqrt{5}a\,.$$

The set C is publicly accessible and can, thus, be read by all parties. A complete secretly sampled continuous Gaussian $\mathbf{d}$ implies $C = \mathbb{R}^{nk}$, whereas $C = \mathbf{d}$ in case $\mathbf{d}$ is completely accessible to the public (see Theorem 7.1).

On the one hand, one observes that public randomness induced by the set C can be viewed and exploited by a potential adversary (and anyone else) in order to launch an attack against the underlying cryptographic primitive. Consequently, the security of any cryptosystem should only be based on secretly sampled random strings that can not be extracted publicly. In fact, we prove in Theorem 7.4 that compressed signatures, that employ public randomness, are secure assuming the hardness to break standard signatures. On the other hand, arbitrary many other signers can take advantage of the available public randomness utilizing it for building own signatures. In Section 7.5.2, we give a description of our multi-signer compression scheme that makes use of this feature. Since each signer operates with its own secret key, that is independently generated, exploiting public randomness has no impact on security. On the contrary, the generation of public random strings can be delegated to other institutions providing the desired distributions on demand. Specifically, in Section 7.3.3 we highlight the usage of a short random seed $\mathbf{r}$ serving as input to a discrete Gaussian sampler acting as a simulator for signatures. The output vector is used in order to extract the required public randomness and more importantly to replace the large centroid $\mathbf{z}^{(2)}$. As a result, it suffices to store the seed instead of the large centroid.

The continuous Gaussian $\mathbf{d}$ was independently sampled and lies in C with probability $P[\ C\ ]$ (see Theorem 7.2). We say C occured, if $\mathbf{d} \in C$. Any other signer can now secretly sample a continuous Gaussian $\mathbf{d}_1 \leftarrow_R \mathcal{D}_1^{nk}$ conditioned on $\mathbf{d}_1 \in C$ according to the probability density function $f(\ \mathbf{x} \mid \mathbf{x} \in C\ )$. Reusing public randomness causes the random vectors $\mathbf{d}_1$ to be distributed following the probability density function $f(\ \mathbf{x} \mid \mathbf{x} \in C\ ) \cdot P[\ C\ ] = e^{-\pi\|\mathbf{x}\|_2^2}$, which perfectly coincides with the required distribution $\mathbf{d}_1 \leftarrow_R \mathcal{D}_1^{nk}$ (see Section 7.2.2). Following this approach, the signer needs only to secretly generate its own continuous Gaussian vector $\mathbf{d}_1$ by sampling from the provided range C, for example with rejection sampling, such that $\mathbf{d}_1 \sim \mathcal{D}_1^{nk}$ is satisfied. In fact, a larger range $C' \supseteq C$ can be selected if desired (see Theorem 7.3). Intuitively, this strategy causes the vectors $\mathbf{z}_1^{(2)}$ to be distributed around the centroid $\mathbf{z}^{(2)}$ (see Figure 7.1). As per construction we have $\mathbf{d}_1 \in C$. By means of Lemma 7.6 it is possible to derive an upper-bound on the norm $\mathbf{z}^{(2)} - \mathbf{z}_1^{(2)}$, where $\mathbf{x} \leftarrow_R \mathcal{D}_{\Lambda^{\perp}_{\mathbf{v}_1}(\mathbf{G}),r}$ and $\mathbf{v}_1$ behave as described in the signing algorithm

$$\begin{aligned} \| \mathbf{z}^{(2)} - \mathbf{z}_1^{(2)} \|_\infty &= \| \sqrt{b} \cdot \tfrac{\mathbf{z}^{(2)}}{\sqrt{b}} - \mathcal{D}_{\Lambda^{\perp}_{\mathbf{v}_1}(\mathbf{G}),r} - \sqrt{b} \cdot \mathbf{d}_1 - \mathcal{D}_{\mathbb{Z}^{n \cdot k} - \sqrt{b} \cdot \mathbf{d}_1, a} \|_\infty \\ &\leq \| \sqrt{b} \cdot (\tfrac{\mathbf{z}^{(2)}}{\sqrt{b}} - \mathbf{d}_1) \|_\infty + \| \mathcal{D}_{\Lambda^{\perp}_{\mathbf{v}_1}(\mathbf{G}),r} - \mathcal{D}_{\mathbb{Z}^{n \cdot k} - \sqrt{b} \cdot \mathbf{d}_1, a} \|_\infty \\ &\leq 2 \cdot 4.7 \cdot \sqrt{5}a < 128 \,. \end{aligned}$$

Each entry of $\mathbf{z}^{(2)} - \mathbf{z}_1^{(2)}$ occupies for $n \leq 2^{70}$ at most 7 bits of memory, except with negligible probability. This value is almost independent of n, which increases the incentive to use higher security parameters and thus causing larger compression factors. On the other hand, a signature is distributed according to a discrete Gaussian with parameter s. Each entry has magnitude of at most $4.7 \cdot s$ except with probability of at most 2^{-100}. □

The following result shows that it is even possible to leave out the first n entries from $\mathbf{z}$, which can always be recovered due to the existence of the identity submatrix in $\mathbf{A}$.

Lemma 7.8. *Suppose* $\mathbf{z} = (\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \mathbf{z}^{(3)})$ *is a signature for a message* msg *with hash value* $H(\mathsf{msg})$ *under public key* $\mathbf{A} = [\, \mathbf{I}_n \mid \bar{\mathbf{A}} \mid \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}]$, *where* $\mathbf{z}^{(1)},\ \mathbf{z}^{(2)} \in \mathbb{Z}^n$. *Then, the signer requires only to output* $(\mathbf{z}^{(2)}, \mathbf{z}^{(3)}) \in \mathbb{Z}^{n(k+1)}$ *in order to ensure correct verification.*

Proof. The verifier computes $\mathbf{t} = H(\mathsf{msg})$ and defines

$$\mathbf{z}^{(1)} := \mathbf{t} - [\bar{\mathbf{A}} \mid \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}] \cdot (\mathbf{z}^{(2)}, \mathbf{z}^{(3)}) \in \mathbb{Z}^n \,.$$

Then, the verifier needs only to check the validity of $\|\mathbf{z}\| \leq s\sqrt{n(k+2)}$, since $\mathbf{A} \cdot \mathbf{z} = H(\mathsf{msg})$ holds per definition of $\mathbf{z}^{(1)}$. □

7.3.2. Conditional Rejection Sampling

In this section we briefly discuss how to perform the rejection sampling step based on conditional probabilities. The principal is identical to the conventional rejection sampling algorithm from previous works. Specifically, we want to sample a vector $\mathbf{d}$ from $C = \frac{\mathbf{z}}{\sqrt{b}} + [-\frac{4.7 \cdot \sqrt{5}a}{\sqrt{b}}, \frac{4.7 \cdot \sqrt{5}a}{\sqrt{b}}]^{nk}$ according to probability density function $f(\mathbf{x} \mid \mathbf{x} \in C) = e^{-\pi \|\mathbf{x}\|_2^2} / P[\, C\,] = \prod_{i=1}^{nk} e^{-\pi x_i^2} / P[\, C_i\,]$ with $C_i = \frac{z_i}{\sqrt{b}} + [-\frac{4.7 \cdot \sqrt{5}a}{\sqrt{b}}, \frac{4.7 \cdot \sqrt{5}a}{\sqrt{b}}]$ and

$$P[C_i] = \int_{C_i} e^{-\pi x^2} dx \,.$$

By means of a simple rejection sampling algorithm, we can sample each entry of $\mathbf{d}$ independently from $C_i = \frac{z_i}{\sqrt{b}} + [-\frac{4.7 \cdot \sqrt{5}a}{\sqrt{b}}, \frac{4.7 \cdot \sqrt{5}a}{\sqrt{b}}]$. For example, one samples a uniform random element d_i from C_i and a second random element u_i from the interval $[\,0,1\,]$. We accept d_i, if $u_i < e^{-\pi d_i^2}$, otherwise we reject and resample. Due to the compact intervals of small width, the rejection sampling algorithm performs very efficient. This conditional rejection sampler can be adapted to other distributions as well.

7.3.3. Single-Signer Compression Scheme in the GPV Setting

Applying the framework introduced in Section 7.2.1 in conjunction with the tools described in Section 7.3.1 (particularly Theorem 7.7 required) empowers an individual signer to compress its own signatures. We, therefore, present the required steps of our single signer compression scheme below. Thus, let GPV signatures be represented by the tuple $\mathbf{z} = (\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \mathbf{z}^{(3)}) \in \mathbb{Z}^{n+n+nk}$.

In fact, our main goal is to give a description of how to replace the large centroid by a short uniform random string that is used to produce vectors being distributed just like GPV signatures. As a result, we have a mechanism to simulate signatures such that the output vectors take over the role of the centroid. In fact, lattice-based GPV signatures are distributed just like discrete Gaussian vectors. Therefore, a discrete Gaussian sampler can be used as a simulator for signatures providing the required public randomness. It is a well-known fact that a discrete Gaussian can be generated by use of rejection sampling or other sampling algorithms such as FastCDT introduced in Section 5.1 that are parametrized by sequences of uniformly distributed numbers [GPV08, Lyu12] supplied, for example, by a cryptographic hash function modeled as random oracle. But it is also possible to produce discrete Gaussians by means of a continuous Gaussian sampler in combination with the technique from [Pei10].
Therefore, suppose we want to sample a vector being distributed negligibly close to a discrete Gaussian with parameter s representing the centroid as assumed by Theorem 7.7. According to the proof, we output a vector $\mathbf{z}^{(2)}$ distributed as $\mathbf{x} + \mathbf{c} + \mathcal{D}_{\mathbb{Z}^{n\cdot k} - \mathbf{c}, a}$, where $\mathbf{c} = \sqrt{s^2 - 5a^2} \cdot \mathbf{d}$, $\mathbf{d} \leftarrow_R \mathcal{D}_1^{n\cdot k}$ and $\mathbf{x} \leftarrow_R \mathcal{D}_{\mathbb{Z}^{n\cdot k}, r}$ holds. Following [Pei10] this is equivalent to first generating a continuous Gaussian vector $\mathbf{d}$ with parameter 1, multiplying it with $\sqrt{b}$ for $b = s^2 - 5a^2$ and rounding each component of the vector to a nearby integer using the randomized rounding operation with parameter a. This produces a vector distributed as $\lceil \sqrt{b} \cdot \mathbf{d} \rfloor_a = \sqrt{b} \cdot \mathbf{d} + \mathcal{D}_{\mathbb{Z}^{n\cdot k} - \sqrt{b}\cdot \mathbf{d}, a}$. Note that the randomized rounding operation behaves in fact like a discrete Gaussian. Thus, for the scheme to work, a potential signer samples a fresh random seed $\mathbf{r}$ of size μ bits as input to a cryptographic hash function modeled as random oracle outputting a sequenence of random numbers that in turn serve as input to a discrete Gaussian sampler. Applying the compression algorithm (Algorithm 2) and using Theorem 7.7, the signer outputs the public seed $\mathbf{r}$, which generates the centroid $\mathbf{z}^{(3)}$, and a compressed signature $(\mathbf{z}_1^{(2)}, \mathbf{z}^{(3)} - \mathbf{z}_1^{(3)})$, where $\mathbf{z}_1^{(2)}$ contains only n entries of the signature $\mathbf{z}_1$ as per Lemma 7.8. The size of the compressed signature amounts to approximately $n(\log(4.7 \cdot s) + k \cdot 7) + \mu$ bits as compared to $n(k+2) \cdot \log(4.7 \cdot s)$ bits without compression. The verifier receives the compressed

signature and computes the discrete Gaussian $\mathbf{z}^{(2)}$ using $\mathbf{r}$. He then uncompresses the signature to $(\mathbf{z}_1^{(1)}, \mathbf{z}_1^{(2)}, \mathbf{z}_1^{(3)})$ and verifies the GPV signature by invoking VerifyGPV (see Section 6.3.5).

7.3.4. Analysis of Compressed Signatures

In this section we analyze the compression rate of the signature scheme. A simple and practical way of comparing compressed signatures is to use the size ratio of signatures before $\mathsf{size}(\mathbf{z}_i)$ and after compression $\mathsf{size}(\mathbf{z}_{CS})$. By

$$\theta(l) = 1 - \frac{\mathsf{size}(\mathbf{z}_{CS})}{\sum_i^l \mathsf{size}(\mathbf{z}_i)}$$

we define the compression rate, which represents the amount of storage that has been saved due to compression, where l denotes the number of signers (resp. signatures). For $l = 1$, we obtain the compression rate for a single signer.

Asymptotical View

For analyzing the compression rate and its asymptotics, we first consider a lower bound on the compression rate starting with the single signer case. Let $\mathbf{z} \leftarrow \mathcal{D}_{\mathbb{Z}^{nk},s}$ be the centroid sampled by a simulator for signatures such as a discrete Gaussian sampler using a seed $\mathbf{r} \in \{0,1\}^\mu$ as input. A compressed signature $(\mathbf{r}, \mathbf{z}_1^{(2)}, \mathbf{z} - \mathbf{z}_1^{(3)})$ consists of $\mathbf{z}_1^{(2)}$ of size $n \cdot \lceil \log(4.7 \cdot s) \rceil$ bits, $\mathbf{z} - \mathbf{z}_1^{(3)}$ of size $n \cdot k \cdot \lceil \log(2 \cdot 4.7 \cdot \sqrt{5}a) \rceil$ bits and a short seed $\mathbf{r}$ of size μ bits. Without compression, however, the size of an individual standard GPV signature amounts to $n \cdot (k+2) \cdot \lceil \log(4.7 \cdot s) \rceil$ bits

$$\begin{aligned}
\theta(1) &= 1 - \frac{n \cdot \lceil \log(4.7 \cdot s) \rceil + n \cdot k \cdot \lceil \log(\sqrt{448.8}a) \rceil + \mu}{n \cdot (k+2) \cdot \lceil \log(4.7 \cdot s) \rceil} && (7.7)\\
&\geq 1 - \left(\frac{1}{k+2} + \frac{\lceil \log(\sqrt{448.8}a) \rceil + 1}{\lceil \log(4.7 \cdot s) \rceil} \right) && (7.8)\\
&= 1 - \left(\frac{1}{k+2} + \frac{o(\log(\ln n))}{O(\log(n))} \right) . && (7.9)
\end{aligned}$$

The compression factor converges for increasing n towards $1 - 1/k+2$, if k is chosen to be constant. But in fact, since the parameter s grows with increasing n, it is required to increase k as a function of n for the scheme to be secure. Typically, one requires $q = 2^k = poly(n)$, which is equivalent to $k = O(\log n)$, implying an improvement factor of approximately $\lg n$. In this case, the compression factor converges towards 1, which is asymptotically unbounded.

Concrete View

A more practical way of measuring the concrete compression rates is to consider the length of a compressed signature and subsequently deduce its storage requirements. Thus, we recall the representation of a compressed signature according to Theorem 7.7, where $\mathbf{x} \leftarrow_R \mathcal{D}_{\Lambda^\perp_{\mathbf{b}}(\mathbf{G}),r}$ for $\mathbf{b} = H(\mathsf{msg})$ and $\mathbf{v} \leftarrow_R \mathcal{D}_{\mathbb{Z}^{nk}-\sqrt{b}\mathbf{d},a}$, then it follows

$$\begin{aligned}\left\| \mathbf{z} - \mathbf{z}_1^{(2)} \right\|_2 &= \left\|\sqrt{b} \cdot \tfrac{\mathbf{z}}{\sqrt{b}} - \mathbf{x} - \sqrt{b} \cdot \mathbf{d} - \mathbf{v}\right\|_2 \\ &\leq \left\|\sqrt{b} \cdot (\tfrac{\mathbf{z}}{\sqrt{b}} - \mathbf{d}) - \mathbf{x}\right\|_2 + \|\mathbf{v}\|_2 \,.\end{aligned}$$

We now consider the expression $\|\mathbf{v}\|_2$, which can be rewritten as $\sqrt{nk}\sqrt{\frac{1}{n\cdot k}\sum_{i=0}^{n\cdot k} v_i^2}$. By the law of large numbers and due to the huge number of samples the estimator $\frac{1}{n\cdot k}\sum_{i=0}^{n\cdot k} v_i^2$ essentially equals to $E[v_i^2] = a^2$ such that $\|\mathbf{v}\|_2$ can be approximated by $\sqrt{n\cdot k} \cdot \sqrt{E[v_i^2]} = a\sqrt{n \cdot k}$. The first expression, however, is a little bit tricky to approximate, since the entries d_i lie in different sets dependend on the entries of $\mathbf{z}$. Signatures produced by the GPV framework basically follow the discrete Gaussian distribution. As a consequence, the random variables $T_i = (\frac{z_i}{\sqrt{b}} - d_i)$ with $z_i \sim \mathcal{D}_{\mathbb{Z},s}$ and $d_i \sim \mathcal{D}_1, d_i \in C_i$ are independent and identically distributed such that the law of large numbers applies. Moreover, the squared entries x_i^2 of $\mathbf{x}$ are of finite variance and independent from T_i. For large enough samples, we obtain

$$\begin{aligned}\tfrac{1}{n\cdot k}\sum_{i=1}^{nk}(\sqrt{b}(\tfrac{z_i}{\sqrt{b}} - d_i) - x_i)^2 &\rightarrow E[(\sqrt{b}(\frac{z_i}{\sqrt{b}} - d_i) - x_i)^2] \\ &= E[x_i^2] + b \cdot E[T_i^2] \\ &= r^2 + b \cdot \sum_{y\in\mathbb{Z}} P[z_i = y] \cdot E[T_i^2 \mid z_i = y] \\ &\leq r^2 + b \cdot \max_{y\in\mathbb{Z}} E[T_i^2 \mid z_i = y]\,.\end{aligned}$$

In order to find the maximum conditional expectation value for each considered parameter selection n and k, we derive an upper bound for

$$\begin{aligned}c^2 &= b \cdot \max_{0\leq i\leq\lceil 4.7\cdot s\rceil} E\left[\left(\frac{i}{\sqrt{b}} - d_i\right)^2 \mid d_i \leftarrow_R \mathcal{D}_1, d_i \in C_i\right], \\ C_i &= \frac{i}{\sqrt{b}} + [-h,h] \text{ for } h = \frac{4.7\cdot\sqrt{5}a}{\sqrt{b}}\,.\end{aligned}$$

For $i > 0$, the conditional expectation is given by

$$\begin{aligned}\frac{1}{P[C_i]}\int_{C_i}\left(\frac{i}{\sqrt{b}}-x\right)^2 e^{-\pi x^2}dx &\leq \frac{1}{P[C_i]}\int_{C_i}\left(\frac{i}{\sqrt{b}}-x\right)^2 \cdot e^{-\pi(\frac{i}{\sqrt{b}}-h)^2}dx \\ &= \frac{2\cdot e^{-\pi(\frac{i}{\sqrt{b}}-h)^2}}{3P[C_i]}\cdot\left(\frac{4.7\sqrt{5}a}{\sqrt{b}}\right)^3,\end{aligned}$$

since $e^{-\pi(\frac{i}{\sqrt{b}}-h)^2} \leq e^{-\pi x^2}$ for $x \in C_i$. As a result, we deduce $\left\|\sqrt{b}\cdot(\frac{\mathbf{z}}{\sqrt{b}}-\mathbf{d})\right\|_2 \leq c\sqrt{n\cdot k}$. Subsequently, we can bound the length of $\mathbf{z}-\mathbf{z}_1^{(2)}$ by $\left\|\mathbf{z}-\mathbf{z}_1^{(2)}\right\|_2 \leq (\sqrt{c^2+r^2}+a)\cdot\sqrt{n\cdot k}$. Following this approach in combination with Lemma 6.2, we can estimate the compression rate more precisely. The compressed signature requires $\lceil n\cdot k\cdot(1+\log(\sqrt{c^2+r^2}+a))\rceil$ bits and the seed $\mathbf{r}$ occupies at most n bits of memory. A standard GPV signature requires $\lceil n(k+2)(1+\log s)\rceil$ bits. It follows

$$\theta(1) = 1-\frac{\lceil n\cdot(1+\log s)\rceil+\lceil n\cdot k\cdot(1+\log(\sqrt{c^2+r^2}+a))\rceil+n}{\lceil n(k+2)(1+\log s)\rceil},$$

where c^2 is upper bounded by $\frac{2be^{-\pi(\frac{i}{\sqrt{b}}-h)^2}}{3P[C_i]}\left(\frac{4.7\sqrt{5}a}{\sqrt{b}}\right)^3$. The storage improvement factor is simply the inverse of the fraction, i.e. $(1-\theta(1))^{-1}$.

Compression Rate in the Multi-Signer Setting

For $l > 1$ the compression factor is slightly higher, because only one seed of size μ bits is required instead of $l\cdot\mu$ bits. Furthermore, the computational costs decrease due to a single call of the discrete Gaussian sampler as opposed to l calls in case without aggregation.

7.3.5. Entropy of Public and Secret Randomness

Measuring the public and secret portion of randomness requires to consider the entropy of the relevant quantities. The entropy $h(X)$ represents a mass for the amount of uncertainty stored in a random variable X. The differential entropy for continuous random variables is, however, a relative measure used for comparison. We aim at comparing the secret and public randomness of the continuous Gaussian vectors sampled in the signing step. Therefore, we have to compute the differential entropy for the distinct randomness portions. The differential entropy of a multivariate continuous Gaussian vector $\mathbf{d}$ with $f(x_1,\dots,x_n) = e^{-\pi\|\mathbf{x}\|_2^2}$ is determined as follows

$$\begin{aligned}
h(\mathbf{d}) &= \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} f(x_1, \ldots, x_m) \cdot \log f(x_1, \ldots, x_m) dx_1 \ldots dx_m \\
&= \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} f(x_1, \ldots, x_m) \cdot \left(-\pi \left\|\mathbf{x}\right\|_2^2\right) dx_1 \ldots dx_m \\
&= \frac{1}{2} \log(e^m) \ \ bits\,.
\end{aligned}$$

When outputting a signature and, hence, revealing the corresponding set C, the entropy of the continuous Gaussian vector decreases, because information is leaked about $\mathbf{d}$. As a consequence, the entropy of $\mathbf{d}$ is now computed based on $C = \frac{\mathbf{z}^{(2)}}{\sqrt{b}} + [-\frac{4.7 \cdot \sqrt{5}a}{\sqrt{b}}, \frac{4.7 \cdot \sqrt{5}a}{\sqrt{b}}]^{nk}$, which corresponds to the secret randomness. However, when sampling perturbations $\mathbf{p}_1$ in the signing step, we require an additional continuous Gaussian vector $\mathbf{d}_1 \leftarrow_R \mathcal{D}_1^{2n}$ that remains completely unexploited. We know from information theory that the conditional entropy of $\mathbf{d}$ given $\mathbf{d} \in C$ is given by

$$\begin{aligned}
h(\mathbf{d} \mid C) &= \sum_{i=1}^{nk} h(d_i \mid C_i) = -\sum_{i=1}^{nk} \int_{C_i} \frac{f(x_i)}{P[\ C_i\ ]} \cdot \log \frac{f(x_i)}{P[\ C_i\ ]} dx_i \\
&= \sum_{i=1}^{nk} \log(P[\ C_i\ ]) - \int_{C_i} \frac{f(x_i)}{P[\ C_i\ ]} \cdot \log f(x_i) dx_i \\
&\approx \sum_{i=1}^{nk} \log 2h = n \cdot k \cdot \log(\frac{2 \cdot 4.7 \cdot \sqrt{5}a}{\sqrt{b}}),
\end{aligned}$$

since $P[\ C_i\ ] \leq 2h$. The first equality follows from the independence of the entries in $\mathbf{d}_1$. The secret portion of randomness has entropy amounting to $h_{\mathsf{secret}} \approx \frac{1}{2}\log(e^{2n}) + n \cdot k \cdot \log(\frac{2 \cdot 4.7 \cdot \sqrt{5}a}{\sqrt{b}})$ bits, whereas the public randomness is lower bounded by $h_{\mathsf{public}} = \frac{1}{2}\log(e^{n(k+2)}) - h_{\mathsf{secret}} \approx n \cdot k \cdot (\ \frac{1}{2}\log(e) + \log(\frac{2 \cdot 4.7 \cdot \sqrt{5}a}{\sqrt{b}})\ )$ bits. One notices that the differential entropy can be negative.

7.4. Implementation and Experimental Results

To scrutinize the performance respectively efficiency of the single signer compression scheme presented in this chapter we modified the optimized implementation for GPV signatures given in Section 6.5.2. In particular, we require 2 further building blocks in order to realize an efficient software implementation of the scheme from Section 7.3.3.

- First, we need to sample the centroids from a uniform random string. This can indeed be realized by use of the salsa20 stream cipher, that stretches an initial seed to the desired output length, in combination with the discrete Gaussian sampler FastCDT introduced in Section 5.1, which allows to sample discrete Gaussian distributed vectors with huge parameters s very efficiently (see Section 6.3.2).

- We need to implement the conditional rejection sampler from Section 7.3.2. The efficiency of this sampler inherently depends on the number of rejections and hence the distribution used to generate samples. This distribution should always be above the target distribution. If the difference is too large, the number of rejections increase. Since we want to sample according to the probability density function $f(x \mid x \in C_i) = e^{-\pi \cdot x^2}/P[\ C_i\ ]$, it is required to find a probability density function $g(x)$ such that $f(x \mid x \in C_i) \leq g(x)$ which is used to generate samples. Surely, we have $f(x \mid x \in C_i) \leq 1/P[\ C_i\ ]$, but this choice is not optimal. Since $e^{-\pi \cdot x^2}$ is monotone decreasing for $x > 0$ and monotone increasing for $x < 0$, we can use the interval boundaries $t_1, t_2 < 1$ of $C = [t_1, t_2]$ as a good advice. For instance, if $t_1, t_2 > 0$,then $e^{-\pi \cdot t_1^2} \geq e^{-\pi \cdot x^2}$ for all $x \in C$. Therefore, we choose $g(x)$ as follows

  1. If $t_1, t_2 > 0$, set $g(x) = e^{-\pi \cdot t_1^2}/P[\ C\ ]$
  2. If $t_1, t_2 < 0$, set $g(x) = e^{-\pi \cdot t_2^2}/P[\ C\ ]$.

  We ignore the case $t_1 < 0 \wedge t_2 > 0$, which does not occur very often. Indeed we omit to compute the probability $P[\ C\ ]$, since it is used for both distributions and can hence be canceled out by multiplication of $P[\ C\ ]$. We note that this does not change the distributions such that we have only to consider the scaled distributions $\bar{g}(x) = g(x) \cdot P[\ C\ ]$ and $\bar{f}(x) = (x \mid x \in C) \cdot P[\ C\ ]$ in the conditional rejection sampler. As a result, we sample an element $x \in C$ uniformly at random and an element $u \in [0, 1]$ uniformly at random and check if $u \cdot \bar{g}(x) \leq \bar{f}(x)$. If this is true, we accept and else reject.

Ring Variant	**Parameters**	**Timings (cycles)**		**Security (bits)**		**Ratio**
	q	**Sign**	**Verify**	**[MR09]**	**[RS10]**	**t_1/t_0**
GPV without Compress.						
n = 512	2^{24}	13395600	464400	> 300	97	-
	2^{27}	14810400	514800	> 300	103	-
	2^{29}	15796800	558000	> 300	107	-
n = 1024	2^{27}	31935600	1117800	> 300	147	-
	2^{29}	34137000	1186200	> 300	153	-
GPV with Compress.						
n = 512	2^{24}	17757000	4264200	> 300	97	1.3 | 9.2
	2^{27}	19630800	4789800	> 300	103	1.3 | 9.0
	2^{29}	21002400	5022000	> 300	107	1.3 | 9.0
n = 1024	2^{27}	42271200	9849600	> 300	147	1.3 | 8.8
	2^{29}	44841600	10542600	> 300	153	1.3 | 8.8

Matrix variant	Parameters	Timings (cycles)		Security (bits)		Ratio
	q	Sign	Verify	[MR09]	[RS10]	T_1/T_0
GPV without Compress.						
n = 512	2^{24}	59862600	9558000	> 300	97	-
	2^{27}	67384800	10733400	> 300	103	-
	2^{29}	73746000	11930400	> 300	107	-
GPV with Compress.						
n = 512	2^{24}	65466000	13699800	> 300	97	1.1 | 1.4
	2^{27}	73236600	15415200	> 300	103	1.1 | 1.4
	2^{29}	79216200	16644600	> 300	107	1.1 | 1.4

The public key of the scheme is instantiated computationally just as described in Section 6.5.2, where the entries of the private key are sample both in the matrix and ring variant according to the discrete Gaussian distribution with parameter $\alpha q = \sqrt{n}$. By T_1/T_0 we denote the ratio of the running time with compression T_0 and without compression T_1. Applying the compression scheme in the ring variant leads to a signature generation engine that is by a factor of 1.3 slower as compared to the signature scheme without compression. And verification is slower by a factor of about 9. However, in the matrix variant both signature generation and verification are essentially as efficient as without compression. On the other hand the signature size is improved by a factor of about 2.5 – 3.1 (see Table 7.1). However, when comparing with the implementation from Section 6.5.1 (see [P9] for timings), verification is still fast. This can be attributed to the process of generating $n \cdot k$ discrete Gaussians with a large parameter s from a seed (see Algorithm 7.3 or Section 7.3.3). We note, that the compression scheme is flexible, that is we can chose to compress less number of entries leading to faster algorithms at the expense of a smaller compression factor, since we need only to generate $l < n \cdot k$ discrete Gaussians, where the remaining entries are built following the traditional approach.

Compression Scheme	Parameters		Compression Factor
	q	αq	
n = 512			
(applied)	2^{29}	$\sqrt{n}$	2.5
(LWE)	2^{29}	$2\sqrt{n}$	2.7
(ring-LWE)	2^{29}	$\sqrt{n \log 2n} \cdot (\frac{n}{2 \log n})^{1/4}$	2.9
n = 1024			
(applied)	2^{30}	$\sqrt{n}$	2.7
(LWE)	2^{30}	$2\sqrt{n}$	2.8
(ring-LWE)	2^{30}	$\sqrt{n \log 2n} \cdot (\frac{n}{2 \log n})^{1/4}$	3.1

Table 7.2.: Compression Rates for different Parameters

7.5. Generic Multi-Signer Compression Strategy

In the following section we introduce a multi-signer compression strategy, if more than one signer agree to share the same source of public randomness. This approach is equivalent to an aggregate signature in its most trivial form, namely bundling signatures together. Due to the fact that public randomness is accessible to all signers viewing signatures (resp. seeds), the same source of public randomness can be exploited by other signers with different keys as well. This observation has already been made in Theorem 7.3 and Theorem 7.7. Our multi-signer compression strategy aims at decreasing the overall computational costs and total signature size if more than one party participate to construct a signature using the same source of public randomness such as a fresh seed $\mathbf{r}$, signature or discrete Gaussian vector following the distribution of signatures. It is well known that the most trivial form of an aggregate signature scheme simply consists of bundling all signatures of all participating signers together. There is no compression in this case and the security of the aggregate signature stems from the unforgeability of each individual signature, because each signature is verified independently from the others (see Theorem 7.10). This approach is taken in the following sections with the difference that each signature is compressed by use of Algorithm 7.3 and a fresh seed $\mathbf{r}$. Analoguous to the uncompressed case, all compressed signatures are bundled together with the seed and subsequently handed over to the verifier who in turn verifies each individual signature independently.

The main advantages of such a multi-signer compression strategy can be summarized as follows. The overall signature size is reduced, since all signers aggree on a single seed $\mathbf{r}$ as opposed to l seeds, each for a different signer. Therefore, it suffices to transmit only $\mathbf{r}$. The reduction of the computational costs at the verifier side are noteworthy. In particular, the verifier has only to call the simulator of signatures once as opposed to l calls, because all signers use the same seed and hence the same centroid. As a result, uncompression of a signature bundle in a multi-signer compression scheme is essentially as fast as uncompressing a single signature. Due to the relationship of our multi-signer compression strategy and standard aggregate signature schemes, which mainly have the goal to reduce the total signature size and to verify that all parties correctly signed the corresponding documents, we refer to Chapter 8 for a related work section on aggregate signature schemes.

7.5.1. Multi-Signer Compression Scheme

Prior to the description of our novel and generic multi-signer compression scheme from public randomness, we start by some definitions. In fact, the idea of the multi-signer compression scheme is strongly related to aggregate signatures in its simplest form, namely bundling l signatures together. We use the terms aggregate signature and bundle of compressed signatures as synonyms throughout this thesis. Therefore, it is reasonable to tailor the definition of aggregate signatures to our multi-signer compression scheme.

Definition 7.9 (**Multi-Signer Compression Scheme (MSC)**). *In a multi-signer compression scheme l signatures $\mathbf{z}_i$ on l messages msg_i from l distinct signers are combined into a signature $\mathbf{z}_{MSC}$ for $1 \leq i \leq l$ such that the resulting aggregate signature $\mathbf{z}_{MSC}$ is significantly smaller than the total size of all individual signatures (compression property). Moreover, each individual standard signature $\mathbf{z}_i$ can efficiently be recovered from $\mathbf{z}_{MSC}$ (uncompression property).*

The generic construction of our multi-signer compression scheme (MSC) from public randomness $\mathbf{r}$ involves 5 algorithms.

$\mathsf{KeyGen}(1^n, i$ with $1 \leq i \leq l)$: Outputs secret key sk_i and public key pk_i to signer i.

$\mathsf{SeedGen}(1^n)$: Outputs a centroid generating seed $\mathbf{r} \in \{0,1\}^\mu$

$\mathsf{Sign}(\mathsf{sk}_i, \mathbf{r} \in \{0,1\}^\mu, \mathsf{msg}_i)$: Outputs a message msg_i of signer i and an individual signature $\mathbf{z}'_i = \mathbf{z} - \mathbf{z}_i$ compressed with respect to $\mathbf{z}$ that is generated by means of the random seed $\mathbf{r}$.

$\mathsf{Bundle}(\overrightarrow{\mathbf{pk}}, \vec{\mathbf{z}}, \mathbf{r}, \overrightarrow{\mathsf{msg}})$: Outputs the aggregate signature $\mathbf{z}_{MSC}$ as a bundle of l compressed signatures including the seed $\mathbf{r}$.

$\mathsf{Verify}(\overrightarrow{\mathbf{pk}}, \mathbf{z}_{MSC}, \mathbf{r}, \overrightarrow{\mathsf{msg}})$: Verifies the aggregate signature $\mathbf{z}_{MSC}$ with the aid of the public keys $\overrightarrow{\mathbf{pk}} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_l)$, the centroid generating seed $\mathbf{r}$ and messages $\overrightarrow{\mathsf{msg}} = (\mathsf{msg}_1, \ldots \mathsf{msg}_l)$. For each valid signature in the bundle $\mathbf{z}_{MSC}$ set the corresponding entry to 1, otherwise set 0. Output **out**.

As already observed, the generic compression algorithms from Section 7.2 naturally induce multi-signer compression schemes, since all parties are allowed to consume the same source of public randomness as per Theorem 7.3. We hereby present a generic approach towards constructing a multi-signer compression scheme that is more efficient than the single-signer approach. The security of the scheme inherently stems from Theorem 7.4.

Theorem 7.10 (Security). *Let $\mathbf{r} \in \{0,1\}^\mu$ be sampled uniformly at random. Then, the bundle of compressed signatures (aggregate signature) in Algorithm 7.4 is secure assuming the hardness to break uncompressed signatures.*

Proof. As per assumption $\mathbf{r}$ is uniform random. Since each compressed signature is recovered and subsequently verified independently from the remaining ones in the bundle, we can directly apply Theorem 7.4. $\square$

```
Algorithm 8: AS Scheme: AggSign
    Data: Distribution of signatures $\mathcal{Z}$, seed $\mathbf{r} \in \{0,1\}^{\mu}$
 1  for $i = 1$ to $l$ do
 2      \\ i-th Signer
 3      Sample $\mathbf{z} \leftarrow \mathcal{Z}$ using input seed $\mathbf{r}$
 4      Set $b = \max_{\mathbf{s},\mathbf{c}} \|f_{\mathbf{s}}(\mathbf{c})\|_{\infty}$
 5      Set $C = \mathbf{z} + [-b, b]^m, P[\ C\ ] := P_{\mathbf{y}\sim\mathcal{Y}}(\mathbf{y} \in C)$
 6      for $i = 1 \rightarrow l$ do
 7          Sample $\mathbf{y}_i \leftarrow \mathcal{Y}(\mathbf{x})/P[\ C\ ],\ \mathbf{x} \in C$
 8          $\mathbf{z}_i = f_{\mathbf{s}_i}(\mathbf{c}_i) + \mathbf{y}_i$
 9      end
10  end
11  Output $(\mathbf{r},\ \mathbf{z} - \mathbf{z}_1, \ldots,\ \mathbf{z} - \mathbf{z}_l),\ \overrightarrow{\mathsf{msg}}$
```

```
Algorithm 9: Verification: AggVerify
    Data: Aggregate signature $(\mathbf{r},\ \mathbf{z}'_1, \ldots,\ \mathbf{z}'_l)$ with
          $\mathbf{z}'_i = \mathbf{z} - \mathbf{z}_i$, messages $\overrightarrow{\mathsf{msg}}$
 1  Sample $\mathbf{z} \leftarrow \mathcal{Z}$ using input seed $\mathbf{r}$
 2  for $i = 1$ to $l$ do
 3      $\mathbf{z}_i = \mathbf{z} - \mathbf{z}'_i$        \\uncompressed signatures
 4      if $\mathit{Verify}(\mathbf{z}_i, \mathit{msg}_i) == 1$ then
 5          $\mathbf{out}_i := 1$
 6      else
 7          $\mathbf{out}_i := 0$
 8      end
 9  end
10  Output out
```

Figure 7.4.: Aggregate Signature Scheme

Indeed, the above described multi-signer compression scheme allows the verifier to recover all individual signatures from $\mathbf{z}_{MSC}$. The centroid associated to $\mathbf{r}$ connects all the individual signatures together. Furthermore, if the seed is made a shared secret, the aggregate signature can only be recovered and verified by the holders of $\mathbf{r}$. Such schemes are interesting within the context of wireless sensor networks, because WSNs are characterized by constrained ressources such that one observes an inherent need for data compression schemes reducing the amount of traffic. Therefore, we consider cluster-based sensor networks in Section 7.6 as a potential application scenario for our scheme.

7.5.2. Multi-Signer Compression Scheme in the GPV Setting

A usable and practical way of instantiating the scheme requires the participating signers to agree on a random string in advance. This is attained, for example, if each signer samples a random salt $\mathbf{r}_i$ and broadcasts it to the remaining parties in order to produce the ultimate seed $\mathbf{r} = H(\mathbf{r}_1, \ldots, \mathbf{r}_l)$ using a cryptographic hash function modeled as random oracle. Each signer maintains a counter that is increased for every compression request. This counter is appended to $\mathbf{r}$ and serves as input to a second hash function, whose output sequence is used to sample the centroid in order to compress GPV signatures. At this point we have to explain how to sample continuous Gaussians in the case when the signers' parameters n_i and k_i differ.
Our goal is to keep the scheme as efficient as with constant parameters. The computation complexity and the number of transmitted seeds should not change. Therefore, one starts by defining the maximum Gaussian parameter $s = \max\limits_i \; s_i$, the maximum dimension $N = \max\limits_i \; n_i$ and the maximum number of entries $M = \max\limits_i n_i k_i$ with s_i, n_i, k_i and $m_i = n_i \cdot k_i$ denoting the parameters of the $i-$th signer. Accordingly, we define $b_i = s_i^2 - 5a^2$ and $B = s^2 - 5a^2 \geq b_i$. Following Theorem 7.7 and Theorem 7.2 each signer samples a continuous Gausssan from a set of proper width. This can be achieved by sampling $\mathbf{d}_i \leftarrow_R \mathcal{D}_1^{m_i}$, $\mathbf{d}_i \in C_i = [\frac{\mathbf{z}^{(3)}}{\sqrt{B}} - \frac{4.7 \cdot \sqrt{5}a}{\sqrt{b_i}}, \frac{\mathbf{z}^{(3)}}{\sqrt{B}} + \frac{4.7 \cdot \sqrt{5}a}{\sqrt{b_i}}]^{m_i}$, where $\mathbf{z}^{(3)}$ is a discrete Gaussian vector with parameter s and $a = \sqrt{\ln\left(2n\left(1+\frac{1}{\epsilon}\right)\right)/\pi}$. The choice of s implies $C \subseteq C_i$ for $C = \frac{\mathbf{z}^{(3)}}{\sqrt{B}} + [-\frac{4.7 \cdot \sqrt{5}a}{\sqrt{B}}, \frac{4.7 \cdot \sqrt{5}a}{\sqrt{B}}]$ and thus provides the required interval width.
Due to differing parameters, the number of signature entries varies among the signers such that $m_i \leq m$. For this reason, one takes as many entries as required from $\mathbf{z}^{(3)}$ starting from the first component of $\mathbf{z}^{(3)}$. Since each signer operates with different parameters b_i and s_i when sampling signatures, we have to derive individual centroids $\mathbf{w}_i$ from $\mathbf{z}^{(3)}$ efficiently. The most reasonable way of doing this requires to set $\mathbf{w}_i = \lceil \frac{\mathbf{z}^{(3)}}{\sqrt{B}} \cdot \sqrt{b_i} \rfloor$, which can be computed efficiently from public parameters, for signer i such that its difference to the center of the scaled sets $\sqrt{b_i} \cdot C_i$ is smaller than 0.5 in each entry. As a result, we still have $\log \parallel \mathbf{w}_i - \mathbf{z}_i^{(3)} \parallel_\infty \leq 7$ except with negligible probability. In case we have $b_i = B$ for all i, $\mathbf{w}_i = \mathbf{z}^{(3)}$ is obtained as the centroid for all signers. In Figure 7.5 we provide the main steps. Here $\mathcal{D}_{\mathbb{Z},s}^M(\mathbf{t})$ denotes the discrete Gaussian sampler simulating signatures with parameter s, input seed $\mathbf{t}$.

Algorithm 10: MS Compression: MCSign

```
Data: Seed $\mathbf{r}$, parameters $s_i, n_i, k_i, B, M, N, s$
$ctr = j$                      \\counter
$\mathbf{t} \leftarrow H(\mathbf{r}, ctr)$             \\actual seed
$\mathbf{z} \leftarrow \mathcal{D}^M_{\mathbb{Z},s}(\mathbf{t})$              \\centroid using $\mathbf{t}$
for $i = 1 \rightarrow l$ do
    \\Compression
    Set $m = n_i \cdot k_i,\ n = n_i, b = b_i$ and $\mathbf{A} = \mathbf{A}_i$
    $\mathbf{w}_i = \lceil \frac{\mathbf{z}}{\sqrt{B}} \cdot \sqrt{b} \rfloor$   \\modified centroid
    Define $C_i = [\frac{\mathbf{z}}{\sqrt{B}} - \frac{4.7 \cdot \sqrt{5}a}{\sqrt{b}}, \frac{\mathbf{z}}{\sqrt{B}} + \frac{4.7 \cdot \sqrt{5}a}{\sqrt{b}}]^m$
    \\Signing
    $\mathbf{d_1} \leftarrow \mathcal{D}_1^{2n}, \mathbf{d_2} \leftarrow \mathcal{D}_1^m \ \wedge\ \mathbf{d_2} \in C_i$
    $\mathbf{z}_i = (\mathbf{z}_i^{(1)}, \mathbf{z}_i^{(2)}, \mathbf{z}_i^{(3)}) \leftarrow f_{\mathbf{A}}^{-1}(H(\mathsf{msg}_i))$
    $\mathbf{y}_i = (\mathbf{z}_i^{(2)}, \mathbf{w}_i - \mathbf{z}_i^{(3)})$, (Lemma 7.8, Theorem 7.7)
end
Output Aggregate $(ctr, \mathbf{y}_1, \dots, \mathbf{y}_l)$
```

Algorithm 11: Verification: $\mathsf{MCVerify}$

```
Data: Seed $\mathbf{r}$, $(ctr, \mathbf{y}_1, \dots, \mathbf{y}_l)$, parameters $s_i, n_i, k_i, B, M, N, S$
$\mathbf{t} \leftarrow H(\mathbf{r}, ctr)$             \\actual seed
$\mathbf{z} \leftarrow \mathcal{D}^M_{\mathbb{Z},S}(\mathbf{t})$              \\centroid
for $i = 1 \rightarrow l$ do
    \\Uncompression
    Set $m = n_i \cdot k_i,\ n = n_i, b = b_i$ and $\mathbf{A} = \mathbf{A}_i$
    $\mathbf{w}_i = \lceil \frac{\mathbf{z}}{\sqrt{B}} \cdot \sqrt{b} \rfloor$   \\modified centroids
    $\mathbf{y}_i = (\mathbf{z}_i^{(2)}, \mathbf{w}_i - \mathbf{z}_i^{(3)}),\ \mathbf{z}_i = (\mathbf{z}_i^{(1)}, \mathbf{z}_i^{(2)}, \mathbf{z}_i^{(3)})$
end
$\mathbf{z} = (\mathbf{z}_1, \dots, \mathbf{z}_l)$          \\uncompressed signatures
out $= (0, \dots, 0)$
for $i = 1 \rightarrow l$ do
    \\Verification
    if $f_{\mathbf{A}_i}(\mathbf{z}_i) = H(\mathsf{msg}_i) \wedge \| \mathbf{z}_i \| < s_i\sqrt{m_i}$ then
        out$_i = 1$, $\mathbf{z}_i$ is valid
    end
end
Output out
```

Figure 7.5.: Multi-Signer Compression Scheme in the GPV Setting

7.6. Application Scenario - Cluster-based Aggregation in Wireless Sensor Networks

An interesting application scenario for the proposed multi-signer compression scheme are wireless sensor networks. In recent years, many research efforts have been spent on minimizing data transmissions within WSNs due to the resource constrained devices forming the topology. At the same time there is a claim for securing the communication flow against adversaries that could attack the network to gather information or to manipulate them. Therefore, a lot of theoretical research has been made on secure data aggregation protocols, which aim at providing a certain level of security as well as mechanisms that improve the lifetime of sensor networks by minimizing the number of transmitted messages. It is a well-known fact [HSW+00] that the transmission of a single bit consumes as much battery power as executing 800-1000 instructions. So it is more convenient to reduce the number bits to be sent at the cost of an increased computation complexity.

A cluster-based sensor network is an appropriate topology to support most of the aggregation schemes. Following this approach, one splits the network into clusters, each of similar size. Typically, one finds such topologies at Logistic Service Providers that monitor the supplied goods on their way from the manufacturer to the client. The products have different features (e.g. the temperature and humidity) that are of interest and thus need to be monitored for the whole delivery period. For instance, the temperature of perishables should remain constant on a reasonable level.
Therefore, one should always keep an eye on the temperature preventing the spoiling of goods. A plausible way to design the topology for goods transported via trucks is to let each truck form a cluster with a small number of sensor nodes. Each cluster is characterized by its cluster head (CH), a dedicated node that acts as a subentity of an aggregator node (similar to a cluster head) or the root. At the top of the topology we have the root that sends its requests to the cluster heads, which in turn forward the request to the cluster participants. After sensing the required values, the cluster participants send them via the cluster heads back to the base station. There are many different ways to implement the signature compression schemes from above. For the sake of simplicity, we consider the generic multi-signer compression scheme from Section 7.5. Therefore, the base station sends a fresh random salt securely to all nodes within the WSN in the setup phase. Whenever the nodes sense the required values they increase an internal counter which serves together with the random salt as the input seed to a discrete Gaussian sampler. Afterwards, they transmit their compressed signatures via the cluster heads back to the base station. The usage of a salt in combination with a counter for creating signatures further allows to capture replay attacks, that can be launched by any adversary eavesdropping the message stream. Moreover, one can allow only privileged members to be the only ones being able to verify signatures. This is attained by making the random salt a secret key shared among the privileged members. By doing this, only those who share the secret key can recover the centroid and subsequently verify the uncompressed signature.

8. Sequential Aggregate Signatures

An aggregate signature (AS) scheme enables a group of signers to combine their signatures on messages of choice such that the combined signature is essentially as large as an individual signature. Aggregate signatures have many application areas such as secure routing protocols [Lyn99] providing path authentication in networks. Moreover, ASs are important mechanisms used in constrained devices, e.g. wireless sensor networks, in order to decrease the amount of transmitted data, which in turn reduces the battery power consumption. Particularly in cluster-based sensor networks, where each cluster consists of a small number of sensor nodes, it is reasonable to apply data aggregation techniques including AS.
The first aggregate signature scheme is due to [BGLS03], which is based on the hardness of the co-Diffie-Hellman problem in the random oracle model. Following this proposal, the aggregation mechanism can be accomplished by any third party since it relies solely on publicly accessible data and the individual signature shares. Conceptually, this scheme is based on bilinear maps. In [LMRS04] Lysyanskaya et al. proposed a new variant of AS, known as sequential aggregate signatures (SAS), which differs from the conventional AS schemes by imposing an order-specific generation of aggregate signatures. In particular, each signer is ordered in a chain and receives the aggregate from its predecessor before the own signature share is added to the aggregate. A characteristical feature of this scheme is to include all previously signed messages and the corresponding public keys in the computation of the aggregate. In practice, one finds, for instance, SAS schemes applied in the S-BGP routing protocol or in certificate chains, where higher level CAs attest the public keys of lower level CAs. The generic SAS construction provided in [LMRS04] is based on trapdoor permutations with proof of security in the random oracle model. However, the SAS scheme suffers from the requirement of certified trapdoor permutations [BY96] for the security proof to go through. But the authors explain how to circumvent the need for certification in the special case of an RSA-based instantiation using large exponents e as public keys. This obviously lacks efficiency due to costly exponentiations. Subsequent works provide similar solutions or improve upon existing ones, e.g. the work [BNN07] removes restrictions on the choice of messages imposed by [BGLS03]. In particular, prior to this improvement the messages were forced to be distinct.
First steps towards eliminating random oracles in the security proof were taken by Lu et al. [LOS^{+}06], who proposed a new SAS variant that is based on bilinear pairings while providing provably security in the standard model. In a following work, the authors of [Nev08] address the requirement of certified trapdoor permutations in [LMRS04] and present a new SAS construction removing the need for certification.

As a main drawback of both schemes, a potential signer has to verify the actual aggregate signature prior to any modification. This issue is investigated in the works [EFG$^+$10] and [BGR12]. The first proposal successfully solved this problem and hence allows to omit verification beforehand when modifying the aggregate. But this approach works as the BGLS-scheme [BGLS03] in a different setting. The SAS construction with lazy verification [BGR12] has the advantage that each signer does not care about the validity of the intermediate aggregate signatures. Therefore, the messages and the corresponding public keys of precedent signers need not to be requested when generating aggregate signatures. Verification of the aggregates is delayed and can be accomplished at any time afterwards. Interestingly, Hohenberger et al. present in [HSW13] the first unrestricted aggregate signature scheme that is based on leveled multilinear maps. The underlying hardness assumptions are, nevertheless, not directly connected to worst-case lattice problems. In this chapter we address the question whether it is possible to build (S)AS schemes that can be based on worst-case lattice problems. This chapter refers to the paper [EB14b], where the author of this thesis was the primary investigator and author of the paper.

Our Results and Contribution

Sequential Aggregation of Signatures from PSTFs.

We present the first lattice-based sequential aggregate signature scheme that is secure in the random oracle model [BR93]. It can be instantiated by any collection of preimage sampleable trapdoor functions [AP09, GPV08, MP12, Pei10, SS11] including identity-based variants such as in [GPV08]. In fact, one can even use different types of trapdoor function families simultaneously. The security model, we adopt, is mainly influenced by [BNN07, LMRS04, Nev08] as it requires the scheme to withstand potential attacks even when the forger controls all but at least one secret key. Inspired by the work [Nev08] we prove by means of a sequential forger that breaking the scheme is as hard as solving hard instances of lattice problems. Specifically, we show that solving the SIS problem can be reduced to the task of successfully forging an aggregate signature. We even prove that our scheme is strongly unforgeable under chosen message attacks [Rüc10]. Interestingly, all results immediately transfer to the ring setting, since the proof is based on a more abstract notion of collision-resistant trapdoor functions [GPV08] subsuming both the matrix and ring variant. In terms of performance, the signing costs of each signer are limited to one run of the GPV signature scheme and $(i-1)$ function evaluations, where i denotes the index of signer S_i in the chain. By applying the framework of [BPW12, SMP08] one can additionally turn any sequential aggregate signature scheme into a proxy signature scheme, where the security is based on the hardness of forging signatures in the underlying SAS scheme.

Instantiation of SAS.

As mentioned before, one can principally use any PSTF family that is suitable for the GPV signature scheme in order to instantiate the SAS scheme, particularly also the trapdoor construction presented in [MP12]. Due to the compressing property of lattice-based PSTFs the range is in many cases much smaller than the domain $\log_2(B_n) > \log_2(R_n)$, where $\log_2(C) := \max_{c \in C} \lceil \log_2 c \rceil$ denotes the maximum bit size of a set C. Consequently, the size of the aggregate signature is larger than an individual one when instantiating the chain of signers with the same security parameters n and q. However, it is always possible to select the parameters n_i and q_i of the signers in such a way that the signature of any signer completely flows in the signature computation of its successor. The resulting aggregate signature is then essentially as large as a signature of the last signer. To achieve this, $\log_2(R_{n_{i+1}}) \geq \log_2(B_{n_i})$ must hold for all $1 \leq i \leq k$. Hence, the aggregate signature of any signer completely flows in the computation of the aggregate signature of the next signer in the chain. A way of measuring the quality or suitability of any PSTF for use in the proposed SAS scheme is the ratio $\log_2(B_n)/\log_2(R_n)$, where a value close to 1 indicates that the sizes of the domains and ranges are of the same order. Based on this selection criterion, we instantiate our construction with the provably secure NTRUSign signature scheme [SS11]. It furtherly allows to achieve asymptotically optimal compression rates that are known from number-theory based SAS schemes such as the schemes provided in [LMRS04, Nev08]. Moreover, we discuss the potential advantages of our construction over RSA-based SAS schemes. In particular, we point out that our proposal is characterized by its flexibility and simplicity of instantiation as compared to the schemes provided in [LMRS04, Nev08]. However, the compressing property of the PSTFs leads inherently to some small SAS overheads, if n and q are chosen to be equal for all signers. A solution to this problem consists in selecting the parameters n_i and q_i in such a way that the signature of any signer completely flows in the signature computation of its successor.

8.1. Our Construction

We introduce our lattice-based sequential aggregate signature scheme in Section 8.1.1. In fact, the construction is based on the more abstract notion of preimage sampleable trapdoor functions introduced in Chapter 6.

8.1.1. Our Basic Signature Scheme

We aim at constructing a SAS scheme from trapdoor functions instead of trapdoor permutations. Inspired by the works of [LMRS04, Nev08] we transfer the core ideas from trapdoor permutations to the lattice-based setting. The main obstacle that needs to be handled is the fact that lattice-based trapdoor functions operate in differing domains and ranges. The input bit size is usually larger than the output one. This is due to the need of collisions. Therefore, we use the encoder-technique enc proposed by [Nev08] that takes these properties into account. In particular, it breaks the signature down into two parts, where the first part is injectively mapped to an element of the image space and subsequently processed in the computation of the modified aggregate signature. The second part is simply handed over to the next signer in the chain. The encoder-technique is originally designed to allow for hiding of additional data like messages in RSA like systems in order to decrease not only the signature sizes but also the total amount of data to be send. The following algorithms provide the main steps of the SAS scheme.

Algorithm 12: $\mathsf{AggSign}(\mathbf{T_i}, m_i, \Sigma_{i-1})$: Signing algorithm of the i-th signer

```
Data: Trapdoor $\mathbf{T_i}$, message $m_i$, $\Sigma_{i-1}$
if $i = 1$ then
    $\Sigma_0 \leftarrow (\epsilon, \epsilon, \epsilon, 0^n)$;
end
Parse $\Sigma_{i-1}$ as $(\overrightarrow{\mathsf{f}_{\mathbf{A}_{i-1}}}, \vec{m}_{i-1}, \vec{\alpha}_{i-1}, \sigma_{i-1}, h_{i-1})$
if $\mathit{AggVerify}(\Sigma_{i-1}) == (\perp, \perp)$ then
    return $\perp$
end
$(\alpha_i, \beta_i) \leftarrow \mathsf{enc}_{f_{\mathbf{A}_i}}(\sigma_{i-1})$
$\vec{\alpha}_i \leftarrow \vec{\alpha}_{i-1} \mid \alpha_i$
$h_i \leftarrow h_{i-1} \oplus \mathbf{H}(\overrightarrow{\mathsf{f}_{\mathbf{A}_i}}, \vec{m}_i, \vec{\alpha}_{i-1}, \sigma_{i-1})$
$g_i \leftarrow \mathbf{G}_{\mathsf{f}_{\mathbf{A}_i}}(h_i)$
$\sigma_i \leftarrow \mathsf{SamplePre}(\mathbf{T}_i, g_i + \beta_i)$

Return $\Sigma_i \leftarrow (\overrightarrow{\mathsf{f}_{\mathbf{A}_i}}, \vec{m}_i, \vec{\alpha}_i, \sigma_i, h_i)$
```

Algorithm 13: $\mathsf{AggVerify}(\Sigma_k)$: Verification algorithm

```
Data: $\Sigma_k$
Parse $\Sigma_k$ as $(\overrightarrow{\mathsf{f}_{\mathbf{A}_k}}, \vec{m}_k, \vec{\alpha}_k, \sigma_k, h_k)$
for $i = k \rightarrow 1$ do
    if $\log_2(R_{n_i}) \leq l$ or $\sigma_i \notin B_{n_i}$ then
        return $(\perp, \perp)$
    end
    $g_i \leftarrow \mathbf{G}_{\mathsf{f}_{\mathbf{A}_i}}(h_i)$
    $\beta_i \leftarrow \mathsf{f}_i(\sigma_i) - g_i$
    $\sigma_{i-1} \leftarrow \mathsf{dec}_{f_{\mathbf{A}_i}}(\alpha_i, \beta_i)$
    $h_{i-1} = h_i \oplus \mathbf{H}(\overrightarrow{\mathsf{f}_{\mathbf{A}_i}}, \vec{m}_i, \vec{\alpha}_{i-1}, \sigma_{i-1})$
end
if $\Sigma_0 = (\epsilon, \epsilon, \epsilon, 0^n)$ then
    return $(\overrightarrow{\mathsf{f}_{\mathbf{A}_k}}, \vec{m}_k)$
end
Else return $(\perp, \perp)$
```

Due to the lack of concrete definitions for sequential aggregate signatures, we fill this gap and present a generic description of SAS schemes.

Definition 8.1. *In a sequential aggregate signature (SAS) scheme k distinct signers, that are ordered in a chain, sequentially put their signature on messages of choice m_i to the aggregate signature such that the resulting aggregate signature σ has the size of an individual signature.*

In Definition 8.2, we give a relaxed version that allows the SAS schemes to be larger than an individual signature, since many proposed constructions do not really match the definition from above in terms of optimality.

Definition 8.2. *We say x-SAS, if the aggregate signature is essentially an individual signature extended by c bits of overhead with $c = (1-x)\cdot\sum_i \mathsf{size}(\sigma_i)$ bits and $x \in [0,1]$.*

For $x = 1$, we immediately obtain the aforementioned definition of the classical SAS scheme.

8.1.2. Informal Description

We give an informal description of the sequential aggregate signature scheme. We focus on the signing and verification steps in Algorithm 12 and Algorithm 13. Each signer S_i in the chain with $1 \le i \le k$ follows the same protocol steps. Let $l > 0$ be a public parameter such that $\log_2(R_{n_i}) > l$ for $1 \le i \le k$, where R_{n_i} denotes the image space of the trapdoor function $f_{\mathbf{A}_i} : B_{n_i} \to R_{n_i}$ and $\log_2(R_{n_i})$ defines the maximum number of bits needed to represent elements of R_{n_i}.
The input to the signing algorithm $\mathsf{AggSign}(\cdot)$ of the i-th signer S_i is its secret key $\mathbf{T}_i$, the message to be signed and a list of data Σ_{i-1} received from signer S_{i-1}. If S_i corresponds to the first signer, the list of data is empty. Otherwise, Σ_{i-1} parses as a list consisting of a sequence of trapdoor functions $f_{\mathbf{A}_1}, \ldots, f_{\mathbf{A}_{i-1}}$ identified by the public keys $\mathbf{A}_1, \ldots, \mathbf{A}_{i-1}$, a sequence of messages $m_1, \ldots, m_{i-1}$, parts of the encoded signatures $\vec{\alpha}_{i-1}$ from signers S_1 to S_{i-2}, an aggregate signature σ_{i-1} of the predecessor S_{i-1} and a hash value $h_{i-1} \in \{0,1\}^l$. Before adding its own signature to Σ_i, the signer checks the validity of the received aggregate by running the verification algorithm on Σ_{i-1}. If the verification succeeds, S_i continues by invoking the encoder on σ_{i-1} resulting in a breakdown (α_i, β_i). The encoder $enc : \{0,1\}^* \to \{0,1\}^* \times R_{n_i}$ is an injective map that splits up the signature into two parts such that β_i can completely be embedded in the computation of σ_i and can always be recovered. The second part α_i is simply appended to the list $\vec{\alpha}_{i-1}$ and plays an important role when recovering the intermediate aggregate signatures. We give a particular instantiation of the proposed splitting algorithm in Section 8.3. The next two steps involve two hash functions $H : \{0,1\}^* \to \{0,1\}^l$ and $G_{f_{\mathbf{A}_i}} : \{0,1\}^l \to R_{n_i}$ which are modeled as random oracles. Similar to [BR96], $H(\cdot)$ is considered the compressor that hashes the message down to l bits, whereas $G_{f_{\mathbf{A}_i}}(\cdot)$ is called the generator and outputs random elements from the image space of $f_{\mathbf{A}_i}$. Regarding the proof of security, such a construction involving $H(\cdot)$ and $G(\cdot)$ avoids the need for certified trapdoor functions satisfying the properties specified in Appendix 6.1.1. By this means, one gets rid of costly checks, because a potential adversary could generate keys leaving out one of

these properties. Finally, the algorithm outputs Σ_i containing the modified aggregate signature. The verification algorithm $\mathsf{AggVerify}(\cdot)$ proceeds in the reverse order and takes Σ_n as input. In each iteration it checks the validity of σ_i and recovers σ_{i-1} with the aid of the decoder $\mathsf{dec}(\cdot)$.

In the following section we present the security model of our scheme including the associated security proof. Subsequently, we show how to instantiate the scheme with PSTFs. To this end, we focus on the provably secure NTRUSign preimage sampleable trapdoor function and provide a comparison with RSA-based SAS schemes as proposed in [LMRS04, Nev08]. Finally, we indicate how to build a proxy signature scheme from any SAS scheme.

8.2. Security Model

We adopt the security model proposed by Neven [Nev08] for sequential aggregate signatures. Moreover, we examine our lattice-based construction in a slightly different setting that is build upon a stronger security assumption $\mathsf{Exp}^{SSAS-SU-CMA}$ and subsumes the former ones [Nev08, LMRS04]. Usually, a sequential aggregate signature scheme is considered to be secure, if it is infeasible to provide existential forgeries of a sequential aggregate signature. The core idea behind these security models is to let the forger F control the private keys and sequential aggregate signatures of all but at least one honest signer. Thus, the forger is allowed to select the public keys of the fake signers. Neven introduces the notion of a sequential forger $\mathcal{S}$ that can be built from a forger F with about the same success probability and running time [Nev08, Lemma 5.3]. Therefore, it is more convenient to consider a sequential forger in our proof of security. The way the sequential forger is constructed out of F can directly be transferred to our setting with some minor changes. The properties of a sequential forger are as follows:

1. Any input to the random oracles $H(\cdot)$ and $G_{f.}(\cdot)$ is queried once, where $f.$ denotes any preimage sampleable trapdoor function. The signing oracle $\mathsf{OAggSign}$ is also queried once for the same input.
2. Each input Q_n to $H(\cdot)$ parses as $Q_n = (\overrightarrow{f_{\mathbf{A}_k}}, \vec{m}_k, \vec{\alpha}_{k-1}, \sigma_{k-1})$ such that $\log_2(R_{n_i}) > l$ holds for $1 \leq i \leq k$ and $k \leq k_{max}$.
3. Before any query $Q_k = (\overrightarrow{f_{\mathbf{A}_k}}, \overrightarrow{m_k}, \vec{\alpha}_{k-1}, \sigma_{k-1})$ to $H(\cdot)$ for $n > 1$, the sequential forger must have made queries Q_i to $H(\cdot)$ for $1 \leq i < k \leq k_{max}$ such that $\mathsf{dec}_{f_{\mathbf{A}_i}}(\alpha_i, f_{\mathbf{A}_i}(\sigma_i) - G_{f_{\mathbf{A}_i}}(h_i)) = \sigma_{i-1}$ for $h_i = h_{i-1} \oplus H(Q_i)$.
4. Preceding any signing query $\mathsf{OAggSign}(\mathbf{T}^*, m_k, \Sigma_{k-1})$ the sequential forger must have made the necessary $H(\cdot)$ and $G_{f.}(\cdot)$ queries in advance with due regard to Property 3. Furthermore, the input query Σ_{k-1} must be valid such that verification algorithm $\mathsf{AggVerify}(\Sigma_{k-1})$ does not fail.

5. Forgeries output by $\mathcal{S}$ must be valid and include the challenge public key at some index i such that $f_{\mathbf{A}_i} = f_{\mathbf{A}^*}$ for $1 \leq i \leq k \leq k_{max}$. We explicitly allow $\mathcal{S}$ to output forgeries on data Σ_{i-1} that has been signed by the signing oracle. The only required restriction is that the signing oracle responses and the forgery must differ on the same input.

According to an adaptive chosen-message attack we permit $\mathcal{S}$ to make arbitrary many sequential aggregate signature queries to the honest signer on messages of its choice. The advantage $\mathsf{AdvAggSign}^*_{\mathcal{S}}$ of $\mathcal{S}$ is the success probability in the following experiments.

Setup
The key generation algorithm is invoked in order to produce the challenge key pair $(\mathbf{T}^*, \mathbf{A}^*)$. The challenge key $f_{\mathbf{A}^*}$ is then handed over to the sequential aggregate forger $\mathcal{S}$.

Queries
The adversary $\mathcal{S}$ has access to the signing oracle $\mathsf{OAggSign}(\mathbf{T}^*, *, *)$. $\mathcal{S}$ acts adaptively and provides to the signing oracle a message m_i to be signed, a sequential aggregate signature σ_{i-1} on a sequence of messages $m_1, \ldots, m_{i-1}$ and data $\alpha_1, \ldots, \alpha_{i-1}$ under public keys $f_{\mathbf{A}_1}, \ldots, f_{\mathbf{A}_{i-1}}$. The oracle returns an aggregate signature under the challenge public key $f_{\mathbf{A}^*}$. Furthermore, we allow $\mathcal{S}$ to have random oracle access to some random functions as required in the random oracle model.

Response
$\mathcal{S}$ eventually outputs a sequential aggregate signature σ_k on k distinct public keys $f_{\mathbf{A}_1}, \ldots, f_{\mathbf{A}_k}$, where one of them corresponds to the challenge key. Moreover, $\mathcal{S}$ outputs k messages $m_1, \ldots, m_k$, each corresponding to one public key.

The forger wins the game $\mathsf{Exp}^{SSAS-EU-CMA}_{\mathcal{A},SAS}(n)$, if he succeeds in outputting a non-trivial valid sequential signature on a sequence of k messages $m_1, \ldots, m_k$ under k distinct public keys $f_{\mathbf{A}_1}, \ldots, f_{\mathbf{A}_k}$ containing the challenge public key $f_{\mathbf{A}_i} = f_{\mathbf{A}_*}$ at some index i. A valid signature is said to be non-trivial, when $\mathcal{S}$ has never made a query to the signing oracle on messages $m_1, \ldots, m_i$ and public keys $f_{\mathbf{A}_1}, \ldots, f_{\mathbf{A}_i}$ before, or he is able to output a forgery that differs from the received signing oracle responses. In the latter case we even allow the forger to use already signed messages to output a forgery as opposed to the security models from [Nev08, LMRS04] focusing on trapdoor permutations. This security notion reflects the strong sequential aggregate signature unforgeability (SAS-SU-CMA), which can be formalized as follows.

```
Experiment $\mathsf{Exp}^{SSAS-SU-CMA}_{\mathcal{A},SAS}(n)$
    $(\mathbf{T}^*, \mathbf{A}^*) \longleftarrow \mathsf{KeyGen}(1^n)$
    $\Sigma = (\overrightarrow{f_{\mathbf{A}_i}}, \overrightarrow{m_i}, \overrightarrow{\alpha_i}, \sigma_{\mathbf{i}}, h_i) \longleftarrow \mathcal{A}^{\mathsf{OAggSign}(\mathbf{T}^*,*,*)}(\overrightarrow{f_{\mathbf{A}^*}})$
    Let $f_{\mathbf{A}_i} = f_{\mathbf{A}^*}$ be the challenge public key in $\overrightarrow{f_{\mathbf{A}_i}} = (f_{\mathbf{A}_1}, \ldots, f_{\mathbf{A}_i})$ and
    $\overrightarrow{m_i} = (m_1, \ldots, m_i)$
    Let $((f_{\mathbf{A}^*}, m_l, \Sigma_{l-1}), \Sigma_l)_{l=1}^{Q_{AS}}$ be query-response tuples of $\mathsf{OAggSign}(\mathbf{T}^*, *, *)$
    Return 1 if $\mathsf{AggVerify}(\Sigma) = (\overrightarrow{\mathsf{f}_{\mathbf{A}_i}},\ \overrightarrow{m_i})$
        and $\Sigma \notin \{\Sigma_l\}_{l=1}^{Q_{AS}}$
```

The adversary is said to be successful in this experiment if he efficiently provides a valid sequential aggregate signature with non-negligible advantage.

8.2.1. Security of our Construction

A collision-finding algorithm $\mathcal{A}$ is said to (t', ϵ')-break a collision-resistant preimage sampleable trapdoor function family (PSTF) if it has running time t' and outputs a collision with probability

$$Pr[f_{\mathbf{B}}(x_1) = f_{\mathbf{B}}(x_2) \mid (\mathbf{B}, \mathbf{T}) \leftarrow \mathsf{TrapGen}(1^n),\ (x_1, x_2) \leftarrow \mathcal{A}(f_{\mathbf{B}})]$$

of at least ϵ'.

Proposition 8.3. *If there exists a sequential forger $\mathcal{S}$ that $(t, q_S, q_H, q_G, k_{max}, \epsilon)$-breaks SAS, then there exists a collision-finding algorithm $\mathcal{A}$ that (t', ϵ')-breaks the collision-resistant PSTF for*

$$\epsilon' \geq \epsilon \cdot (1 - \frac{q_H(q_H + q_G)}{2^l}) - \frac{q_H}{2^{\omega(\log n)}}$$
$$t' \leq t + (q_H + k_{max}) \cdot t_{f.} + q_H \cdot t_{\mathsf{SampleDom}}\,.$$

Proof : By contradiction, we assume that there exists a successful sequential forger $\mathcal{S}$ that breaks the SAS with non-negligible probability ϵ. Using $\mathcal{S}$, we construct a poly-time algorithm $\mathcal{A}$ that finds a collision in the collision-resistant trapdoor function $f_{\mathbf{A}_i} : B_{n_i} \longrightarrow R_{n_i}$ with probability negligibly close to ϵ. Given the challenge public key $\mathbf{A}^*$ of the trapdoor function $f_{\mathbf{A}^*}$, $\mathcal{A}$ runs $\mathcal{S}$ on public key $\mathbf{A}^*$ with $f_{\mathbf{A}^*} : B_n \longrightarrow R_n$ and simulates the environment as follows:

Setup : At the beginning of this game algorithm $\mathcal{A}$ sets up the empty lists $HT[*]$ and $GT[*, *]$.

H-Random oracle query $H(Q_i)$: After parsing the input Q_i as $(\overrightarrow{f_{\mathbf{A}_i}},\ \overrightarrow{m}_i,\ \overrightarrow{\alpha}_{i-1},\ \sigma_{i-1})$, $\mathcal{A}$ checks the index i. If $i = 1$, $\mathcal{A}$ sets $h_0 \leftarrow 0^n$. In case $i > 1$, following Property 3 of a sequential forger there exists a unique sequence of random oracle queries $Q_1, \ldots, Q_{i-1}$ with table entry $H(Q_{i-1}) = (\sigma_{i-1}, h_{i-1})$.
If the public key $f_{\mathbf{A}_i}$ does not correspond to the challenge public key $f_{\mathbf{A}^*}$, then $\mathcal{A}$ continues as follows:

- $h \leftarrow_R \{0,1\}^l, \ h_i = h \oplus h_{i-1}$ and sets $HT[Q_i] \leftarrow (\perp, h_i)$.

Otherwise, if $f_{\mathbf{A}_i} = f_{\mathbf{A}^*}$, then $\mathcal{A}$ performs the following tasks:

- $h \leftarrow_R \{0,1\}^l, \ h_i = h \oplus h_{i-1}$
- $(\alpha_i, \beta_i) \leftarrow \mathsf{enc}_{f_{\mathbf{A}_i}}(\sigma_{i-1})$
- $\sigma_i \leftarrow \mathsf{SampleDom}(1^n)$ and compute $g \leftarrow f_{\mathbf{A}^*}(\sigma_i) - \beta_i \in R_{n_i}$, since R_{n_i} is additive.
  (By Property 1 of PSTFs according to Section 6.1.1, $f_{\mathbf{A}^*}(\sigma_i) \sim \mathcal{U}(R_{n_i})$)
  If $G_{f_{\mathbf{A}^*}}(h_i)$ has not been defined, he sets $G[f_{\mathbf{A}^*}, h_i] \leftarrow g$, $HT[Q_i] \leftarrow (\sigma_i, h_i)$ and outputs h to $\mathcal{S}$, otherwise BAD_1 occured and $\mathcal{A}$ aborts.

G-Random oracle query 1 $G_{f_{\mathbf{A}_i}}(h)$: On input $f_{\mathbf{A}_i}$ and h algorithm $\mathcal{A}$ checks the entry $GT[f_{\mathbf{A}_i}, h]$. If it is not defined, it selects $g \leftarrow_R R_{n_i}$ uniformly at random, sets $GT[f_{\mathbf{A}_i}, h] = g$ and returns g to $\mathcal{S}$. By Property 1 of a sequential forger it does not make the same query again.

Sequential signing query $\mathsf{OAggSign}(\mathbf{T}^*, m_i, \Sigma_{i-1})$: $\mathcal{A}$ extracts the values $\overrightarrow{f_{\mathbf{A}_{i-1}}}$, $\vec{m}_{i-1}$, $\vec{\alpha}_{i-1}$, σ_{i-1}, h_{i-1} from Σ_{i-1}. As per Property 4 he finds a non-empty entry $HT[Q_{i-1}] = (\sigma_i, h_i)$ with $Q_{i-1} = (\overrightarrow{f_{\mathbf{A}_{i-1}}} \mid f_{\mathbf{A}^*}, \ \vec{m}_{i-1} \mid m_i, \ \vec{\alpha}_{i-1}, \ \sigma_{i-1})$. Then $\mathcal{A}$ returns $\Sigma_i = (\overrightarrow{f_{\mathbf{A}_i}}, \ \vec{m}_i, \ \vec{\alpha}_i, \ \sigma_i, h_i)$ with $(\alpha_i, \beta_i) \leftarrow \mathsf{enc}_{f_{\mathbf{A}_i}}(\sigma_{i-1})$

Finally, the forger $\mathcal{S}$ outputs a valid forgery $\Sigma'_k = (\overrightarrow{f_{\mathbf{A}_k}}, \ \vec{m}_k, \ \vec{\alpha}_k, \ \sigma_k)$ with probability ϵ as per property 5. Since Σ'_k is valid, we have $\mathsf{AggVerify}(\Sigma'_k) = (\overrightarrow{f_{\mathbf{A}_k}}, \vec{m}_k)$ and $\overrightarrow{f_{\mathbf{A}_k}}$ includes the challenge public key $f_{\mathbf{A}_i} = f_{\mathbf{A}^*}$ at index $1 \leq i \leq k$. During the execution of $\mathsf{AggVerify}(\Sigma'_k)$ we get m_i and Σ'_i containing σ'_i. $\mathcal{A}$ proceeds as follows in order to obtain a collision. We now have to differ two cases:

1. If $\mathcal{S}$ already made a signature query on (m_i, Σ_{i-1}), it received back Σ_i containing the signature σ^*. Since Σ'_k is a forgery, we have $\sigma'_i \neq \sigma^*$ such that $f_{\mathbf{A}^*}(\sigma'_i) = f_{\mathbf{A}^*}(\sigma^*)$
2. In the case, $\mathcal{S}$ did not request a signature on (m_i, Σ_{i-1}) from the signing oracle, by Property 4 there exists an entry $HT[Q_{i-1}] = (\sigma^*, h_i)$ with $\sigma^* \longleftarrow \mathsf{SampleDom}(1^n)$ and $GT[f_{\mathbf{A}^*}, h_i] = g_i$ such that $f_{\mathbf{A}^*}(\sigma^*) = g_i + \beta_i = f_{\mathbf{A}_i}(\sigma'_i)$ and $h = h_i \oplus h_{i-1}$ is returned to $\mathcal{S}$. If $\sigma^* = \sigma'_i$, then BAD_2 occured and $\mathcal{A}$ aborts.

In both cases $\mathcal{A}$ found a collision in $f_{\mathbf{A}^*}$ (which is infeasible due to Property 4 of trapdoor functions from Section 6.1.1).

Analysis and security:

We define by $\neg BAD_i$ the event that BAD_i does not occur. $\mathcal{S}$'s environment is perfectly simulated as in the real system, when the events BAD_1 and BAD_2 do not occur. Thus, we have

$$P[\mathcal{S} \text{ ouputs forgery} \mid \neg BAD_1 \wedge \neg BAD2] = \epsilon\,.$$

$\mathcal{A}$ wins the game when $\mathcal{S}$ succeeds in providing a valid forgery and the events BAD_1 and BAD_2 do not happen. Therefore, we need to estimate an upper bound for the probability of a successful forger:

$$P[\mathcal{A} \text{ wins}] \quad = \quad \epsilon \cdot P[\neg BAD_1] - P[BAD_2]\,.$$

$$\begin{aligned}
P[\mathcal{A} \text{ wins}] &= P[\mathcal{S} \text{ outp. forgery} \wedge \neg BAD_1 \wedge \neg BAD_2] \\
&= P[\mathcal{S} \text{ outp. forgery} \mid \neg BAD_1 \wedge \neg BAD_2] \cdot P[\neg BAD_1 \wedge \neg BAD_2] \\
&= P[\mathcal{S} \text{ outp. forgery} \mid \neg BAD_1 \wedge \neg BAD_2] \cdot (1 - P[BAD_1 \vee BAD_2]) \\
&\geq P[\mathcal{S} \text{ outp. forgery} \mid \neg BAD_1 \wedge \neg BAD_2] \cdot (1 - \sum_i P[BAD_i]) \\
&= P[\mathcal{S} \text{ outp. forgery} \mid \neg BAD_1 \wedge \neg BAD_2] \cdot (P[\neg BAD_1] - P[BAD_2]) \\
&\geq P[\mathcal{S} \text{ outp. forgery} \mid \neg BAD_1 \wedge \neg BAD_2] \cdot P[\neg BAD_1] - P[BAD_2] \\
&= \epsilon \cdot P[\neg BAD_1] - P[BAD_2]
\end{aligned}$$

The event BAD_1 occurs when algorithm $\mathcal{A}$ chooses a fresh random value $h \leftarrow_R \{0,1\}^l$ in the H-Random oracle query step and attempts to set a table entry $GT[*, h_k]$ that is already defined, where $h_k = h \oplus h_{k-1}$. The probability of this event is

$$P[BAD_1] = \frac{|GT|}{2^l} \leq \frac{q_H(q_H + q_G)}{2^l}$$

where the last term follows by summation over all H-queries to the simulation. The event BAD_2 occurs when the forger $\mathcal{S}$ outputs a valid forgery σ_i' that is equal to the corresponding table entry $HT(Q_{i-1}) = (\sigma^*, *)$. Based on the conditional min-entropy property of σ^* given $f_{\mathbf{A}^*}(\sigma^*)$ the probability of BAD_2 to happen is

$$P[BAD_2] \leq \frac{q_H}{2^{\omega(\log n)}},$$

which is negligible. Therefore, we obtain

$$\epsilon' \geq \epsilon \cdot (1 - \frac{q_H(q_H + q_G)}{2^l}) - \frac{q_H}{2^{\omega(\log n)}}\,.$$

We derive an upper bound for the running time of $\mathcal{S}$ taking into account only function evaluations and invocations of SampleDom. Each verification requires at most k_{max} function evaluations. Invoking $H(\cdot)$ implies at most one execution of

SampleDom and two function evaluations, thereof one evaluation to identify the sequence $Q_{k-1}, \ldots, Q_1$. Therefore, the running time is upper bounded by:

$$t' \leq t + (2q_H + k_{max}) \cdot t_f + q_H \cdot t_{\mathsf{SampleDom}} .$$

$\square$

Proposition 8.4. *The proposed sequential aggregate signature scheme is strongly existentially unforgeable under chosen-message attack.*

Proof. By Proposition 8.3 finding collisions for preimage sampleable trapdoor functions can be reduced to the hardness of forging sequential aggregate signatures in the SAS described above. The authors of [GPV08] give the corresponding algorithms of how to instantiate preimage sampleable trapdoor functions by means of lattices satisfying the required properties and show by [GPV08, Theorem 5.9] that the task of finding collisions is as hard as solving $SIS_{q,n,2s\sqrt{m}}$. $\square$

The security proof of the unstateful probabilistic FDH scheme is almost identical to the stateful one. One notices, that the extended message $m||r$ to be signed always differs for repeated request queries on the same message m due to the random salt r. As in Proposition 8.4 one reduces collision-resistance to the unforgeability of sequential aggregate signatures.

8.3. Instantiation

In general, one can use any collision-resistant trapdoor function that is suitable for the GPV signature scheme. In particular, one can instantiate the SAS scheme with the trapdoor constructions from [GPV08, AP09, Pei10, MP12]. In this section we analyze the proposed sequential aggregate signature scheme in conjunction with NTRUSign. Therefore, let $f_{\mathbf{A}_i} : B_{n_i} \longrightarrow R_{n_i}, 1 \leq i \leq k$ be a family of preimage-sampleable trapdoor functions, each corresponding to the public key $\mathbf{A}_i$ of signer S_i. B_{n_i} denotes the domain of the trapdoor function $f_{\mathbf{A}_i}$ and can be represented by vectors of bit size $\log_2(B_{n_i}) := \max_{b \in B_i} \lceil \log_2 b \rceil$. Analogously, one defines the maximum bit size $\log_2(R_{n_i})$ of the image space. For instance, if we choose $B_{n_i} = \{\mathbf{z} \in \mathbb{Z}^{m_i} \mid \|\mathbf{z}\| \leq s_i\sqrt{m_i}\}$ and $R_{n_i} = \mathbb{Z}_{q_i}^{n_i}$, we have $\log_2(R_{n_i}) = n_i \cdot \lceil \log_2(q_i) \rceil$ and respectively $\log_2(B_{n_i}) \leq m_i \cdot (\lceil \log_2(s_i) + 1 \rceil)$ with overwhelming probability according to Lemma 6.2. The encoding function $\mathsf{enc}(\cdot)$ can, therefore, be built as follows. The range R_{n_i} is converted into a large bit string that is subsequently split into blocks of size $\lceil \log_2(4.7 \cdot s_i) + 1 \rceil$ bits. Each block is then filled with an entry from the signature σ_i. There are many possibilities to handle the last block as it may contain less bits. Finally, the bit string is converted back to the vector presentation β_{i+1}. The remaining signature bits are stored in the vector α_{i+1}, which is appended to the aggregate signature.

Security and Performance

The bit security of this scheme mainly depends on the bit security of each signers key and the system parameter l. Hence, the security of our construction is upper bounded by $\min\limits_{1\leq i\leq k}(c_i, l)$, where c_i denotes the bit security of the i-th signer. To determine the performance we ignore all operations beside function evaluations and preimage samplings. The signing costs of the i-th signer amount to one call of $\mathsf{SamplePre}(\cdot)$ and $(i-1)$ function evaluations $f_{\mathbf{A}}$. Verification requires k function evaluations.

In what follows, we will focus particularly on the trapdoor construction provided in [SS11] since it has some nice properties which can be utilized in the proposed SAS construction. A crucial factor for our choice is a low ratio $\log_2(B_{n_i})/\log_2(R_{n_i})$ as compared to other lattice-based PSTFs. This ratio implicitly affects the compression rate, since a ratio equal to or smaller than 1 implies optimal compression rates for equal parameters n_i, meaning that signatures completely fit into the image space without wrapping around.

Efficient Instantiation with provably secure NTRUSign

The provably secure NTRU-Sign signature scheme proposed by Stehlé et al. [SS11] is a full domain hash scheme satisfying the properties of collision-resistant PSTFs from Section 6.1.1.

$\mathsf{KeyGen}(q, n, 1^n)$ It returns public key $\mathbf{A} = \mathbf{g}/\mathbf{f} \in \mathcal{R}_q^{\times}$ and trapdoor $\mathbf{T} = \begin{bmatrix}\mathbf{f} & \mathbf{g}\\ \mathbf{F} & \mathbf{G}\end{bmatrix}$ for $\mathsf{f}_{\mathbf{A}}(\sigma^{(1)}, \sigma^{(2)}) = \mathbf{A}\sigma^{(1)} - \sigma^{(2)}$, where $\mathsf{f}_{\mathbf{A}} : B_n \to R_n = \mathcal{R}_q$ with $B_n = \{(\sigma^{(1)}, \sigma^{(2)}) \in \mathcal{R}^2 : \left\|(\sigma^{(1)}, \sigma^{(2)})\right\| \leq s \cdot \sqrt{2n}\}$.

$\mathsf{Sign}(\mathbf{T}, m)$ The signing algorithm computes the hash value $H(m||r)$ of the extended message $m||r$ with a random seed $r \leftarrow_R U(\{0,1\}^d)$. Then it samples $\sigma = (\sigma^{(1)}, \sigma^{(2)}) \leftarrow \mathsf{SamplePre}(\mathbf{T}, H(m||r))$ and outputs $(r, \sigma^{(1)})$ as the signature.

$\mathsf{Verify}(\sigma, m)$ The verification algorithm computes $\mathbf{t} = H(m||r) \in \mathcal{R}_q$ and determines $\sigma^{(2)} = \mathbf{A}\sigma^{(1)} - \mathbf{t}$. If the conditions $\sigma \in B_n$ and $r \in \{0,1\}^d$ are valid, it outputs 1, otherwise 0.

When instantiating the SAS scheme with this trapdoor construction, we obtain compression factors of strictly larger than 50 %. For the sake of simplicity, assume we have public keys $\mathbf{A}_i \in \mathcal{R}_q$ with identical parameters q, $\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$ for n a power of two, which is obviously different from RSA where the moduli $N = p \cdot q$ have to be distinct since otherwise they would share the same secret. An NTRUSign signature is a vector $(r, \sigma^{(1)}, \sigma^{(2)})$ such that the bit size of $\sigma^{(j)}$ is bounded by $n \cdot (\lceil \log_2(s) \rceil + 1) < \log_2(R_n)$ with overwhelming probability and $r \in \{0,1\}^n$. Any vector of the image space occupies at most $\log_2(R_n) = n \cdot \lceil \log_2(q) \rceil$ bits of memory. In general, one can use Algorithm 12 and Algorithm 13 in order to instantiate the NTRUSign SAS scheme. Since we consider the probabilistic FDH approach using a random seed r, one simply replaces messages m_i by the extended messages $m_i||r_i$.

8.3.1. Comparison with RSA-based SAS

RSA based sequential signatures due to [LMRS04, Nev08] are less flexible compared to the proposed construction. In particular, the public keys $N_i = p_i \cdot q_i$ of RSA based instances have to be distinct and satisfy more restrictive conditions in order to make the scheme work. For example in [LMRS04], the hash space of $H(\cdot)$ requires to be specified before starting aggregation. This can be attributed to the differing domains $\mathbb{Z}_{N_i}^{\times}$ as a result of different moduli N_i. For instance, the hash space is chosen to be a proper subset of $\mathbb{Z}_{N_1}^{\times}$. However, this is not the case in our construction, since we can use equal domains and ranges without any security concerns. Thus, one allows the corresponding hash functions $\mathbf{G}_{\mathbf{f}_{\mathbf{A}_i}}(\cdot)$ to be equal. In order to achieve high compression without blowing up the aggregate signatures too much, the bit sizes of public keys have to be identical or are ordered to be increasing in RSA based SAS schemes. This is due to the fact that the signatures are uniform random elements in $\mathbb{Z}_{N_i}^{\times}$ and can only be fully embedded in $\mathbb{Z}_{N_{i+1}}^{\times}$ if $b_i \leq b_{i+1}$ or $N_i < N_{i+1}$ is satisfied for $b_i = \lceil \log(N_i) \rceil$ and $1 \leq i \leq k$. Indeed, this also holds for lattice-based constructions. Specifically, one has to increase the parameters n_{i+1} or q_{i+1} such that $\log_2(D_{n_i}) \leq \log_2(R_{n_{i+1}})$. By this, we have aggregate signatures being as large as individual ones.

8.3.2. Analysis

We want to derive a measure for the quality of the SAS scheme. Therefore, we consider the compression rate measuring the storage savings due to the SAS scheme. One simply relates the bit size of the aggregate signature to the total size of all individual signatures, which corresponds to the case one does not employ SAS schemes. By [SS11, Theorem 4.2] an NTRUSign signature is distributed as a discrete Gaussian vector with parameter $s = \omega(n^2 \cdot \sqrt{\ln n \cdot \ln(8nq)} \cdot q^{1/2+\epsilon})$ and $\epsilon \in (0, \frac{\ln\ n}{\ln q})$. In principal, it is possible to choose the parameters q_i and n_i of the signers in such a way that the aggregate signature has the size of an individual signature. Since there is a wide choice of selecting the chain of signers, which result in different compression rates, we restrict to the case, where q_i and n_i are equal for all signers. The aggregate signature is of the form $(\sigma_i, \vec{\alpha}_i, h_i)$ consisting of σ_i of size $2n\lceil \log_2(4.7 \cdot s) \rceil$ bits, $\vec{\alpha}_i$ of size $(i-1) \cdot n(2\lceil \log_2(4.7 \cdot s) \rceil - \lceil \log_2(q) \rceil)$ bits and h_i occupying l bits of memory. Each signer in the chain produces $n(2\lceil \log_2(4.7 \cdot s) \rceil - \lceil \log_2(q) \rceil)$ bits of overhead.

Since the length of the signature strongly depends on q, we consider two cases for the choice of q. First, we let $q = n^{\omega(1)}$ to be slightly superpolynomial in n such that $\log_2(n) = o(\log_2(q))$. For the compression rate we then have:

$$\begin{aligned}
\theta(i) &= 1 - \frac{\lceil 2n\log_2(4.7\cdot s)\rceil + (i-1)\cdot n(2\lceil\log_2(4.7\cdot s)\rceil - \lceil\log_2(q)\rceil) + l}{i\cdot 2n\lceil\log_2(4.7\cdot s)\rceil} \\
&\geq 1 - \left(\frac{1}{i} + \frac{2n(\lceil\log_2(4.7\cdot s) - \log_2(q^{1/2})\rceil + 1) + l/i}{2n\lceil\log_2(4.7\cdot s)\rceil}\right) \\
&\geq 1 - \left(\frac{1}{i} + \frac{\lceil\log_2(4.7\cdot n^2\sqrt{\ln(n)\ln(8nq)})\rceil + l/(2\cdot n\cdot i) + 1}{\lceil\log_2(4.7\cdot s)\rceil}\right) \\
&= 1 - \left(\frac{1}{i} + \frac{o(\log_2(q))}{o(\log_2(q)) + \log_2(q^{1/2})}\right).
\end{aligned}$$

Thus the compression rate converges towards $1 - 1/i$ which is asymptotically optimal, meaning that in average aggregate signatures and individual signatures are of equal size. We note that similar to [BPR12, AKPW13] solving γ-Ideal-SVP with slightly superpolynomial factors γ appears to be exponentially hard given present best attack-algorithms.

Secondly, we let $q = Poly(n)$. By a trivial computation using $q = n^{2c}$ we have $\mathsf{rate}(\sigma_i) \approx 1 - 1/i - 1/c$, meaning that each signer produces $1/c \cdot \mathsf{size}(\sigma_i)$ of overhead per signature. As a result, we obtain an $(1 - 1/c)$-SAS scheme. So, choosing c large enough returns an almost optimal SAS scheme.

8.3.3. Proxy Signatures

From the aforementioned sequential aggregate signature scheme one can immediately build a proxy signature scheme using the generic construction from [SMP08]. The core idea of a proxy signature scheme is to allow a potential signer, called delegator, to delegate its signing rights to a subentity, called proxy, which is enabled to sign documents on behalf of the delegator. Any verifier can figure out whether a signature is indeed produced by a proxy signer and if he received the signing rights from the delegator. The security of the proxy signature scheme is related to the security of the SAS scheme as stated in Theorem 8.5.

Theorem 8.5. ***([SMP08, Theorem 2])*** *Let $\mathcal{AS}$ be a (t, q_s, ϵ)-unforgeable sequential aggregate signature scheme. Then, the above construction provides a $(t_0, q_s', q_d', \epsilon')$-unforgeable proxy signature scheme where $\epsilon = \epsilon'/2qd$, $t = t'$ and $qs = q_s' + q_d'$.*

Part III.

Lattice Representations

9. Representation Formula for Lattice Problems

In this chapter we provide a different view to lattice problems by use of tools from complex analysis. We aim at formalizing the solutions of lattice problems by a general formula. In particular, we introduce generalizations of Cauchy integrals to higher dimensional complex spaces as a main building block that decide membership of a mathematical object within a domain. This allows us to answer the question whether a lattice point is lying inside a domain by evaluating generalized Cauchy integrals. Subsequently, we can deduce a simple and easy to understand representation formula for solutions of well-known lattice problems and their approximated versions. Such a formula has the benefit of abstracting from certain algorithmic views and further extends the theoretical framework for analyzing lattice problems. In fact, we show that putting lattice points into the denominator of a series allows to derive interesting results such as a formula representing a lattice via its poles. Once a function has its simple poles at lattice-points the residue theorem from complex analysis is applicable offering the opportunity to deduce lattice relations. In particular, we consider the one-dimensional and two-dimensional case, which are closely related to elliptic functions. For higher dimensional lattices, we present some basic ideas, but point out that such simple relations seem hard to obtain requiring an introduction into the theory of abelian functions or higher dimensional complex tori.

The number of lattice-points inside a domain has many application areas such as predicting the length of the shortest vector in a lattice. Due to the hardness to determine the respective quantity, the Gauss heuristic is often applied instead with the goal to estimate the attack complexity of lattice attack algorithms. This heuristic is easy to compute, but has the major disadvantage to be non-precise, directly affecting subsequent computations. The proposed approach may be considered as an initial step towards resolving this issue. The abstraction to general formulas also allows to analyze lattice problems from a different point of view. Once having a simple expression in the integral, it is possible to derive interesting properties such as conditions of how to select parameters. For instance, in the one-dimensional case we can directly translate the lattice periodicity into trigonometric functions resulting in a closed expression, which allows to derive additional features from the properties of well-studied trigonometric functions. But also in combination with quantum algorithms such as Grover's search algorithm it might speed-up the search for solutions particularly for approximated versions of lattice problems. Similarly, we proceed with the two-dimensional case, where the Weierstrass zeta function rep-

resents a converging function with simple poles at all lattice-points.
This chapter refers to the paper [EB15c], where the author of this thesis was the primary investigator and author of the publication.

9.1. Cauchy Integrals

In this section we present one of the main building blocks of this chapter. More specifically, we introduce the Cauchy integral and its generalizations to arbitrary dimensions. In fact, we give a formula for many lattice problems that is purely mathematical rather than algorithmical. It can be represented as a finite sum of Cauchy integrals. The use of Cauchy integrals and its generalizations may raise further applications. For instance, it is conceivable to integrate over seemingly complicated surfaces, where the corresponding integrals are computed efficiently. We start by introducing the standard Cauchy integral from complex analysis, which has the nice property that function values of a holomorphic function can be computed via path integration. In addition, Cauchy integrals offer much flexibility concerning the underlying function such that different functions can still have the same integral value.

9.1.1. Complex Space

The complex vector space $\mathbb{C}^n$ is a Euclidean vector space with $\mathbb{C}^n \cong \mathbb{R}^{2n}$. The topology in $\mathbb{C}^n$ is given by the metric $d(\mathbf{z}, \mathbf{y}) = |\mathbf{z} - \mathbf{y}|$, where $|\mathbf{z}| = \sqrt{\langle \mathbf{z}, \bar{\mathbf{z}} \rangle}$ for the inner product map $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{n} x_i y_i$ and $\bar{\mathbf{z}} = (\bar{z}_1, \ldots, \bar{z}_n)$ with $z_i \in \mathbb{C}$. For $\mathbf{z} \in \mathbb{C}^n$, we define $Re(\mathbf{z}) := (Re(z_1), \ldots, Re(z_n)) \in \mathbb{R}^n$ and $Im(\mathbf{z}) \in \mathbb{R}^n$ accordingly. By $\bar{D}$ we denote the topological closure of a set $D \subseteq \mathbb{C}^n$.

In the following, we provide a definition of holomorphic functions in multidimensional complex spaces, since it is an important requirement for many of the theorems that follow.

Definition 9.1 (Holomorphic Function). *Let $D \subseteq \mathbb{C}^n$ be an open set. A complex-valued function f defined on D is said to be holomorphic on D if $f \in \mathcal{C}^1(D)$ (differentiable) and*

$$\frac{\partial f}{\partial \bar{z}_j}(\mathbf{z}) = 0 \textit{ for every } \mathbf{z} \in D \textit{ and } j = 1, \ldots, n\,.$$

The next definition plays an important role particularly for the Bochner-Martinelli integral formula introduced in the following section.

Definition 9.2 (Smooth Boundary). *Suppose that $D \subset \mathbb{C}^n$ is a domain. ∂D is said to be a class $\mathcal{C}^1$ boundary, if there exists a real-valued class $\mathcal{C}^1$ function $\rho : \mathbb{C}^n \longrightarrow \mathbb{R}$ such that $D = \{\mathbf{x} \in \mathbb{C}^n \mid \rho(\mathbf{x}) < 0\}$ and $\partial D = \{\mathbf{x} \in \mathbb{C}^n \mid \rho(\mathbf{x}) = 0\}$ and the gradient vector $\nabla \rho \neq 0$ on ∂D.*

9.1.2. Integral Formulas

We first state the classical Cauchy integral formula from complex analysis and subsequently present a description of the multidimensional setting.

Theorem 9.3. *Let f be holomorphic in a simply connected domain D. Let $a \in D$ and $B \subseteq D$ be a region inside D. Then*

$$\frac{1}{2\pi i} \int\limits_{\partial B} \frac{f(z)}{z-a} dz = \begin{cases} f(a), & \text{if } a \in B \\ 0, & \text{if } a \notin \bar{B}\,. \end{cases}$$

In fact, for any path $\gamma \subset D$ the integral $\frac{1}{2\pi i} \int\limits_{\gamma} \frac{f(z)}{z-a} dz$ evaluates to $f(a)$ in case γ encircles a, and otherwise the integral equals to zero, in case a lies outside the circle. Amazingly, one observes that integration is performed only along the contour γ or boundary of D in order to compute $f(a)$.

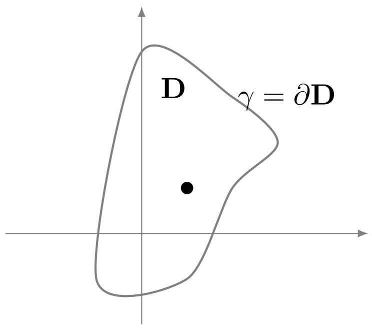

Figure 9.1.: A point inside a region.

Example 9.4. *Let γ be a circle around $3i$ with radius 1. Suppose we want to compute the integral $\int\limits_{\gamma} \frac{1}{z^2+9} dz$. We let $\int\limits_{\gamma} \frac{1}{z^2+9} dz = \int\limits_{\gamma} \frac{1}{(z+3i)(z-3i)} dz = \int\limits_{\gamma} \frac{f(z)}{z-3i} dz$. According to Cauchy's integral formula we have $\int\limits_{\gamma} \frac{1}{z^2+9} dz = 2\pi i f(3i) = \frac{\pi}{3}$.*

Based on Cauchy's integral formula one can define the so-called winding number $\eta(\gamma, z)$ about z with $f(z) = 1$:

Corollary 9.5 (Winding Number). *Let γ be a closed path of integration, and let z be a point that does not lie on γ, the winding number $\eta(\gamma, z)$ about z is*

$$\eta(\gamma, z) = \frac{1}{2\pi i} \int\limits_{\gamma} \frac{d\xi}{\xi - z}$$

Example 9.6. *Let $\gamma : [0, 2\pi] \to c + re^{it}$ be a function that is continuously differentiable with $\gamma(0) = \gamma(2\pi)$. Let $t \in [0, 2\pi]$ and define $g(t) = \frac{1}{2\pi i} \int\limits_{a}^{t} \frac{\gamma'(t)}{\gamma(t)-z}$. Then*

$g(0) = 0$ *and* $g(2\pi) = \eta(\gamma, z)$.

Closely related to the Cauchy integral we introduce the Residue Theorem.

Theorem 9.7 (**Residue Theorem**). *Let D be a simply connected domain and D_f a discrete subset of D. Furthermore, let $f : D\backslash D_f \to \mathbb{C}$ be holomorphic in $D\backslash D_f$ and $\gamma : I \to D\backslash D_f$ be a closed path in $D\backslash D_f$, then*

$$\frac{1}{2\pi i} \int\limits_{\gamma} f(z)dz = \sum_{a\in D_f} \eta(\gamma, a) \cdot \mathsf{Res}_a f$$

Here, the residue $\mathsf{Res}_a f$ of a function f at a point $a \in D_f$ is the coefficient c_{-1} of the corresponding Laurent series on a neighborhood $U = U_r(a)\backslash\{a\}$ around a, on which f is holomorphic. In this case,

$$\mathsf{Res}_a f = \frac{1}{2\pi i} \int\limits_{\partial U} f(z)dz\,.$$

Since we are interested in higher dimensional problems, we have to consider generalizations of those theorems. The next theorem, which can also be found in the literature, represents a straight forward generalization of Cauchy's formula to multidimensional complex spaces. It is very simple and can be used once having understood the basic formula in Theorem 9.3. This statement dates back to Hörmander in 1966, who proved Cauchy's integral formula solely for polydiscs. This can be very useful in order to deduce an expression for the solution of LWE instances.

Theorem 9.8. *Let $B_1, \ldots, B_n \subset \mathbb{C}$ be n open discs and the polydisc $B = \prod_{i=1}^{n} B_i \subset \mathbb{C}^n$ its cartesian product. Furthermore, let $f(\mathbf{z})$ be holomorphic in a region $D \supset B$. Then the Cauchy integral states*

$$\frac{1}{(2\pi i)^n} \int\limits_{\partial B_1} \cdots \int\limits_{\partial B_n} \frac{f(\mathbf{z})}{(z_1 - a_1)\ldots(z_n - a_n)} dz_1 \ldots dz_n = \begin{cases} f(\mathbf{a}), & \text{if } \mathbf{a} \in B \\ 0, & \text{if } \mathbf{a} \notin \bar{B} \end{cases}$$

A disc B_i is a set parameterized by a real $r_i > 0$, which contains all complex numbers satisfying $B_i = \{z \in \mathbb{C} \mid Re(z)^2 + Im(z)^2 = r_i^2\}$. However, if one considers more complicated domains resp. surfaces, the theorem above is insufficient and hence not applicable in order to deduce a representation formula to the solution of an arbitrary CVP instance. In 1938 resp. 1943 Bochner and Martinelli proved a generalization of Cauchy's integral formula with respect to arbitrary surfaces. However, this requires to introduce the Martinelli kernel, which is defined as follows.

Definition 9.9. *The Bochner-Martinelli kernel $U(\mathbf{z},\mathbf{a})$ in $\mathbb{C}^n$ is a differential form of bidegree $(n, n-1)$ given by*

$$U(\mathbf{z},\mathbf{a}) = (-1)^{n(n-1)/2}\frac{(n-1)!}{(2\pi i)^n}\sum_{j=1}^{n}\frac{(-1)^{j-1}(\bar{z}_j-\bar{a}_j)}{|\mathbf{z}-\mathbf{a}|^{2n}}d\bar{\mathbf{z}}[j]\wedge d\mathbf{z},$$

where $d\bar{\mathbf{z}}[j] = d\bar{z}_1 \wedge \ldots \wedge d\bar{z}_{j-1} \wedge d\bar{z}_{j+1} \wedge \ldots \wedge d\bar{z}_n$ and $d\mathbf{z} = dz_1 \wedge \ldots \wedge dz_n$ are $(0, n-1)$ resp. $(n, 0)$ differential forms.

The factor $(-1)^{n(n-1)/2}$ represents the orientation in $\mathbb{C}^n$. Differential forms are considered to be an important approach towards defining integrands over higher dimensional manifolds such as curves. In fact, it gained much interest due to its independence from concrete choices of coordinates.

Example 9.10. *For $n = 1$, we have $U(z,a) = \frac{1}{2\pi i}\frac{1}{z-a}dz$ corresponding to the standard Cauchy kernel from Theorem 9.3.*

Theorem 9.11. *Let $D \subset \mathbb{C}^n$ be a bounded domain with piecewise smooth boundary. Furthermore, let f be holomorphic in D and $a \in \mathbb{C}^n$, then*

$$\int_{\partial D} f(\mathbf{z})\ U(\mathbf{z},\mathbf{a}) = \begin{cases} f(\mathbf{a}), & \text{if } \mathbf{a} \in D \\ 0, & \text{if } \mathbf{a} \notin \bar{D} \end{cases},$$

where $U(\mathbf{z},\mathbf{a})$ denotes the Bochner-Martinelli kernel.

As an immediate consequence, we obtain the following corollary from Theorem 9.8 in conjunction with Theorem 9.11.

Corollary 9.12. *Let B and f be defined as in Theorem 9.8, then we have*

$$\int_{\partial B} f(\mathbf{z})\ U(\mathbf{z},\mathbf{a}) = \frac{1}{(2\pi i)^n}\int_{\partial B_1}\cdots\int_{\partial B_n}\frac{f(\mathbf{z})}{(z_1-a_1)\ldots(z_n-a_n)}dz_1\ldots dz_n\,.$$

9.2. Representation Formulas for Lattice Problems

In the following section we show how to represent the solution of a multidimensional CVP or LWE problem as a finite sum of integrals using the generalizations introduced in the previous section. This formalization is from a mathematical point of view very interesting as it allows a CVP or LWE problem from discrete mathematics to be solved via a formula from complex analysis. In fact, the Cauchy integral formula makes use of various properties, if the underlying function and the domain D are properly chosen. This opens new directions towards analyzing lattice problems and constructing cryptographic schemes. For instance, one could keep the domain secret, that can be represented by a low number of bits such as low degree polynomials.

Throughout this thesis we will use the following isomorphism in order to embed arbitrary vectors from $\mathbb{R}^m$ into $\mathbb{C}^k$ for some $k \in \mathbb{N}$ with $k \geq \lceil m/2 \rceil$. This mainly follows from the relationship $\mathbb{C} \cong \mathbb{R}^2$.

Definition 9.13. *Let $m \in \mathbb{N}$ and $k = \lceil m/2 \rceil$. Define by $\phi : \mathbb{R}^m \to \mathbb{C}^k$ the isomorphism that embeds vectors from $\mathbb{R}^m$ into $\mathbb{C}^k$ such that*

- $\phi(\mathbf{x}) = (x_1 + i \cdot x_{m/2+1}, \ldots, x_{m/2} + i \cdot x_m)$ *for even m.*
- $\phi(\mathbf{x}) = (x_1 + i \cdot x_{(m-1)/2+1}, \ldots, x_{(m-1)/2} + i \cdot x_{m-1}, x_m)$ *for odd m.*

and inverse $\phi^{-1}(\mathbf{z}) = [(\mathbf{z} + \bar{\mathbf{z}})/2, (\mathbf{z} - \bar{\mathbf{z}})/(2i)]$ for even m. The inverse for an odd integer is straightforward.

9.2.1. Number of lattice points inside a domain

We start by giving a formula that returns the exact number of lattice points inside an arbitrary bounded domain. To this end, one has only to evaluate a finite number of Cauchy integrals.

Theorem 9.14. *Let $m > 0$, $k = \lceil m/2 \rceil$ and $D \subset \mathbb{C}^k$ be a domain with smooth boundary. Moreover, let $\Lambda \in \mathbb{Z}^m$ be a lattice with basis $\mathbf{B} \in \mathbb{Z}^{m \times m}$ and $\phi : \mathbb{R}^m \to \mathbb{C}^k$ be an isomorphism as defined in Definition 9.13. The number of lattice points inside D is given by*

$$\sum_{\mathbf{a} \in \Lambda} \int_{\partial D} U(\mathbf{z}, \phi(\mathbf{a}))$$

.

Proof. The map ϕ is bijective as by construction with inverse map $\phi^{-1}(\mathbf{z}) = [(\mathbf{z} + \bar{\mathbf{z}})/2, (\mathbf{z} - \bar{\mathbf{z}})/(2i)]$. Define $f(\mathbf{x}) = 1$, then we have

$$\int_{\partial D} U(\mathbf{z}, \phi(\mathbf{a})) = \begin{cases} 1, & \text{if } \mathbf{a} \in D \\ 0, & \text{if } \mathbf{a} \notin \bar{D} \end{cases},$$

according to Theorem 9.11, since f is holomorphic. As a result, the sum provides the number of lattice points inside D. Direct evaluation of the integrals rather than checking the domains allows to compute the number of lattice points inside D. □

Example 9.15. *If we consider an m-dimensional sphere as required in many cryptographic applications, we get the number of lattice points inside the ball $B_r(0) \in \mathbb{R}^m$ of radius r with center $\mathbf{0} \in \mathbb{R}^m$ by*

$$\sum_{\mathbf{a} \in \Lambda} \int_{\partial \phi(B_r(0))} U(\mathbf{z}, \phi(\mathbf{a})) \overset{heuristic}{\approx} \frac{Vol(B_r(0))}{\det(\Lambda)},$$

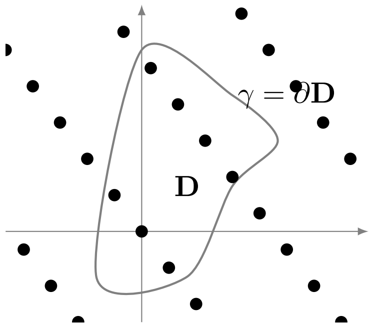

Figure 9.2.: Number of lattice points inside an arbitrary region D.

where the last term stems from the Gaussian heuristic with $Vol(B_r(0)) = \frac{\pi^n r^m}{\Gamma(m+1)}$*. In order to show that the ball has a smooth boundary, we define* $\rho(\mathbf{x}) = \sum_{i=1}^{m} x_i^2 - 1$*. The open ball is then identified by the set* $B_r(\mathbf{0}) = \{\mathbf{x} \in \mathbb{R}^m \mid \rho(\mathbf{x}) < 0\}$ *and its boundary* $\partial B_r(\mathbf{0}) = \{\mathbf{x} \in \mathbb{R}^m \mid \rho(\mathbf{x}) = 0\}$*. A quick view to the constituents of* $\rho(\mathbf{x})$ *also reveals that the gradient vector* $\nabla\rho \neq 0$ *such that the boundary* $\partial B_r(\mathbf{0})$ *is indeed smooth. Note that* $\partial\phi(B_r(0)) = \phi(\partial B_r(0))$*, since* ϕ *is continuous and bijective. The number of lattice points to be considered can be derived with the aid of the basis matrix* $\mathbf{B}$ *using, for instance, singular value decomposition. Following this, one can even give a range. Therefore, let* $C = \{\mathbf{v} \in \mathbb{Z}^m \mid s.t.\ \|\mathbf{B}\mathbf{v}\| \leq r\}$*. Then, the number of lattice points is given by the finite sum*

$$\sum_{\mathbf{a} \in C} \int_{\partial\phi(B_r(0))} U(\mathbf{z}, \phi(\mathbf{a})) = \int_{\partial\phi(B_r(0))} \sum_{\mathbf{a} \in C} U(\mathbf{z}, \phi(\mathbf{a})).$$

Clearly, the number of lattice points to be considered is finite in case one has to compute the number of lattice points inside a domain for q-ary lattices. In this case the amount of lattice points is bounded by q^m*.*

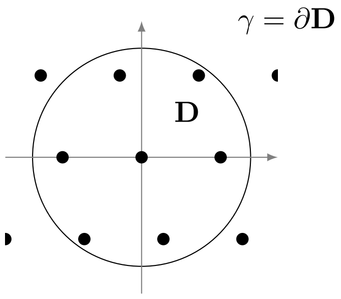

Figure 9.3.: Number of lattice points inside a circle.

In the following section, we give some interesting results that can help to understand the significant properties of Cauchy integrals more appropriately. Once the lattice-points are placed in the denominator, the corresponding series has its simple poles at the corresponding lattice points such that the Cauchy integral does not vanish if the poles are surrounded by a loop. An interesting research question is whether the series converges towards some formula or known representation, if we change the integral with the sum. We know that this is not true for the trivial case, where lattice-points are placed in the numerator. We give a positive answer to this question and explain this at the example of one-dimensional and two-dimensional lattices, which are shown in [BLP$^+$13] to be relevant for cryptography. The one-dimensional case is also closely related to the hidden number problem (HNP). For higher dimensional lattices it is required to analyze abelian functions which is beyond the scope of this thesis.

9.2.2. One-dimensional Lattices

First, we note that one-dimensional lattices considered in [BLP$^+$13] are generated by an element $a \in \mathbb{Z}_q$, where $\Lambda(a) = \{z \in \mathbb{Z} \mid z = a \cdot x \bmod q, x \in \mathbb{Z}_q\}$. Prior to considering such lattices interesting for cryptographic applications, we start with the basic case, namely the lattice $\Lambda = \mathbb{Z}$, which provides a very interesting relationship to trigonometric functions. In this case, we obtain the following result.

Lemma 9.16. *([Shu15])* $\frac{1}{z} + \sum\limits_{d=1}^{\infty} \frac{1}{z-d} + \frac{1}{z+d} = \pi \cot(\pi z)$

By use of this interesting relation between lattice points acting as simple poles and trigonometric functions, we can state the following theorem for one-dimensional lattices using the Cauchy integral formula.

Theorem 9.17. *Let $\Lambda = \mathbb{Z}$. The number of lattice points inside a domain D is given by*

$$\frac{1}{2\pi i} \int\limits_{\partial D} \pi \cot(\pi z) dz$$

Proof. For $\Lambda = \mathbb{Z}$, we construct the series $\frac{1}{z} + \sum\limits_{i=1}^{\infty} \frac{1}{z-i} + \frac{1}{z+i}$. This series has all its simple poles at the lattice points. But it is known from Lemma 9.16 that this series converges towards $\frac{1}{z} + \sum\limits_{d=1}^{\infty} \frac{1}{z-d} + \frac{1}{z+d} = \pi \cot(\pi z)$. The statement is now proven via

$$\begin{aligned} \frac{1}{2\pi i} \int\limits_{\partial D} \frac{1}{z} dz + \frac{1}{2\pi i} \sum_{d=1}^{\infty} \int\limits_{\partial D} \frac{1}{z-d} + \frac{1}{z+d} dz &= \frac{1}{2\pi i} \int\limits_{\partial D} \pi \cot(\pi z) dz \\ &= \sum_{a \in \mathbb{Z}} \eta(\partial D, a) \cdot \mathsf{Res}_a f, \end{aligned}$$

which evaluates for $f(z) = \pi \cot(\pi z)$ to the number of lattice points via the Cauchy integral formula or the Residue theorem. This shows that if the series is convergent it is in fact possible to deduce a closed expression that is independent from the number of lattice points. □

We return to the question of how to establish a closed expression for arbitrary one-dimensional lattices. Arbitrary lattices are of the form $\Lambda(a) = \{a \cdot s \mid s \in \mathbb{Z}\} = a\mathbb{Z}$. Therefore, we can generalize Lemma 9.16 and Theorem 9.17 as follows.

Theorem 9.18. *Let $\Lambda = a\mathbb{Z}$ for an arbitrary integer a. The number of lattice points inside a domain D is given by*

$$\frac{1}{2\pi i} \int\limits_{\partial D} \pi \cot\left(\pi \frac{z}{a}\right) dz$$

Proof. Let $h(z) = \frac{1}{z/a} + \sum\limits_{d=1}^{\infty} \frac{1}{z/a-d} + \frac{1}{z/a+d}$. Then, $f(z)$ has its poles at $a \cdot d$ for integers $d \in \mathbb{Z}$. The set of poles, hence, generates $a \cdot \mathbb{Z}$. Now, set $y = z/a$ and substitute this in the series $h(z)$, which leads to $\frac{1}{y} + \sum\limits_{d=1}^{\infty} \frac{1}{y-d} + \frac{1}{y+d} = \pi \cot(\pi y)$ as per Lemma 9.16. Substituting y back by z/a proves the claim. □

Example 9.19. *Suppose that we have an instance $b = a \cdot s + e$ for a large error term (possibly exponential) in the range $[-r, r]$ with $r > 0$, where both s and e are not known. Note, that we do not have an LWE instance, since b is not reduced modulo some exponential integer q, otherwise we could solve LWE easily, which is known to be hard. Assuming that there exists only one combination s, e for b within this range, we can use Theorem 9.22 in order to obtain both s and e. That is, we compute the Cauchy integral for $f(z) = z$ following Theorem 9.3*

$$\begin{aligned} b - e &= \frac{1}{2\pi i} \int\limits_{\partial B_r(b)} z\pi \cot\left(\pi \frac{z}{a}\right) dz \\ &= \frac{1}{2\pi i} \int\limits_{\partial B_r(b)} z \cdot \left(\frac{1}{z/a} + \sum_{d=1}^{\infty} \frac{1}{z/a-d} + \frac{1}{z/a+d}\right) dz \\ &= \sum_{d \in a\mathbb{Z}} d \cdot \eta(\partial B_r(b), d) \cdot \mathsf{Res}_d\, g, \end{aligned}$$

where $\partial B_r(b)$ is the boundary of a circle of radius r around b and $g(z) = \pi \cot\left(\pi \frac{z}{a}\right)$. One can furtherly parametrize the circle via $y = r \cdot e^{i\phi}$.

Example 9.20. *Now, suppose we have an LWE instance $b = a \cdot s + e \bmod q$ and everything else remains as in the previous example. A straight forward approach to solve the search version of the LWE problem is to find an integer $0 \le j \le d =$*

$\min\{a, 2^{\lceil \log c \rceil}\}$ *such that*

$$b - e' = \frac{1}{2\pi i} \int\limits_{\partial B_r(j \cdot q + b)} z\pi \cot\left(\pi \frac{z}{a}\right) dz \;=\; \frac{1}{2\pi i} \int\limits_{\partial B_r(0)} z\pi \cot\left(\pi \frac{z - (j \cdot q + b)}{a}\right) dz,$$

is non-zero, where r is bounded by the error size and c denotes the width of s. In case, the solution is unique or the number of non-zero elements is small (polynomial) we can use the following expression

$$\begin{aligned} b - e' &= \frac{1}{2\pi i} \int\limits_{\bigcup\limits_{\substack{j=0\\ j<d}} \partial B_r(j \cdot q + b)} z\pi \cot\left(\pi \frac{z}{a}\right) dz \\ &= \frac{1}{2\pi i} \sum_{j=0}^{d} \int\limits_{\partial B_r(j \cdot q + b)} z\pi \cot\left(\pi \frac{z}{a}\right) dz \\ &= \frac{1}{2\pi i} \int\limits_{\partial B_r(0)} \sum_{j=0}^{d} z\pi \cot\left(\pi \frac{z - (j \cdot q + b)}{a}\right) dz \end{aligned}$$

in order to find at least an approximate version of s and e. We observe, that the width of s has to be exponential, since otherwise there exists a polynomial-time algorithm that solves the problem.

In the following theorem, we show that LWE gets easy when $a \mid q$ even for an exponentially large modulus q.

Theorem 9.21. *Let a and q be integers such that $a \mid q$. Define by $\Lambda = a\mathbb{Z}$ the lattice generated by a. Let $b = a \cdot s + e \bmod q$ be an LWE instance with unique solution s, e. Then, the solution is given by*

$$b - e = \frac{1}{2\pi i} \int\limits_{\partial B_r(0)} z\pi \cot\left(\pi \frac{z - b}{a}\right) dz$$

Proof. Following Example 9.20, we have for any choice of $0 \leq j \leq q$ and large enough $r > 0$

$$\begin{aligned}
\frac{1}{2\pi i}\int\limits_{\partial B_r(j\cdot q+b)} z\pi\cot(\pi\frac{z}{a})dz &= \frac{1}{2\pi i}\int\limits_{\partial B_r(0)} z\pi\cot\left(\pi\frac{z-(j\cdot q+b)}{a}\right)dz\\
&= \frac{1}{2\pi i}\int\limits_{\partial B_r(0)} z\pi\cot\left(\pi\frac{z-b}{a}-\pi\frac{j\cdot q}{a}\right)dz\\
&= \frac{1}{2\pi i}\int\limits_{\partial B_r(0)} z\pi\cot\left(\pi\frac{z-b}{a}-\pi k\right)dz, k\in\mathbb{Z}\\
&= \frac{1}{2\pi i}\int\limits_{\partial B_r(0)} z\pi\cot\left(\pi\frac{z-b}{a}\right)dz\,.
\end{aligned}$$

The last equation follows from the π-periodicity of the cotangens function and is hence independent from j. □

We now analyze the case that $\frac{q}{\mathsf{gcd}(q,a)}$ is polynomial in the security parameter λ.

Lemma 9.22. *Let a and q be integers such that $\frac{q}{\mathsf{gcd}(q,a)} = k_1 = poly(\lambda)$. Furthermore, let $k_2 = \frac{a}{\mathsf{gcd}(q,a)}$ and define by $\Lambda = a\mathbb{Z}$ the lattice generated by a. Let $b = a\cdot s + e \bmod q$ be an LWE instance with unique solution s, e. Then, there exists an $0 \leq j \leq k_2$ such that*

$$b - e = \frac{1}{2\pi i}\int\limits_{\partial B_r(0)} z\pi\cot\left(\pi\frac{z-b}{a}-\pi\frac{j\cdot k_1}{k_2}\right)dz$$

solves the LWE-problem.

Proof. First we note, that since $\frac{q}{\mathsf{gcd}(q,a)} = k_1$ is polynomial, we also have that $\frac{a}{\mathsf{gcd}(q,a)} = k_2$ is polynomial due to $a \in \mathbb{Z}_q$. Therefore, we have for large enough $r > 0$

$$\begin{aligned}
\frac{1}{2\pi i}\int\limits_{\partial B_r(j\cdot q+b)} z\pi\cot\left(\pi\frac{z}{a}\right)dz &= \frac{1}{2\pi i}\int\limits_{\partial B_r(0)} z\pi\cot\left(\pi\frac{z-(j\cdot q+b)}{a}\right)dz\\
&= \frac{1}{2\pi i}\int\limits_{\partial B_r(0)} z\pi\cot\left(\pi\frac{z-b}{a}-\pi\frac{j\cdot q}{a}\right)dz\\
&= \frac{1}{2\pi i}\int\limits_{\partial B_r(0)} z\pi\cot\left(\pi\frac{z-b}{a}-\pi\frac{j\cdot k_1}{k_2}\right)dz\,.
\end{aligned}$$

Hence, the last equation is periodic in j with period k_2, that is $j \in \mathbb{Z}_{k_2}$. As a result, we need to evaluate at most $k_2 = poly(\lambda)$ integrals in order to find a solution. □

From the theorems above we deduce that prime moduli represent the preferred choice.

9.2.3. Two-dimensional Lattices

Two dimensional lattices are more complicated than the one-dimensional case. To this end, one considers doubly periodic functions or elliptic functions which have two periods or $f(z+\omega_1) = f(z+\omega_2)$ with two linearly independent vectors $\omega_1, \omega_2 \in \mathbb{C}$. The corresponding lattice is defined as the integer linear combination of the periods $\Lambda = \{i \cdot \omega_1 + j \cdot \omega_2 \mid (i,j) \in \mathbb{Z}^2\}$. Hence, the periods of $f(z)$ are all lattice-points in Λ. For the one-dimensional case, one easily verifies that $\pi \cot(\pi z)$ is periodic in $\mathbb{Z}$, that is $\pi \cot(\pi z) = \pi \cot(\pi(z+d))$ for all $d \in \mathbb{Z}$. Doubly periodic functions can be represented by Weierstrass functions. There exists a general theory about elliptic functions.

Definition 9.23 (**Weierstass Elliptic Function**). *Let $\omega_1, \omega_2 \in \mathbb{C}$ be linearly independent complex numbers. Then, the Weierstass Elliptic Function is defined by*

$$\wp(z) = \frac{1}{z^2} + \sum_{\omega \in \Lambda, \omega \neq 0} \left(\frac{1}{(z-\omega)^2} - \frac{1}{\omega^2} \right)$$

for $\Lambda = \{i \cdot \omega_1 + j \cdot \omega_2 \mid (i,j) \in \mathbb{Z}^2\}$.

First, one easily verifies by inspection that $\wp(z) = \wp(z+\omega)$ for all $\omega \in \Lambda$. Furthermore, the elliptic function has poles of order 2 at all its lattice points. Weierstrass proved that the last term $-\frac{1}{\omega^2}$ plays an important role. In particular, it forces the series to converge for arbitrary z, since otherwise the series diverges. We note that

$$\begin{aligned}\lim_{z\to 0} \left(\wp(z) - \frac{1}{z^2} \right) &= \lim_{z\to 0} \sum_{\omega \in \lambda, \omega \neq 0} \left(\frac{1}{(z-\omega)^2} - \frac{1}{\omega^2} \right) \\ &= \sum_{\omega \in \Lambda, \omega \neq 0} \lim_{z \to 0} \left(\frac{1}{(z-\omega)^2} - \frac{1}{\omega^2} \right) = 0.\end{aligned}$$

Furthermore, it has a limiting behaviour as z approximates the origin. That is, elliptic functions additionally have a Laurent representation such that about the origin, we can write

$$\wp(z) = \frac{1}{z^2} + a_2 z^2 + a_4 z^4 + \dots \ .$$

However, for our purposes it is not useful to integrate over elliptic functions, because the residue vanishes at the lattice points due to poles of order 2. Therefore, we have to consider the antiderivative of $\wp(z)$, which is by the general theorem of complex analysis a single-valued function, hence having simple poles at the lattice points. The antiderivative of $\wp(z) = -\zeta'(z)$ is defined by the Weierstrass zeta function.

Definition 9.24 (Weierstass Zeta Function). *Let $\omega_1, \omega_2 \in \mathbb{C}$ be linearly independent complex numbers. Then, the Weierstass Zeta Function is defined by*

$$\zeta(z) = \frac{1}{z} + \sum_{\omega \in \Lambda, \omega \neq 0} \left(\frac{1}{z-\omega} + \frac{1}{\omega} - \frac{z}{\omega^2} \right)$$

for $\Lambda = \{i \cdot \omega_1 + j \cdot \omega_2 \mid (i,j) \in \mathbb{Z}^2\}$.

The zeta function $\zeta(-z) = -\zeta(z)$ is an odd function that *converges* as well due to the convergence of $\wp(z)$. Furthermore, $\zeta(z)$ has simple poles at all lattice points in Λ. As a result, we can use the residue theorem.

Theorem 9.25. *Let $\omega_1, \omega_2 \in \mathbb{C}$ be linearly independent complex numbers and let $\zeta(z)$ be defined as in Definition 9.24. The number of lattice points inside a domain D is given by*

$$\frac{1}{2\pi i} \int_{\partial D} \zeta(z) dz$$

Proof. Since $\zeta(z)$ is a converging function with simple poles at all lattice points in Λ, we can apply the residue theorem as per Theorem 9.7 such that the integral evaluates to

$$\frac{1}{2\pi i} \int_{\partial D} \zeta(z) dz = \sum_{\omega \in \Lambda} \eta(\partial D, \omega) \cdot \mathsf{Res}_{\omega} \zeta$$

□

Integration of $\wp(z + 2\omega_i) = \wp(z)$ leads to $\zeta(z + 2\omega_i) = \zeta(z) + 2\eta_i$, from which we deduce $\eta_i = \zeta(\omega_i)$ for $z = -\omega_i$ and $i = 1, 2$, since $\zeta(z)$ is an odd function. Whittaker and Watson proved in 1990 that

$$2\eta_1\omega_2 - 2\eta_2\omega_1 = \pi \cdot i\,.$$

It is also common to identify the zeta function by its series expansion

$$\zeta(z) = \frac{1}{z} - \sum_{j=2}^{\infty} \frac{g_j z^{2j-1}}{2j-1},$$

where $g_2 = \frac{t_2}{20}, g_3 = \frac{t_3}{28}$ and $g_k = f(t_2, t_3, k)$ for $k > 3$. We call g_2 and g_3 the fingerprint of the lattice, since the series is determined by these two elements. However, the fingerprint is computed via the Eisenstein series.

In the following sections, we only show how to apply the Cauchy integral in general and arbitrary lattices without caring of convergence. Following this, we do not derive formulas as in the 1-dimensional and 2-dimensional case.

9.2.4. CVP Representation Formula for Arbitrary Lattices

Before starting to consider CVP representations, we recall the Voronoi cell $\mathcal{V}(\Lambda)$ of a lattice according to [MV10], which contains all vectors from $\mathbb{R}^m$ that are strictly closer to the origin than to any other lattice point. By $\bar{\mathcal{V}}(\Lambda)$ we denote its topological closure. More formally, the Voronoi cell is defined by the set

$$\mathcal{V}(\Lambda) = \{\mathbf{x} \in \mathbb{R}^m \mid \|\mathbf{x} - \mathbf{v}\|_2 > \|\mathbf{x}\|_2 \,, \ \forall \mathbf{v} \in \Lambda \backslash \{\mathbf{0}\}\} \,.$$

Any Voronoi cell of a lattice is a polytope that can be described by at most $2(2^m - 1)$ facets [MV10]. The facets induce the same number of half-spaces whose intersections yield the Voronoi cell. The half-spaces can be expressed in terms of lattice vectors $\mathbf{v}$, namely the so-called Voronoi relevant vectors, where $\mathbf{v}/2$ represents the center of a half-space. Let V be the set of Voronoi relevant vectors in accordance to [MV10]. For any $\mathbf{v} \in V$, we define the corresponding half-space by

$$H_{\mathbf{v}} = \{\mathbf{x} \in \mathbb{R}^m \mid \|\mathbf{x} - \mathbf{v}\|_2 > \|\mathbf{x}\|_2\} \,.$$

The Voronoi cell is then determined via $\mathcal{V}(\Lambda) = \bigcap\limits_{\mathbf{v} \in V} H_{\mathbf{v}}$.

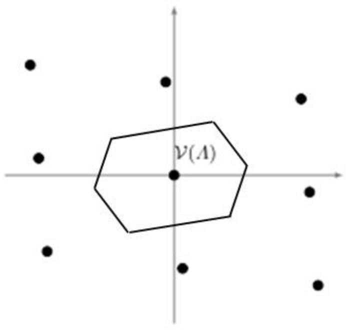

Figure 9.4.: Voronoi cell of a lattice.

Any polytope can be represented by finite degree polynomials (e.g., see [AH11, Hen06, GH03]). In Theorem 9.26 we express the solution of a CVP instance by use of the Bochner-Martinelli integral.

Theorem 9.26 (CVP solution). *Let $\Lambda \in \mathbb{Z}^m$ be a lattice with basis $\mathbf{B} \in \mathbb{Z}^{m \times m}$ and $\mathbf{w} + \mathcal{V}(\Lambda)$ its Voronoi cell shifted by $\mathbf{w} \in \mathbb{R}^m$. Furthermore, let ϕ be denote a bijective map according to Definition 9.13. Then, the solution to the CVP problem with target vector $\mathbf{t} \in \mathbb{R}^m$ is given by*

$$S_{CVP} = \sum_{\substack{\mathbf{a} \in \Lambda \\ \|\mathbf{a}\|_2 \leq \|\mathbf{t}\|_2 + \lambda_1}} \int_{\partial\phi(\mathbf{t} + \mathcal{V}(\Lambda))} \mathbf{z} \; U(\mathbf{z}, \phi(\mathbf{a}))$$

.

Proof. The map ϕ is bijective as by construction with inverse map $\phi^{-1}(\mathbf{z}) = [(\mathbf{z}+\bar{\mathbf{z}})/2, (\mathbf{z}-\bar{\mathbf{z}})/2]$. Define $f(\mathbf{z}) = \mathbf{z}$ to be the identity, which is holomorphic in $\mathbb{C}^n$ since each component $f_j(\mathbf{z}) = z_j$ is holomorphic according to Definition 9.1. The shifted Voronoi cell $\mathbf{t} + \mathcal{V}(\Lambda)$ contains exactly one lattice point, because otherwise it contains no lattice point or at least two lattice points $\mathbf{w}_1$ and $\mathbf{w}_2$ inside $\mathbf{t} + \mathcal{V}(\Lambda)$ such that $\mathbf{t} \in \mathbf{w}_i + \mathcal{V}(\Lambda)$ for $i = 1, 2$, which is a contradiction to the definition of a Voronoi cell. Conversely, $\mathbf{t} + \mathcal{V}(\Lambda)$ must contain a lattice point since otherwise $\mathbf{t}$ is not contained in any translation $\mathbf{w} + \mathcal{V}(\Lambda)$ for $\mathbf{w} \in \Lambda$. Using the Bochner-Martinelli integral, we obtain

$$\sum_{\substack{\mathbf{a}\in\Lambda \\ \|\mathbf{a}\|_2 \le \|\mathbf{t}\|_2 + \lambda_1}} \int\limits_{\partial\phi(\mathbf{t}+\mathcal{V}(\Lambda))} \mathbf{z}\, U(\mathbf{z}, \phi(\mathbf{a})) = \begin{cases} \mathbf{a}, & \text{if } \phi(\mathbf{a}) \in \phi(\mathbf{t} + \mathcal{V}(\Lambda)) \\ 0, & \text{if } \phi(\mathbf{a}) \notin \phi(\mathbf{t} + \bar{\mathcal{V}}(\Lambda)) \end{cases},$$

since $\mathbf{t} + \mathcal{V}(\Lambda)$ is a polytope that can be described by a polynomial such that the boundary $\partial\phi(\mathbf{t} + \mathcal{V}(\Lambda))$ is piecewise smooth. □

One can also solve the CVP problem differently by taking another approach. Following this, we recall the winding number $\eta(\gamma, z)$ from Corollary 9.5 but extended to higher dimensional spaces. We express by $c \in \mathbb{N}$ the number of windings $c \cdot \eta(\gamma, z)$. The following bijective encoder allows to relate a lattice point to the winding number c. From the winding number we can retrieve the corresponding lattice point back.

Definition 9.27 (Cantor's Pairing Function). *Cantor's pairing function is an invertible map defined by*

$$\beta : \mathbb{N} \times \mathbb{N} \to \mathbb{N}, (x_1, x_2) \to (x_1 + x_2)(x_1 + x_2 + 1)/2 + x_2 \,.$$

In order to compute the inverse, one performs the following steps

1. *Set* $t = x_1 + x_2$ *and* $d = \frac{t(t+1)}{2}$
2. *Then, we have* $\beta(x_1, x_2) = d + x_2$ *and* $t^2 + t - 2d = 0$
3. *As a result,* $t = \frac{\sqrt{8d+1}-1}{2}$.
4. $d = \lfloor \frac{\sqrt{8\beta(x_1,x_2)+1}-1}{2} \rfloor$
5. $x_2 = \beta(x_1, x_2) - d$ *and* $x_1 = t - x_2$

Cantor's pairing function can inductively be generalized to the Cantor tuple function

$$\beta_n(x_1, \ldots, x_n) = \beta(\beta_{n-1}(x_1, \ldots, x_{n-1}), x_n), \text{ for } n > 2 \,.$$

Cantor's pairing function is used in some applications as a space filling curve. All points satisfying $x_1 + x_2 = t$ lie on the diagonal line between $(\frac{t^2+t}{2}, 0)$ and $(0, \frac{t^2+3t}{2})$. Moreover, small elements t lead to small function values, a desirable feature in cryptographic applications. Since we can represent elements of $\mathbb{Z}$ by a tuple $(x, y) \in \mathbb{N}^2$, where the first component resp. second component is set to zero if the element is negative or positive, we can extend β_n as follows in order to allow for vectors with entries from $\mathbb{Z}$ as required for lattice-based applications.

Definition 9.28 (Extended Cantor's Pairing Function)**.** *Let $\mathbf{z} \in \mathbb{Z}^n$ and $\beta_n(\mathbf{x})$ be a Cantor tuple function. Define by $\psi := \beta_m(T^m(\mathbf{z})) : \mathbb{Z}^m \to \mathbb{N}$ the extended Cantor tuple function, where $T^m(\mathbf{z}) := \mathbb{Z}^m \to \mathbb{N}^{2m}, (T(z_1), \ldots, T(z_m))$ with*

$$T(z_i) = \begin{cases} (0, |z_i|), & \text{if } z_i < 0 \\ (z_i, 0), & \text{if } z_i > 0 \end{cases},$$

Using the extended version of Cantor's pairing function we can restate Theorem 9.26 as follows.

Theorem 9.29 (CVP Solution)**.** *Let $\Lambda \in \mathbb{Z}^m$ be a lattice with basis $\mathbf{B} \in \mathbb{Z}^{m \times m}$ and $\mathbf{w} + \mathcal{V}(\Lambda)$ be its Voronoi cell shifted by $\mathbf{w} \in \mathbb{R}^m$. Let ϕ be a bijective map following Definition 9.13 and ψ be defined as in Definition 9.28. Then, the solution to the CVP problem with target vector $\mathbf{t} \in \mathbb{R}^m$ is given by*

$$\begin{aligned} S_{CVP} &= \sum_{\substack{\mathbf{a} \in \Lambda \\ \|\mathbf{a}\|_2 \leq \|\mathbf{t}\|_2 + \lambda_1}} \int_{\psi(\mathbf{a}) \cdot \partial\phi(\mathbf{t} + \mathcal{V}(\Lambda))} U(\mathbf{z}, \phi(\mathbf{a})) \\ &= \int_{\partial\phi(\mathbf{t} + \mathcal{V}(\Lambda))} \sum_{\substack{\mathbf{a} \in \Lambda \\ \|\mathbf{a}\|_2 \leq \|\mathbf{t}\|_2 + \lambda_1}} \psi(\mathbf{a})\, U(\mathbf{z}, \phi(\mathbf{a}))\,. \end{aligned}$$

Proof. We let $f(\mathbf{z}) = 1$ be a constant function, which is trivially holomorphic. The number of windings $\psi(\mathbf{a})$ depends bijectively on the entries of the respective lattice points $\mathbf{a}$. Since each shifted Voronoi cell contains at most one lattice point, this sum of integrals evaluates to $\psi(\mathbf{a})$ the number of windings for a lattice point $\mathbf{a} \in \Lambda$ with $\phi(\mathbf{a}) \in \phi(\mathbf{t} + \mathcal{V}(\Lambda))$. This function can be inverted using the inversion algorithm from Definition 9.27 in combination with Definition 9.28. $\square$

The approximate version of the CVP problem is a little bit tricky, because more than one solution is expected. However, one desires to obtain all solutions encoded by one number. We resolve this problem by means of the extended version of Cantor's pairing function in the exponent. Such a strategy further allows to have some kind of an order among the solutions.

Theorem 9.30 (Approximate-CVP solution)**.** *Let $\Lambda \in \mathbb{Z}^m$ be a lattice with basis $\mathbf{B} \in \mathbb{Z}^{m\times m}$ and $B_r(\mathbf{w})$ be a ball of radius r centered at $\mathbf{w} \in \mathbb{R}^m$. Let ϕ be a bijective map according to Definition 9.13 and ψ be defined as in Definition 9.28. Then, the list of closest vectors with maximum distance to the target vector $\mathbf{t} \in \mathbb{R}^m$ is given by*

$$S_{CVP-List} = \sum_{\substack{\mathbf{a}\in\Lambda \\ \|\mathbf{a}\|_2 \le \|\mathbf{t}\|_2 + r}} 2^{\psi(\mathbf{a})} \int_{\partial\phi(B_r(\mathbf{t}))} U(\mathbf{z}, \phi(\mathbf{a}))\,.$$

Proof. We let $f(\mathbf{z}) = 1$ be a constant function, which is trivially holomorphic. In Example 9.15 we already showed that the ball has a smooth boundary. The number of windings $2^{\psi(\mathbf{a})}$ depends bijectively on the entries of the respective lattice points $\mathbf{a}$. In case a lattice point lies outside the closure of $B_r(\mathbf{t})$, the integral is zero. Hence, the sum of integrals evaluates to an integer, where the position of non-zero bits are encodings $\psi(\mathbf{a})$ of all closest vector points $\mathbf{a} \in \Lambda$ within distance of at most r to $\mathbf{t}$. As a result, we obtain all lattice points using the inversion algorithm from Definition 9.27 and Definition 9.28 applied on the positions of all non-zero bits. □

Grover's Search Algorithm Grover's algorithm is a quantum algorithm that takes a function $g_1 : \{0,1\}^m \to \{0,1\}$ as input and searches for a solution $\mathbf{x}$ such that $g_1(\mathbf{x}) = 1$ is satisfied. The algorithm requires $O(\sqrt{2^m})$ iterations to solve the search problem in an unstructured list. This corresponds to a quadratic speed up as compared to the classical case. We can build such a function as follows

$$g_1(\mathbf{a}, \mathbf{t}) = \int_{\partial\phi(\mathbf{t}+\mathcal{V}(\Lambda))} U(\mathbf{z}, \phi(\mathbf{a})) = \begin{cases} 1, & \text{if } \phi(\mathbf{a}) \in \phi(\mathbf{t} + \mathcal{V}(\Lambda)) \\ 0, & \text{if } \phi(\mathbf{a}) \notin \phi(\mathbf{t} + \bar{\mathcal{V}}(\Lambda)) \end{cases},$$

where g_1 is also parametrized by the target vector $\mathbf{t}$. In case one considers approximate versions of the CVP problem such as the γ-CVP problem, the number of solutions to the equation $g_2(\mathbf{x}) = 1$ for

$$g_2(\mathbf{a}, \mathbf{t}) = \int_{\partial\phi(B_\alpha(\mathbf{t}))} U(\mathbf{z}, \phi(\mathbf{a})) = \begin{cases} 1, & \text{if } \mathbf{a} \in \phi(B_\alpha(\mathbf{t})) \\ 0, & \text{if } \mathbf{a} \notin \phi(\bar{B}_\alpha(\mathbf{t})) \end{cases},$$

increases to $N > 1$ for a given length such as $\alpha = \gamma \cdot \lambda_1$, where $\min_{\mathbf{y}\in\Lambda} \|\mathbf{y} - \mathbf{t}\|_2$ is upper bounded by λ_1 denoting the shortest vector in the lattice. In this case, Grover's optimal bound for the number of iterations in order to find a solution is given by $\frac{\pi}{4}\sqrt{\frac{2^m}{N}}$. By use of polar coordinates, one can parametrize the surface area of an n-dimensional ball $B_\alpha(0)$. Based on the solver for γ-CVP, one can build a solver for γ-SVP. In fact, applying Grover's algorithm on n different sublattices allows to find the desired result. Following [GMSS99], one solves γ-CVP for target vectors $\mathbf{b}_i$ and sublattices $\Lambda(\mathbf{B}^{(i)})$ generated by the basis matrices $\mathbf{B}^{(i)} = [\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}, 2\mathbf{b}_i, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_n]$. Selecting the shortest vector among the solutions provides a solution to γ-SVP.

References

[ABB10a] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 553–572. Springer, May 2010.

[ABB10b] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 98–115. Springer, August 2010.

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO*, volume 5677 of *LNCS*, pages 595–618. Springer, 2009.

[AD97] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *29th Annual ACM Symposium on Theory of Computing*, pages 284–293. ACM Press, May 1997.

[AFG13] Martin R. Albrecht, Robert Fitzpatrick, and Florian Göpfert. On the efficacy of solving lwe by reduction to unique-svp. In *ICISC 2013*, 2013.

[AGV09] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 474–495. Springer, March 2009.

[AH11] Gennadiy Averkov and Martin Henk. Representing simple d-dimensional polytopes by d polynomials. *Mathematical Programming*, 126(2):203–230, 2011.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th Annual ACM Symposium on Theory of Computing*, pages 99–108. ACM Press, May 1996.

[Ajt98] Miklós Ajtai. The shortest vector problem in L2 is NP-hard for randomized reductions (extended abstract). In *30th Annual ACM Symposium on Theory of Computing*, pages 10–19. ACM Press, May 1998.

[Ajt99] Miklós Ajtai. Generating hard instances of the short basis problem. In *ICALP*, LNCS, pages 1–9. Springer, 1999.

[AKPW13] Joël Alwen, Stephan Krenn, Krzysztof Pietrzak, and Daniel Wichs. Learning with rounding, revisited. In Ran Canetti and JuanA. Garay, editors, *Advances in Cryptology CRYPTO 2013*, volume 8042 of *Lecture Notes in Computer Science*, pages 57–74. Springer, 2013.

[AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *33rd Annual ACM Symposium on Theory of Computing*, pages 601–610. ACM Press, July 2001.

[AKS02] Miklos Ajtai, Ravi Kumar, and D. Sivakumar. Sampling short lattice vectors and the closest lattice vector problem. CCC '02, pages 53–. IEEE Computer Society, 2002.

[AP09] Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. In *STACS*, volume 3 of *LIPIcs*, pages 75–86. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009.

[Ban93] W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(4):625–635, 1993.

[Ban95] W. Banaszczyk. Inequalities for convex bodies and polar reciprocal lattices in r^n. *Discrete Computational Geometry*, 13(1):217–231, 1995.

[BCHK07] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM Journal on Computing*, 36(5):1301–1328, 2007.

[BDK$^+$07] Johannes Buchmann, Erik Dahmen, Elena Klintsevich, Katsuyuki Okeya, and Camille Vuillaume. Merkle signatures with virtually unlimited signature capacity. In *ACNS 2007*, LNCS, pages 31–45. Springer, 2007.

[BF11] Dan Boneh and David Mandell Freeman. Homomorphic signatures for polynomial functions. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 149–168. Springer, May 2011.

[BG14] Shi Bai and StevenD. Galbraith. An improved compression technique for signatures based on learning with errors. In Josh Benaloh, editor, *CT-RSA 2014*, LNCS, pages 28–47. Springer, 2014.

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer, May 2003.

[BGR12] Kyle Brogle, Sharon Goldberg, and Leonid Reyzin. Sequential aggregate signatures with lazy verification from trapdoor permutations. ASIACRYPT'12, pages 644–662. Springer-Verlag, 2012.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, pages 309–325, 2012.

[Bl0] Johannes Blömer. Closest vectors, successive minima, and dual hkz-bases of lattices. In *Proceedings of the 27th International Colloquium on Automata, Languages and Programming*, ICALP '00, pages 248–259. Springer-Verlag, 2000.

[BLP$^+$13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *STOC*, pages 575–584, 2013.

[BNN07] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Unrestricted aggregate signatures. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *ICALP 2007: 34th International Colloquium on Automata, Languages and Programming*, volume 4596 of *Lecture Notes in Computer Science*, pages 411–422. Springer, July 2007.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 719–737. Springer, April 2012.

[BPW12] Alexandra Boldyreva, Adriana Palacio, and Bogdan Warinschi. Secure proxy signature schemes for delegation of signing rights. *Journal of Cryptology*, 25(1):57–115, 2012.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73. ACM Press, November 1993.

[BR96] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In Ueli M. Maurer, editor, *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416. Springer, May 1996.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume

7417 of *Lecture Notes in Computer Science*, pages 868–886. Springer, August 2012.

[BS99] Johannes Blömer and Jean-Pierre Seifert. On the complexity of computing short linearly independent vectors and short bases in a lattice. In *31st Annual ACM Symposium on Theory of Computing*, pages 711–720. ACM Press, May 1999.

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd Annual Symposium on Foundations of Computer Science*, pages 97–106. IEEE Computer Society Press, October 2011.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524. Springer, August 2011.

[BY96] Mihir Bellare and Moti Yung. Certifying permutations: Noninteractive zero-knowledge based on any trapdoor permutation. *Journal of Cryptology*, 9(3):149–166, 1996.

[BZ13] Dan Boneh and Mark Zhandry. Secure signatures and chosen ciphertext security in a quantum computing world. In Ran Canetti and JuanA. Garay, editors, *Advances in Cryptology CRYPTO 2013*, volume 8043 of *LNCS*, pages 361–379. Springer, 2013.

[CHJ$^+$02] Jean-Sébastien Coron, Helena Handschuh, Marc Joye, Pascal Paillier, David Pointcheval, and Christophe Tymen. GEM: A generic chosen-ciphertext secure encryption method. In Bart Preneel, editor, *Topics in Cryptology – CT-RSA 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 263–276. Springer, February 2002.

[CHK04] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 207–222. Springer, May 2004.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 523–552. Springer, May 2010.

[CKN03] Ran Canetti, Hugo Krawczyk, and Jesper Buus Nielsen. Relaxing chosen-ciphertext security. In Dan Boneh, editor, *Advances in Cryp-*

tology – CRYPTO 2003, volume 2729 of *Lecture Notes in Computer Science*, pages 565–582. Springer, August 2003.

[CN99] Jin-Yi Cai and Ajay Nerurkar. Approximating the svp to within a factor (1+1/dime) is np-hard under randomized reductions. *J. Comput. Syst. Sci.*, 59(2):221–239, 1999.

[Cor00] Jean-Sébastien Coron. On the exact security of full domain hash. In Mihir Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235. Springer, August 2000.

[CS98] J. H. Conway and N. J. A. Sloane. *Sphere packings, lattices and groups*, volume 3. Springer Verlag, 1998.

[DD12] Léo Ducas and Alain Durmus. Ring-Lwe in polynomial rings. PKC'12, pages 34–51. Springer, 2012.

[DDLL13] Lo Ducas, Alain Durmus, Tancrde Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In Ran Canetti and JuanA. Garay, editors, *CRYPTO 2013*, volume 8042 of *LNCS*, pages 40–56. Springer Berlin Heidelberg, 2013.

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.

[DKRS03] I. Dinur, G. Kindler, R. Raz, and S. Safra. Approximating cvp to within almost-polynomial factors is np-hard. *Combinatorica*, 23(2):205–243, 2003.

[DMQ13] Nico Döttling and Jörn Müller-Quade. Lossy codes and a new variant of the learning-with-errors problem. In Thomas Johansson and PhongQ. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 18–34. Springer Berlin Heidelberg, 2013.

[DN12] Léo Ducas and PhongQ. Nguyen. Learning a zonotope and more: Cryptanalysis of ntrusign countermeasures. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 433–450. Springer, 2012.

[DSGKM12] Dana Dachman-Soled, Rosario Gennaro, Hugo Krawczyk, and Tal Malkin. Computational extractors and pseudorandomness. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 383–403. Springer, March 2012.

[EB13] Rachid El Bansarkhani and Johannes Buchmann. Improvement and efficient implementation of a lattice-based signature scheme. In Lange Tanja, Kristin Lauter, and Petr Lisonek, editors, *Selected Areas in Cryptography*, LNCS. Springer, 2013.

[EB14a] Rachid El Bansarkhani and Johannes Buchmann. LCPR: High performance compression algorithm for lattice-based signatures, Submitted, Cryptology ePrint Archive, Report 2014/334, 2014. http://eprint.iacr.org/.

[EB14b] Rachid El Bansarkhani and Johannes Buchmann. Towards lattice based sequential aggregate signatures. In David Pointcheval and Damien Vergnaud, editors, *AFRICACRYPT 2014*, volume 8469 of *LNCS*, pages 336–355. Springer, 2014.

[EDB15] Rachid El Bansarkhani, Özgür Dagdelen, and Johannes Buchmann. Augmented learning with errors: The untapped potential of the error term. In *Financial Crypto 2015*, LNCS. Springer, 2015.

[EB15a] Rachid El Bansarkhani and Johannes Buchmann. High performance lattice-based CCA-secure encryption, Submitted, Cryptology ePrint Archive, Report 2015/042, 2015. http://eprint.iacr.org/.

[EB15b] Rachid El Bansarkhani and Johannes Buchmann. Efficient Lattice-based Encryption via A-LWE in the Standard Model, Submitted, 2015

[EB15c] Rachid El Bansarkhani and Johannes Buchmann. Representation Formulas for Lattice Problems via Cauchy Integrals, Submitted, 2015

[EFG$^+$10] Oliver Eikemeier, Marc Fischlin, Jens-Fabian Götzmann, Anja Lehmann, Dominique Schröder, Peter Schröder, and Daniel Wagner. History-free aggregate message authentication codes. In Juan A. Garay and Roberto De Prisco, editors, *SCN 10: 7th International Conference on Security in Communication Networks*, volume 6280 of *Lecture Notes in Computer Science*, pages 309–328. Springer, September 2010.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, August 1987.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178. ACM, 2009.

[GG91] Allen Gersho and Robert M. Gray. *Vector quantization and signal compression.* Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[GGH97a] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Eliminating decryption errors in the Ajtai-Dwork cryptosystem. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 105–111. Springer, August 1997.

[GGH97b] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 112–131. Springer, August 1997.

[GH03] Martin Grötschel and Martin Henk. The representation of polyhedra by polynomial inequalities. *Discrete Computational Geometry*, 29(4):485–504, 2003.

[GH11] Craig Gentry and Shai Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In Rafail Ostrovsky, editor, *52nd Annual Symposium on Foundations of Computer Science*, pages 107–109. IEEE Computer Society Press, October 2011.

[GJSS01] Craig Gentry, Jakob Jonsson, Jacques Stern, and Michael Szydlo. Cryptanalysis of the NTRU signature scheme (NSS) from Eurocrypt 2001. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 1–20. Springer, December 2001.

[GLP12] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In *CHES*, volume 7428 of *LNCS*. Springer, 2012.

[GMSS99] O. Goldreich, D. Micciancio, S. Safra, and J. P. Seifert. Approximating shortest lattice vectors is not harder than approximating closet lattice vectors. *Inf. Process. Lett.*, 71(2):55–61, 1999.

[GN08] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 31–51. Springer, April 2008.

[GOPS13] Tim Güneysu, Tobias Oder, Thomas Pöppelmann, and Peter Schwabe. Software speed records for lattice-based signatures. In Philippe Gaborit, editor, *Post-Quantum Cryptography*, volume 7932 of *LNCS*. Springer, 2013.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206. ACM Press, May 2008.

[Gra84] R.M. Gray. Vector quantization. IEEE ASSP Mag., pages 4–29, 1984.

[Gro04] Jens Groth. Rerandomizable and replayable adaptive chosen ciphertext attack secure cryptosystems. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 152–170. Springer, February 2004.

[Hel85] Bettina Helfrich. Algorithms to construct minkowski reduced and hermite reduced lattice bases. *Theor. Comput. Sci.*, 41(2-3):125–139, 1985.

[Hen06] M. Henk. *Polynomdarstellungen von Polyedern.* Preprint. Univ., Fak. für Mathematik, 2006.

[HHGP$^+$03] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, JosephH. Silverman, and William Whyte. Ntrusign: Digital signatures using the ntru lattice. In *Topics in Cryptology CT-RSA 2003*, volume 2612 of *LNCS*, pages 122–140. SPRINGER, 2003.

[HL93] Amir Herzberg and Michael Luby. Pubic randomness in cryptography. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 421–432. Springer, August 1993.

[HLM03] Jonathan Herzog, Moses Liskov, and Silvio Micali. Plaintext awareness via key registration. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 548–564. Springer, August 2003.

[HSW$^+$00] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System architecture directions for networked sensors. *SIGPLAN Not.*, 35(11):93–104, 2000.

[HSW13] Susan Hohenberger, Amit Sahai, and Brent Waters. Full domain hash from (leveled) multilinear maps and identity-based aggregate signatures. In Ran Canetti and JuanA. Garay, editors, *CRYPTO 2013*, volume 8042 of *LNCS*, pages 494–512. Springer, 2013.

[Kan87] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12(3):415–440, 1987.

[Kho04] Subhash Khot. Hardness of approximating the shortest vector problem in lattices. In *45th Annual Symposium on Foundations of Computer Science*, pages 126–135. IEEE Computer Society Press, October 2004.

[Kil06] Eike Kiltz. Chosen-ciphertext security from tag-based encryption. In Shai Halevi and Tal Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 581–600. Springer, March 2006.

[Lar12] Ron Larson. *Brief Calculus: An Applied Approach*, volume 9. 2012.

[Len83] Hendrik W. Lenstra. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8:538–548, 1983.

[LLL82] A.K. Lenstra, H.W.jun. Lenstra, and Lászlo Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:515–534, 1982.

[LM08] Vadim Lyubashevsky and Daniele Micciancio. Asymptotically efficient lattice-based digital signatures. In Ran Canetti, editor, *TCC 2008: 5th Theory of Cryptography Conference*, volume 4948 of *Lecture Notes in Computer Science*, pages 37–54. Springer, March 2008.

[LMRS04] Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Hovav Shacham. Sequential aggregate signatures from trapdoor permutations. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 74–90. Springer, May 2004.

[LOS$^+$06] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 465–485. Springer, May / June 2006.

[LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In Aggelos Kiayias, editor, *Topics in Cryptology – CT-RSA 2011*, volume 6558 of *Lecture Notes in Computer Science*, pages 319–339. Springer, February 2011.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, May 2010.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-lwe cryptography. In Thomas Johansson and PhongQ. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 35–54. Springer, 2013.

[LV08] Benoît Libert and Damien Vergnaud. Unidirectional chosen-ciphertext secure proxy re-encryption. In Ronald Cramer, editor, *PKC 2008: 11th International Conference on Theory and Practice of Public Key Cryptography*, volume 4939 of *Lecture Notes in Computer Science*, pages 360–379. Springer, March 2008.

[Lyn99] Charles Lynn. Secure border gateway protocol (s-bgp). In *ISOC Network and Distributed System Security Symposium – NDSS'99*. The Internet Society, February 1999.

[Lyu08] Vadim Lyubashevsky. Towards practical lattice-based cryptography, 2008.

[Lyu09] Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 598–616. Springer, December 2009.

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 738–755. Springer, April 2012.

[Mic98] Daniele Micciancio. The shortest vector in a lattice is hard to approximate to within some constant. pages 92–98. FOCS, 1998.

[Mic07] Daniele Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. *Computational Complexity*, 16(4):365–411, 2007.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718. Springer, April 2012.

[MP13] Daniele Micciancio and Chris Peikert. Hardness of sis and lwe with small parameters. In *CRYPTO (1)*, pages 21–39, 2013.

[MR04] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th Annual Symposium on Foundations of Computer Science*, pages 372–381. IEEE Computer Society Press, October 2004.

[MR08] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In DanielJ. Bernstein, Johannes Buchmann, and Erik Dahmen, editors, *Post-Quantum Cryptography*, pages 147–191. Springer, 2008.

[MR09] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In DanielJ. Bernstein, Johannes Buchmann, and Erik Dahmen, editors, *Post-Quantum Cryptography*. Springer, 2009.

[MT84] G. Marsaglia and W. Tsang. A fast, easily implemented method for sampling from decreasing or symmetric unimodal density functions.

SIAM Journal on Scientific and Statistical Computing, 5(2):349–359, 1984.

[MV10] Daniele Micciancio and Panagiotis Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. In Leonard J. Schulman, editor, *42nd Annual ACM Symposium on Theory of Computing*, pages 351–358. ACM Press, June 2010.

[Nev08] Gregory Neven. Efficient sequential aggregate signed data. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 52–69. Springer, April 2008.

[NR06] Phong Q. Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 271–288. Springer, May / June 2006.

[NR09] Phong Q. Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. *Journal of Cryptology*, 22(2):139–160, April 2009.

[Odl90] A. M. Odlyzko. The rise and fall of knapsack cryptosystems. In *In Cryptology and Computational Number Theory*, pages 75–88. A.M.S, 1990.

[Pei07] Chris Peikert. Limits on the hardness of lattice problems in lp norms. In *In IEEE Conference on Computational Complexity*, pages 333–346, 2007.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 333–342. ACM Press, May / June 2009.

[Pei10] Chris Peikert. An efficient and parallel gaussian sampler for lattices. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 80–97. Springer, August 2010.

[Pei14] Chris Peikert. Lattice cryptography for the internet. Cryptology ePrint Archive, Report 2014/070, 2014. http://eprint.iacr.org/.

[Pol71] John M. Pollard. The Fast Fourier Transform in a finite field. *Mathematics of Computation*, 25(114):365–374, 1971.

[PR07] Manoj Prabhakaran and Mike Rosulek. Rerandomizable RCCA encryption. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 517–534. Springer, August 2007.

[PSNT06] Duong Phan, Reihaneh Safavi-Naini, and Dongvu Tonien. Generic construction of hybrid public key traitor tracing with full-public-traceability. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP 2006: 33rd International Colloquium on Automata, Languages and Programming, Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 264–275. Springer, July 2006.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571. Springer, August 2008.

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 187–196. ACM Press, May 2008.

[Reg04] Oded Regev. New lattice-based cryptographic constructions. *J. ACM*, 51:899–942, 2004.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93. ACM Press, May 2005.

[RS10] Markus Rückert and Michael Schneider. Estimating the security of lattice-based cryptosystems. 2010. http://eprint.iacr.org/.

[Rüc10] Markus Rückert. Strongly unforgeable signatures and hierarchical identity-based signatures from lattices without random oracles. In Nicolas Sendrier, editor, *Post-Quantum Cryptography*, volume 6061 of *LNCS*, pages 182–200. Springer, 2010.

[Sch86] C.P. Schnorr. A more efficient algorithm for lattice basis reduction. In Laurent Kott, editor, *Automata, Languages and Programming*, volume 226 of *LNCS*, pages 359–369. Springer, 1986.

[Sch87] C. P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theor. Comput. Sci.*, 53(2-3):201–224, 1987.

[SE94] C. P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Math. Program.*, 66(2):181–199, 1994.

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[Shu15] Jerry Shurman. Lecture notes on complex analysis, 2015. http://people.reed.edu/ jerry/311/cotan.pdf.

[SMP08] JacobC.N. Schuldt, Kanta Matsuura, and KennethG. Paterson. Proxy signatures secure against proxy key exposure. In Ronald Cramer, editor, *PKC*, volume 4939 of *LNCS*, pages 141–161. Springer, 2008.

[SS11] Damien Stehlé and Ron Steinfeld. Making ntru as secure as worst-case problems over ideal lattices. In *Advances in Cryptology EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 27–47. Springer, 2011.

[SSTX09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 617–635. Springer, December 2009.

[vEB81] P. van Emde-Boas. *Another NP-complete partition problem and the complexity of computing short vectors in a lattice*. Report. Department of Mathematics. University of Amsterdam. Department, Univ., 1981.

[Win96] Franz Winkler. *Polynomial Algorithms in Computer Algebra (Texts and Monographs in Symbolic Computation)*. Springer, 1 edition, 1996.

[XF07] Rui Xue and Dengguo Feng. Toward practical anonymous rerandomizable RCCA secure encryptions. In Sihan Qing, Hideki Imai, and Guilin Wang, editors, *ICICS 07: 9th International Conference on Information and Communication Security*, volume 4861 of *Lecture Notes in Computer Science*, pages 239–253. Springer, December 2007.

[Zha10] Fuzhen Zhang. *The Schur Complement and Its Applications*, volume 4. Springer, 2010.