Universidade Estadual de Campinas
Instituto de Computação

Jheyne Nayara Ortiz

Theoretical and practical contributions to Ring-LWE cryptography: twisted embeddings and the discrete Galois transform

Contribuições teóricas e práticas para criptografia baseada no problema Ring-LWE: mergulhos torcidos e transformada discreta de Galois

CAMPINAS
2022

Jheyne Nayara Ortiz

Theoretical and practical contributions to Ring-LWE cryptography: twisted embeddings and the discrete Galois transform

Contribuições teóricas e práticas para criptografia baseada no problema Ring-LWE: mergulhos torcidos e transformada discreta de Galois

Tese apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Doutora em Ciência da Computação.

Thesis presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Supervisor/Orientador: Prof. Dr. Ricardo Dahab
Co-supervisor/Coorientador: Prof. Dr. Diego de Freitas Aranha

Este exemplar corresponde à versão final da Tese defendida por Jheyne Nayara Ortiz e orientada pelo Prof. Dr. Ricardo Dahab.

CAMPINAS
2022

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

Or8t

Ortiz, Jheyne Nayara, 1992-
Theoretical and practical contributions to Ring-LWE cryptography : twisted embeddings and the discrete Galois transform / Jheyne Nayara Ortiz. – Campinas, SP : [s.n.], 2022.

Orientador: Ricardo Dahab.
Coorientador: Diego de Freitas Aranha.
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Criptografia pós-quântica. 2. Reticulados algébricos. 3. Criptografia homomórfica. I. Dahab, Ricardo, 1957-. II. Aranha, Diego de Freitas, 1982-. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: Contribuições teóricas e práticas para criptografia baseada no problema Ring-LWE : mergulhos torcidos e transformada discreta de Galois
Palavras-chave em inglês:
Post-quantum cryptography
Algebraic lattices
Homomorphic encryption
Área de concentração: Ciência da Computação
Titulação: Doutora em Ciência da Computação
Banca examinadora:
Ricardo Dahab [Orientador]
Daniel Nelson Panario Rodríguez
Rafael Misoczki
Sueli Irene Rodrigues Costa
Julio César López Hernández
Data de defesa: 30-03-2022
Programa de Pós-Graduação: Ciência da Computação

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0001-7152-2103
- Currículo Lattes do autor: http://lattes.cnpq.br/2041434857835392

Universidade Estadual de Campinas
Instituto de Computação

Jheyne Nayara Ortiz

Theoretical and practical contributions to Ring-LWE cryptography: twisted embeddings and the discrete Galois transform

Contribuições teóricas e práticas para criptografia baseada no problema Ring-LWE: mergulhos torcidos e transformada discreta de Galois

Banca Examinadora:

- Prof. Dr. Ricardo Dahab
  IC/Unicamp
- Prof. Dr. Daniel Nelson Panario Rodríguez
  Carleton University
- Dr. Rafael Misoczki
  Google
- Profa. Dra. Sueli Irene Rodrigues Costa
  IMECC/Unicamp
- Prof. Dr. Julio Cesar López Hernández
  IC/Unicamp

A ata da defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria do Programa da Unidade.

Campinas, 30 de março de 2022

Acknowledgements

Ao concluir este trabalho, agradeço a Deus, em primeiro lugar, pelas oportunidades e pessoas com quem compartilhei o período da pós-graduação.

Aos meus pais, Aidê e Jorge, que me transmitiram os valores que sustentam a minha resiliência e busca pela independência pessoal através dos estudos, da perseverança e da fé. Ao meu marido, Douglas, que me proveu apoio ininterrupto, compartilhando com carinho e respeito do processo de constante aprimoramento. À minha irmã, Nayane, e ao meu cunhado, Rafael, por terem feito parte deste processo, me ouvindo e gerando momentos de descontração.

Ao meu amigo e coautor, Pedro, que dividiu, desde o primeiro semestre, as dificuldades de cada estágio da nossa formação como pesquisadores no Brasil. Obrigada por ter sido meu olhar pragmático e pela atenção à minha saúde mental. Ao meu amigo e coautor, Robson, pela paciência em transmitir os seus conhecimentos de álgebra e reticulados. A sua colaboração foi essencial para a conclusão desta tese.

Aos meus orientadores, Prof. Ricardo Dahab e Prof. Diego F. Aranha, pelas oportunidades, apoio e sabedoria. Aos colegas dos laboratórios LASCA e LMCAD no IC/Unicamp pela companhia no dia a dia e nas conferências, e pelas conversas e diversos cafés. Aos demais professores e funcionários do IC, pelo convívio e todo o apoio operacional.

A todos que direta ou indiretamente contribuíram para a realização deste trabalho.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001 and by The Brazilian National Council for Scientific and Technological Development (CNPq), grant 164489/2018-5.

Resumo

Diversos trabalhos têm caracterizado instâncias fracas do problema LWE sobre anéis (Ring-LWE) explorando vulnerabilidades provenientes de estruturas algébricas. Apesar dessas instâncias fracas não serem englobadas pelos teoremas de pior caso, permitir a instanciação de outros anéis aumenta o escopo de aplicações possíveis e favorece a diversificação de hipóteses de segurança. A primeira parte desta tese é dedicada a estender o problema Ring-LWE na Criptografia Baseada em Reticulados para incluir reticulados algébricos construídos através de mergulhos torcidos. Primeiramente, uma nova classe de problemas, intitulada *Twisted Ring-LWE*, é definida substituindo o mergulho canônico por uma versão estendida. A segurança do problema Twisted Ring-LWE é demonstrada através de uma redução computacional do Ring-LWE para o Twisted Ring-LWE nas variantes de busca e decisão. Além disso, demonstra-se que o fator de torção não interfere nos fatores de aproximação nas reduções de pior caso para caso médio. Assim, o Twisted Ring-LWE mantém a segurança consolidada do Ring-LWE e aumenta o escopo de reticulados algébricos que podem ser considerados para aplicações de criptografia. Depois disso, essa tese foca na construção algébrica do reticulado $\mathbb{Z}^n$ através de mergulhos torcidos. Assim, apresenta-se como amostrar eficientemente de distribuições Gaussianas esféricas e como conectar instâncias do Twisted Ring-LWE construídas sobre corpos de números distintos. Por fim, um criptossistema de chave pública é modificado para incorporar mergulhos torcidos, culminando em uma discussão de como suas principais sub-rotinas podem ser executadas.

A segunda parte desta tese compreende uma análise do uso da Transformada Discreta de Galois (DGT) para multiplicação polinomial em anéis ciclotômicos com condutor potência de dois. A DGT é comparada com técnicas usadas por candidatos na segunda rodada do projeto de padronização do NIST intitulado *Post-Quantum Cryptography* na arquitetura x64. Conclui-se que a DGT não apresenta melhora de desempenho se comparada às formulações da Transformada Numérica (NTT). Além disso, o algoritmo *four-step* de Bailey, originalmente formulado sobre a Transformada Rápida de Fourier, é modificado para acelerar a DGT em esquemas de encriptação homomórfica em GPUs. A nova formulação recursiva da DGT resulta em uma multiplicação homomórfica até 3,6 vezes mais rápida do que o estado da arte. Um experimento posterior indicou que a DGT recursiva induz uma melhora de até 15% na multiplicação homomórfica se comparada com uma abordagem equivalente da NTT.

Abstract

Several works have characterized weak instances of the Ring-LWE Problem by exploring vulnerabilities arising from algebraic structures. Although worst-case hardness theorems do not address these weak instances, enabling other ring instantiations enlarges the scope of possible applications and favors the diversification of security assumptions. The first part of this thesis is devoted to extending the Ring-LWE Problem in lattice-based cryptography to include algebraic lattices realized through twisted embeddings. First, we define the class of problems *Twisted Ring-LWE*, replacing the canonical embedding with an extended form. We prove that Twisted Ring-LWE is secure by providing a reduction from Ring-LWE to Twisted Ring-LWE in search and decision forms. It is also shown that the torsion factor does not affect the asymptotic approximation factors in the worst-case to average-case reductions. Thus, Twisted Ring-LWE maintains the consolidated hardness guarantee of Ring-LWE and increases the existing scope of algebraic lattices that can be considered for cryptographic applications. Next, we focus on the algebraic construction of rotated $\mathbb{Z}^n$-lattices via twisted embeddings. We show how to sample efficiently from spherical Gaussian distributions and connect Twisted Ring-LWE instances built on distinct number fields. Finally, we modify a public-key cryptosystem to integrate twisted embeddings and discuss how to perform its primary algorithmic operations.

The second part of this thesis comprehends the Discrete Galois Transform (DGT) evaluation for polynomial multiplication in power-of-two cyclotomic rings. We compare the DGT with techniques used by round-two candidates of NIST's Post-Quantum Cryptography standardization project in the x64 architecture. We conclude that the DGT does not improve efficiency compared to Number-Theoretic Transform (NTT) formulations. Then, we modify Bailey's four-step algorithm based on the Fast Fourier Transform to improve the DGT for homomorphic encryption schemes on GPUs. The new recursive formulation of DGT allows up to 3.6 times faster homomorphic multiplication than the state-of-the-art. A further experiment indicated that the recursive DGT induces an improvement of up to 15% in homomorphic multiplication compared to a similar approach of NTT.

List of Abbreviations and Acronyms

R-LWE$_{q,\Psi^{(\tau)}}$ Search version of the Twisted Ring-LWE Problem. 40

R-LWE$_{q,\Psi}$ Search version of the Ring-LWE Problem. 33

R-LWE$_{q,\Upsilon^{(\tau)}}$ Average-case decision version of the Twisted Ring-LWE Problem. 40

R-LWE$_{q,\Upsilon}$ Average-case decision version of the Ring-LWE Problem. 33

AVX2 Advanced Vector Extensions 2. 21, 67, 86, 91, 109, 124

CRT Chinese Remainder Theorem. 54, 55, 77, 109

CUDA Compute Unified Device Architecture. 19–21, 67, 93, 98, 99, 101, 102, 104, 106, 109

CVP Closest Vector Problem. 14, 15, 62

DFT Discrete Fourier Transform. 70, 71, 73, 82, 98

DGS Discrete Gaussian Sampling Problem. 33–35, 43

DGT Discrete Galois Transform. 6, 7, 9, 17, 20–22, 67–69, 74–76, 80–84, 86–91, 93, 96–98, 100, 106, 109, 110, 123, 124

DIF Decimation-In-Frequency. 70, 71, 85

DIT Decimation-In-Time. 70, 71, 82

FFT Fast Fourier Transform. 9, 21, 67, 69, 70, 72, 73, 80, 93, 94, 96, 98, 109

FHE Fully Homomorphic Encryption. 19

GapSVP Promise Shortest Vector Problem. 15, 34

GIVP Generalized Independent Vectors Problem. 33, 34

GPU Graphics Processing Unit. 6, 7, 19–21, 67, 91, 93, 96–106, 109

HE Homomorphic Encryption. 19, 99, 108–110

IDGT Inverse Discrete Galois Transform. 9, 76, 82, 85, 90, 91, 97

IFFT Inverse Fast Fourier Transform. 94

INTT Inverse Number-Theoretic Transform. 9, 74, 91, 95

LR Logistic Regression. 93, 104–106

LWE Learning With Errors. 6, 17, 32–36, 41, 42, 110

NIST U.S. National Institute of Standards and Technology. 6, 7, 14–18, 20, 86, 88, 108, 109

NTT Number-Theoretic Transform. 6, 7, 9, 17, 21, 67, 68, 73, 74, 76, 80, 86, 88–91, 93–96, 98, 106, 109, 110, 123, 124

PHE Partially Homomorphic Encryption. 19

PKE Public-Key Encryption scheme. 51

RDGT Recursive DGT. 21, 22, 67, 93, 96–98, 101–106, 109, 110

RFFT Recursive FFT. 67, 93, 94, 98

RIDGT Recursive IDGT. 97

RINTT Recursive INTT. 95

RNS Residue Number System. 69, 101–103

RNTT Recursive NTT. 21, 67, 93–98, 101–106, 109, 110

RSA Rivest-Shamir-Adleman public-key cryptosystem. 19

SIMD Single Instruction Multiple Data. 19, 97, 109

SIS Short Integer Solution Problem. 32

SIVP Shortest Independent Vectors Problem. 14, 15, 33, 34, 36, 43

SM Streaming Multiprocessors. 101, 102

SPOG Secure Processing on GPGPUs. 99–101

SVP Shortest Vector Problem. 14, 15, 33

List of Symbols

(s_1, s_2) Number field signature. 30

D_r One-dimensional continuous Gaussian distribution of width r. 28

$D_{\Lambda,r,\mathbf{c}}(\mathbf{a})$ n-dimensional discrete Gaussian distribution over Λ. 28

$D_{\mathbf{r}}$ Elliptical n-dimensional Gaussian distribution with parameter $\mathbf{r}$. 28

$K_{\mathbb{R}}$ Tensor product field $K_{\mathbb{R}} = K \otimes_{\mathbb{Q}} \mathbb{R}$. 32

K An algebraic number field. 29

$[K : \mathbb{Q}]$ Number field degree. 29

$\langle a, b \rangle_\tau$ Inner product in $K_{\mathbb{R}}$ induced by the twisted embedding σ_τ. 37

$\mathrm{Gal}(K/\mathbb{Q})$ Galois group of a number field K over $\mathbb{Q}$. 30

$\mathcal{I}^\vee$ Dual ideal of a fractional ideal $\mathcal{I}$ of K. 31

$\mathcal{I}_q$ The quotient $\mathcal{I}_q = \mathcal{I}/q\mathcal{I}$ for an integer q. 33

$\mathcal{I}$ Ideal of the ring of integers $\mathcal{O}_K$. 31

N_K Norm of a Galois number field K. 31

$\mathcal{O}_K$ Ring of integers of a number field K. 31

$\Phi_m(x)$ m-th cyclotomic polynomial. 29

$\Psi_p(x)$ Minimal polynomial of $\zeta_p + \zeta_p^{-1}$. 52

Ψ A family of distributions over $K_{\mathbb{R}}$. 33

$\mathbb{Q}(\zeta_m)$ m-th cyclotomic number field. 29

Tr_K Trace of a Galois number field K. 31

Υ A distribution over a family of error distributions, each over $K_{\mathbb{R}}$. 33

$\mathbf{G} = \mathbf{M}^\top \mathbf{M}$ Gram-matrix of Λ with respect to the generator matrix $\mathbf{M}$. 27

$\eta_\epsilon(\Lambda)$ Smoothing parameter of a lattice Λ. 28, 34

gcd Greatest common divisor. 30

$\lambda_1^{(p)}(\Lambda)$ Minimum distance of a lattice Λ in the ℓ_p-norm. 26

$\lambda_k^{(p)}(\Lambda)$ Successive minimum of a lattice Λ in the ℓ_p-norm. 26

$\mathbb{Q}(\zeta_m + \zeta_m^{-1})$ Maximal real subfield of the m-th cyclotomic number field. 29

$\mathbb{T}$ The quotient $\mathbb{T} = K_{\mathbb{R}}/R^\vee$. 33

$\mathcal{A}_{s,\psi_\tau}$ Twisted Ring-LWE distribution over $R_q \times \mathbb{T}$. 40

$\mathcal{A}_{s,\psi}$ Ring-LWE distribution over $R_q \times \mathbb{T}$. 33

ψ An error distribution over $K_{\mathbb{R}}$. 33

$\rho_{r,\mathbf{c}}$ Gaussian function over H centered at $\mathbf{0}$. 28

σ_τ A twisted embedding parameterized by a totally positive element $\tau \in K$. 37

σ_i Number field monomorphism from K to $\mathbb{C}$. 30

σ Canonical embedding from a number field into the space H. 32

τ A totally positive element in a number field K called the torsion factor. 37

$\varphi(\cdot)$ Euler's totient function. 29

ζ_m Primitive m-th root of unity. 29

Contents

Chapter 1

Introduction

This thesis is organized into two parts, aiming to advance the area of lattice-based cryptosystems and, in particular, Ring-LWE cryptography. Cryptanalysis efforts of ideal-lattice cryptography inspired the first part of this thesis. The second part is influenced by NIST's Post-Quantum Cryptography standardization process [101] and also by the consortium for Homomorphic Encryption Standardization [5]. In the next section, we motivate and give a short introduction to the problems that we have tackled in this work. Our contributions are summarized in Section 1.2, and the organization of this thesis is given in Section 1.3.

1.1 Motivation and Context

The terminology post-quantum cryptography emerged in mid-2006 to describe the response to advances in quantum computing. Post-quantum cryptosystems are conjectured to be resistant to quantum attacks, especially the variants of Shor's algorithm [128], that break factorization- and discrete logarithm-based primitives. Quantum-resistant cryptosystems are categorized according to the class of their underlying hard problem. Currently, the most relevant classes are based on hard problems over lattices, error-correcting codes, hash functions, multivariate quadratic equations, and elliptic-curve isogenies.

Lattices can be informally described as repeating arrangements of points in an n-dimensional space in a grid pattern. The $\mathbb{Z}^n$-lattice in dimension two is depicted in Figure 1.1. Some hard computational problems are defined over this structure, and their conjectured hardness against classical and quantum computers forms the theoretical foundation of a handful of cryptographic constructions. These constructions gave rise to the field of lattice-based cryptography, whose security is based on the Shortest Independent Vectors Problem (SIVP), the Shortest Vector Problem (SVP), and the Closest Vector Problem (CVP) [3, 114].

Ajtai [3] was the first to explore the computational hardness of lattices for cryptography. He presented a public-key cryptosystem whose security is based on a worst-case to average-case reduction. In other words, it means that breaking the cryptosystem, on average, is as hard to break as solving worst-case instances of certain NP-hard lattice problems. In cryptography, average-case complexity means that the reduction works for

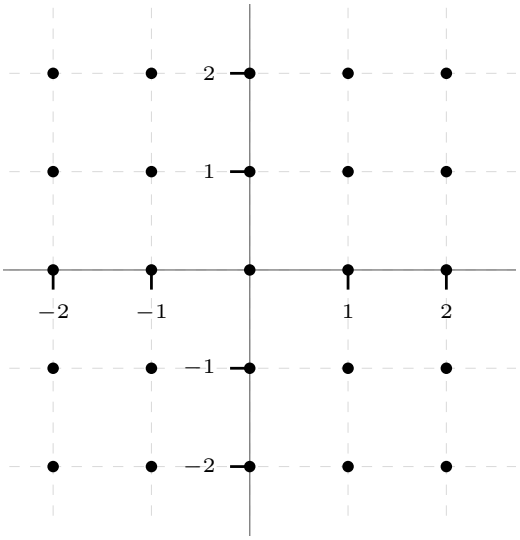

Figure 1.1: The integer lattice $\mathbb{Z}^2$.

all but a negligible fraction of instances.

No efficient algorithms are known for solving the SVP, CVP, and SIVP problems using either classical or quantum computers. Because of that, in lattice-based cryptography, the security proofs consider approximations for such problems, in which an approximation factor γ determines a distance from the optimal solution. Particularly, the SVP problem is commonly defined as a promise problem as follows.

Definition 1 (GapSVP$_\gamma$). *For an approximation factor $\gamma = \gamma(n) \geq 1$, the GapSVP$_\gamma$ is: given a lattice Λ and length $d > 0$, output YES if $\lambda_1(\Lambda) \leq d$ and NO if $\lambda_1(\Lambda) > \gamma d$.*

Promise problems generalize decision problems, where the input is promised to belong to a particular subset of all possible inputs. However, the behavior is not specified for the complement of this subset, and the algorithm may return any answer [93]. Promise problems are indicated by the predicate "gap".

Lattice-based cryptography offers fast and energy-saving primitives as well as quantum resistance. This claim is reinforced by practical experimentation such as Saarinen's [105], which evaluated the bandwidth, latency, and energy consumption of round-two submissions to NIST's Post-Quantum Cryptography standardization project on an ARM Cortex M4 processor. His results also take into account measurements for ECDH and ECDSA algorithms, which are elliptic-curve cryptosystems for key-encapsulation and digital signatures. Tables 1.1 and 1.2 present selected results for ECC- and lattice-based cryptosystems. Notice the superior performance of lattices compared to current cryptosystems in terms of running time, at the expense of increased memory consumption. Saarinen's complete evaluation also indicates that the considered lattice-based cryptosystems consume less energy than elliptic-curve cryptography.

Table 1.1: Bandwidth and latency, in bytes and 10^6 clock cycles, respectively, of post-quantum KEM algorithms on an ARM Cortex M4 STM32F at 96 Mhz [105].

KEM Algorithm	**Post-Quantum Level**	pk	ct	KGen	Enc	Dec
ECDH-secp256k1	none	64	64	4.108	8.215	4.108
ECDH-secp256r1	none	64	64	5.814	11.63	5.815
Kyber512	L1	800	736	0.516	0.654	0.623
Kyber768	L3	1184	1088	0.978	1.150	1.100
Kyber1024	L5	1568	1568	1.575	1.784	1.714
NTRU-HPS2048509	L1	699	699	78.79	0.634	0.546
NTRU-HPS2048677	L3	930	930	141.3	0.943	0.849
NTRU-HRSS701	L3	1138	1138	154.2	0.398	0.898
NTRU-HPS4096821	L5	1230	1230	212.0	1.189	1.079
LightSaber	L1	672	736	0.457	0.651	0.677
Saber	L3	992	1088	0.899	1.170	1.209
FireSaber	L5	1312	1472	1.455	1.791	1.854

Table 1.2: Bandwidth and latency, in bytes and 10^6 clock cycles, respectively, of post-quantum signature schemes on an ARM Cortex M4 STM32F at 96 Mhz [105].

Signature Scheme	**Post-Quantum Level**	pk	ct	KGen	Sig	Vf
ECDSA-secp256k1	none	64	64	4.109	4.475	4.546
ECDSA-secp256r1	none	64	64	5.814	6.185	6.639
Dilithium2	L1	1184	2044	1.328	4.663	1.389
Dilithium3	L2	1472	2701	2.172	7.212	2.116
Dilithium4	L3	1760	3366	2.930	7.263	2.997
FALCON-512	L1	897	690	182.2	39.57	0.493
FALCON-512-tree	L1	897	690	200.9	18.19	0.492
FALCON-1024	L5	1793	1330	380.2	79.36	1.013

1.1.1 NIST's Post-Quantum Cryptography Standardization Project

In 2016, NIST initiated a process for standardization of public-key quantum-resistant cryptographic algorithms. The process, named Post-Quantum Cryptography, requested nominations of key-encapsulation mechanisms, public-key encryption algorithms, and schemes for digital signatures [101]. In the past few years, this process drew strong interest from the academic community and the industry to develop and implement post-quantum cryptosystems. In November 2017, 82 candidates, whose categorization is depicted in Figure 1.2, were submitted for consideration by NIST. Among these, 69 submissions met the minimum acceptance criteria and were accepted as first-round candidates. The first round lasted until January 2019, and 26 algorithms were selected to advance to the second round. The second-round candidates are organized by their underlying classes of problems in Figure 1.3.

In Figure 1.4, we detail the second-round NIST candidates. The magenta nodes indicate the submissions that did not match the minimum acceptance criteria to advance to the third round. The third round announcement occurred in July 2020, classifying the

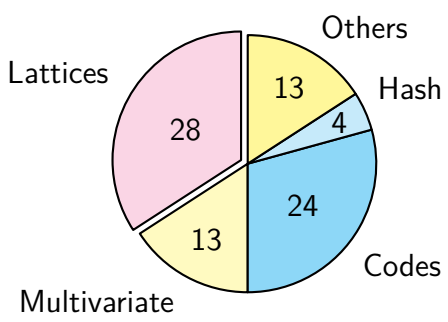

Figure 1.2: Categorization of submissions submitted to the first round of NIST's standardization project.

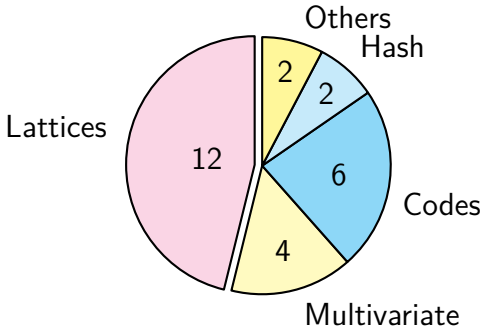

Figure 1.3: Categorization of submissions accepted for the second round of NIST's standardization project.

submissions as third-round finalists and alternate candidates, represented by gray and cyan nodes, respectively. From the beginning of the standardization project, lattice-based cryptography has been the most prominent class of problems. Currently, it represents five of the seven third-round finalists.

The main computational problem in the foundation of more recent lattice-based cryptosystems is the Learning With Errors (LWE) Problem [121]. Since its introduction in the cryptographic realm in 2005, algebraically structured variants of LWE have been proposed, such as LWE over rings [88], denoted Ring-LWE, Poly-LWE [62], and Module-LWE [35, 83, 7], among others [116]. These algebraic structured lattices, called ideal lattices, arise from a ring R, which is usually taken as a polynomial ring $R = \mathbb{Z}[x]/(f(x))$. In the Ring-LWE Problem, this ring is usually isomorphic to the ring of integers of a cyclotomic number field.

Among the third-round NIST candidates, the hardness of CRYSTALS-Dilithium [63, 64], CRYSTALS-Kyber [31, 18], and Saber [60, 25] is based on LWE variants. Moreover, Kyber, Dilithium, Saber, and FALCON [69] operate on polynomial rings of the form $R = \mathbb{Z}[x]/(x^n + 1)$ for n a power of two. In particular, $x^n + 1$ is maximally sparse, allowing polynomial multiplication using the Number-Theoretic Transform (NTT), which provides linear-time multiplication in the transform domain.

The evaluation criteria adopted throughout NIST's standardization process evaluate three aspects of the candidates: *i*) security; *ii*) cost and performance; and *iii*) algorithm and implementation characteristics [98]. At the beginning of the third round, the remaining candidates survived a considerable amount of cryptanalysis, which reinforced their security claims. This belief motivated the development of efficient and secure implementations in various platforms [102, 103]. In particular, the tasks of polynomial multiplication and error sampling are the most computationally expensive operations in lattice-based cryptography. Following this branch of cryptographic engineering, we designed experiments on the x64 architecture for evaluating the suitability of the Discrete Galois Transform (DGT) for polynomial multiplication in those candidates that adopt the NTT. The results are discussed in Chapter 6, Part II.

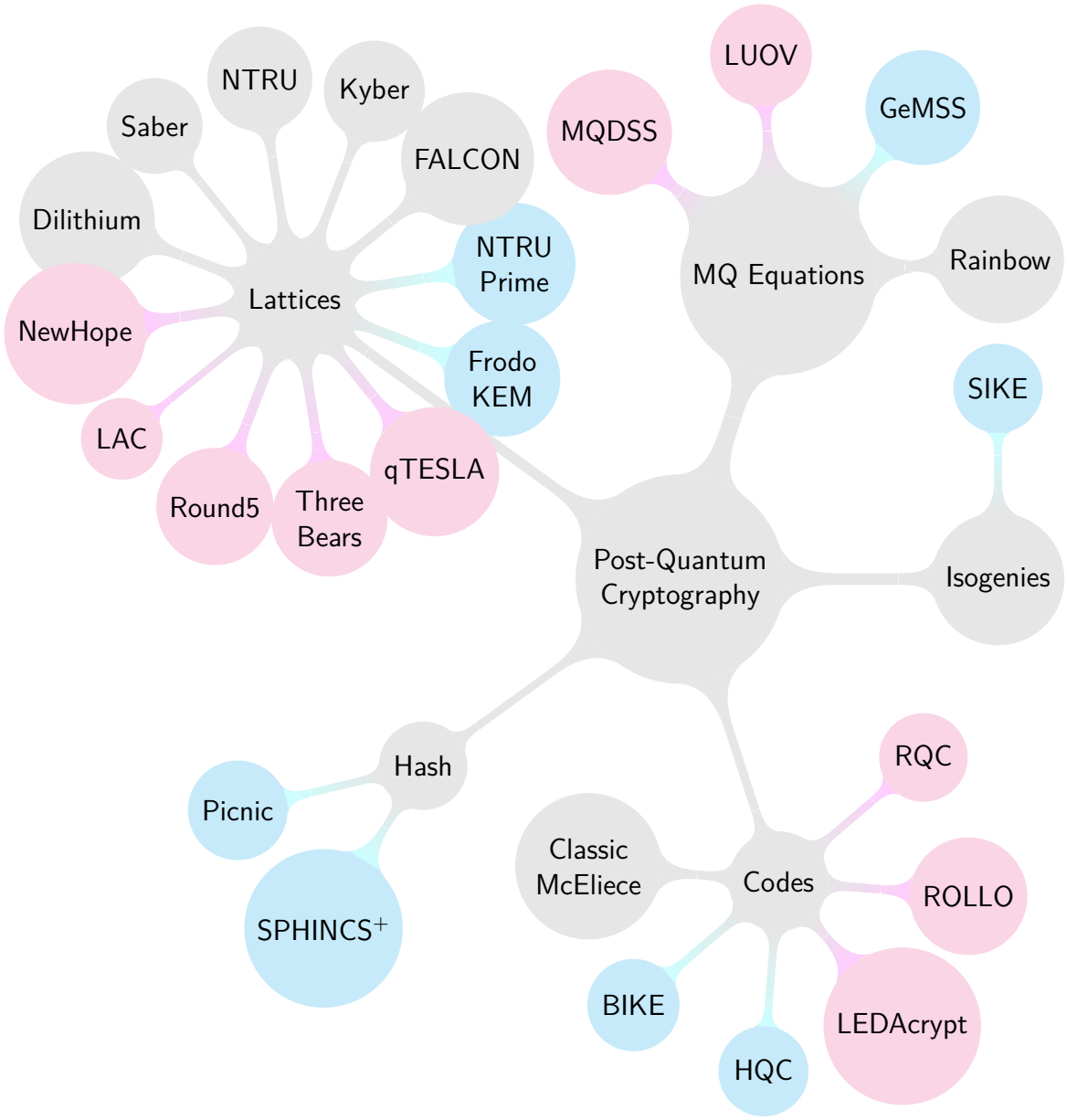

Figure 1.4: Second- and third-round NIST's candidates. The magenta nodes indicate the submissions that did not match the minimum acceptance criteria to advance to the third round. Third-round finalists and alternate candidates are represented by gray and cyan nodes, respectively.

1.1.2 Homomorphic Encryption Standardization

Apart from the algorithms for key generation, encryption, and decryption, a homomorphic encryption scheme also defines homomorphic addition and multiplication. These additional algorithms allow the computation over encrypted data without requiring decryption. A basic framework for homomorphic encryption is as follows. Consider an encryption scheme defined as a tuple of algorithms (KGen, Enc, Dec). Also, consider two messages m_0 and m_1 given in plaintext. A homomorphic encryption scheme is endowed

with a property which satisfies

$$\mathsf{Dec}(\mathsf{Enc}(m_0, \mathsf{pk}) \bullet \mathsf{Enc}(m_1, \mathsf{pk}), \mathsf{sk}) = \mathsf{Dec}(\mathsf{Enc}(m_0 \diamond m_1, \mathsf{pk}), \mathsf{sk}).$$

In the RSA cryptosystem, both operators refer to multiplication. However, in Paillier's cryptosystem [112], $\diamond$ and $\bullet$ are the usual addition and multiplication, respectively.

In 1978, Rivest, Adleman, and Dertouzos [123] conceived the notion of Homomorphic Encryption (HE) schemes. The idea was to preserve some mathematical structure after encryption that enabled the evaluation of arithmetic circuits over ciphertexts without decryption or knowledge of the secret key. The first HE schemes had limited capability, supporting either additions or multiplications, and because of that, they were called Partially Homomorphic Encryption (PHE) schemes. Examples of PHE schemes are due to ElGamal [70] and Paillier [112]. These schemes do not support addition and multiplication simultaneously, making their suitability to real-world implementations limited.

Around thirty years later, Gentry [72] introduced the first practical construction of Fully Homomorphic Encryption (FHE), supporting an unlimited number of additions and multiplications. The early proposals were not attractive due to high latency or memory consumption, but Gentry's work served as a blueprint for many FHE schemes [73, 35, 32, 86, 68, 47, 45]. His main contribution was the proposal of a bootstrapping operation that homomorphically evaluates the decryption procedure to remove the upper bound on the complexity of supported functions [72].

Homomorphic encryption has enabled new services for secure data computation for the medical, health, and financial sectors. Because of that, a consortium of industry, government, and academia was created in 2017 to standardize homomorphic encryption schemes. The standard [5] summarizes the security of most current FHE schemes, recommending security parameters for several security levels. In particular, the primary schemes for implementation of HE are the BGV [35] and BFV [68]. The CKKS [45] is referred to as an alternative scheme. Notice that most libraries for homomorphic encryption implement schemes based on the Ring-LWE Problem, displaying common choices of underlying rings, the most common being $R = \mathbb{Z}[x]/(x^n + 1)$ with n a power of two.

The standard recommends parameter sets depending on the cost model and error distribution. In general, the ring dimension ranges from 1024 to 32 768. The modulus size can be as small as $\log 15$ or as large as $\log 829$. In this sense, depending on the application, HE schemes require high-performance computers such as Graphics Processing Units (GPUs), which deliver memory bandwidth of around one terabyte per second and thousands of cores.

Although the overhead on computation over ciphertexts has been significantly reduced in modern schemes [125, 48], their implementation relies on polynomial arithmetic. Thus, libraries of HE require efficient techniques to reduce the overhead of costly operations such as polynomial multiplication and division. In particular, the CUDA paradigm[1] can be used to explore the Single Instruction Multiple Data (SIMD) capability of GPUs for accelerating the implementation of polynomial multiplication [58, 80, 22, 2].

[1]Compute Unified Device Architecture (CUDA) is a SIMD architecture developed and maintained by NVIDIA to employ GPU parallelism in tasks beyond graphical processing.

In this context, we present in Chapter 7, Part II, CUDA-enabled implementations of the BFV [68] and CKKS [45] homomorphic encryption schemes. We focus our implementation efforts on accelerating polynomial multiplication using formulations of the Discrete Galois Transform (DGT). The results are evaluated in NVIDIA Tesla GPUs.

1.1.3 Alternative Instantiations for Ring-LWE Cryptography

Several works have been exploring properties of number fields used in the foundation of cryptosystems based on ideal lattices. An example is a quantum polynomial-time algorithm to find a small generator of a principal ideal in the ring of algebraic integers of cyclotomic rings [36], which applies to a few schemes, including the fully-homomorphic encryption scheme of Smart and Vercauteren [129]. Moreover, a sequence of works has characterized weak instances of Ring-LWE and Poly-LWE problems and proposed attacks using special properties for specific parameters [65, 66, 44, 38, 37, 43, 40, 41, 42, 130]. Another motivation for searching for alternative number fields is the inflexibility of system parameters that grow as a power-of-two. When it is required to increase the security level in such cryptosystems, it may be necessary to increase the lattice dimension, which implies doubling its size. However, a more suitable dimension could be a value much smaller than the next power of two. A ring dimension ranging from 700 to 800 suffices for 128-bit security [6].

Although these weak instances are not addressed by worst-case hardness theorems [115], new proposals adopting non-conventional rings have emerged as alternatives, thus favoring the diversification of security assumptions. For NTRU-based schemes, examples are the NTTRU [90], the third-round NTRU submission [39] in the NIST Post-Quantum Cryptography contest [101], and NTRU Prime [29]. For Ring-LWE, the instantiations have been restricted to cyclotomic number fields. Lyubashevsky, Peikert, and Regev introduced a toolkit with techniques for secure implementation of Ring-LWE primitives over any cyclotomic number field [89], allowing applications to work on cyclotomic rings with non-power-of-two dimension. Later on, this toolkit was implemented in software in two distinct libraries [92, 57]. An alternative instantiation could be the adoption of the polynomial ring $\mathbb{Z}[x]/(x^p - x - 1)$ for p prime, which was proposed for NTRU Prime [29], and suggested for the Ring-LWE setting [118].

We conjecture that the Ring-LWE Problem can be parameterized by number fields other than the cyclotomic for cryptographic applications. Furthermore, we extended the Ring-LWE class of problems to embrace more general algebraic constructions of lattices which allow additional factors on the embedding coordinates. Part I of this thesis is dedicated to the advances in this investigation.

1.2 Contributions

In the last few years, the standardization and practicality of cryptographic constructions based on the Ring-LWE Problem advanced rapidly, motivated by NIST's Post-Quantum Cryptography standardization process and the Homomorphic Encryption Standardization consortium. However, these cryptosystems' security was not submitted to an extensive

test of time since Ring-LWE is dated from 2010 [88]. Also, homomorphic encryption schemes usually require huge ring instantiations, requiring high-computational power for arithmetic operations. This thesis aims to provide evidence of the security of Ring-LWE cryptography and propose alternative algorithms for its efficient implementation. We achieve the proposed objectives through theoretical and practical contributions summarized as follows.

DGT on x64 architecture. We re-implement the polynomial multiplication in the reference code of selected cryptosystems targeting the x64 architecture, replacing the NTT with two variations of the Discrete Galois Transform (DGT). One variation was introduced by Badawi et al. [4]. The second is proposed in Section 6.5.1 and merges the twisting procedure with the computation of the transform. The experimental evaluation is done in an Intel Skylake processor and reported in Section 6.5.2. Complementary experimental results on an Intel Haswell processor are given in Appendix A. Unfortunately, our results indicate that both DGT's variants do not explore well the x64 architecture in comparison with the NTT, even in an AVX2-optimized implementation.

Recursive transforms on GPUs. Motivated by a recursive formulation of the Fast Fourier Transform (FFT), we define the Recursive DGT (RDGT) in Chapter 7, which is compared to the straightforward DGT in CUDA-enabled implementations [22, 2] of homomorphic encryption schemes in GPUs. Then, we contrast the RDGT with a recursive formulation of NTT in the same context. As a result, the experiments indicate that the high arithmetic density of DGT is natively suitable to the GPU architecture. In particular, the homomorphic multiplication in the BFV scheme [68] using the RDGT is up to 3.6 times faster than the state-of-the-art. Also, the homomorphic multiplication in the CKKS scheme [45] is up to 15% faster than an implementation using the Recursive NTT.

Twisted Ring-LWE. Part I is dedicated to our proposal, the Twisted Ring-LWE Problem. We extend the Ring-LWE Problem in Chapter 3 by replacing the canonical embedding with twisted embeddings on both the search and decision variants. We explore some applications of Twisted Ring-LWE in Chapter 4, including a special case of ideal lattices in which the spherical Gaussian sampling can be performed directly in the number field without any loss in the sphericity or standard deviation. Further, we show that twisted embeddings can be used for converting instances between distinct number fields if the corresponding lattices in $\mathbb{R}^n$ are equivalent. The applicability of Twisted Ring-LWE for cryptography is discussed in Chapter 5, examining a public-key encryption scheme over a maximal real cyclotomic number field. As a result, algebraic constructions from coding theory via twisted embeddings can also be used in cryptographic applications based on the Ring-LWE Problem. Also, we conjecture that algebraic properties of equivalent number fields can be used to assert the actual security of a ring instantiation.

Publications

Most of the content described in Part I was published in the open-access journal Entropy [109] as a result of collaborative work with Robson R. de Araujo, Diego F. Aranha, Sueli I. R. Costa, and Ricardo Dahab. Early versions of this work [109] were presented at LAWCI 2018 [111] and CrossFyre 2019 [108], and posted online at the IACR Cryptology ePrint Archive [110]. Moreover, the results in Section 4.2, and 5.3 are contemporary to these publications.

Part II results from joint work with Pedro Geraldo M. R. Alves and Diego F. Aranha. The experimental results on Chapter 6 evaluating the DGT on x64 architecture were not published. However, the results obtained for the BFV homomorphic encryption scheme in Section 7.3 can be found in the proceedings of the International Conference on Financial Cryptography and Data Security 2021 [13]. Finally, the experimental evaluations of RDGT in the CKKS homomorphic encryption scheme presented in Section 7.4 were submitted to the Journal of Cryptographic Engineering in December 2021.

1.3 Thesis Organization

Part I describes our advances in introducing twisted embeddings in Ring-LWE cryptography. Particularly, Chapter 2 presents concepts of lattices and algebraic number theory and the Ring-LWE Problem. The proposed class of problems, named Twisted Ring-LWE, is given in Chapter 3. Chapter 4 is dedicated to applications of twisted embeddings for Ring-LWE cryptography, focusing on the algebraic construction of rotated $\mathbb{Z}^n$-lattices under twisted embeddings. Finally, Chapter 5 discusses how to perform the main algorithmic tasks of a public-key encryption scheme assuming that the geometrical analysis of Ring-LWE is done via twisted embeddings.

Experimental results with the Discrete Galois Transform for polynomial multiplication in the ring $\mathbb{Z}[x]/(x^n+1)$ are given in Part II. Chapter 6 discuss the applicability of DGT for polynomial multiplication in a key-encapsulation mechanism and selected digital signature schemes. Experiments with DGT in homomorphic encryption schemes are detailed in Chapter 7. Lastly, Chapter 8 presents final remarks and directions for future works.

Part I

Twisted Embeddings

Introduction to Part I

Figure 1.5 illustrates the organization of the first part of this thesis. We start Chapter 2 by covering introductory concepts on Euclidean vector spaces. Then, we define lattices on these vector spaces and properties relevant to this thesis, such as duality and equivalence. Next, we cover algebraic number theory, relating algebraic number fields to lattices using field monomorphisms. Finally, we present the Ring-LWE Problem and hard lattice problems related to its security results.

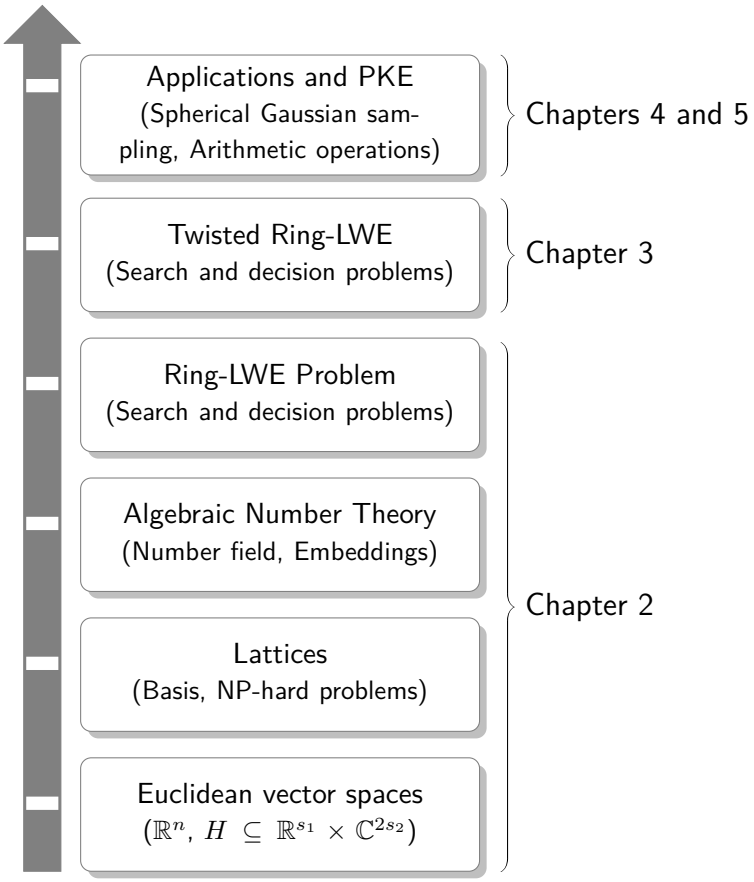

Figure 1.5: Structure of the first part of this thesis.

After that, the primary contribution of this thesis, named *Twisted Ring-LWE*, is given in Chapter 3 using results from algebraic number theory to extend the Ring-LWE Problem. Finally, in Chapters 4 and 5, we conclude the first part by presenting applications of Twisted Ring-LWE and a public-key encryption scheme defined on the proposed class of problems.

Chapter 2

Preliminaries

Notations. First, we state the notation used throughout this thesis. For a positive integer number m, denote by $[m]$ the set $\{1, \ldots, m\}$. We represent vectors using bold lower-case letters, for example, $\mathbf{a} = (a_1, \ldots, a_n)$. We use bold capital letters for matrices, for example, $\mathbf{M}$. In particular, the n-by-n identity matrix is denoted as $\mathbf{I}_n$. We denote the set of integer, real, and complex numbers as $\mathbb{Z}$, $\mathbb{R}$, and $\mathbb{C}$, respectively. For $1 \leq p < \infty$, the ℓ_p-norm of a vector $\mathbf{a}$ in $\mathbb{R}^n$ or $\mathbb{C}^n$ is $\|\mathbf{a}\|_p = \left(\sum_{i=1}^{n} |a_i|^p\right)^{1/p}$, and the ℓ_∞-norm is $\|\mathbf{a}\|_\infty = \max_i |a_i|$. We denote as $a \xleftarrow{\$} E$ that e is sampled from the uniform distribution over E. For a complex number $a \in \mathbb{C}$, we denote its real and imaginary parts as $\Re(a)$ and $\Im(a)$, respectively. The conjugate of a complex number $a \in \mathbb{C}$ is $\overline{a} = \Re(a) - i\Im(a)$ in which $i = \sqrt{-1}$.

2.1 Euclidean Vector Spaces

An *Euclidean vector space* $(E, \langle , \rangle_E)$ is an n-dimensional $\mathbb{R}$-vector space E with an inner product $\langle , \rangle_E$, which is isometric to $\mathbb{R}^n$ with the standard inner product. In the Ring-LWE context, lattices are conveniently defined in a specific subspace of $\mathbb{C}^n$ isometric to $\mathbb{R}^n$: the *space* H.

Definition 2 (Space H). *Let s_1 and s_2 be non-negative integer numbers such that $n = s_1 + 2s_2 > 0$. The* subspace $H \subseteq \mathbb{C}^n$ *is defined as*

$$H = \left\{(a_1, \ldots, a_n) \in \mathbb{R}^{s_1} \times \mathbb{C}^{2s_2} : a_{j+s_1+s_2} = \overline{a_{j+s_1}},\ \forall\ j \in [s_2]\right\}.$$

We consider H endowed with the inner product obtained as a restriction of the standard inner product of two vectors $\mathbf{a}, \mathbf{b}$ in $\mathbb{C}^n$:

$$\langle \mathbf{a}, \mathbf{b} \rangle_H := \sum_{i=1}^{n} a_i \overline{b_i} = \sum_{i=1}^{s_1} a_i b_i + \sum_{j=1}^{s_2} (a_{j+s_1} b_{j+s_1+s_2} + a_{j+s_1+s_2} b_{j+s_1}) \in \mathbb{R}.$$

In this sense, the norm (usually ℓ_2-norm) of $\mathbf{a} \in H$ is defined as $\|\mathbf{a}\| = \sqrt{\langle \mathbf{a}, \mathbf{a} \rangle_H}$.

Denote by $\mathbf{u}_i$ the vector with all zero coordinates except for the i-th position, which is equal to one, for $1 \leq i \leq n$. We consider $\{\mathbf{u}_1, \ldots, \mathbf{u}_n\}$ the canonical basis of $\mathbb{R}^n$ (over $\mathbb{R}$)

and $\mathbb{C}^n$ (over $\mathbb{C}$). An orthonormal basis[1] for H can be defined in terms of the canonical basis of $\mathbb{C}^n$ as follows.

Definition 3 (Canonical basis of H). *Let s_1 and s_2 be non-negative integer numbers such that $n = s_1 + 2s_2 > 0$. For $i \in [s_1]$, define $\mathbf{h}_i = \mathbf{u}_i$. For $i \in [s_2]$, define $\mathbf{h}_{i+s_1} = \frac{1}{\sqrt{2}}(\mathbf{u}_{i+s_1} + \mathbf{u}_{i+s_1+s_2})$ and $\mathbf{h}_{i+s_1+s_2} = \frac{i}{\sqrt{2}}(\mathbf{u}_{i+s_1} - \mathbf{u}_{i+s_1+s_2})$. Then, the set $\mathbf{B} = \{\mathbf{h}_1, \ldots, \mathbf{h}_n\}$ is an orthonormal basis of H, which we call the* canonical basis *of H as an n-dimensional $\mathbb{R}$-vector space.*

Notice that any vector $\mathbf{a} \in H \subseteq \mathbb{C}^n$ can be written as an $\mathbb{R}$-linear combination of the vectors of the canonical basis $\mathbf{B}$ of H as

$$\mathbf{a} = \sum_{i=1}^{s_1} a_i \mathbf{h}_i + \sum_{j=1}^{s_2} \sqrt{2}\Re(a_{j+s_1})\mathbf{h}_{j+s_1} + \sum_{j=1}^{s_2} \sqrt{2}\Im(a_{j+s_1})\mathbf{h}_{j+s_1+s_2}.$$

The linear map $\kappa\left(\sum_{i=1}^{n} b_i \mathbf{h}_i\right) := \sum_{i=1}^{n} b_i \mathbf{u}_i$, with $b_i \in \mathbb{R}$, defines an isomorphism between the $\mathbb{R}$-vector spaces H and $\mathbb{R}^n$, such that $\langle \mathbf{a}, \mathbf{b} \rangle_H = \langle \kappa(\mathbf{a}), \kappa(\mathbf{b}) \rangle$, where $\langle , \rangle$ denotes the standard inner product in $\mathbb{R}^n$. Then, it follows that H and $\mathbb{R}^n$ are isometric, that is, H is an Euclidean space. In particular, the norm of an element $\mathbf{a} \in H$ coincides with the usual norm of $\kappa(\mathbf{a}) \in \mathbb{R}^n$, that is, $\|\mathbf{a}\|_2 = \|\kappa(\mathbf{a})\|_2$.

2.2 Lattices

Consider an Euclidean vector space $(E, \langle , \rangle_E)$ with an orthonormal basis $\mathbf{B}(E) = \{\mathbf{e}_1, \ldots, \mathbf{e}_n\}$ of E. A set $\Lambda \subset E$ is said to be a *full-rank lattice* (or simply *lattice*), if Λ is a discrete additive subgroup of E with rank n. Equivalently, $\Lambda \subset E$ is a lattice if there exists a set of linearly independent vectors $\mathbf{B} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\} \subset E$ such that

$$\Lambda = \Lambda(\mathbf{B}) = \left\{ \sum_{i=1}^{n} a_i \mathbf{v}_i \ : \ a_i \in \mathbb{Z} \right\}.$$

The set $\mathbf{B}$ is called a *basis* (or a $\mathbb{Z}$-basis) of Λ. For each $\mathbf{v}_j \in \mathbf{B}$, it can be written in terms of the orthonormal basis $\mathbf{B}(E)$ as $\mathbf{v}_j = \sum_{i=1}^{n} v_{ij}\mathbf{e}_i$ for $v_{ij} \in \mathbb{R}$. In Figure 2.1, the hexagonal lattice is presented as an example of lattice in $\mathbb{R}^2$, for which a basis is given by the set of vectors $\mathbf{B} = \{(1, 0), (1/2, \sqrt{3}/2)\}$.

The *minimum distance* of a lattice Λ in the ℓ_p-norm, denoted $\lambda_1^{(p)}(\Lambda)$, is the length of a shortest nonzero lattice vector, that is, $\lambda_1^{(p)}(\Lambda) = \min_{\mathbf{0} \neq \mathbf{x} \in \Lambda} \|\mathbf{x}\|_p$. Similarly, for any $k \leq n$, the k-th *successive minimum* of a lattice Λ, denoted $\lambda_k^{(p)}(\Lambda)$, is the smallest $\hat{r} > 0$ such that Λ contains at least k linearly independent vectors of norm at most $\hat{r}$. These quantities will be used to define Gaussian measures over lattices and compute the approximation factors in security reductions.

[1] A subset $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ of an Euclidean vector space $(E, \langle , \rangle_E)$ is called orthonormal if $\langle \mathbf{v}_i, \mathbf{v}_j \rangle_E = 0$ when $i \neq j$. Also, all vectors $\mathbf{v}_i$ are required to have unit length.

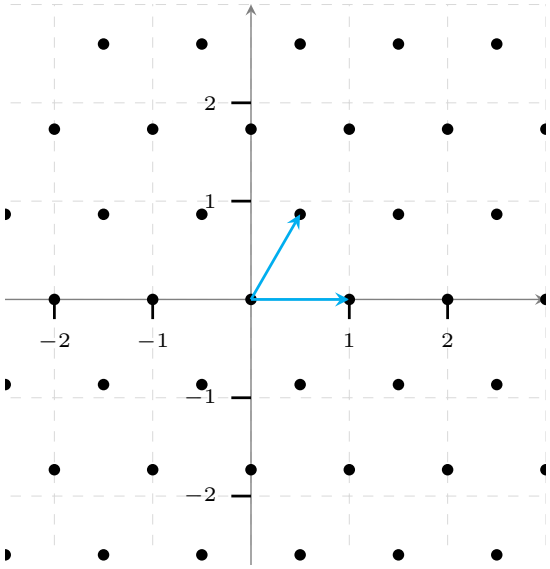

Figure 2.1: Hexagonal lattice and a basis $\mathbf{B} = \{(1,0),(1/2,\sqrt{3}/2)\}$.

The matrix $\mathbf{M} = [v_{ij}]_{n\times n}$, for which the j-th column is given by the coefficients of $\mathbf{v}_j$ written in the orthonormal basis $\mathbf{B}(E)$, is called a *generator matrix* of Λ. Two basis generate the same lattice if and only if the associated generator matrices $\mathbf{M}$ and $\mathbf{M}'$ are related as $\mathbf{M}' = \mathbf{MU}$, where $\mathbf{U}$ is unimodular[2]. The matrix $\mathbf{G} = \mathbf{M}^\top\mathbf{M}$ is called the *Gram matrix* of Λ with respect to $\mathbf{M}$. Since the basis $\mathbf{B}(E)$ of the Euclidean vector space is orthonormal, then $\mathbf{G} = [\langle \mathbf{v}_i, \mathbf{v}_j\rangle_E]_{n\times n}$. The determinant of $\mathbf{G}$ is called the *determinant* of Λ and is denoted by $\det(\Lambda)$. Clearly, $\det(\Lambda) = \det(\mathbf{M})^2$ does not depend of a particular basis of Λ.

The *dual lattice* of Λ is the lattice $\Lambda^* = \{\mathbf{a} \in E \ : \ \langle \mathbf{a}, \mathbf{b}\rangle_E \in \mathbb{Z}, \forall\ \mathbf{b} \in \Lambda\}$. It is known that $(\Lambda^*)^* = \Lambda$ and if Λ has generator matrix $\mathbf{M}$, then $(\mathbf{M}^\top)^{-1}$ is a generator matrix for Λ^* and therefore $\det(\Lambda^*) = \det(\Lambda)^{-1}$. We represent the dual of the hexagonal lattice in Figure 2.2.

A lattice $\Lambda \subset E$ is called *integral* if $\langle \mathbf{a}, \mathbf{b}\rangle_E \in \mathbb{Z}$ for all $\mathbf{a}, \mathbf{b} \in \Lambda$. Equivalently, Λ is an integral lattice if and only if $\Lambda \subset \Lambda^* \subset (\Lambda/\det(\Lambda))$. An integral lattice is called *unimodular*, or *self-dual*, if $\det(\Lambda) = 1$ or $\Lambda = \Lambda^*$.

Two lattices Λ and Λ' are said to be *equivalent* if one can be obtained from the other through a rotation, a reflection, or a change of scale. Two Gram matrices $\mathbf{G}$ and $\mathbf{G}'$ of two equivalent lattices Λ and Λ', respectively, are related as $\mathbf{G}' = c^2\mathbf{U}^\top\mathbf{GU}$, where $c \neq 0$ is a real constant and $\mathbf{U}$ is unimodular. Notice that the dual lattice in Figure 2.2 is equivalent to the hexagonal lattice up to a rotation and a scaling factor [52, Exercise 2.10].

We say that a lattice Λ in $(E, \langle , \rangle_E)$ is *orthogonal* if it has a basis $\mathbf{B} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ such that $\langle \mathbf{v}_i, \mathbf{v}_j\rangle = 0$ if $i \neq j$, for all $i, j \in [n]$. This means that Λ has a diagonal Gram matrix. Moreover, if the basis $\mathbf{B}$ satisfies $\langle \mathbf{v}_i, \mathbf{v}_j\rangle = 0$ if $i \neq j$ and $\langle \mathbf{v}_i, \mathbf{v}_j\rangle = c$ if $i = j$, for all $i, j \in [n]$ and some $c \in \mathbb{R}$, then Λ is equivalent to the $\mathbb{Z}^n$-lattice. In this case, Λ has a Gram matrix $\mathbf{G} = c\,\mathbf{I}_n$. In particular, when $c = 1$, we say that Λ is an *orthonormal* lattice.

[2]A matrix $\mathbf{U}$ is said to be unimodular if it has integer entries and $\det(\mathbf{U}) = \pm 1$.

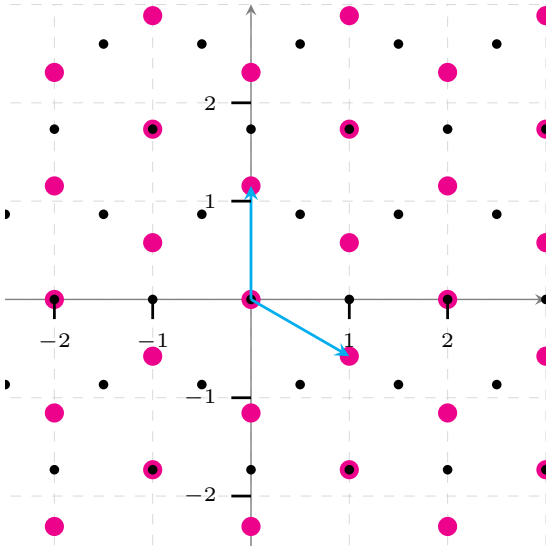

Figure 2.2: Hexagonal lattice and its dual lattice represented in black and magenta dots, respectively. A basis for the dual lattice is $\{(1, -1/\sqrt{3}), (0, 2/\sqrt{3})\}$. Notice that, in this case, the dual lattice is a rotated and expanded version of the original lattice and contains the double of it.

2.2.1 Gaussian Measures

For $r > 0$, define the Gaussian function $\rho_{r,\mathbf{c}} : H \to (0, 1]$ centered at $\mathbf{c}$ as

$$\rho_{r,\mathbf{c}}(\mathbf{a}) = \exp(-\pi\|\mathbf{a} - \mathbf{c}\|^2/r^2). \tag{2.1}$$

The subscript $\mathbf{c}$ is taken to be $\mathbf{0}$ when omitted. The Gaussian mass of a coset $\mathbf{c} + \Lambda$ is defined as $\rho_r(\mathbf{c} + \Lambda) = \sum_{\mathbf{x} \in \mathbf{c}+\Lambda} \rho_r(\mathbf{x})$. By normalizing the Gaussian function, we obtain the continuous Gaussian probability distribution D_r of width r, whose density is given by $r^{-n} \cdot \rho_r(\mathbf{x})$.

We extend this definition to elliptical Gaussian distributions in $\{\mathbf{h}_i\}_{i \in [n]}$, the canonical basis of H, as follows. Let $\mathbf{r} = (r_1, \ldots, r_n) \in (\mathbb{R}^+)^n$ be a vector of positive real numbers such that $r_{j+s_1+s_2} = r_{j+s_1}$ for each $j \in [s_2]$. Then, a sample from the n-dimensional distribution $D_{\mathbf{r}}$ is given by $\sum_{i=1}^{n} x_i \mathbf{h}_i$, where the terms x_i are chosen independently from the one-dimensional Gaussian distribution D_{r_i} over $\mathbb{R}$.

For any $\mathbf{c} \in \mathbb{R}^n$, real $r > 0$, and an arbitrary lattice Λ with dimension n, normalizing the Gaussian function $\rho_{r,\mathbf{c}}(\mathbf{a})$ gives the *discrete Gaussian distribution* over Λ as

$$D_{\Lambda,r,\mathbf{c}}(\mathbf{a}) = \frac{\rho_{r,\mathbf{c}}(\mathbf{a})}{\rho_{r,\mathbf{c}}(\Lambda)},$$

for all $\mathbf{a} \in \Lambda$.

The smoothing parameter is a lattice parameter defining the width beyond which a discrete Gaussian starts to behave similarly to a continuous distribution [96].

Definition 4 (Smoothing parameter). *For an n-dimensional lattice Λ and positive real $\epsilon > 0$, the* smoothing parameter $\eta_\epsilon(\Lambda)$ *is the smallest r such that $\rho_{1/r}(\Lambda^* \setminus \{\mathbf{0}\}) \leq \epsilon$.*

It relates to the minimum distance and the successive minimum of a lattice, and it will be used to derive the approximation factors in the worst-case to average-case reduction for the Twisted Ring-LWE Problem.

2.3 Algebraic Number Theory

In this section, we summarize concepts and results from algebraic number theory, presenting the case of cyclotomic number fields and their maximal real subfields. Details can be found in [126, 131, 107].

An (*algebraic*) *number field* K is a finite extension of the field $\mathbb{Q}$, meaning that $\mathbb{Q} \subset K$ and K is a $\mathbb{Q}$-vector space with finite dimension. The *degree* of K, denoted $[K : \mathbb{Q}]$, is the dimension of the $\mathbb{Q}$-vector space K. In general, if K and L are number fields such that $K \subset L$, the symbol $[L : K]$ is defined to be the integer number $[L : \mathbb{Q}]/[K : \mathbb{Q}]$ and is called the degree of the extension L/K.

By the Primitive Element Theorem, there exists an element $\theta \in K$ such that $K = \mathbb{Q}(\theta)$, which is equivalent to say that $\{1, \ldots, \theta^{n-1}\}$, with $n = [K : \mathbb{Q}]$, is a *power basis* of K over $\mathbb{Q}$. In this sense, an element $a \in K$ can be represented in the power basis as

$$a = c_1 + c_2\theta + \ldots + c_n\theta^{n-1}, \ \forall\ c_i \in \mathbb{Q}.$$

An *algebraic number* θ is a root of a nonzero polynomial with rational coefficients. If θ satisfies no similar equation of degree up to n, then θ is said to be an algebraic number of degree n.

If $f(x)$ is the minimal polynomial of θ over $\mathbb{Q}$, then K is isomorphic to $\mathbb{Q}[x]/(f(x))$ and $K = \mathbb{Q}(\theta')$ for some root θ' of $f(x)$. The roots of $f(x)$ are called the conjugates of θ.

Example 2.3.1. The field $\mathbb{Q}(\sqrt{2}) = \{c_1 + c_2\sqrt{2} \ : \ c_1, c_2 \in \mathbb{Q}\}$ is a *quadratic* number field, that is, it is a number field of degree two over $\mathbb{Q}$. By associating $\sqrt{2}$ with the indeterminate x, $\mathbb{Q}(\sqrt{2})$ is isomorphic to the polynomial ring $\mathbb{Q}[x]/(x^2 - 2)$.

Example 2.3.2 (Cyclotomic number field). A number field of particular interest is $\mathbb{Q}(\zeta_m)$, the *m-th cyclotomic field*, where $\zeta_m = \exp(2\pi i/m)$ is a primitive m-th root of unity for any integer number $m \geq 1$. The degree of $\mathbb{Q}(\zeta_m)$ is $\varphi(m)$, where $\varphi(\cdot)$ denotes Euler's totient function. The minimal polynomial of ζ_m, called the *m-th cyclotomic polynomial*, is $\Phi_m(x) = \prod_{k \in \mathbb{Z}_m^*}(x - \zeta_m^k)$, where $\mathbb{Z}_m^*$ denotes the group of invertible elements in $\mathbb{Z}_m$.

Example 2.3.3 (Maximal real subfield). For $m \not\equiv 2 \pmod 4$, $m > 1$, the number field $\mathbb{Q}(\zeta_m + \zeta_m^{-1}) \subset \mathbb{R} \cap \mathbb{Q}(\zeta_m)$ is the *maximal real subfield* of $\mathbb{Q}(\zeta_m)$ and has degree $\varphi(m)/2$.

Let K be a number field. A map $^- \colon K \to K$ is called an *involution* of K if $\overline{a + b} = \overline{a} + \overline{b}$, $\overline{a \cdot b} = \overline{a} \cdot \overline{b}$, and $\overline{\overline{a}} = a$, for all $a, b \in K$. If $K = \mathbb{C}$, the complex conjugation is an example of involution. If $K = \mathbb{Q}(\zeta_m)$ is a cyclotomic number field, then $\overline{\zeta_m} = \zeta_m^{-1}$ is the same involution given by the complex conjugation. In this work, whenever the cyclotomic number field is used, we implicitly assume this involution. For the maximal real subfield $\mathbb{Q}(\zeta_m + \zeta_m^{-1})$, we consider the involution given by the identity map.

The subfield $F = \{a \in K f \overline{a} = a\}$, called the fixed field by involution of K, satisfies $[K : F] \leq 2$. When $[K : F] = 1$ (or $F = K$), we say that the involution is *trivial* (it is the identity); otherwise, the involution is said to be *non-trivial*. If $K = \mathbb{Q}(\zeta_m)$, the fixed field by the involution $\overline{\zeta_m} = \zeta_m^{-1}$ of K is its maximal real subfield [27].

2.3.1 Field Monomorphisms

Definition 5 ([107]). *Let L_1, L_2 be two field extensions of a field K. A* field monomorphism *σ from L_1 to L_2 is a non-trivial field homomorphism, that is, a map from L_1 to L_2 such that*

$$\begin{aligned} \sigma(a \cdot b) &= \sigma(a) \cdot \sigma(b) \\ \sigma(a + b) &= \sigma(a) + \sigma(b) \\ \sigma(1) &= 1 \\ \sigma(0) &= 0 \end{aligned}$$

for all $a, b \in L_1$.

These monomorphisms are $\mathbb{Q}$-monomorphisms, that is, $\sigma(a) = a$ for all $a \in \mathbb{Q}$. Let K be a number field of degree n. There are exactly n distinct field monomorphisms σ_i from K to $\mathbb{C}$. If $K = \mathbb{Q}(\theta)$ and $f(x)$ is the minimal polynomial of θ, these monomorphisms are defined as $\sigma_i(\theta) = \theta_i$ for $i \in [n]$, where θ_i are all the distinct roots of $f(x)$.

A monomorphism $\sigma_i : K \to \mathbb{C}$ is said to be *real* if $\sigma_i(K) \subset \mathbb{R}$. Otherwise, it is said to be *complex*. If σ_i is a complex monomorphism, then $\overline{\sigma_i}$ is another complex monomorphism defined by $\overline{\sigma_i}(a) = \overline{\sigma_i(a)}$. So, the number field degree n can be written as $n = s_1 + 2s_2$, where $s_1 \geq 0$ is the number of real monomorphisms and $2s_2 \geq 0$ is the number of complex monomorphisms from K to $\mathbb{C}$. The pair (s_1, s_2) is called the *signature* of K. We say that K is *totally real* when $s_2 = 0$, and that K is *totally complex* when $s_1 = 0$. The number field K is said to be a *CM-field* if it is totally complex and has degree two over its fixed field by the involution F [27].

Any cyclotomic number field $K = \mathbb{Q}(\zeta_m)$, with $m \geq 3$, is totally complex. Their monomorphisms are defined as $\sigma_i(\zeta_m) = \zeta_m^i$ for each $i \in [m]$ such that $\gcd(i, m) = 1$. In turn, any maximal real cyclotomic subfield $\mathbb{Q}(\zeta_m + \zeta_m^{-1})$ is totally real. Their monomorphisms are defined as $\sigma_i(\zeta_m + \zeta_m^{-1}) = \zeta_m^i + \zeta_m^{-i}$ for each $1 \leq i \leq \lfloor m/2 \rfloor$ such that $\gcd(i, m) = 1$. Note that $\mathbb{Q}(\zeta_m)$ is a CM-field once $\mathbb{Q}(\zeta_m)$ is a totally complex field of degree two over $\mathbb{Q}(\zeta_m + \zeta_m^{-1})$.

The number field K is said to be a *Galois* number field if, for every $x \in K$, the minimal polynomial of x over $\mathbb{Q}$ has all its roots in K. In this case, the set of automorphisms $\sigma : K \to K$, where $\sigma(a) = a$ for all $a \in \mathbb{Q}$, constitutes a group under the composition, called *Galois group* of K over $\mathbb{Q}$, denoted $\mathrm{Gal}(K/\mathbb{Q})$. If $K \subset \mathbb{C}$ is a Galois number field, then the monomorphisms from K to $\mathbb{C}$ are exactly the elements of $\mathrm{Gal}(K/\mathbb{Q})$. An important fact is that any Galois number field is totally real or totally complex. Cyclotomic number fields and their maximal real subfields are Galois number fields.

2.3.2 Ring of Integers and Its Ideals

Let K be a Galois number field. For every $a \in K$, the *trace* and *norm* of any element $a \in K$ can be defined, respectively, as

$$\mathrm{Tr}_K(a) = \sum_{\sigma \in \mathrm{Gal}(K/\mathbb{Q})} \sigma(a) \qquad \text{and} \qquad \mathrm{N}_K(a) = \prod_{\sigma \in \mathrm{Gal}(K/\mathbb{Q})} \sigma(a).$$

For all $a \in K$, $\mathrm{Tr}_K(a)$ and $\mathrm{N}_K(a)$ are elements of $\mathbb{Q}$.

The set of all elements in a number field K that are the root of a monic polynomial in $\mathbb{Z}[x]$ is a ring called the *ring of integers* of K, denoted $\mathcal{O}_K$. If K is a number field of degree n, its ring of integers has a $\mathbb{Z}$-basis with n elements, which is called an *integral basis* of K. If $a \in \mathcal{O}_K$, then $\mathrm{Tr}_K(a)$ and $\mathrm{N}_K(a)$ are elements of $\mathbb{Z}$.

If $\mathcal{I}$ is a nonzero (integral) ideal of $\mathcal{O}_K$, then $\mathcal{I}$ has a $\mathbb{Z}$-basis with n elements. The same holds if $\mathcal{I}$ is a *fractional ideal* of K, which is a subset of K satisfying the condition that $d\mathcal{I} \subset \mathcal{O}_K$ is an integral ideal for some element $d \in \mathcal{O}_K$. Note that every integral ideal is also fractional ($d = 1$). Also, any $\mathbb{Z}$-basis of some nonzero fractional ideal of K, including its ring of integers, is a $\mathbb{Q}$-basis of K.

In the case of cyclotomic number fields $\mathbb{Q}(\zeta_m)$ of degree $n = \varphi(m)$, $\mathcal{O}_K = \mathbb{Z}[\zeta_m]$, which is the set of all $\mathbb{Z}$-linear combinations of powers of ζ_m:

$$\mathcal{O}_K = \{c_1 + c_2\zeta_m + \ldots + c_n\zeta_m^{n-1} \ : \ c_i \in \mathbb{Z}\}.$$

Similarly, the ring of integers of $\mathbb{Q}(\zeta_m + \zeta_m^{-1})$ is $\mathbb{Z}[\zeta_m + \zeta_m^{-1}]$. In general, the ring of integers of a number field $K = \mathbb{Q}(\theta)$ does not have the form $\mathbb{Z}[\theta]$. When this is the case, we say that K is a *monogenic* number field.

The fractional ideal $\mathcal{D}_K^{-1} = \{a \in K : \mathrm{Tr}_K(a\mathcal{O}_K) \subset \mathbb{Z}\}$ is the *codifferent ideal*, that is, the dual ideal of the ring of integers. Frequently, the codifferent ideal is also denoted by $\mathcal{O}_K^\vee$. Note that $\mathcal{O}_K \subset \mathcal{D}_K^{-1}$. If $\mathcal{O}_K = \mathbb{Z}[\theta]$ for some $\theta \in K$, then $\mathcal{O}_K^\vee = (p'(\theta))^{-1}\mathcal{O}_K$, where $p'(x)$ is the derivative of the minimal polynomial $f(x)$ of θ [122, Section 13.2, J]. The inverse ideal of the codifferent, that is, $\mathcal{D}_K = (\mathcal{D}_K^{-1})^{-1}$, is an ideal of $\mathcal{O}_K$ called *different* of K. In general, the *dual ideal* of any fractional ideal $\mathcal{I}$ of K is the fractional ideal $\mathcal{I}^\vee$ of K, defined as

$$\mathcal{I}^\vee := \{a \in K : \mathrm{Tr}_K(a\mathcal{I}) \subset \mathbb{Z}\} = \mathcal{I}^{-1} \cdot \mathcal{O}_K^\vee.$$

If $\mathcal{I}$ is any ideal of the ring $\mathcal{O}_K$, then the *congruence relation modulo* $\mathcal{I}$ is defined as follows. For $a, b \in \mathcal{O}_K$, let $a \equiv b \pmod{\mathcal{I}}$ when $a - b \in \mathcal{I}$. The equivalence class of an element $a \in \mathcal{O}_K$ is given by

$$a + \mathcal{I} := \{a + r \ : \ r \in \mathcal{I}\}.$$

If $\mathcal{I} \neq \mathcal{O}_K$, the congruent classes modulo $\mathcal{I}$ are identified with the elements of the quotient ring $\mathcal{O}_K/\mathcal{I}$ [122].

For a nonzero fractional ideal $\mathcal{I}$ of $\mathcal{O}_K$, the norm of $\mathcal{I}$ is $\mathrm{N}(\mathcal{I}) = |\mathcal{O}_K/\mathcal{I}|$, that is, the cardinality of the quotient of additive groups. If $\mathcal{I}$ and $\mathcal{J}$ are ideals of $\mathcal{O}_K$, then $\mathrm{N}(\mathcal{I}\mathcal{J}) = \mathrm{N}(\mathcal{I})\,\mathrm{N}(\mathcal{J})$, where $\mathcal{I}\mathcal{J}$ denotes the product of $\mathcal{I}$ and $\mathcal{J}$, that is, the set all finite

sums of products ab for $a \in \mathcal{I}$ and $b \in \mathcal{J}$. If $\mathcal{I}$ is a principal ideal generated by some $a \in K$, then $\mathrm{N}(\mathcal{I}) = |\,\mathrm{N}_K(a)|$.

2.4 Ideal Lattices

In 2006, Lyubashevsky and Micciancio [87] introduced the concept of ideal lattices as a generalization of cyclic lattices in the context of the Short Integer Solution Problem over rings (Ring-SIS). In a nutshell, a lattice is cyclic if it is invariant under cyclic rotations of coordinates [94]. Consider a lattice Λ spanned by the vectors of $\mathbf{B}$.

Definition 6 ([87]). *An* ideal lattice *is an integer lattice* $\Lambda(\mathbf{B}) \subseteq \mathbb{Z}^n$ *such that* $\Lambda(\mathbf{B}) = \{g(x) \mod f(x) : g(x) \in \mathcal{I}\}$ *for some monic polynomial* $f(x)$ *of degree* n *and ideal* $\mathcal{I} \subseteq \mathbb{Z}[x]/(f(x))$.

Later, in 2010, Lyubashevsky, Peikert, and Regev [88] introduced a ring-based variant of the Learning With Errors (LWE) Problem. Their main focus was on rings of the form $\mathbb{Z}[x]/(\Phi_m(x))$ in which $\Phi_m(x)$ is the m-th cyclotomic polynomial. However, in contrast with Definition 6, they chose the ring homomorphism as the canonical embedding rather than the coefficient embedding. Notice that, in the coefficient embedding, the lattice vectors are precisely the vector containing the coefficients of elements in the ring.

For a number field K, consider the n distinct field monomorphisms σ_i from K to $\mathbb{C}$. It follows that fractional ideals in K yields lattices under the canonical embedding, also known as the Minkowski embedding.

Definition 7. *The* canonical embedding *from* K *into the subspace* H *is the homomorphism*

$$\sigma(a) = (\sigma_1(a), \ldots, \sigma_n(a))\,.$$

Consider a fractional $\mathcal{I}$ with $\mathbb{Z}$-basis $\{b_1, \ldots, b_n\}$. The image of the canonical embedding in the space H is an ideal lattice with rank n and basis $\{\sigma(b_1), \ldots, \sigma(b_n)\} \subset H$. Thus, an ideal in K can be identified by its embedded lattice $\sigma(\mathcal{I})$ in which geometrical norms are taken. In particular, for any $a \in K$ and any $p \in [1, \infty]$, the ℓ_p-norm of a is $\|a\|_p = \|\sigma(a)\|_p = \left(\sum_{i=1}^{n} |\sigma_i(x)|^p\right)^{1/p}$ for $p < \infty$ and $\max_i |\sigma_i(a)|$ when $p = \infty$.

The fact that addition and multiplication are taken component-wise in the canonical embedding started to be used to design efficient algorithmic tasks and bound the error growth after multiplications. Especially, the expansion of an element resulting from multiplication is given by

$$\|a \cdot b\|_p \leq \|a\|_\infty \cdot \|b\|_p,$$

for any $a, b \in K$. In the coefficient embedding, such analysis requires computing the expansion factor of the underlying ring, which depends on the minimal polynomial $f(x)$. Section 5.2 will give details on the computation of the expansion factor.

Finally, the canonical embedding allows defining Gaussian distributions over ideals in K using the fact that the space H is isomorphic to the tensor field $K_{\mathbb{R}} = K \otimes_{\mathbb{Q}} \mathbb{R}$. The field $K_{\mathbb{R}}$ represents the number field K by extending rational coefficients to reals.

2.5 The Ring-LWE Problem

In 2005, Regev introduced the Learning With Errors (LWE) Problem along with a reduction from worst-case lattice problems such as the decision version of the Shortest Vector Problem (SVP) and the Shortest Independent Vectors Problem (SIVP) [121]. In 2010, Lyubashevsky, Peikert, and Regev proposed an algebraic variant of LWE called Ring-LWE that exploits algebraic structure to make cryptographic applications more efficient in terms of memory and running time [88].

For defining the Ring-LWE distribution and its associated computational problems, consider a number field K with ring of integers $R = \mathcal{O}_K$. Recall that $R^\vee$ is the (fractional) codifferent ideal of K, and let $\mathbb{T} = K_\mathbb{R}/R^\vee$. Let $q \geq 2$ be an integer modulus and, for any fractional ideal $\mathcal{I}$ of K, let $\mathcal{I}_q = \mathcal{I}/q\mathcal{I}$.

Definition 8 ([88] Ring-LWE Distribution). *For $s \in R_q^\vee$, called the secret, and an error distribution ψ over $K_\mathbb{R}$, a sample from the Ring-LWE distribution $\mathcal{A}_{s,\psi}$ over $R_q \times \mathbb{T}$ is generated by choosing $a \xleftarrow{\$} R_q$ uniformly at random, choosing $e \leftarrow \psi$, and outputting $(a, b = (a \cdot s)/q + e \mod R^\vee)$.*

Definition 9 ([88] Ring-LWE, Search). *Let Ψ be a family of distributions over $K_\mathbb{R}$. The search version of the Ring-LWE Problem, denoted R-LWE$_{q,\Psi}$, is defined as follows: given access to arbitrarily many independent samples from $\mathcal{A}_{s,\psi}$, for some arbitrary $s \in R_q^\vee$ and $\psi \in \Psi$, find s.*

Definition 10 ([88, 117] Ring-LWE, Average-case Decision). *Let Υ be a distribution over a family of error distributions, each over $K_\mathbb{R}$. The average-case Ring-LWE decision problem, denoted R-LWE$_{q,\Upsilon}$, is to distinguish (with non-negligible advantage) between independent samples from $\mathcal{A}_{s,\psi}$ for a random choice of $(s, \psi) \leftarrow U(R_q^\vee) \times \Upsilon$, and the same number of uniformly random and independent samples from $R_q \times \mathbb{T}$.*

Lyubashevsky, Peikert, and Regev proved the hardness of Ring-LWE in two components [88]. The first component is a quantum reduction from approximate SVP on ideal lattices in a ring R to the search variant of Ring-LWE. This reduction works for any number field. The second component is a classical reduction from the search problem to the decision variant of Ring-LWE. The authors provided two versions for this reduction: one with a nonspherical error distribution in the canonical embedding and the other with a spherical error distribution but with a limited number of samples. However, this second component only works for prime moduli that split completely and cyclotomic number fields.

In 2017, Peikert, Regev, and Stephens-Davidowitz extended to decision the hardness results for the search variant of Ring-LWE [117]. In particular, they gave a polynomial-time quantum reduction from worst-case approximate SIVP directly to the decision variant of (Ring-)LWE. In this sense, the hardness results of both search and decision variants follow the same template: a worst-case to average-case reduction from the Discrete Gaussian Sampling Problem (DGS) to Ring-LWE and a reduction from the Generalized Independent Vectors Problem (GIVP), which is a generalization of SIVP, to DGS.

We now formally define the computational problems which form the foundation of the (Ring)-LWE hardness, namely approximations for GapSVP, SIVP, and DGS, which is denoted K-DGS when the underlying lattice is taken over a number field K [88].

Definition 11 ($\mathbf{GapSVP}_\gamma$). *For an approximation factor $\gamma = \gamma(n) \geq 1$, the $GapSVP_\gamma$ is: given a lattice Λ and length $d > 0$, output YES if $\lambda_1(\Lambda) \leq d$ and NO if $\lambda_1(\Lambda) > \gamma d$.*

Definition 12 ($\mathbf{SIVP}_\gamma$). *For an approximation factor $\gamma = \gamma(n) \geq 1$, the $SIVP_\gamma$ is: given a lattice Λ, output n linearly independent lattice vectors of length at most $\gamma(n) \cdot \lambda_n(\Lambda)$.*

By seeing a fractional ideal $\mathcal{I}$ of an arbitrary number field K as a lattice using the canonical embedding, let $D_{\mathcal{I},r}$ denote the discrete Gaussian distribution of width r over $\mathcal{I}$ in $K_\mathbb{R} = K \otimes_\mathbb{Q} \mathbb{R}$, which is isomorphic to the space H.

Definition 13 (K-$\mathbf{DGS}_\gamma$). *For a function γ that maps lattices to nonnegative reals, the K-DGS_γ problem is: given an ideal $\mathcal{I}$ in K and a parameter $r \geq \gamma = \gamma(\mathcal{I})$, output an independent sample from a distribution that is within negligible distance of $D_{\mathcal{I},r}$.*

Alternatively, for the purpose of the worst-case to average-case reduction for (Ring-)LWE, the DGS problem can be stated as follows: given an n-dimensional lattice Λ and a number $r \geq \sqrt{2n} \cdot \eta_\epsilon(\Lambda)/\alpha$, output a sample from $D_{\Lambda,r}$.

Regev gave reductions from both SIVP and GapSVP to DGS [121, Section 3.3], providing a quantum algorithm for solving these hard lattice problems. Particularly, the hardness result from SIVP is given in Lemma 2.1.

Lemma 2.1 ([121, Lemma 3.17]). *For any $\epsilon = \epsilon(n) \leq \frac{1}{10}$ and any $\varphi(\Lambda) \geq \sqrt{2}\eta_\epsilon(\Lambda)$, there is a polynomial time reduction from $GIVP_{2\sqrt{n}\varphi}$ to DGS_φ.*

For completeness, we present the theorems that state the hardness results for both search and decision variants of Ring-LWE. These theorems will be used in Section 3.3.1 for analyzing the approximation factors in the presence of torsion factors. First, we define the class of error distributions $\Psi_{\leq\alpha}$ for which the search Ring-LWE is proved to be hard.

Definition 14 ([88, Definition 3.4]). *For a positive real $\alpha > 0$, the family $\Psi_{\leq\alpha}$ is the set of all elliptical Gaussian distributions $D_\mathbf{r}$ (over $K_\mathbb{R}$) where each parameter $r_i \leq \alpha$.*

Theorem 2.2 ([88, Theorem 4.1]). *Let K be an arbitrary number field of degree n and $R = \mathcal{O}_K$. Let $\alpha = \alpha(n) > 0$, and let $q = q(n) \geq 2$ be such that $\alpha q \geq 2 \cdot \omega(\sqrt{\log n})$[3]. For some negligible $\epsilon = \epsilon(n)$, there is a probabilistic polynomial-time quantum reduction from K-DGS_γ to R-$LWE_{q,\Psi_{\leq\alpha}}$, where*

$$\gamma = \max\left\{\eta_\epsilon(\mathcal{I}) \cdot (\sqrt{2}/\alpha) \cdot \omega\left(\sqrt{\log n}\right), \sqrt{2n}/\lambda_1(\mathcal{I}^\vee)\right\}.$$

Similarly, we define the distribution over a family of error distributions Υ_α for which instances of the decision variant of Ring-LWE are supported by hardness results.

[3]In this context, $\omega(f(n))$ denotes an arbitrary function that grows asymptotically faster than $f(n)$. Conversely, $O(f(n))$ denotes an arbitrary function upper bounded by the function $f(n)$.

Definition 15 ([117]). *Fix an arbitrary $f(n) = \omega(\sqrt{\log n})$. For $\alpha > 0$, a distribution sampled from Υ_α is an elliptical Gaussian $D_{\mathbf{r}}$, where $\mathbf{r} \in G$ is sampled as follows: for $i \in [s_1]$, sample $x_i \leftarrow D_1$ and set $r_i^2 = \alpha^2(x_i^2 + f^2(n))/2$. For $i = s_1 + 1, \ldots, s_1 + s_2$, sample $x_i, y_i \leftarrow D_{1/\sqrt{2}}$ and set $r_i^2 = r_{i+s_2}^2 = \alpha^2(x_i^2 + y_i^2 + f^2(n))/2$*

Theorem 2.3 ([117, Theorem 6.2]). *Let K be an arbitrary number field of degree n and $R = \mathcal{O}_K$. Let $\alpha = \alpha(n) \in (0, 1)$, and let $q = q(n) \geq 2$ be an integer such that $\alpha q \geq 2 \cdot \omega(1)$. There is a polynomial-time quantum reduction from K-DGS$_\gamma$ to (average-case, decision) R-LWE$_{q,\Upsilon_\alpha}$ for any*

$$\gamma = \max\left\{\eta(\mathcal{I}) \cdot \sqrt{2}/\alpha \cdot \omega(1), \sqrt{2n}/\lambda_1(\mathcal{I}^\vee)\right\}.$$

Chapter 3

Twisted Ring-LWE

In this chapter, we describe the main contribution of this thesis: the class of problems *Twisted Ring-LWE*. The proposed class of problems generalizes the original Ring-LWE Problem [88, 117] by replacing the canonical embedding with twisted embeddings.

The Ring-LWE Problem was introduced in the cryptographic realm in 2010, aiming at exploiting extra algebraic structure to reduce the quadratic overhead of the LWE problem [121]. The Ring-LWE was proposed focusing on polynomial rings of the form $\mathbb{Z}[x]/(\Phi_m(x))$ for $\Phi_m(x)$ the m-th cyclotomic polynomial, which is a representation for the ring of integers of the m-th cyclotomic number field. By fixing the underlying ring R as the ring of integers, the canonical embedding σ maps R to a lattice $\sigma(R)$ in $\mathbb{R}^n$ (Figure 3.1), in which approximated lattice problems are proved to be NP-hard. Also, the canonical embedding is a ring homomorphism from the number field to a space isomorphic to $\mathbb{R}^n$, meaning that the ring structure of R is preserved in $\sigma(R)$.

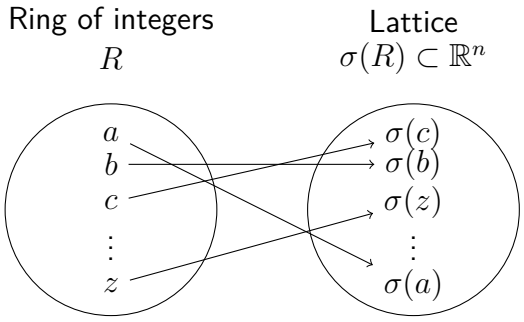

Figure 3.1: The canonical embedding depicted.

By noticing that twisted embeddings generalize the canonical embedding, we adapted the Ring-LWE Problem to incorporate algebraic constructions of rotated lattices. This modification enlarges the scope of lattices considered for cryptography, either to allow new ring instantiations or to provide additional information for cryptoanalysis.

We start this chapter by collecting necessary results on algebraic lattices obtained through twisted embeddings (Section 3.1). Then, Section 3.2 presents the class of problems Twisted Ring-LWE. The hardness of Twisted Ring-LWE is demonstrated by security reductions from the original Ring-LWE Problem. Also, we recompute the approximation factors in the worst-case to average-case reduction from SIVP, considering the torsion

factor which defines the twisted embedding (Section 3.3). Applications and practical aspects of using twisted embeddings in a cryptographic scheme are elaborated in Chapters 4 and 5.

3.1 Twisted Embeddings

In this section consider the following setting. Let K be an algebraic number field with degree n, signature (s_1, s_2), and $^-$ a fixed involution. Consider F to be the fixed field by the involution of K. Let σ_i be the real monomorphisms for $i \in [s_1]$, and σ_{i+s_1} be the complex monomorphisms for $i \in [2s_2]$ from K to $\mathbb{C}$, where $\sigma_{i+s_1+s_2} = \overline{\sigma_{i+s_1}}$ for all $i \in [s_2]$. The twisted embeddings defined next are a generalization of the canonical embedding [27]. An element $\tau \in K$ is said to be *totally positive* if $\tau \in F$ and $\tau_i = \sigma_i(\tau)$ is a positive real number for all $i \in [n]$.

Definition 16 (Twisted embeddings). *For any totally positive $\tau \in F$, the* twisted embedding σ_τ *is the homomorphism $\sigma_\tau : K \to H$, defined as*

$$\sigma_\tau(a) = \Big(\sqrt{\tau_1}\sigma_1(a), \ldots, \sqrt{\tau_{s_1}}\sigma_{s_1}(a), \sqrt{\tau_{1+s_1}}\sigma_{1+s_1}(a), \ldots, \sqrt{\tau_{2s_2+s_1}}\sigma_{2s_2+s_1}(a)\Big).$$

Since the element $\tau = 1$ in F is totally positive, then $\sigma_{\tau=1} = \sigma$ and twisted embeddings are generalizations of the canonical embedding. Twisted embeddings provide a way to obtain a variety of lattices in $H \simeq \mathbb{R}^n$ in addition to the ones obtained via canonical embedding, as a consequence of Proposition 3.1 [27].

Proposition 3.1 ([27]). *If M is a free $\mathbb{Z}$-module of rank n in K (particularly, if M is the ring of integers of K or any fractional ideal of K), then $\sigma_\tau(M)$ is a full-rank lattice in H.*

Twisted embeddings can be extended from K to $K_\mathbb{R}$ as follows. For any totally positive element $\tau \in F$, the $\mathbb{R}$-vector space $\sigma_\tau(K_\mathbb{R})$ is isomorphic to $H \simeq \mathbb{R}^n$. If $\mathfrak{B}$ is a $\mathbb{Q}$-basis of the number field K, then $\mathfrak{B}$ is an $\mathbb{R}$-basis of $K_\mathbb{R}$. So, for all totally positive $\tau \in F$, $\sigma_\tau(\mathfrak{B})$ is an $\mathbb{R}$-basis of H.

Consider the natural extension of the trace function $\mathrm{Tr}_K : K \to \mathbb{Q}$ to $\mathrm{Tr}_K : K_\mathbb{R} \to \mathbb{R}$. For any totally positive $\tau \in F$, we can define an inner product in $K_\mathbb{R}$ as

$$\langle a, b\rangle_\tau := \langle \sigma_\tau(a), \sigma_\tau(b)\rangle_H = \mathrm{Tr}_K(\tau a\bar{b}), \qquad a, b \in K_\mathbb{R}.$$

By considering the inner product $\langle , \rangle_\tau$, the $\mathbb{R}$-vector space $K_\mathbb{R}$ is an Euclidean vector space of dimension n isometric to both $(H, \langle , \rangle_H)$ and $(\mathbb{R}^n, \langle , \rangle)$.

Recall that, for each $a \in K_\mathbb{R}$, the ℓ_p-norms of a under the canonical embedding are $\|a\|_p = \|\sigma(a)\|_p = (\sum_{i=1}^n |\sigma_i(a)|^p)^{1/p}$ for $p < \infty$, and $\max_i |\sigma_i(a)|$ for $p = \infty$. Similarly, the ℓ_p-norms induced from $\mathbb{C}^n$ under twisted embeddings are defined as

$$\|a\|_{p,\tau} := \|\sigma_\tau(a)\|_p = \left(\sum_{i=1}^n |\sqrt{\tau_i}\sigma_i(a)|^p\right)^{1/p}$$

for $p < \infty$, and the ℓ_∞-norm is

$$\|a\|_{\infty,\tau} := \|\sigma_\tau(a)\|_\infty = \max_i |\sqrt{\tau_i}\sigma_i(a)|$$

in which $\tau_i = \sigma_i(\tau)$ for a totally positive element $\tau \in F$. Thus, any free $\mathbb{Z}$-module[1] M of rank n can be seen as a full-rank lattice directly in the Euclidean vector space $(K_\mathbb{R}, \langle , \rangle_\tau)$. However, the image of $\sigma_\tau(M)$ is frequently considered as in $(H, \langle , \rangle_H)$.

Using the fact that $\sigma_\tau(a \cdot b) = \sigma(a) \odot \sigma_\tau(b) = \sigma_\tau(a) \odot \sigma(b)$ for any $a, b \in K_\mathbb{R}$, where $\odot$ is the component-wise multiplication in the space H, it follows that

$$\|a \cdot b\|_{p,\tau} \leq \|a\|_\infty \|b\|_{p,\tau} \qquad \text{and} \qquad \|a \cdot b\|_{p,\tau} \leq \|a\|_p \|b\|_{\infty,\tau}. \tag{3.1}$$

Notice that, since multiplication of elements in $K_\mathbb{R}$ is mapped to coordinate-wise multiplication in H, for any element $a \in K_\mathbb{R}$, we have that the distribution of $a \cdot D_\mathbf{r}$ is $D_{\mathbf{r}'}$, where $r'_i = r_i \cdot \left|\sqrt{\tau_i}\sigma_i(a)\right|$ for $i \in [n]$. Because of the induced norms from $\mathbb{C}$, which maps elements of K to H, an elliptical distribution defined in the space H can be seen as a distribution directly over $K_\mathbb{R}$. For practical applications, sampling from an error distribution in $K_\mathbb{R}$ is done by generating the error in H and mapping it to its corresponding element in $K_\mathbb{R}$, via twisted embeddings. However, in some special cases, an error can be efficiently sampled directly in $K_\mathbb{R}$ without requiring the computation of the inverse of the Vandermonde matrix with respect to σ_τ [62].

Since $K_\mathbb{R} \simeq \mathbb{R}^n$ under twisted embeddings, it follows that $K_\mathbb{R}$ admits an orthonormal basis. Thus, for any $\mathbb{Z}$-basis $\mathfrak{B} = \{v_1, \ldots, v_n\}$ of the free $\mathbb{Z}$-module M of rank n in K, the matrix $[\langle v_i, v_j \rangle_\tau]_{n \times n}$ is a Gram matrix of the lattice M in $(K_\mathbb{R}, \langle , \rangle_\tau)$, which coincides with the Gram matrix of $\sigma_\tau(M)$ in $(H, \langle , \rangle_H)$ with respect to the basis $\{\sigma_\tau(v_1), \ldots, \sigma_\tau(v_n)\}$. It should be clear that, for different totally positive elements, the lattices obtained from M may not be equivalent, as can be seen below.

Example 3.1.1. Let $K = \mathbb{Q}(\sqrt{3}) = \{c_1 + c_2\sqrt{3} \ : \ a, b \in \mathbb{Q}\}$ be a totally real number field with degree two. It follows that the fixed field by the usual involution is $F = K$. For any totally positive element $\tau \in F$, consider the lattice $M_\tau = \mathcal{O}_K = \mathbb{Z}[\sqrt{3}]$ in the inner product space $(K_\mathbb{R}, \langle , \rangle_\tau)$. The set $\{1, \sqrt{3}\}$ is a $\mathbb{Z}$-basis of M_τ and the Gram matrix of the lattice M_τ is given by

$$\mathbf{G}_\tau = \begin{bmatrix} \mathrm{Tr}_K(\tau) & \mathrm{Tr}_K(\tau\sqrt{3}) \\ \mathrm{Tr}_K(\tau\sqrt{3}) & \mathrm{Tr}_K(3\tau) \end{bmatrix}.$$

For example, for $\tau = 1$ and $\tau = 2 + \sqrt{3}$, the Gram matrices are

$$\mathbf{G}_1 = \begin{bmatrix} 2 & 0 \\ 0 & 6 \end{bmatrix} \qquad \text{and} \qquad \mathbf{G}_{2+\sqrt{3}} = \begin{bmatrix} 4 & 6 \\ 6 & 12 \end{bmatrix}. \tag{3.2}$$

Suppose that these two lattices are equivalent. Then, there exists a square matrix $\mathbf{U}$ with integer entries and determinant ± 1, and a real number $k \neq 0$ such that $\mathbf{G}_{2+\sqrt{3}} =$

[1] In general, if A is a commutative ring, then an A-module is a set M, endowed with the operations of addition $+ : M \times M \to M$ and scalar multiplication $\cdot : M \times M \to M$, which satisfy precisely the same axioms as the addition and scalar multiplication of a vector space, except that the scalars are taken from a ring instead of a field [82, Chapter 3].

$k^2\mathbf{U}^\top\mathbf{G}_1\mathbf{U}$. Since the determinant of both matrices in (3.2) is equal to 12, then $k = \pm 1$. Now, consider $\mathbf{U}$ to be a matrix for which the rows are given by the vectors $(a, b) \in \mathbb{Z}^2$ and $(c, d) \in \mathbb{Z}^2$. So, the system of equations $\mathbf{G}_{2+\sqrt{3}} = \mathbf{U}^\top\mathbf{G}_1\mathbf{U}$ has no solution $(a, b, c, d) \in \mathbb{Z}^4$ because the equation $2 = a^2 + 3c^2$, provided by the first entry, has no solution $(a, c) \in \mathbb{Z}^2$. This gives a contradiction. Therefore, the lattices given by the same module $M = \mathcal{O}_K$ in the two different inner product spaces $(K_\mathbb{R}, \langle , \rangle_1)$ and $(K_\mathbb{R}, \langle , \rangle_{2+\sqrt{3}})$ are not equivalent.

Any full-rank lattice M in $(K_\mathbb{R}, \langle , \rangle_\tau)$ is said to be an *algebraic lattice*. If $M = \mathcal{I}$ is a fractional ideal in K and the lattice $\mathcal{I}$ is integral – that is, $\langle a, b\rangle_\tau \in \mathbb{Z}$ for all $a, b \in \mathcal{I}$ –, then $\mathcal{I}$ can be called an *ideal lattice* in $(K_\mathbb{R}, \langle , \rangle_\tau)$. Since $\langle a, b\rangle_\tau = \mathrm{Tr}_K(\tau a\bar{b})$, an ideal $\mathcal{I}$ of K constitutes an ideal lattice in $(K_\mathbb{R}, \langle , \rangle_\tau)$ if and only if $\tau\mathcal{I}\overline{\mathcal{I}} \subset \mathcal{D}_K^{-1}$ $(= \mathcal{O}_K^\vee)$. Ideal lattices can be obtained if and only if K is either a totally real number field or a CM-field. In particular, ideal lattices can be obtained via cyclotomic number fields and their maximal real subfields.

Let $\mathcal{I}$ be a fractional ideal of K. It is known that $\sigma(\mathcal{I}^\vee) = \overline{\sigma(\mathcal{I})^*}$ in H under the canonical embedding. However, the same does not hold for twisted embeddings in general, as can be inferred from Proposition 3.2.

Proposition 3.2. *Let $\tau \in F$ be a totally positive element and let $\mathcal{I}$ be a fractional ideal of K. Then, in the Euclidean vector space $(K_\mathbb{R}, \langle , \rangle_\tau)$, it follows that:*

(i) $\mathcal{I}^* = \tau^{-1}\overline{\mathcal{I}}^\vee$; and

(ii) $\mathcal{I}$ is an unimodular (self-dual) lattice in $(K_\mathbb{R}, \langle , \rangle_\tau)$ if and only if $\tau\mathcal{I}\overline{\mathcal{I}} = \mathcal{D}_K^{-1}$.

Proof. By definition, $a \in \mathcal{I}^*$ if and only if $\mathrm{Tr}_K(\tau a\overline{\mathcal{I}}) \subset \mathbb{Z}$, which occurs if and only if $\tau a \in \overline{\mathcal{I}}^\vee$, which is equivalent to $a \in \tau^{-1}\overline{\mathcal{I}}^\vee$. This proves (i). Secondly, $\mathcal{I}$ is unimodular when $\mathcal{I}$ is integral and $\mathcal{I} = \mathcal{I}^*$. The lattice $\mathcal{I}$ is integral if and only if $\tau\mathcal{I}\overline{\mathcal{I}}^{-1} \subset \mathcal{D}_K^{-1}$. In turn, by (i), $\mathcal{I} = \mathcal{I}^*$ if and only if $\mathcal{I} = \tau^{-1}\overline{\mathcal{I}}^\vee = \tau^{-1}\overline{\mathcal{I}}^{-1}\mathcal{D}_K^{-1}$, which is equivalent to $\tau\mathcal{I}\overline{\mathcal{I}} = \mathcal{D}_K^{-1}$. Therefore, $\mathcal{I}$ is unimodular if and only if $\tau\mathcal{I}\overline{\mathcal{I}} = \mathcal{D}_K^{-1}$. □

3.2 Twisted Ring-LWE Problem

This section introduces a new class of problems, named *Twisted Ring-LWE*, that extends the Ring-LWE Problem, adopting twisted embeddings instead of canonical embedding. We prove that solving the Twisted Ring-LWE Problem is at least as hard as solving the original Ring-LWE Problem [88] by providing a polynomial-time reduction from Ring-LWE to Twisted Ring-LWE.

In the Ring-LWE distribution (Definition 8), the error e is randomized by a distribution ψ over the space $(K_\mathbb{R}, \langle , \rangle_{\tau=1})$. In this sense, an error in $K_\mathbb{R}$ can be seen as the inverse image of a sample from the distribution ψ in $H \simeq \mathbb{R}^n$ via canonical embedding. In our general case, we consider K a number field with an involution, F its associated fixed field, $\tau \in F$ a totally positive element, and σ_τ the twisted embedding. Thus, the error e is randomized by a distribution ψ over $(K_\mathbb{R}, \langle , \rangle_\tau)$. In the following, it is assumed $q \geq 2$ is an integer number, $R = \mathcal{O}_K$, and $\mathcal{I}_q = \mathcal{I}/q\mathcal{I}$ for any fractional ideal $\mathcal{I}$ of K.

Definition 17 (Twisted Ring-LWE Distribution). *For a totally positive element $\tau \in F$, let ψ_τ denote an error distribution over the inner product $\langle , \rangle_\tau$ and $s \in R_q^\vee$, called the secret, be an uniformly randomized element. The* Twisted Ring-LWE distribution *$\mathcal{A}_{s,\psi_\tau}$ produces samples of the form*

$$(a,\ b = a \cdot s + e \mod qR^\vee) \in R_q \times K_\mathbb{R}/qR^\vee, \tag{3.3}$$

where $a \xleftarrow{\$} R_q$ is sampled uniformly at random and the error e is randomized by ψ_τ in $(K_\mathbb{R}, \langle , \rangle_\tau)$.

Originally, Ring-LWE was defined in the space $K_\mathbb{R}$ provided with the inner product associated with the canonical embedding [88]. Analogously, we can define the search and decision variants of Ring-LWE in $(K_\mathbb{R}, \langle , \rangle_\tau)$. We modify the definition of the family of error distributions Ψ and Υ previously given in (14) and (15) for the space $(K_\mathbb{R}, \langle , \rangle_\tau)$. Consider that we strictly follow the search problem as defined by Lyubashevsky et al. [88] and the decision problem which was further defined by Peikert et al. [117].

Definition 18. *For a positive real $\alpha > 0$, the family $\Psi_{\leq\alpha}^{(\tau)}$ is the set of all elliptical Gaussian distributions $D_\mathbf{r}$ over $(K_\mathbb{R}, \langle , \rangle_\tau)$, where each parameter $r_i \leq \alpha$.*

Definition 19 (Twisted Ring-LWE, Search). *Let $\Psi^{(\tau)}$ be a family of distributions over the inner product space $(K_\mathbb{R}, \langle , \rangle_\tau)$. The search version of the Twisted Ring-LWE Problem, denoted R-LWE$_{q,\Psi^{(\tau)}}$, is defined as follows: given access to arbitrarily many independent samples from $\mathcal{A}_{s,\psi_\tau}$ for some arbitrary $s \in R_q^\vee$ and $\psi_\tau \in \Psi^{(\tau)}$, find s.*

Definition 20. *Fix an arbitrary $f(n) = \omega\left(\sqrt{\log n}\right)$. For $\alpha > 0$, a distribution sampled from $\Upsilon_\alpha^{(\tau)}$ is an elliptical Gaussian $D_\mathbf{r}$ in $(K_\mathbb{R}, \langle , \rangle_\tau)$, where $\mathbf{r}$ is sampled as follows: for $i \in [s_1]$, sample $x_i \leftarrow D_1$ and set $r_i^2 = \alpha^2(x_i^2 + f^2(n))/2$. For $i = s_1 + 1, \ldots, s_1 + s_2$, sample $x_i, y_i \leftarrow D_{1/\sqrt{2}}$ and set $r_i^2 = r_{i+s}^2 = \alpha(x_i^2 + y_i^2 + f^2(n))/2$.*

Notice that, in Definition 20, sampling $x_i \leftarrow D_1$ for $i \in [s_1]$ and $x_i, y_i \leftarrow D_{1/\sqrt{2}}$ for $i = s_1 + 1, \ldots, s_1 + s_2$ is done according to the Gaussian function given in Equation 2.1, using the norm induced by the corresponding twisted embedding σ_τ.

Definition 21 (Twisted Ring-LWE, Average-case Decision). *Let $\Upsilon^{(\tau)}$ be a distribution over a family of error distributions, each in the inner product space $(K_\mathbb{R}, \langle , \rangle_\tau)$. The average-case Twisted Ring-LWE decision problem, denoted R-LWE$_{q,\Upsilon^{(\tau)}}$, is to distinguish, with non-negligible advantage, between arbitrarily many independent samples from $\mathcal{A}_{s,\psi_\tau}$, for a random choice of $(s, \psi_\tau) \leftarrow U(R_q^\vee) \times \Upsilon^{(\tau)}$, and the same number of uniformly random and independent samples from $R_q \times K_\mathbb{R}/R^\vee$.*

Generally speaking, the Twisted Ring-LWE distribution and the search and decision variants of Twisted Ring-LWE collapse to their original definitions in the Ring-LWE Problem when $\tau = 1$.

3.3 Hardness of the Twisted Ring-LWE

In this section, we provide evidence for the hardness of Twisted Ring-LWE. First, we give reductions from Ring-LWE to the Twisted Ring-LWE Problem. By doing so, Twisted Ring-LWE is proven to be at least as hard as NP-hard lattice problems. Moreover, these are indeed self reductions, in the sense that they preserve the secret term $s \in R_q^\vee$, only distorting the error distribution over $K_\mathbb{R}$.

Theorem 3.3. *Let K be an arbitrary number field with $R = \mathcal{O}_K$ and $\tau \in K$ be totally positive. Let (s, ψ) be randomly chosen from $(U(R_q^\vee) \times \Psi)$ in $(K_\mathbb{R}, \langle , \rangle_{\tau=1})$. Then, there is a polynomial-time reduction from R-LWE$_{q,\Psi}$ to R-LWE$_{q,\Psi^{(\tau)}}$.*

Proof. We assume the existence of an oracle for R-LWE$_{q,\Psi^{(\tau)}}$ that, given a set of independent samples from $\mathcal{A}_{s,\psi_\tau}$, for some arbitrary $s \in R_q^\vee$ and $\psi_\tau \in \Psi^{(\tau)}$, recovers the secret term s. Given a set of independent samples from the Ring-LWE distribution $\mathcal{A}_{s,\psi}$, solving the search version of Ring-LWE amounts to finding the secret s. In order to evoke the R-LWE$_{q,\Psi^{(\tau)}}$ oracle to solve Ring-LWE, we must ensure that the error terms from the input samples follow a Gaussian distribution $\psi_\tau \in \Psi^{(\tau)}$. Let the input samples from $\mathcal{A}_{s,\psi}$ be represented as

$$(a_i, b_i = a_i \cdot s + e_i \mod qR^\vee) \in R_q \times \mathbb{T},$$

where $e_i \xleftarrow{\psi} K_\mathbb{R}$. Thus, we use the fact that $e_i = \sigma^{-1}(\tilde{\mathbf{e}}_i)$, for some $\tilde{\mathbf{e}}_i$ obtained from the Gaussian distribution ψ over H. The Twisted Ring-LWE samples are obtained by first computing the corresponding representatives of each pair (a_i, b_i) in H as

$$\{(\sigma(a_i), \sigma(b_i))\} = \{(\sigma(a_i), \sigma(a_i) \cdot \sigma(s) + \tilde{\mathbf{e}}_i)\}.$$

By applying the inverse transformation σ_τ^{-1}, we obtain that

$$\left\{\left(\sigma_\tau^{-1}\left(\sigma\left(a_i\right)\right), \sigma_\tau^{-1}\left(\sigma(b_i)\right)\right)\right\} = \left\{\left(\sigma_\tau^{-1}\left(\sigma\left(a_i\right)\right), \sigma_\tau^{-1}\left(\sigma\left(a_i\right)\right) \cdot s + \sigma_\tau^{-1}(\tilde{\mathbf{e}}_i)\right)\right\}. \tag{3.4}$$

Notice that s was unchanged by the transformations, so it is a randomized element over $R_q^\vee$. Because a_i was sampled according to a uniform distribution over R_q and both σ and σ_τ^{-1} transformations are injective, $\sigma_\tau^{-1}(\sigma(a_i))$ is also uniform in R_q. And, finally, since $e_i' = \sigma_\tau^{-1}(\tilde{\mathbf{e}}_i)$ is randomized by ψ_τ in $(K_\mathbb{R}, \langle , \rangle_\tau)$, the set of samples in (3.4) follows the distribution $\mathcal{A}_{s,\psi^\tau}$. Given the set of samples (3.4) as input for the Twisted Ring-LWE solver, it finds the secret s. Then, mapping the solution to the Ring-LWE instance of the R-LWE$_{q,\Psi^{(\tau)}}$ solution is done by the identity transformation. Since the computation of the transformations σ and σ_τ^{-1} can be seen as vector-matrix multiplications, the reduction costs $O(n^2)$ operations. Thus, the given reduction from R-LWE$_{q,\Psi}$ to R-LWE$_{q,\Psi^{(\tau)}}$ runs in polynomial time. This concludes the proof. □

Theorem 3.4. *Let K be an arbitrary number field with $R = \mathcal{O}_K$ and $\tau \in K$ be a totally positive element. Let (s, ψ) be randomly chosen from $(U(R_q^\vee) \times \Upsilon)$ in $(K_\mathbb{R}, \langle , \rangle_{\tau=1})$. There is a polynomial-time reduction from R-LWE$_{q,\Upsilon}$ to R-LWE$_{q,\Upsilon^{(\tau)}}$.*

Proof. Given a set of m pairs of the form $(a_i, b_i) \in R_q \times \mathbb{T}$, each drawn either from $\mathcal{A}_{s,\psi}$ or from a uniform distribution over $R_q \times \mathbb{T}$, we prove that the (decision) Ring-LWE Problem

can be solved using only an oracle for (decision) Twisted Ring-LWE and a polynomial-time function for mapping the input instances. As in the reduction for the search variant, we apply the transformations σ and σ_τ^{-1}, in this order, to each pair $(a_i, b_i) \in R_q \times \mathbb{T}$. As a result, those pairs drawn from $(U(R_q), U(\mathbb{T}))$ are still uniformly distributed over $R_q \times \mathbb{T}$, since both σ and σ_τ^{-1} are injective maps. On the other hand, the pairs drawn from $\mathcal{A}_{q,\psi}$ now follow the Twisted Ring-LWE distribution $\mathcal{A}_{q,\psi_\tau}$. Thus, given an algorithm that solves (decision) R-LWE$_{q,\Upsilon^{(\tau)}}$, it distinguishes in two different sets the $m/2$ samples drawn from $\mathcal{A}_{q,\psi_\tau}$ and those $m/2$ uniformly distributed. Since mapping Ring-LWE to Twisted Ring-LWE instances preserves distributions, the solution for (decision) Ring-LWE problem is done by an identity transformation. Finally, the computation of the transformations σ and σ_τ^{-1} costs $O(n^2)$ operations; thus, the reduction runs in polynomial time. This concludes the proof. □

3.3.1 Computing the Approximation Factors

Consider an arbitrary number field K of degree n with a ring of integers $R = \mathcal{O}_K$, and a fractional ideal $\mathcal{I}$ in K. This section analyzes the effect of redefining the inner product in the Ring-LWE security reductions. Note that the same fractional ideal leads to different lattices up to the change of ring homomorphism.

By strictly following the setting of Lyubashevsky et al. [88], we start by deriving upper bounds for the smoothing parameter concerning the ℓ_p-norm under twisted embeddings. From the inequalities in (3.1), we are able to relate the ℓ_p-norm under twisted embeddings with the infinity norm under the canonical embedding as

$$\|a\|_\infty \geq \frac{\|a\|_{p,\tau}}{\left(\sum_{i=1}^{n} \tau_i^{p/2}\right)^{\frac{1}{p}}}.$$

We can also relate ℓ_p-norms under both embeddings in H as

$$\frac{1}{\max_i \tau_i} \cdot \|a\|_{p,\tau} \leq \|a\|_p \leq \frac{1}{\min_i \tau_i} \cdot \|a\|_{p,\tau}.$$

Using the above inequalities, Lemmas 3.5 and 3.6 present upper bounds for the smoothing parameter associated with twisted embeddings, which are a simple adaptation of Lemmas 2.7 and 3.5 from [113]. Notice that, when $\tau = 1$, these upper bounds are exactly the same as presented in [113]. Consider that $\lambda_n^{(p,\tau)}(\Lambda)$ and $\lambda_1^{(p,\tau)}(\Lambda)$ denotes the k-th successive minimum and the minimum distance of a lattice Λ in the ℓ_p-norm, respectively, under a twisted embedding denoted σ_τ.

Lemma 3.5. *Let K be an arbitrary number field and $\tau \in K$ be totally positive. For any $p \in [2, \infty]$, any n-dimensional lattice Λ in $(K_\mathbb{R}, \langle , \rangle_\tau)$, and any $\epsilon > 0$,*

$$\eta_\epsilon(\Lambda) \leq \lambda_n^{(p,\tau)}(\Lambda) \cdot \frac{n^{1/2-1/p}}{\min_i \tau_i} \cdot \sqrt{\log(2n(1+1/\epsilon))/\pi}.$$

In particular, for any $\omega\left(\sqrt{\log n}\right)$ *function, there is a negligible function* $\epsilon(n)$ *for which*

$$\eta_\epsilon(\Lambda) \leq \lambda_n^{(p,\tau)}(\Lambda) \cdot \frac{n^{1/2-1/p}}{\min_i \tau_i} \cdot \omega\left(\sqrt{\log n}\right).$$

Lemma 3.6. *Let* K *be an arbitrary number field and* $\tau \in K$ *totally positive. For any* $p \in [1, \infty]$*, any* n*-dimensional lattice* Λ *in* $(K_{\mathbb{R}}, \langle , \rangle_\tau)$*, and any* $\epsilon > 0$*,*

$$\eta_\epsilon(\Lambda) \leq \frac{\max_i \tau_i \cdot n^{1/p} \cdot \sqrt{\log(2n(1+1/\epsilon))/\pi}}{\lambda_1^{(p,\tau)}(\Lambda^*)}.$$

In particular, for any $\omega\left(\sqrt{\log n}\right)$ *function, there is a negligible function* $\epsilon(n)$ *such that*

$$\eta_\epsilon(\Lambda) \leq \max_i \tau_i \cdot n^{1/p} \cdot \omega\left(\sqrt{\log n}\right) / \lambda_1^{(p,\tau)}(\Lambda^*).$$

The hardness of search Ring-LWE consists of two reductions: (i) a worst-case to average-case reduction from DGS to Ring-LWE (Theorem 2.2); and (ii) a reduction from SIVP to DGS (Lemma 2.1). Thus, we use the inequalities for the smoothing parameter η_ϵ derived in Lemmas 3.5 and 3.6 to recompute the approximation factors in both reductions. We start by computing the approximated factor γ from Theorem 2.1. As long as $\alpha < \sqrt{\log n/n}$, it follows that the K-DGS$_\gamma$ parameter is

$$\gamma = \eta_\epsilon(\mathcal{I}) \cdot \left(\sqrt{2}/\alpha\right) \cdot \omega\left(\sqrt{\log n}\right) = \eta_\epsilon(\mathcal{I}) \cdot \tilde{O}(1/\alpha).$$

Using the inequality $\eta_\epsilon(\mathcal{I}) \leq \lambda_n^{(p,\tau)}(\Lambda) \cdot \frac{n^{1/2-1/p}}{\min_i \tau_i} \cdot \omega\left(\sqrt{\log n}\right)$ from Lemma 3.5, we obtain that the parameter φ in Lemma 2.1 is

$$\varphi \leq \lambda_n^{(p,\tau)}(\Lambda) \cdot \frac{n^{1/2-1/p}}{\min_i \tau_i} \cdot \omega\left(\sqrt{\log n}\right) \cdot \tilde{O}(1/\alpha).$$

Now, using the above inequality for φ, we define the upper bound for the SIVP parameter to be μ, for which

$$\mu = 2\sqrt{n}\varphi \leq 2\sqrt{n} \cdot \lambda_n^{(p,\tau)}(\Lambda) \cdot \frac{n^{1/2-1/p}}{\min_i \tau_i} \cdot \omega\left(\sqrt{\log n}\right) \cdot \tilde{O}(1/\alpha).$$

Remark. *Notice that, regardless of the* ℓ_p*-norm,* $\mu = \tilde{O}\left(\sqrt{n}/\alpha\right)$*. Since* $\tilde{O}\left(\sqrt{n}/\alpha\right)$ *is the approximation factor for the search version of the Ring-LWE Problem [88, Section 4], we conclude that the approximation factors remain unchanged concerning the change of embeddings due to the asymptotic notation. Moreover, since the torsion factor does not depend on the number field degree* n*, the approximation factors for the decision version of the Twisted Ring-LWE Problem also remain unchanged.*

Chapter 4

Applications of Twisted Ring-LWE

In this chapter, we present two applications of twisted embeddings. In both cases, we focus primarily on the algebraic construction of rotated $\mathbb{Z}^n$-lattices, mainly because the ring of integers of power-of-two cyclotomic number fields builds the $\mathbb{Z}^n$-lattice in the canonical embedding and are widely adopted in the literature [31, 12, 60, 10, 63, 69, 45].

As summarized by Peikert [115], some weak instances of the Ring-LWE Problem presented in the literature in the last few years [65, 66, 44, 38, 43] are insecure because the error distributions are not covered by hardness results. In general, a cryptosystem based on the intractability of Ring-LWE performs its algorithmic tasks in the polynomial representation of the ring of integers in $K_{\mathbb{R}}$. Moreover, for efficiency reasons, the error distribution is taken as a spherical Gaussian distribution. However, the error distribution must remain spherical when seen in H under the canonical embedding. When the number field is a power-of-two cyclotomic, the format of error distribution is preserved in H, but the width is enlarged by a factor related to the field conductor. For other choices of number fields, neither the format nor the standard deviation may be preserved.

Informally speaking, the main result of Section 4.1 states that when the ideal lattice is a rotated $\mathbb{Z}^n$-lattice, a spherical Gaussian distribution over $K_{\mathbb{R}}$ is totally preserved when seen in the space H. This result embraces the power-of-two cyclotomic case since the canonical embedding is a specialization of twisted embeddings.

Theorem (Informal). *If an ideal lattice is congruent to $\mathbb{Z}^n$ under a twisted embedding, it is possible to sample from a spherical Gaussian distribution in $K_{\mathbb{R}}$ with a hardness guarantee.*

In Section 4.2 we restrict the transformation used in the Ring-LWE to Twisted Ring-LWE reductions to lattices equivalent in the space H. In the security reductions, we assume that the Ring-LWE distribution is defined over a number field K. Then, we use a totally positive element $\tau \in K$ to compute $\sigma_\tau^{-1}(\sigma(b_i = a_i \cdot s + e_i))$ to adjust the error terms e_i. After these transformations, the error terms follow an error distribution under the twisted embedding σ_τ. As a result, the Twisted Ring-LWE instance is composed of a pair of elements in K. However, if we could choose a torsion factor in a distinct number field L, for which a fractional ideal $\mathcal{I}$ leads to a lattice equivalent to $\sigma(\mathcal{O}_K)$, then we can convert elements of $\mathcal{O}_K$ into $\mathcal{I}$. In this sense, we can switch samples of Twisted Ring-LWE from the number field K to L by computing the unimodular matrix that relates

their lattices in H. This result is summarized in the following statement.

Theorem (Informal). *If two ideal lattices are equivalent in H, then the Twisted Ring-LWE instances upon them are efficiently convertible into each other.*

Moreover, if we take K as a power-of-two cyclotomic number field and L as its maximal real subfield, we verify that instantiating the Ring-LWE Problem over K is equivalent to instantiating the Twisted Ring-LWE Problem over L. This result may provide additional information for asserting the hardness of using power-of-two cyclotomic number fields for cryptography.

4.1 Efficient Spherical Error Sampling

Ducas and Durmus [62] showed how to sample efficiently from a spherical Gaussian distribution in cyclotomic polynomial rings. This section extends this result to a more general class of number fields, taking the algebraic realization of $\mathbb{Z}^n$-lattices as an example. Consequently, if a lattice is equivalent to $\mathbb{Z}^n$, we can sample from a spherical Gaussian distribution in the tensor field $K_{\mathbb{R}}$ with a hardness guarantee.

Durmus and Ducas proved a special case when a spherical Gaussian distribution with width s in the power basis corresponds to a spherical Gaussian distribution with width $s\sqrt{m'}$ over the space H [62, Theorem 4.1]. In the statement of Theorem 4.1, let $m' = m$ if m is odd and $m' = m/2$ if m is even. Also, let β represent the polynomial reduction from $\mathbb{Q}[x]/(\Theta_m(x))$ to $\mathbb{Q}[x]/(\Phi_m(x))$, where $\Theta_m(x) = x^{m'} - 1$ if m is odd, and $\Theta_m(x) = x^{m'} + 1$ if m is even. Let the linear operator $\mathbf{T} : H \to H$ with matrix in the canonical basis of H be:

$$\mathbf{T} = \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{I}_{\phi(m)/2} & i\,\mathbf{I}_{\phi(m)/2} \\ \mathbf{I}_{\phi(m)/2} & -i\,\mathbf{I}_{\phi(m)/2} \end{pmatrix}, \quad \text{with } i = \sqrt{-1}. \tag{4.1}$$

Theorem 4.1 ([62, Theorem 5]). *Let $v \in \mathbb{Q}[x]/(\Theta_m(x))$ be a random variable distributed as $\psi_s^{m'}$ in the power basis. Then, the distribution of $(\mathbf{T}^{-1} \circ \sigma \circ \beta)(v)$, seen in the canonical basis of H, is the spherical Gaussian $\psi_{s\sqrt{m'}}^{\phi(m)}$.*

In order to sample directly over the cyclotomic ring $\mathbb{Q}[x]/(\Phi_m(x))$, leading to the correct distribution in the embedding representation, they sample the error polynomial in the ring $\mathbb{Q}[x]/(\Theta_m(x))$. Then, the reduction modulo Φ_m leads to the correct distribution under the canonical embedding. This method avoids resorting to complex embeddings and the inverse of the Vandermonde matrix.

The shape of the distribution is preserved because the transformation $\mathbf{T}^{-1} \circ \sigma$ is, in fact, a scaled-orthogonal map from the power basis of $\mathbb{Q}[x]/(\Phi_m(x))$ to the space H, where $\mathbf{T}^{-1}$ is Hermitian ($\mathbf{T}^{-1} = \overline{\mathbf{T}}^{\top}$). The proof for Theorem 4.1 reduces to proving that $\mathbf{M} \in \mathbb{C}^{\phi(m)\times m'}$, the matrix representing the linear map γ from the power basis of $\mathbb{Z}[x]/(\Theta_m(x))$ to the canonical basis of $\mathbb{C}^{\phi(m)}$ satisfies $\mathbf{C} = \mathbf{M}\overline{\mathbf{M}}^{\top} = m'\,\mathbf{I}_{\phi(m)}$. The coefficients of $\mathbf{M}$ are given by $m_{i,j} = \sigma_j(x^i) = \zeta_m^{ij}$. Then, for all $i, j \in \mathbb{Z}_m^*$, we have that

$$c_{i,j} = \sum_{k=1}^{m'} \zeta_m^{ik}\overline{\zeta_m^{jk}} = \sum_{k=1}^{m'} \left(\zeta_m^{i-j}\right)^k = \begin{cases} m' & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

Thus, $\mathbf{E} = \mathbf{T}^{-1}\mathbf{M} = \overline{\mathbf{E}}$, so $\mathbf{E}\mathbf{E}^\top = \mathbf{E}\overline{\mathbf{E}}^\top = \mathbf{T}^{-1}\mathbf{M}\overline{\mathbf{M}}^\top\mathbf{T} = m'\,\mathbf{I}_{\phi(m)}$. This last equation implies that, if a random variable $v \in \mathbb{Q}[x]/(\Theta_m(x))$ has covariance matrix $s^2\,\mathbf{I}_{m'}$, then the covariance matrix of $(\mathbf{T}^{-1} \circ \gamma)(v)$ is $s^2\mathbf{E}\,\mathbf{I}_{m'}\,\overline{\mathbf{E}}^\top = s^2 m'\,\mathbf{I}_{\phi(m)}$, and the distribution of $(\mathbf{T}^{-1} \circ \gamma)(v)$ is the spherical Gaussian $\psi^{\phi(m)}_{s\sqrt{m'}}$.

In the following, we discuss how the shape of spherical Gaussian distributions may be preserved when seen in the space H for special algebraic constructions under twisted embeddings. Following Ducas and Durmus' approach, we are interested in lattices equivalent to $\mathbb{Z}^n$, whose Gram matrices have the form $c\,\mathbf{I}_n$ for $c \in \mathbb{R}$. In this sense, the matrix mapping elements of $K_\mathbb{R}$ to the space H is a scaled-orthogonal map [62]. It follows that any algebraic realization of the $\mathbb{Z}^n$-lattice preserves the shape of an error distribution over $K_\mathbb{R}$ when seen as in H.

In Theorem 4.2, we prove that fractional ideals realizing lattices equivalent to $\mathbb{Z}^n$ in an orthonormal basis, which are the particular case when the Gram matrix is simply $\mathbf{I}_n$, preserve both format and standard deviation of spherical Gaussian distributions. We recall that ideal lattices can be obtained if and only if K is a totally real number field or if K is a CM-field. Recall that a CM-field is a totally imaginary quadratic extension of a totally real number field [27].

Theorem 4.2. *Let K be a number field with an involution. Consider $\tau \in K$ totally positive and $\mathcal{I} \subset \mathcal{O}_K$ a fractional ideal such that $\mathcal{I}$ is an ideal lattice in $(K_\mathbb{R}, \langle , \rangle_\tau)$. If $\mathcal{I}$ is a lattice equivalent to $\mathbb{Z}^n$, then both the format and the standard deviation of a spherical Gaussian distribution in an orthonormal basis of $\mathcal{I} \subset K_\mathbb{R}$ are preserved when seen in the canonical basis of the space H (via the twisted embedding σ_τ).*

Proof. Let n be the degree of K and let $v \in \mathcal{I}$ be a random variable over the spherical Gaussian distribution with covariance matrix $s^2\,\mathbf{I}_n$ in an orthonormal $\mathbb{Z}$-basis of $\mathcal{I}$, for some real number s. Since the twisted embedding $\sigma_\tau : K_\mathbb{R} \to H$ is a linear transformation, the covariance matrix of $\sigma_\tau(v)$ in the canonical basis of H is $\mathbf{E}s^2\,\mathbf{I}_n\,\mathbf{E}^\top$, where $\mathbf{E} = \mathbf{T}^{-1}\mathbf{M}$, with $\mathbf{T}$ as in (4.1) and $\mathbf{M}$ is the generator matrix of $\sigma_\tau(\mathcal{I})$. Since $\mathbf{M}\mathbf{M}^\top = \mathbf{M}^\top\mathbf{M} = \mathbf{I}_n$, and because $\mathbf{M}\mathbf{M}^\top$ is the Gram matrix of the $\mathbb{Z}^n$-equivalent lattice $\mathcal{I}$ in $(K_\mathbb{R}, \langle , \rangle_\tau)$, the covariance matrix of $\sigma_\tau(v)$ is

$$\mathbf{E}s^2\,\mathbf{I}_n\,\mathbf{E}^\top = \mathbf{T}^{-1}\mathbf{M}s^2\,\mathbf{I}_n\,\mathbf{M}^\top\mathbf{T} = s^2\,\mathbf{I}_n,$$

which proves that $\sigma_\tau(v)$ is randomized in the spherical Gaussian distribution over the canonical basis of H with the same standard deviation as v over $K_\mathbb{R}$ in the orthonormal basis of $\mathcal{I}$. This concludes the proof. □

Examples of ideal lattices equivalent to $\mathbb{Z}^n$ are those obtained from cyclotomic number fields $\mathbb{Q}(\zeta_{2^k})$ [27], and their maximal real subfields [16], and the maximal real subfields $\mathbb{Q}(\zeta_p + \zeta_p^{-1})$ for any prime $p \geq 5$. The case of the power-of-two cyclotomic number fields were previously addressed by Lyubashevsky et al. [88], and Ducas and Durmus [62]. In the following, we discuss the family of lattices equivalent to $\mathbb{Z}^n$ built on $\mathbb{Q}(\zeta_p + \zeta_p^{-1})$, for any $p \geq 5$ prime.

Let $p \geq 5$ be a prime number, $n = (p-1)/2$, and $\zeta = \zeta_p = \exp(-2i\pi/p)$. The cyclotomic construction of the $\mathbb{Z}^n$-lattice (Proposition 4.3) is on the ring of integers of

the maximal real subfield of a cyclotomic number field whose integral basis is $\mathcal{C} = \{e_j = \zeta^j + \zeta^{-j} : 1 \leq j \leq n\}$.

Proposition 4.3 ([106, Proposition 1]). *Let $p \geq 5$ be a prime number, and let $K = \mathbb{Q}(\zeta_p + \zeta_p^{-1})$ and $\tau = \frac{1}{p}(1-\zeta_p)(1-\zeta_p^{-1})$. Then $\mathcal{O}_K$ in $(K_{\mathbb{R}}, \langle , \rangle_\tau)$ is a lattice equivalent to $\mathbb{Z}^n$ with basis $\mathcal{C}' = \{e'_1, \ldots, e'_n : e'_n = e_n \text{ and } e'_j = e_j + e'_{j+1}\}$, where $\mathcal{C} = \{e_1, \ldots, e_n\}$ is an integral basis of K.*

The generator matrix of the $\mathbb{Z}^n$-lattice in $H = \mathbb{R}^n$, realized in Proposition 4.3, is given by

$$\mathbf{M} = \mathbf{D}\mathbf{M}'\mathbf{U}, \tag{4.2}$$

where $\mathbf{D} = \text{diag}\left[\sqrt{\frac{\sigma_k(\tau)}{p}}\right]_{n\times n}$, $\mathbf{M}' = [\sigma_i(\zeta^j + \zeta^{-j})]_{i,j\in[n]\times[n]}$ and

$$\mathbf{U} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{pmatrix}_{n\times n}.$$

Notice that, in this context, $H = \mathbb{R}$ because K is a totally real number field. As an immediate consequence of Theorem 4.2, in Corollary 4.3.1 we prove that the construction for the $\mathbb{Z}^n$-lattice mentioned above, in fact, does not change the shape of the error distribution and, more importantly, the standard deviation is the same when the distribution is seen over H.

Corollary 4.3.1. *Let $K = \mathbb{Q}(\zeta_p + \zeta_p^{-1})$ for $p \geq 5$ prime and let $v \in \mathcal{O}_K$ be a random variable distributed as ψ_s^n in the basis $\mathcal{C}'$. Then, the distribution of $(\mathbf{T}^{-1} \circ \sigma_\tau)(v)$ for $\tau = \frac{1}{p}(1-\zeta_p)(1-\zeta_p^{-1})$, seen in the canonical basis of H, is the spherical Gaussian ψ_s^n.*

Proof. In the realization of the $\mathbb{Z}^n$-lattice (Proposition 4.3), the matrix representing the linear map σ_τ from the basis $\mathcal{C}'$ of $\mathcal{O}_K$ to the canonical basis of $\mathbb{R}^n$ is given by $\mathbf{M}$ (4.2). Since $\mathcal{O}_K$ is a lattice equivalent to $\mathbb{Z}^n$ in the basis $\mathcal{C}'$, the result follows immediately from Theorem 4.2. This concludes the proof. □

4.2 Connecting Twisted Ring-LWE Instances

In this section, we relate distinct number fields by applying the transformation used to reduce an instance of the Ring-LWE Problem to the Twisted Ring-LWE. The basic idea is as follows. If two fractional ideals, each on a distinct number field, lead to the same lattice in H under twisted embeddings, they are equivalent lattices. So, there is a unimodular matrix related to their generator matrices, which we use to convert elements between the two fractional ideals.

Again, we take the case of lattices equivalent to $\mathbb{Z}^n$ as an example. It is known that the rings of integers of three distinct number fields leads to rotated versions of

the $\mathbb{Z}^n$-lattice under twisted embeddings. In particular, we could assume that K is a power-of-two cyclotomic number field, L is the maximal real subfield of a power-of-two cyclotomic number field, and M is the maximal real subfield of a p-th cyclotomic number field. However, the value of n only equals a power of two in the three cases when $p = 5$. Recall that, for the number field M, $n = (p-1)/2$. Because of that, in Figure 4.1, we only consider the number fields K and L as defined above. For K, the twist factor is $\tau_1 = 1$, corresponding to the canonical embedding. For the maximal real subfield L, $\tau_2 = 2-(\zeta_{2^r}+\zeta_{2^r}^{-1})$, in which $r \geq 3$ and $n = [L:\mathbb{Q}] = 2^{r-2}$ [16, Section 3.1]. In both cases, the construction of the rotated $\mathbb{Z}^n$-lattice is done from the respective ring of integers.

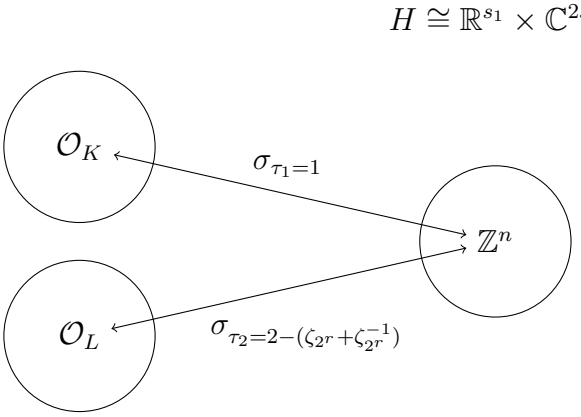

Figure 4.1: Distinct number fields that lead to a rotation of the $\mathbb{Z}^n$-lattice.

The generator matrix of the $\mathbb{Z}^n$-lattice built from K is $\mathbf{M}_1 = [\sigma_i(\zeta_m^j)]_{i,j\in[n]}$ for $m = 2^\ell$, $\ell \in \mathbb{Z}$, and $n = m/2$, which is given in terms of the integral basis $\{\zeta_m^j\}$ for $j \in [n]$, the power basis of $\mathcal{O}_K$. The Gram matrix of $\mathbf{M}_1$ is $\mathbf{G}_1 = \mathbf{M}_1^\top \mathbf{M}_1 = n\,\mathbf{I}_n$, which is equivalent to

$$\mathrm{Tr}(x\overline{x}) = \sum_{i=1}^{n} \sigma_i(\zeta_m^j)\overline{\sigma_i(\zeta_m^k)} = \sum_{i=1}^{n} \zeta_m^{ij}\overline{\zeta_m^{ik}} = \sum_{i=1}^{n} (\zeta_m^{j-k})^i = \begin{cases} n & \text{if } j = k, \\ 0 & \text{otherwise.} \end{cases}$$

On the other hand, the generator matrix of the $\mathbb{Z}^n$-lattice associated with the ring of integers of L is given by

$$\mathbf{M}_2 = \frac{1}{\sqrt{2^{r-1}}}\mathbf{D}\mathbf{M}'\mathbf{U},$$

where $\mathbf{D} = \mathrm{diag}\left[\sqrt{\sigma_i(2-(\zeta_{2^r}+\zeta_{2^r}^{-1}))}\right]_{n\times n}$, $\mathbf{M}' = \left[\sigma_i(\zeta_{2^r}^j + \zeta_{2^r}^{-j})\right]_{i,j\in[n]\times[n]}$, and

$$\mathbf{U} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{pmatrix}_{n\times n}.$$

In this case, an integral basis of $\mathcal{O}_L$ is given by $e_1 = 1$ and $e_i = \zeta_{2^r}^i + \zeta_{2^r}^{-i}$ for $2 \leq i \leq n$. The construction of the $\mathbb{Z}^n$-lattice is done in the basis $\{e_i'\}_{i\in[n]}$ in which $e_i' = \sum_{j=1}^{i} e_j$ for

$i \in [n]$. Since both $\sigma_{\tau_1}(\mathcal{O}_K)$ and $\sigma_{\tau_2}(\mathcal{O}_L)$ are rotated versions of the $\mathbb{Z}^n$-lattice, they are also equivalent to each other. In this sense, there exists a unimodular matrix $\mathbf{U}'$ such that $\mathbf{M}_1 = \mathbf{U}'\mathbf{M}_2$. Since the vectors in $\mathbf{M}_2$ are linearly independent, $\det(\mathbf{M}_2) \neq 0$ and $\mathbf{M}_2$ is invertible. Thus, the matrix $\mathbf{U}'$ can be determined by computing

$$\mathbf{M}_1\mathbf{M}_2^{-1} = \mathbf{U}'\mathbf{M}_2\mathbf{M}_2^{-1} = \mathbf{U}'.$$

Notice that the matrix $\mathbf{M}_2^{-1}$ can be precomputed, assuming that the values for m and r are fixed such that $[K : \mathbb{Q}] = [L : \mathbb{Q}]$. In this sense, the conversion of elements between $\mathcal{O}_K$ and $\mathcal{O}_L$ can be performed efficiently in $O(n^2)$ operations.

We formalize this transformation in Theorem 4.4 and use it to convert Twisted Ring-LWE instances defined over two distinct number fields. The particular case of power-of-two cyclotomic number fields and their maximal real subfields is addressed in Proposition 4.5.

Theorem 4.4. *Consider any two Twisted Ring-LWE instances over distinct fractional ideals $\mathcal{I} \subseteq \mathcal{O}_K$ and $\mathcal{J} \subseteq \mathcal{O}_L$. If the lattices $\sigma_{\tau_1}(\mathcal{I})$ and $\sigma_{\tau_2}(\mathcal{J})$ are equivalent in H for totally positive elements $\tau_1 \in K$ and $\tau_2 \in L$, then a Twisted Ring-LWE instance over $\mathcal{I}$ can be efficiently converted into an instance over $\mathcal{J}$.*

Proof. If two lattices $\sigma_\tau(\mathcal{I})$ and $\sigma_\nu(\mathcal{J})$ are equivalent, then one can be obtained from the other through a rotation, a reflection, or a change of scale as follows. Let $\mathbf{M}_1$ and $\mathbf{M}_2$ be the generator matrices of $\sigma_{\tau_1}(\mathcal{I})$ and $\sigma_{\tau_2}(\mathcal{J})$, respectively. There exists a unimodular matrix $\mathbf{U}$ such that $\mathbf{M}_1 = \mathbf{U}\mathbf{M}_2$ which can be computed as $\mathbf{M}_1\mathbf{M}_2^{-1}$. The existence of $\mathbf{M}_2^{-1}$ is guaranteed since $\mathbf{M}_2$ is a basis. Given a set of samples from the Twisted Ring-LWE distribution $\mathcal{A}_{s,\tau_1}$ as

$$(a_i, b_i = a_i \cdot s + e_i) \in R_q \times K_{\mathbb{R}}/R^\vee,$$

where $e_i \xleftarrow{\psi_{\tau_1}} K_{\mathbb{R}}$, their representatives in H under the twisted embedding σ_{τ_1} are computed by performing the matrix multiplications $\mathbf{M}_1^\top \cdot \mathbf{a}$ and $\mathbf{M}_1^\top \cdot \mathbf{b}$, in which $\mathbf{a}$ and $\mathbf{b}$ are the coefficient vectors of a_i and b_i in the power basis of K. The corresponding set of samples from the Twisted Ring-LWE distribution $\mathcal{A}_{s,\psi_{\tau_2}}$ is obtained by computing the change of basis given by $\mathbf{U}$, which is done by left multiplying the coefficient vectors by $\mathbf{U}^\top$. Notice that

$$\mathbf{M}_1^\top \cdot \mathbf{a} = (\mathbf{U}\mathbf{M}_2)^\top \cdot \mathbf{a} = \mathbf{M}_2^\top\mathbf{U}^\top \cdot \mathbf{a}.$$

In this sense, the coefficient vector of an element $a \in K_{\mathbb{R}}$ in the field $L_{\mathbb{R}} = L \otimes_{\mathbb{Q}} \mathbb{R}$ is given by $\mathbf{U}^\top \cdot \mathbf{a}$. Since a matrix multiplication can be performed in $O(n^2)$ operations, the above conversion of Twisted Ring-LWE instances is efficiently computable. This concludes the proof. □

Proposition 4.5. *Consider the number fields $K = \mathbb{Q}(\zeta_m)$ with $m \geq 4$ a power of two, $n = m/2$ and ring of integers $\mathcal{O}_K = \mathbb{Z}[\zeta_m]$, and $L = \mathbb{Q}(\zeta_{2^r} + \zeta_{2^r}^{-1})$ with $r \geq 3$, $n = 2^{r-2}$, and ring of integers $\mathcal{O}_L = \mathbb{Z}[\zeta_{2^r} + \zeta_{2^r}^{-1}]$. A Twisted Ring-LWE instance over $\mathcal{O}_K$ can be efficiently converted into an instance over $\mathcal{O}_L$, and vice-versa.*

Proof. The ring of integers $\mathcal{O}_K$ in the power basis $\{\zeta_m^i\}_{i\in[n]}$ leads to a construction of the $\mathbb{Z}^n$-lattice in the canonical embedding [27, Proposition 2.8]. Recall that the canonical

embedding corresponds to the twisted embedding σ_{τ_1} when $\tau_1 = 1$. Similarly, the ring of integers $\mathcal{O}_L$ leads to a construction of the $\mathbb{Z}^n$-lattice in the basis $\{e'_i\}_{i\in[n]}$ under the twisted embedding σ_{τ_2} with $\tau_2 = 2 - (\zeta_{2^r} + \zeta_{2^r}^{-1})$. The basis $\{e'_i\}_{i\in[n]}$ is $e'_i = \sum_{j=1}^{i} e_j$ for $i \in [n]$, where $\{e_i\}_{i\in[n]}$ is an integral basis given by $e_1 = 1$ and $e_i = \zeta_{2^r}^{i} + \zeta_{2^r}^{-i}$ for $2 \leq i \leq n$ [16, Proposition 4]. Since both $\sigma_{\tau_1}(\mathcal{O}_K)$ and $\sigma_{\tau_2}(\mathcal{O}_L)$ are equivalent to the $\mathbb{Z}^n$-lattice, they are also equivalent to each other. Thus, by Theorem 4.4, a Twisted Ring-LWE instance over $\mathcal{O}_K$ can be efficiently converted into a Twisted Ring-LWE instance over $\mathcal{O}_L$, and vice-versa. □

Notice that, in comparison with the notation used in Section 3.3, we restrict the transformation used in both Theorem 4.4 and Proposition 4.5 to the coefficient vector representation. Consequently, we do not compute the integral basis of the number fields or explicitly perform the transformations σ and σ_τ^{-1}. This simplification results from the related generating matrices being equivalent. The coefficient representation is used in the cyclotomic toolkit of Lyubashevsky, Peikert, and Regev [89] and Chapter 5.

An immediate consequence of Proposition 4.5 is that instantiating Ring-LWE using power-of-two cyclotomic number fields may be equivalent to considering only its maximal real subfield. The equivalence of these two number fields can be used for assessing the hardness of widely used ring instantiations for cryptography in future works. Recall that power-of-two number fields have being used in cryptographic applications ranging from public-key schemes [31, 12, 60] and digital signature schemes [10, 63, 69] to more complex systems as fully homomorphic encryption [45]. In this direction, algebraic properties of maximal real subfields could be explored for cryptoanalysis of power-of-two number fields in the light of previous efforts [53, 54, 56].

Analogously to Proposition 4.5, similar results can be obtained for other classes of algebraic lattices such as D_n-lattices [33, 79, 78, 61]. In coding theory, algebraic lattice constructions have been used to obtain lattices with good properties for Gaussian and Rayleigh fading channels. In this sense, by fixing the target lattice in $\mathbb{R}^n$, the equivalence of distinct algebraic constructions can be used to provide evidence of the hardness of ring instantiations for cryptographic use.

Chapter 5

The Twisted Ring-LWE on a Public-Key Cryptosystem

In this chapter, we adapt the compact Public-Key Encryption scheme (PKE) of Lyubashevsky, Peikert, and Regev [89, Section 8.2] to deal with twisted embeddings. Originally, this public-key cryptosystem was designed for general cyclotomic number fields in the canonical embedding.

In this context, we are interested in algebraic lattices equivalent to $\mathbb{Z}^n$, which are known to be constructed from power-of-two cyclotomic number fields [27] and their maximal real subfields [16], and from the maximal real subfields of p-th cyclotomic number fields for any prime $p \geq 5$ [26]. Recall from Chapter 4 that lattices equivalent to $\mathbb{Z}^n$ lead to efficient sampling from spherical Gaussian distributions. In this sense, our motivation to provide a practical evaluation of twisted embeddings on maximal real subfields of p-th cyclotomic number fields is two-fold. First, power-of-two cyclotomic number fields are already widely adopted in practical protocols, e.g. [60, 31, 63, 10, 12]. Thus, enabling an alternative ring instantiation for the Ring-LWE Problem increases the existing scope of algebraic lattices that can be considered for cryptographic applications. Second, when p is a prime, the ring dimension is $n = (p-1)/2$. Considering that the dimension does not increase as a power of two, one may want to find a ring instantiation to better achieve a target security level. For example, to obtain a ring dimension between 700 and 800, the required for achieving 128-bit security [90], possible choices for the value of p range from the 223-th to the 252-th prime numbers, comprehending twenty-nine possible choices.

Notations. In the following sections, the symbol $\lfloor \cdot \rceil$ denotes a valid discretization to cosets of $R^\vee$ or $p'R^\vee$. For any $\overline{a} \in \mathbb{Z}_q$, let $[\![\overline{a}]\!]$ denote the unique representative $a \in (\overline{a} + q\mathbb{Z}) \cap [-q/2, q/2)$, that is, a is the representative of $\overline{a}$ in the balanced representation of $\mathbb{Z}_q$, which is entry-wise extended to vectors and polynomials. Finally, the notation $x \xleftarrow{\psi_\tau} K_{\mathbb{R}}$ indicates that x is sampled according to the error distribution ψ_τ over the field $K_{\mathbb{R}}$.

5.1 The Public-Key Cryptosystem

Consider the number field $K = \mathbb{Q}(\zeta_p + \zeta_p^{-1})$ for $p \geq 5$ a prime with $\zeta = \zeta_p = e^{-2i\pi/p}$ a primitive p-th root of unity in $\mathbb{C}$, and $n = (p-1)/2$. We denote the minimal polynomial of $\zeta_p + \zeta_p^{-1}$ as $\Psi_p(x)$. The ring of integers of K is $\mathcal{O}_K = \mathbb{Z}[\zeta_p + \zeta_p^{-1}]$ and its dual is $\mathcal{O}_K^\vee = \langle t^{-1} \rangle$, in which $t = \Psi_p'(\zeta_p + \zeta_p^{-1})$ [122].

Our modified public-key cryptosystem is $\mathsf{PKE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ as defined in Algorithms 1, 2, and 3. It is parameterized by the ring of integers of K, which is denoted by $R = \mathcal{O}_K$, and two coprime integers p' and q. The message space is defined as $R_{p'}$, and q is the Twisted Ring-LWE modulus. Consider that ψ_τ is a spherical Gaussian distribution over $(K_\mathbb{R}, \langle , \rangle_\tau)$ for a totally positive element $\tau = \frac{1}{p}(1-\zeta_p)(1-\zeta_p^{-1}) \in K$.

Algorithm 1: PKE.KGen: Key pair generation algorithm

Output: A public key $\mathsf{pk} = (a, b) \in R_q \times R_q$ and a secret key $\mathsf{sk} = x \in R^\vee$

$a \xleftarrow{\$} R_q$
$x \xleftarrow{\psi_\tau} K_\mathbb{R}$
$x = \lfloor x \rceil_{R^\vee}$
$e \xleftarrow{\psi_\tau} K_\mathbb{R}$
$e = \lfloor p' \cdot e \rceil_{p'R^\vee}$
$b = a \cdot x + e \mod qR$
return $(\mathsf{pk} = (a, b), \mathsf{sk} = x)$

Algorithm 2: PKE.Enc: Encryption algorithm

Input: A message $\mu \in R_{p'}$ and the public key $\mathsf{pk} = (a, b) \in R_q \times R_q$

Output: The ciphertext $\mathsf{ct} = (u, v) \in R_q \times R_q^\vee$

$z \xleftarrow{\psi_\tau} K_\mathbb{R}$
$z = \lfloor z \rceil_{R^\vee}$
$e' \xleftarrow{\psi_\tau} K_\mathbb{R}$
$e' = \lfloor p' \cdot e' \rceil_{p'R^\vee}$
$e'' \xleftarrow{\psi_\tau} K_\mathbb{R}$
$e'' = \lfloor p' \cdot e'' \rceil_{t^{-1}\mu + p'R^\vee}$
$u = a \cdot z + e' \mod qR$
$v = z \cdot b + e'' \in R_q^\vee$
return $ct = (u, v)$

Algorithm 3: PKE.Dec: Decryption algorithm

Input: A ciphertext $\mathsf{ct} = (u, v) \in R_q \times R_q^\vee$ and the secret key $\mathsf{sk} = x \in R^\vee$

Output: The message $\mu \in R_{p'}$

$d = v - u \cdot x \mod qR^\vee$
$d = \mathsf{Decode}(\llbracket d \rrbracket) \in R^\vee$
$\mu = t \cdot d \mod p'R$
return μ

In comparison with the original public-key cryptosystem, we define the error distribution in PKE as a spherical Gaussian distribution under twisted embeddings, denoted ψ_τ, in which $\tau \in K$ is the torsion factor. Recall that, when $\psi = 1$, the corresponding twisted embedding collapses to the canonical embedding. Also, we removed the scaling factor $\hat{m}$ from both PKE.KGen and PKE.Dec algorithms. For $R = \mathbb{Z}[\zeta_m]$, $t = \hat{m}/g \in R$, where $\hat{m} = m/2$ if m is even, otherwise $\hat{m} = m$, in which m is the cyclotomic index and

$g = \prod_p (1 - \zeta_p) \in R$ [89, Definition 2.17]. Thus, the use of $\hat{m}$ is particular to the case of ring of integers of cyclotomic number fields. In our case, $\hat{m} = 1$; thus, we suppressed this parameter from our cryptosystem.

In particular, we are interested in efficiently performing the following algorithmic operations when K is the maximal real subfield of a cyclotomic number field.

- Uniform sampling on R_q.
- Sampling from the spherical Gaussian distribution ψ_τ.
- Discretizing error samples to cosets of $R^\vee$ or $R_{p'}^\vee$.
- Multiplying in R_q, $R_q^\vee$, and $R_{p'}$.
- Decoding noise terms in the decryption algorithm.

Uniform sampling. Uniform sampling is required by the PKE.KGen algorithm for generating the public element $a \in R_q$. Uniformly random integers modulo q can be obtained by rejection sampling on a uniformly-random byte array, as performed by the key-encapsulation mechanisms CRYSTALS-Kyber [31] and Saber [60].

Sampling from the spherical Gaussian distribution. Since R is an algebraic lattice equivalent to $\mathbb{Z}^n$ under twisted embeddings, we can sample from a spherical Gaussian distribution directly on both $K_\mathbb{R}$ and H in an orthonormal basis with security hardness guarantee.

The following sections discuss how to evaluate the remaining algorithmic operations using polynomial and coefficient vector representations. First, in Section 5.2, we analyze the use of the polynomial representation, which is widely adopted in the literature related to the Ring-LWE Problem. Then, we discuss the format and the expansion factor of the defining polynomial $\Psi_p(x)$, which helps in determining the size of the integer modulus q. Finally, we conclude that the exponential growth of the polynomial coefficients after a reduction modulo $\Psi_p(x)$ turns the polynomial representation impractical for cryptographic use. As a consequence, in Section 5.3, we investigate the coefficient representation, as adopted by Lyubashevsky et al. [89]. The coefficient representation generalizes the polynomial representation, allowing the replacement of the power basis. As a result, we use an orthonormal basis for all algorithmic operations in the PKE, not restricting to the spherical Gaussian sampling operation.

5.2 The Polynomial Representation

In Ring-LWE cryptosystems over power-of-two cyclotomic number fields [60, 31, 63, 10, 12, 88], arithmetic operations such as addition and multiplication are performed in the polynomial representation of the ring of integers. The ring of integers of the maximal real

subfield $\mathbb{Q}(\zeta_p + \zeta_p^{-1})$ is $\mathbb{Z}[\zeta_p + \zeta_p^{-1}]$. Thus, associating $\zeta_p + \zeta_p^{-1}$ with indeterminate x yields an isomorphism between $\mathbb{Z}[\zeta_p + \zeta_p^{-1}]$ and $\mathbb{Z}[x]/(\Psi_p(x))$, in which $\Psi_p(x)$ is the minimal polynomial of $\zeta_p + \zeta_p^{-1}$. Considering that sampling from a spherical Gaussian distribution is done in an orthonormal basis $\mathcal{C}'$, performing arithmetic operations in the polynomial representation requires a change of basis from $\mathcal{C}'$ to the power basis $\{(\zeta_p + \zeta_p^{-1})^j : 0 \leq j < n\}$ for $n = (p-1)/2$.

The coefficients of the defining polynomial $\Psi_p(x)$ vary according to the choice of p. Aranés and Arenas provided a closed formula for the coefficients of $\Psi_{p^\upsilon}(x)$:

$$\Psi_{p^\upsilon}(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1} + a_n x^n,$$

for p prime, $\upsilon \geq 1$, $a_n = 1$, and $n = \frac{1}{2}\varphi(p^\upsilon) = p^{\upsilon-1}(p-1)/2$, which is given in Theorem 5.2. First, consider that, for strictly positives r and k, $A_r(k)$ are the determinants of order k, defined in Theorem 5.1. Details can be found in [17].

Theorem 5.1 ([17, Theorem 1]). *For any strictly positive integers r and k, we have that*

$$A_r(k) = \binom{r+k-2}{k} + \binom{r+k-3}{k-1},$$

in which $\binom{r}{k}$ denotes the binomial coefficient $\frac{r!}{k!(r-k)!}$.

Theorem 5.2 ([17, Theorem 2]). *The coefficients a_j of the polynomial $\Psi_{p^\upsilon}(x)$ are given as follows. If p is odd,*

$$a_j = \begin{cases} 0, & \text{if } j > n - p^{\upsilon-1}; \\ \sum\limits_{\substack{k=1 \\ k \equiv 1 \pmod 2}}^{\left\lfloor \frac{n-j}{p^{\upsilon-1}} \right\rfloor} (-1)^{(n-j-kp^{\upsilon-1})/2} A_{j+2}\left(\frac{n-j-kp^{\upsilon-1}}{2}\right), & \text{if } n+j \equiv 1 \pmod 2; \\ (-1)^{\frac{n-j}{2}} \sum\limits_{k=0}^{\left\lfloor \frac{n-j}{2p^{\upsilon-1}} \right\rfloor} (-1)^k A_{j+2}\left(\frac{n-j}{2} - kp^{\upsilon-1}\right), & \text{if } n+j \equiv 0 \pmod 2; \end{cases}$$

and in the case $p = 2$ and $\upsilon \geq 3$:

$$a_j = \begin{cases} (-1)^{\frac{n-j}{2}} A_{j+2}\left(\frac{n-j}{2}\right), & \text{if } j \text{ is even}; \\ 0, & \text{otherwise}. \end{cases}$$

Notice that, in our case, $\upsilon = 1$; thus, all coefficients are always non-zero. For example, when $p = 31$, we have that $n = 15$ and the defining polynomial $\Psi_p(x)$ is

$$\begin{aligned} \Psi_{31}(x) = x^{15} + x^{14} - 14x^{13} - 13x^{12} + 78x^{11} + 66x^{10} - 220x^9 - 165x^8 \\ + 330x^7 + 210x^6 - 252x^5 - 126x^4 + 84x^3 + 28x^2 - 8x - 1, \end{aligned}$$

which is very dense, and the coefficients are not restricted to the set $\{0, 1\}$. However, depending on the choice of value for the coefficient's modulus q, the defining polynomial may have a complete factorization modulo q, which allows algorithms based on the Chinese Remainder Theorem (CRT) for efficient polynomial multiplication. For example, for

$p = 31$ and $q = 61$, the defining polynomial factors in 15 distinct degree-one polynomials as follows:

$$\Psi_{31}(x) \mod 61 = (x+5)(x+6)(x+15)(x+16)(x+21)(x+22)(x+24)(x+27)$$
$$(x+29)(x+36)(x+38)(x+41)(x+48)(x+49)(x+51).$$

Thus, $f(x) = \Psi_p(x)$ can be factored as $f(x) = \prod_{i=1}^{k} f_i(x) \pmod q$, in which $f_i(x)$ are polynomials of small degree. The multiplication $a(x) \cdot b(x)$ modulo $f(x)$ is done by computing $a_i(x) = a(x) \mod f_i(x)$ and $b_i(x) = b(x) \mod f_i(x)$ for $1 \leq i \leq k$, computing the component-wise multiplication $a_i(x) \cdot b_i(x)$ and, finally, using the inverse operation to obtain the polynomial $c(x)$ such that $c(x) \mod f_i(x) = a_i(x)b_i(x) \mod f_i(x)$, as discussed by Lyubashevsky and Seiler [90]. Although the asymptotic cost of an algorithm based on this technique is $O(n \log n)$, the hidden constants may be large due to the increased number of reductions modulo q in comparison with CRT-based algorithms for power-of-two cyclotomic number fields [90, 49].

Another important aspect of the defining polynomial is captured by the expansion factor, a property introduced by Lyubashevsky and Micciancio [87]. The *expansion factor* of a polynomial $f(x)$ is

$$\mathrm{EF}(f(x), k) = \max_{g(x) \in \mathbb{Z}[x], \deg(g(x)) \leq k(\deg(f(x))-1)} \|g(x)\|_{f(x)} / \|g(x)\|_\infty,$$

in which $\|g(x)\|_f$ is the norm of the polynomial $g(x)$ after reduction modulo $f(x)$. In this context, we compute the norm of a polynomial by taking its coefficient vector, that is, $\|f(x)\| := \|\mathbf{f}\|$ in which $f(x) = \sum_{i=0}^{n-1} f_i x^i \in R$ and $\mathbf{f} = [f_0, \ldots, f_{n-1}] \in \mathbb{Z}^n$. By computing the expansion factor of $\Psi_p(x)$, we can measure the increase in magnitude of the maximum coefficient of $\|g(x)\|_{\Psi_p(x)}$. Also, the expansion factor helps us in choosing a value for q such that the coefficients do not wrap around after arithmetic operations, avoiding the occurrence of decryption errors.

We analyze the expansion factor of $\Psi_p(x)$ by comparing it with $x^n + 1$, the defining polynomial of cyclotomic polynomial rings with dimension a power of two, which is widely adopted in practical applications. For that, we recall Lemma 5.3, which defines an upper bound for the size of the coefficients of a polynomial $g(x) \in \mathbb{Z}[x]$ after a reduction modulo $f(x)$.

Lemma 5.3. *If $g(x)$ is a polynomial in $\mathbb{Z}[x]$ and $f(x)$ is a monic polynomial in $\mathbb{Z}[x]$ such that $\deg(g(x)) \geq \deg(f(x))$, then $\|g(x)\|_{f(x)} \leq \|g(x)\|_\infty (2\|f(x)\|_\infty)^{\deg(g(x))-\deg(f(x))+1}$.*

For the case $f(x) = \Psi_p(x)$, it is sufficient to analyze the value of $\|f(x)\|_\infty$. First, for $f(x) = x^n + 1$, we have that $\|f(x)\|_\infty = 1$. On the other hand, when $f(x) = \Psi_p(x)$, $\|f(x)\|_\infty$ takes the maximum value of a_j according to Theorem 5.2. For example, for $p = 31$, $\|f(x)\|_\infty = 330$, leading to an exponential growth of coefficients, which is roughly $330^{\deg(g(x))-\deg(f(x))+1}$ times bigger with respect to the case when $f(x) = x^{16} + 1$. Such growth of coefficients requires an increased value for the choice of the modulus q to avoid the coefficients wrapping around after polynomial operations. However, this also leads to an increase in the length of system parameters and consequently memory and bandwidth

requirements for transmission of public parameters.

Unfortunately, the exponential growth of the expansion factor makes the polynomial representation of limited interest since coefficients' modulus may become very large to avoid the occurrence of decryption errors. Because of that, in the next section, we evaluate performing the PKE's algorithmic tasks using the coefficient vector representation, as previously done by Lyubashevsky, Peikert, and Regev in the toolkit for arbitrary cyclotomic rings [89].

5.3 The Coefficient Vector Representation

In this section, we evaluate performing the algorithmic tasks of PKE, described in Algorithms 1, 2, and 3, using coefficient vectors as representatives of number field elements.

Commonly, elements in R are represented in the power basis $\{(\zeta_p+\zeta_p^{-1})^j : 0 \leq j < n\}$ or, equivalently, as residue polynomials in $\mathbb{Z}[x]/(\Psi_p(x))$ by associating $\zeta_p + \zeta_p^{-1}$ with x. However, ring elements can also be represented by vectors of integer coefficients in any $\mathbb{Z}$-basis $\mathfrak{B} \subset R$. For an element $a \in R$, it follows that

$$a = a_1 b_1 + a_2 b_2 + \ldots + a_n b_n,$$

in which $a_i \in \mathbb{Z}$ for $1 \leq i \leq n$ and $\mathfrak{B} = \{b_1, \ldots, b_n\} \in R$. In other words, $a \in R$ is represented by the inner product

$$a = \langle \mathfrak{B}, \mathbf{a} \rangle = \mathbf{M}^\top \cdot \mathbf{a},$$

in which $\mathbf{a} = [a_1, \ldots, a_n] \in \mathbb{Z}^n$ and $\mathbf{M}$ is a generator matrix.

When K is the maximal real subfield of a p-th cyclotomic number field for $p \geq 5$, an orthonormal basis is $\mathfrak{B} = \mathcal{C}' = \{\zeta_p^j + \zeta_p^{-j}\}_{j=1}^n$ and the generator matrix of $\sigma_\tau(R)$ in the space H is

$$\mathbf{M} = \mathbf{D}\mathbf{M}'\mathbf{U},$$

in which $\mathbf{D} = \operatorname{diag}\left[\sqrt{\frac{\sigma_i(\tau)}{p}}\right]_{n\times n}$, $\mathbf{M}' = [\sigma_i(\zeta^j + \zeta^{-j})]_{i,j\in[n]\times[n]}$, and

$$\mathbf{U} = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 & 0 \\ 1 & 1 & 0 & \ldots & 0 & 0 \\ 1 & 1 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & 1 & \ldots & 1 & 1 \end{pmatrix}.$$

Computing with the matrix $\mathbf{M}$ requires floating-point representations and matrix-vector multiplication, for which a naïve algorithm has a quadratic cost. Because of that, we can choose to keep the coefficient vector representation in the space H, avoiding the computation of $\mathbf{M}^{-1}$. For example, consider the n field embeddings from K into $\mathbb{R}$:

$$\sigma_k(\zeta^j + \zeta^{-j}) = \zeta^{kj} + \zeta^{-kj} = 2\cos\left(\frac{2\pi kj}{p}\right).$$

For $p = 7$, the generator matrix $\mathbf{M}$ is

$$\mathbf{M} = \begin{pmatrix} -0.3279852776056815 & -0.7369762290995779 & -0.5910090485061034 \\ -0.591009048506103 & -0.32798527760568197 & 0.736976229099578 \\ -0.7369762290995778 & 0.5910090485061037 & -0.3279852776056813 \end{pmatrix}.$$

Notice that symmetries on the matrix $\mathbf{M}$ can be explored to reduce the quadratic cost of multiplying $\mathbf{M}$ by some vector. In our case, we fix the representation basis of R as $\sigma_\tau(\mathcal{C}')$ for all PKE's operations and the computations are done on coefficient vectors. Thus, we avoid changing the representation between various bases and converting elements between the space H and the field $K_\mathbb{R}$.

In the following, we present Lemma 5.4 before defining the basis for the dual ring $R^\vee$. In this context, consider that R is a Dedekind domain, K is its field of quotients, and $L|K$ is a separable extension of degree n. Moreover, notice that the ring of integers of a number field is a Dedekind domain [122].

Lemma 5.4 ([122, Chapter 13, J.]). *Let $L = K(\theta)$, where θ is integral over R and*

$$f(x) = b_0 + b_1 x + \cdots + b_{n-2}x^{n-2} + x^{n-1} \in R[x]$$

is the minimal polynomial of θ over K, and let $f'(x)$ denote its derivative. Then,

1. $\mathrm{Tr}_{L|K}\left(\frac{\theta^i}{f'(\theta)}\right) = 0$ *when* $0 \leq i \leq n-3$ *and* $\mathrm{Tr}_{L|K}\left(\frac{\theta^{n-2}}{f'(\theta)}\right) = 1$;
2. $R[\theta]^\vee = \frac{1}{f'(\theta)}R[\theta]$.

From Lemma 5.4, by taking $R = \mathcal{O}_K$, we have that the elements in $R^\vee$ can be represented in the basis $\mathcal{C}^\star$:

$$\mathcal{C}^\star = \Psi'_p(\zeta_p + \zeta_p^{-1})^{-1} \cdot \mathcal{C}',$$

in which $\Psi'_p(x)$ is the derivative of the minimal polynomial of $\zeta_p + \zeta_p^{-1}$ and $\mathcal{C}'$ is a $\mathbb{Z}$-basis for R [122].

Definition 22. *For $R = \mathbb{Z}[\zeta_p + \zeta_p^{-1}]$, define $g = t^{-1} \in K$, where $t = \Psi'_p(\zeta_p + \zeta_p^{-1}) \in R$.*

Consider that $\Psi_p(x) = b_0 + b_1 x + \cdots + b_{n-1}x^{n-1} + x^n$ with $n = (p-1)/2$. The derivative of $\Psi_p(x)$, denoted $\Psi'_p(x)$, is $\Psi'_p(x) = b_1 + 2b_2 x + 3b_3 x^2 + \cdots + (n-1)b_{n-1}x^{n-2} + nx^{n-1}$. We can write $\Psi'(\zeta_p + \zeta_p^{-1})$ in the power basis of R as

$$b_1 + 2b_2(\zeta_p + \zeta_p^{-1}) + 3b_3(\zeta_p + \zeta_p^{-1})^2 + \cdots + (n-1)b_{n-1}(\zeta_p + \zeta_p^{-1})^{n-2} + n(\zeta_p + \zeta_p^{-1})^{n-1},$$

and, thus, $t \in R$.

Notice that, in PKE, some system parameters such as the public key pk, the message μ, and the ciphertext ct all lie on a quotient ring. For that, we prove that a $\mathbb{Z}$-basis for a nonzero ideal $\mathcal{I} \subseteq \mathcal{O}_K$ is also a $\mathbb{Z}_q$-basis for the quotient ideal $\mathcal{I}/q\mathcal{I}$, for $q \geq 2$ an integer.

Proposition 5.5. *Let $\mathfrak{B}$ be a $\mathbb{Z}$-basis for a nonzero ideal $\mathcal{I} \subseteq \mathcal{O}_K$. Then, for an integer $q \geq 2$, $\mathfrak{B}' = \{\mathbf{b}'_1, \ldots, \mathbf{b}'_n\}$ is a $\mathbb{Z}_q$-basis for the quotient $\mathcal{I}/q\mathcal{I}$, in which $\mathbf{b}'_i = \mathbf{b}_i + q\mathcal{I}$.*

Proof. We first prove the linear independence of $\mathfrak{B}'$. Consider that $\sum_{i=1}^{n} a_i \mathbf{b}'_i = q\mathcal{I}$ in $\mathcal{I}/q\mathcal{I}$ with $a_i \in \mathbb{Z}$. It follows that,

$$\begin{aligned} &\sum_{i=1}^{n} a_i(\mathbf{b}_i + q\mathcal{I}) = q\mathcal{I} \\ \Rightarrow &\sum_{i=1}^{n} a_i \mathbf{b}_i + \sum_{i=1}^{n} a_i q\mathcal{I} = q\mathcal{I} \\ \Rightarrow &\sum_{i=1}^{n} a_i \mathbf{b}_i = q\mathcal{I} - \sum_{i=1}^{n} a_i q\mathcal{I} = q\mathcal{I}. \end{aligned}$$

Thus, we have that $\sum_{i=1}^{n} a_i \mathbf{b}_i = \sum_{i=1}^{n} q c_i \mathbf{b}_i$ for $c_i \in \mathcal{I}$ and $a_i = q c_i$, $\forall\, i$. Then, $a_i \equiv 0 \pmod{q}$, and finally $a_i = 0 \in \mathbb{Z}_q$, $\forall\, i$. Now, let $v \in \mathcal{I}$. It follows that,

$$\begin{aligned} v &= \sum_{i=1}^{n} a_i \mathbf{b}_i \qquad a_i \in \mathbb{Z} \\ \Leftrightarrow v &= \sum_{i=1}^{n} a_i(\mathbf{b}'_i - q\mathcal{I}) \\ \Leftrightarrow v &= \sum_{i=1}^{n} a_i \mathbf{b}'_i - \sum_{i=1}^{n} a_i q\mathcal{I} \\ \Leftrightarrow v &= \sum_{i=1}^{n} a_i \mathbf{b}'_i - \sum_{i=1}^{n} a_i q\mathcal{I} \\ \Leftrightarrow v &= \sum_{i=1}^{n} a_i \mathbf{b}'_i \qquad \in \mathcal{I}/q\mathcal{I}. \end{aligned}$$

Since the vectors in $\mathfrak{B}'$ are linearly independent and $\mathfrak{B}'$ spans $\mathcal{O}_K/q\mathcal{I}$, $\mathfrak{B}'$ is a basis for $\mathcal{I}/q\mathcal{I}$. This concludes the proof. □

By Proposition 5.5, since $\mathcal{C}' \subset R$ and $\mathcal{C}^\star \subset R^\vee$ are $\mathbb{Z}$-basis for R and $R^\vee$, respectively, then $\mathcal{C}'$ and $\mathcal{C}^\star$ are also $\mathbb{Z}_q$-basis for the quotients $R_q = R/qR$ and $R_q^\vee = R^\vee/qR^\vee$, for $q \geq 2$ an integer.

5.3.1 Discretization

In Ring-LWE applications, the discretization procedure rounds an element of the complex space H to some randomly determined nearby element of H in a lattice coset $\Lambda + \mathbf{c}$ [100]. Specifically, the PKE requires converting a continuous Gaussian into a discrete Gaussian-like distribution. However, the discretization introduces an additional error in the samples, increasing its length. The notion of the subgaussian parameter captures the quality of the discretized sample.

In this section, we present how the discretization is done, using the coordinate-wise randomized rounding for sampling the intermediate vector $\mathbf{f}$. Then, we show that the discretized error samples are 0-subgaussian with parameter $\sqrt{2\pi}$ when K is the maximal real subfield of a p-th cyclotomic number field.

First, we formally define supplementary statistic notions required by the main result of this section.

Definition 23. *The* n-th moment *of a random variable* X *is* $\mathbb{E}[X^n]$. *Also, the* n-th central moment *of* X *is* $\mathbb{E}[(X - \mathbb{E}[X])^n]$.

The *expectation* of a random variable X is the first moment of a random variable X denoted by $\mathbb{E}[X]$. Also, the second central moment is the variance of X.

A subgaussian random variable is a random variable that is bounded in a certain technical sense by a normal random variable [99]. Micciancio and Peikert [95] introduced the concept of δ-subgaussian random variable as a relaxation of a subgaussian random variable. We use moment-generating functions to compute a distribution's moments and δ-subgaussian to measure the length of the offset vector produced by the discretization algorithm.

Definition 24. *The* moment-generating function *of a random variable* X *is a function* $M_X(s)$ *defined as*

$$M_X(s) = \mathbb{E}[\exp(sX)].$$

We say that the moment-generating function of X *exists if there is an* $h > 0$ *such that* $M_X(s)$ *is finite for all* $s \in [-h, h]$.

Definition 25. *For any* $\delta \geq 0$, *we say that a random variable* X *(or its distribution) over* $\mathbb{R}$ *is* δ-subgaussian *with parameter* $s > 0$ *if for all* $t \in \mathbb{R}$, *the (scaled) moment-generating function satisfies*

$$\mathbb{E}[\exp(2\pi tX)] \leq \exp(\delta) \cdot \exp(\pi s^2 t^2)$$

Notice that the $\exp(\pi s^2 t^2)$ term on the right is the (scaled) moment-generating function of the one-dimensional Gaussian distribution of parameter s over $\mathbb{R}$.

Finally, we define the concept of the largest singular value of a matrix $\mathbf{M}$, also known as the spectral norm, which is denoted by $s_1(\mathbf{M})$.

Definition 26. *For a square matrix* $\mathbf{M}$, *the square roots of the eigenvalues of* $\overline{\mathbf{M}}^\top \mathbf{M}$ *are called* singular values.

Consider a lattice $\Lambda = \mathcal{L}(\mathbf{B})$ represented by a good basis $\mathbf{B} = \{\mathbf{b}_i\}_{i \in [n]}$ – that is, a basis containing reasonably short and orthogonal vectors –, a point $\mathbf{x} \in H$, and a point $\mathbf{c} \in H$ representing a lattice coset $\Lambda + \mathbf{c}$. The goal is to discretize $\mathbf{x}$ to a point $\mathbf{y} \in \Lambda + \mathbf{c}$, denoted $\mathbf{y} \leftarrow \lfloor x \rceil_{\Lambda + \mathbf{c}}$, so that the subgaussian parameter of $\mathbf{y} - \mathbf{x}$ is not too large.

The discretization is done by sampling a relatively short offset vector $\mathbf{f}$ from the coset $\mathbf{c}' = \Lambda + (\mathbf{c} - \mathbf{x})$, and outputting $\mathbf{y} = \mathbf{x} + \mathbf{f}$. In PKE, we can use the coordinate-wise randomized rounding in an orthonormal basis for sampling the vector $\mathbf{f}$. The coordinate-wise randomized rounding works as follows: given a coset $\Lambda + \mathbf{c}'$, we represent $\mathbf{c}'$ in the basis $\mathbf{B}$ as $\mathbf{c}' = \sum_{i=1}^{n} a_i \mathbf{b}_i \mod \Lambda$ for some coefficients $a_i \in [0, 1]$, then randomly and independently choose each f_i from $\{a_i - 1, a_i\}$ to have expectation zero, and output $\mathbf{f} = \sum_{i=1}^{n} f_i \mathbf{b}_i \in \Lambda + \mathbf{c}'$. In our setting, the vector $\mathbf{c}'$ is already given in H. So, there is no need to explicitly convert it so some basis in H to obtain the representative of $\mathbf{c}'$ in H.

Corollary 5.5.1 (Corollary 2.3 [89]). *Let $\delta_i, \mathbf{s}_i \geq 0$ and $\mathbf{X}_i$ be random vectors in $\mathbb{R}^n$ (or in H), and let $\mathbf{A}_i$ be $n \times n$ matrices for $i = 1, \ldots, k$. Suppose that for every i, when conditioning on any values of $\mathbf{X}_1, \ldots, \mathbf{X}_{i-1}$, the random vector $\mathbf{X}_i$ is δ_i-subgaussian with parameter $\mathbf{s}_i$. Then, $\sum_{i=1}^{n} \mathbf{A}_i \mathbf{X}_i$ is $(\sum \delta_i)$-subgaussian with parameter $\lambda_{\max}\left(\sum \mathbf{s}_i^2 \mathbf{A}_i \mathbf{A}_i^\top\right)^{1/2}$, where $\lambda_{\max}$ denotes the largest eigenvalue.*

In the coordinate-wise randomized rounding method, each f_i has expectation zero and is bounded by one in magnitude. So, it is 0-subgaussian with parameter $\sqrt{2\pi}$, and hence so is the entire vector of f_i values [89, Section 2.4.2]. For $R = \mathbb{Z}[\zeta_p + \zeta_p^{-1}]$ in the orthonormal basis $\mathcal{C}' = \{e'_1, \ldots, e'_n : e'_n = e_n \text{ and } e'_j = e_j + e'_{j+1}\}$, where $\{e_j = \zeta_p^j + \zeta_p^{-j}\}_{j=1}^n$, we have that $\sigma_\tau(R)$ is a lattice equivalent to $\mathbb{Z}^n$. Thus, $\overline{\mathbf{M}}^\top \mathbf{M} = \mathbf{I}_n$, where $\mathbf{M}$ is the generator matrix of $\sigma_\tau(R)$, and $s_1(\mathbf{M}) = 1$. Finally, by Corollary 5.5.1, the output vector $\mathbf{f}$ is 0-subgaussian with parameter $\sqrt{2\pi} \cdot s_1(\mathbf{M}) = \sqrt{2\pi}$ for any value of p.

In contrast, for $\mathbb{Z}[\zeta_m]$ in the decoding basis $\vec{d}$ [89], the spectral norm is $s_1(\vec{d}) = \sqrt{\text{rad}(m)/m}$, where $\text{rad}(m)$ represents the radical[1] of the cyclotomic index m. In this case, $s_1(\vec{d})$ can be as large as one. In conclusion, the discretization method does not produce worst samples than in the cyclotomic case. The error samples are enlarged by a constant factor of $\sqrt{2\pi}$ regardless of the choice of the number field index p.

5.3.2 Arithmetic Operations

In PKE, we need to perform addition, subtraction, and multiplication of ring elements. For that, we assume that all arithmetic operations are performed over H in the basis $\sigma_\tau(\mathcal{C}')$ and $\sigma_\tau(\mathcal{C}^\star)$ of R and $R^\vee$, respectively.

By linearity, the addition of two elements in a vector space corresponds to the component-wise addition of the respective coordinate vectors, at least if both elements are represented in the same basis [92]. In particular, additions in the PKE are done between elements of the dual ring $R^\vee$. For that, we can use the fact that $\sigma_\tau(a+b) = \sigma_\tau(a) + \sigma_\tau(b)$ to perform the ring additions as the component-wise addition of the coefficient vectors of $a, b \in R^\vee$ in the basis $\sigma_\tau(\mathcal{C}^\star)$.

The PKE computes three distinct multiplications. Firstly, it performs $p' \cdot e$, for $p' \in \mathbb{Z}$ being the modulus defining the message space, and $e \in K_\mathbb{R}$ an error drawn from the spherical Gaussian distribution ψ_τ. In this case, the multiplication is simply a scaling of an error term by p'. Secondly, we have computations of the form $a \cdot x$, for $a \in R_q$, and $x \in R^\vee$ a discretized sample from the spherical Gaussian distribution ψ_τ. Finally, in the decryption algorithm, we have $t \cdot d$, for $t \in R$ being the factor for which $R^\vee = \langle t^{-1} \rangle$, and $d \in R^\vee$ being the result from the decoding procedure in the decryption algorithm.

In summary, we still need to compute the ring multiplication between elements of R and $R^\vee$. Since a $\mathbb{Z}$-basis for R is also a $\mathbb{Z}_q$-basis for R_q, both multiplications can be performed using the same procedure. For that, both elements must be represented on the same basis. Assuming that $x \in R$ and $y \in R^\vee$, the multiplication $x \cdot y$ can be done by

[1] The radical of an integer m, denoted $\text{rad}(m)$, is the product of distinct prime numbers dividing m, that is, $\text{rad}(m) = \prod_{\text{prime } p \mid m} p$.

scaling y by $t \in R$, so that $x \cdot y$ lies in the ring R. Thus, the multiplication in the ring R can be performed using the fact that

$$\sigma_\tau(a \cdot b) = \sigma(a) \odot \sigma_\tau(b) = \sigma_\tau(a) \odot \sigma(b),$$

for any $a, b \in K_\mathbb{R}$, with $\odot$ denoting the component-wise product. Notice that performing the multiplication of ring elements under twisted embeddings requires having one of the operands in the canonical embedding. Concretely, one of the operands needs to be transformed from the basis $\sigma_\tau(\mathcal{C}')$ to $\sigma(\mathcal{C}')$. Recall that $a = \langle \mathfrak{B}, \mathbf{a} \rangle$ for any element $a \in R$, which is equal to say that $a = \mathfrak{B}^\top \cdot \mathbf{a}$, for a basis $\mathfrak{B}$ and some coefficient vector $\mathbf{a}$ over $\mathbb{Z}$ [89, Section 4]. In particular, for $\mathfrak{B} = \sigma_\tau(\mathcal{C}')$, we have that

$$\begin{aligned} a &= \mathbf{M}^\top \cdot \mathbf{a} \\ &= (\mathbf{D}\mathbf{M}'\mathbf{U})^\top \cdot \mathbf{a} \\ &= \mathbf{U}^\top \mathbf{M}'^\top \mathbf{D}^\top \cdot \mathbf{a}. \end{aligned}$$

On the other hand, for $\mathfrak{B} = \sigma(\mathcal{C}')$, it follows that

$$\begin{aligned} a &= (\mathbf{M}'\mathbf{U})^\top \cdot \mathbf{a} \\ &= \mathbf{U}^\top \mathbf{M}'^\top \cdot \mathbf{a}. \end{aligned}$$

In this sense, the coefficient vector of an element $a \in R$ under the canonical embedding is given by $\mathbf{D}^\top \cdot \mathbf{a}$.

In summary, the multiplication of elements in R and $R^\vee$ requires the multiplication of an operand by $\mathbf{D}^\top$ and the multiplication of the element in $R^\vee$ by t. In other words, the multiplication by the diagonal matrix $\mathbf{D}$ corresponds to the component-wise multiplication by $\left(\sqrt{\frac{\sigma_1(\tau)}{p}}, \ldots, \sqrt{\frac{\sigma_n(\tau)}{p}}\right)$. Moreover, we have that

$$\sigma_\tau(t \cdot x \cdot y) = \sigma(t) \odot \sigma_\tau(x) \odot \sigma(y),$$

in which $t \in R$, $x, y \in K_\mathbb{R}$, and $\sigma(t)$ can be precomputed in an offline phase. Finally, the multiplication in R can be done as

$$\sigma_\tau(t \cdot x \cdot y) = \sigma(t) \odot \sigma_\tau(x) \odot t_\tau \odot \sigma(y),$$

in which $t, x \in R$, $y \in R^\vee$, and t_τ is the linear transformation $t_\tau = \left(\sqrt{\frac{\sigma_1(\tau)}{p}}, \ldots, \sqrt{\frac{\sigma_n(\tau)}{p}}\right)$. Notice that we can precompute $\sigma(t)$ for later use since $t \in R$ is a fixed parameter with respect to the choice of number field and prime p. Thus, the overall cost for computing a multiplication under twisted embeddings is exactly $3n$ multiplications in $\mathbb{R}$.

Consequently, the expansion of the coefficients after multiplications is independent of the expansion factor of $f(x)$. The length of the elements relies on the inequalities taken under twisted embeddings:

$$\|a \cdot b\|_{p,\tau} \leq \|a\|_\infty \|b\|_{p,\tau} \quad \text{and} \quad \|a \cdot b\|_{p,\tau} \leq \|a\|_p \|b\|_{\infty,\tau},$$

for any $a, b \in K_{\mathbb{R}}$. In contrast with the polynomial representation, discussed at the end of Section 5.2, the error growth is bounded by the infinity norm under twisted embeddings. Thus, the coefficient representation does not require an exponential value of modulus q to avoid decryption errors. We leave as future work providing a close formula to the choice of value for the modulus q and the correctness and hardness proofs for our modified PKE.

5.3.3 Decoding $R^\vee$

The main operation in the decryption algorithm is the decoding of the term $d = v - u \cdot x \mod qR^\vee$ in the dual ring $R^\vee$. The algorithm asks to recover an unknown short vector $\mathbf{x} \in H$ given $\mathbf{t} = \mathbf{x} \mod \Lambda$, which is essentially a solution for the Bounded Distance Decoding (BDD) Problem. For completeness, we define the BDD Problem in the following.

Definition 27 ($\mathbf{BDD}_\alpha$). *Given a n-dimensional lattice Λ and a vector $\mathbf{t}$ such that $dist(\mathbf{t}, \Lambda) < \alpha \cdot \lambda_1(\Lambda)$, find a lattice vector $\mathbf{x}$ such that $\|\mathbf{x} - \mathbf{t}\| < \alpha \cdot \lambda_1(\Lambda)$.*

For the Ring-LWE context, Lyubashevsky et al. [89, Section 2.4.1] define an extension of Babai's round-off algorithm [19]. Babai's round-off algorithm is given in Theorem 5.6. The theorem as presented next is an adaptation of both [77, Theorem 7.34] and [91, Theorem 3]. Notice that Babai's round-off algorithm can be executed with an arbitrary basis $\mathbf{B} = \{\mathbf{b}_i\} \in \mathbb{R}^n$ for the lattice Λ. A 'good basis' consists of a set of vectors reasonably orthogonal to one another, whereas a 'bad basis' contains highly non-orthogonal vectors. Thus, when the algorithm is executed using a good basis, it is likely to be successful in solving CVP, effectively returning a lattice vector close to the target vector $\mathbf{t} \in \mathbb{R}^n$. On the other hand, for a bad basis $\mathbf{B}$, the algorithm does not work properly, outputting a lattice vector far from the target $\mathbf{t}$.

Theorem 5.6. *For $n \in \mathbb{N}$, let the set $\{\mathbf{b}_i\} \in \mathbb{R}^n$ be the basis for the lattice $\Lambda \subset \mathbb{R}^n$ and let $\mathbf{t} \in \mathbb{R}^n$ be a target vector. If the basis vectors $\{\mathbf{b}_i\}$ are sufficiently orthogonal to one another, the procedure of Babai's round-off method given in Algorithm 4 solves CVP.*

Algorithm 4: Babai's round-off method

Input: A basis $\{\mathbf{b}_i\}$ for a lattice $\Lambda \subset \mathbb{R}^n$, and a target vector $\mathbf{t} \in \mathbb{R}^n$.
Output: A lattice vector $\mathbf{x} \in \Lambda$.

1 Write $\mathbf{t}$ in the basis $\{\mathbf{b}_i\}$ as $\mathbf{t} = \sum_i c_i \mathbf{b}_i$ such that $\mathbf{t} = \mathbf{B}^\top \cdot \mathbf{c}$
2 Compute the coefficient vector as $\mathbf{c} = (\mathbf{B}^\top)^{-1} \cdot \mathbf{t}$
3 Round each entry of $\mathbf{c}$ to the nearest integer as $c_i = [\![c_i]\!] \in \mathbb{Z}$
4 Compute the lattice vector $\mathbf{x}$ as $\mathbf{x} = \mathbf{B}^\top \cdot \mathbf{c}$
5 **return** $\mathbf{x} \in \Lambda$

In general, if the vectors in the basis are reasonably orthogonal to one another, then the algorithm solves some version of approximation CVP, but if the basis vectors are highly non-orthogonal, then the lattice vector returned by the algorithm is generally far from the closest vector to the target vector $\mathbf{t} \in \mathbb{R}^n$.

The extended algorithm of Lyubashevsky et al. [89] starts by fixing a set $\{\mathbf{v}_i\}$ of n linearly independent, and typically short, vectors in the dual lattice $\Lambda^\vee$. Notice that this

definition generalizes the original Babai's round-off algorithm, which requires the input set of vectors to be a basis of $\Lambda^\vee$. Then, the target vector $\mathbf{t} = \mathbf{x} \mod \Lambda$ is represented in the basis $\{\mathbf{b}_i\}$ as $\sum_i c_i \mathbf{b}_i$, where $c_i \in \mathbb{R}/\mathbb{Z}$[2]. In this case, $\{\mathbf{b}_i\}$ is the dual basis of $\{\mathbf{v}_i\}$, which generates the superlattice $\Lambda' \supseteq \Lambda$. Claim 1 [89, Claim 2.10] follows from the fact that $c_i = \langle \mathbf{x}, \overline{\mathbf{v}_i} \rangle \mod 1$.

Claim 1. *Let $\Lambda \subset H$ be a lattice, let $\{\mathbf{v}_i\} \subset \Lambda^\vee$ be a set of linearly independent vectors in its dual, and let $\{\mathbf{b}_i\} \subset \Lambda$ denote the dual basis of $\{\mathbf{v}_i\}$. The above round-off algorithm, given input $\mathbf{x} \mod \Lambda$, outputs $\mathbf{x}$ if and only if all the coefficients $a_i = \langle \mathbf{x}, \overline{\mathbf{v}_i} \rangle \in \mathbb{R}$ in the expansion $\mathbf{x} = \sum_i a_i \mathbf{b}_i$ are in $[-1/2, 1/2)$.*

In summary, Claim 1 states that the success of the decoding process depends inversely on the length of the vectors in the dual basis $\{\overline{\mathbf{v}_i}\}$. Recall that, in PKE, the decoding process is performed in the dual ring $R^\vee$ in the basis $\sigma_\tau(\mathcal{C}^\star)$, where $\mathcal{C}^\star = t^{-1} \cdot \mathcal{C}'$. In this sense, we need to determine the dual vectors of $\sigma_\tau(\mathcal{C}^\star)$ and also their maximum length.

From Proposition 3.2, item (i), we have that $\mathcal{I}^* = \tau^{-1}\overline{\mathcal{I}}^\vee$ in the space $(K_\mathbb{R}, \langle , \rangle_\tau)$. Equivalently,

$$\sigma_\tau(\mathcal{I})^* = \sigma_\tau(\tau^{-1}\overline{\mathcal{I}}^\vee),$$

from which we derive how to obtain the vectors dual to $\sigma_\tau(\overline{\mathcal{I}}^\vee)$ as

$$\begin{aligned}
&\sigma_\tau(\mathcal{I})^* = \sigma_\tau(\tau^{-1}\overline{\mathcal{I}}^\vee) \\
\Leftrightarrow\ &\sigma_\tau^{-1}(\sigma_\tau(\mathcal{I})^*) = \tau^{-1}\overline{\mathcal{I}}^\vee \\
\Leftrightarrow\ &\tau \cdot \sigma_\tau^{-1}(\sigma_\tau(\mathcal{I})^*) = \overline{\mathcal{I}}^\vee \\
\Leftrightarrow\ &\sigma_\tau(\tau \cdot \sigma_\tau^{-1}(\sigma_\tau(\mathcal{I})^*)) = \sigma_\tau(\overline{\mathcal{I}}^\vee) \\
\Leftrightarrow\ &\sigma_\tau(\tau \cdot \sigma_\tau^{-1}(\sigma_\tau(\mathcal{I})^*))^* = \sigma_\tau(\overline{\mathcal{I}}^\vee)^*.
\end{aligned}$$

In contrast with the canonical embedding, $\sigma(\mathcal{I}^\vee)$ is simply $\overline{\sigma(\mathcal{I})^*}$. When $\mathcal{I} = R$, the ring of integers of K, it follows that $\sigma(R^\vee)^* = \sigma(R)$.

For any $\mathbb{Q}$-basis $\mathfrak{B} = \{b_j\}$ of K, its dual basis is denoted by $\mathfrak{B}^\vee = \{b_j^\vee\}$, which is characterized by the fact that $\mathrm{Tr}(b_i \cdot b_j^\vee) = \delta_{i,j}$, the Kronecker delta. In our case, we need to characterize the vectors dual to $\sigma_\tau(R^\vee)$, which can be done by finding a basis dual to $\sigma_\tau(\mathcal{C}^\star)$.

Notice that, by construction, $\frac{1}{p}\mathrm{Tr}(\tau e_i' e_j') = \delta_{i,j}$, in which $\mathcal{C}' = \{e_j'\}$ is given by $e_n' = e_n$ and $e_j' = e_j + e_{j+1}'$ for $1 \leq j < n$, and $\{e_i = \zeta^i + \zeta^{-i}\}_{i=1}^n$ is an integral basis of R. In particular, since we are working with the inner product induced by twisted embeddings, we need to determine a basis dual to $\mathcal{C}^\star$ as

$$\mathrm{Tr}(\tau \cdot \mathcal{C}^\star \cdot (\mathcal{C}^\star)^\vee) = \delta_{i,j}.$$

[2]Any two elements $a, b \in \mathbb{R}$ in $\mathbb{R}/\mathbb{Z}$ are said to be equivalent if and only if $a - b \in \mathbb{Z}$. In this sense, if a and b are equivalent, their representative is the fractional part which lies in $[0, 1)$. Formally, the quotient $\mathbb{R}/\mathbb{Z}$ is $\{\mathbb{Z} + r : r \in [0, 1)\}$.

In fact, such basis dual to $\mathcal{C}^\star$ is $\frac{1}{p} \cdot t \cdot \mathcal{C}'$, for which it follows that

$$\begin{aligned} & \operatorname{Tr}(\tau \cdot \mathcal{C}^\star \cdot (\mathcal{C}^\star)^\vee) \\ = \ & \operatorname{Tr}(\tau \cdot (t^{-1} \cdot \mathcal{C}') \cdot (p^{-1} \cdot t \cdot \mathcal{C}')) \\ = \ & \operatorname{Tr}(\tau \cdot \mathcal{C}' \cdot (p^{-1} \cdot \mathcal{C}')) \\ = \ & \frac{1}{p} \operatorname{Tr}(\tau e_i' e_j') \\ = \ & \delta_{i,j}. \end{aligned}$$

In this sense, the ℓ_2-norm of the basis dual to $\mathcal{C}^\star$ can be computed as

$$\begin{aligned} \|(\mathcal{C}^\star)^\vee\|_2 &= \left\| \frac{1}{p} \cdot t \cdot \mathcal{C}' \right\|_2 \\ &= \left\| \sigma_\tau(p^{-1} \cdot t \cdot \mathcal{C}') \right\|_2 \\ &= \left(\left\langle \sigma_\tau(p^{-1} \cdot t \cdot \mathcal{C}'), \sigma_\tau(p^{-1} \cdot t \cdot \mathcal{C}') \right\rangle \right)^{\frac{1}{2}} \\ &= \left(\operatorname{Tr}(\tau \cdot p^{-1} \cdot t \cdot \mathcal{C}') \right)^{\frac{1}{2}} \\ &= \left(\frac{1}{p} \operatorname{Tr}(\tau \cdot t \cdot \mathcal{C}') \right)^{\frac{1}{2}} \\ &= \left(\frac{1}{p} \cdot \sum_{j=1}^{n} \sigma_j(\tau)\sigma_j(t)\sigma_j(e_j') \right)^{\frac{1}{2}} \\ &= \left(\frac{1}{p} \cdot \sum_{j=1}^{n} \sigma_j(2 - (\zeta + \zeta^{-1}))\sigma_j(t)\sigma_j(e_j') \right)^{\frac{1}{2}}, \end{aligned}$$

and the ℓ_∞-norm is given by

$$\max_{1 \le i \le n} \left| \sqrt{\sigma_i(2 - (\zeta + \zeta^{-1}))} \sigma_i(t) \sigma_i(e_j') \right|, \forall \ j \in [n].$$

The computation of both ℓ_p and ℓ_∞-norms use the fact that the n field embeddings of K are

$$\sigma_k(e_j) = \zeta^{kj} + \zeta^{-kj} = 2\cos\left(\frac{2\pi kj}{p}\right),$$

and the element $t \in R$ is

$$t = b_1 + 2b_2(\zeta_p + \zeta_p^{-1}) + 3b_3(\zeta_p + \zeta_p^{-1})^2 + \cdots + (n-1)b_{n-1}(\zeta_p + \zeta_p^{-1})^{n-2} + n(\zeta_p + \zeta_p^{-1})^{n-1},$$

for $b_j \in \mathbb{Z}$. Recall from Section 5.2 that the coefficients $b_i \in \mathbb{Z}$ of the defining polynomial $\Psi_p(x)$ increase rapidly according to the prime number p. Thus, the maximum length of the dual vectors may render the basis $\mathcal{C}^\star$ of $R^\vee$ impractical for decoding error terms. However, the fact that $\operatorname{Tr}(\zeta^k + \zeta^{-k}) = \sum_{j=1}^{n} \sigma_j(\zeta^k + \zeta^{-k}) = -1$ for any $1 \le k \le n$, and the relations inherent to the vectors e_j' may still be explored to simplify the computation of the norms. For details on these properties, see the work of Oggier et al. [106].

Moreover, in light of the work of Lyubashevsky et al. [89], in which each algorithmic task is performed on a distinct basis, one may find a basis for $R^\vee$ which suits better the decoding task considering the length of the vectors dual to $R^\vee$. These opportunities may be explored as future work.

5.4 Discussion

In light of the ongoing NIST's standardization process, we informally contrast PKE with similar third-round finalists, assuming the main algorithmic tasks, as discussed in previous sections.

The uniform sampling of elements in R_q may proceed as CRYSTALS-Kyber and Saber. However, the exact rejection rate depends on the integer modulus used for arithmetic operations and its bit size. Thus, a proper analysis requires an actual instantiation of PKE, that is, choosing values for the parameters p, p', and q. In practice, sampling from a spherical Gaussian distribution may reduce to obtain samples from a binomial distribution, as was first done for New Hope [12] and further extended to other public-key encryption schemes. Although it requires a formal proof, switching to a binomial distribution makes the discretization procedure unnecessary, simplifying the PKE design.

In PKE, one may consider representing ring elements using coefficient vectors. Assuming that all algorithmic operations are done on the same basis, arithmetic operations such as addition and multiplication have linear cost, not requiring the computation of NTTs or a Toom-Cook-based algorithm for multiplication. Notice, however, that this analysis assumes the precomputation of $t \in R$ and the diagonal matrix $\mathbf{D}$. Finally, the most expensive operation is decoding error terms in $R^\vee$, which is done through Babai's round-off algorithm. This step is more involving and requires additional research.

Part II

Algorithms for Polynomial Multiplication

Introduction to Part II

Figure 5.1 depicts the organization of Part II of this thesis, which is composed of two chapters, 6 and 7, each represented by a descriptive diagram. This part aims at exploring the suitability of using the Discrete Galois Transform (DGT) for polynomial multiplication in two distinct architectures, namely x64 desktop machines and Graphics Processing Units (GPUs). In particular, we consider the polynomial ring $R = \mathbb{Z}[x]/(x^n + 1)$ with n a power of two.

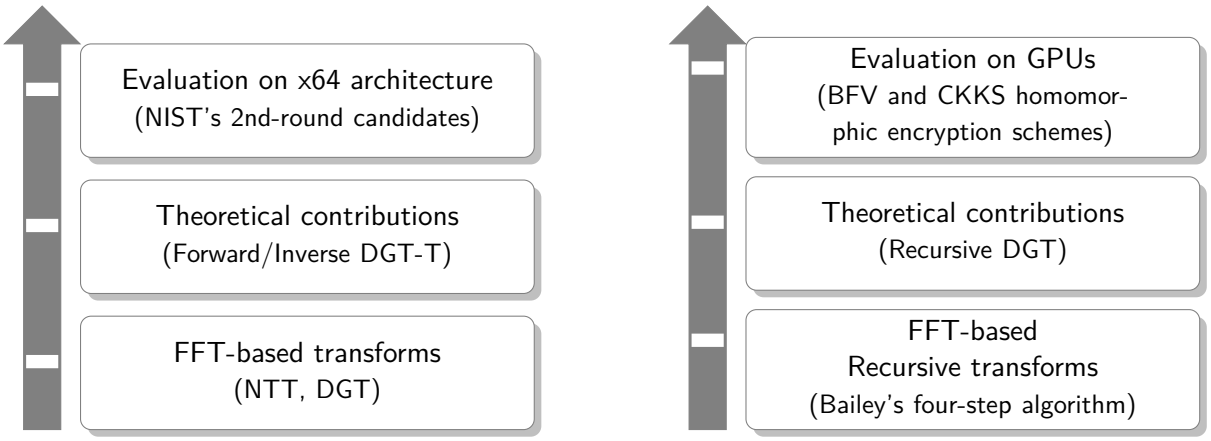

Figure 5.1: Structure of the second part of this thesis. The left part of this diagram refers to Chapter 6. The right part is a representation of the content in Chapter 7.

Chapter 6 starts by revisiting Fast Fourier Transform (FFT) algorithms, specially radix-2 FFT algorithms. Next, we present the Number-Theoretic Transform (NTT), a widely adopted algorithm for polynomial multiplication in the quotient ring $R_p = R/pR$, for p a prime number. Then, we present the Discrete Galois Transform, which we evaluate for its use in lattice-based cryptosystems based on the Ring-LWE Problem. Motivated by NIST's Post-Quantum Cryptography standardization project, we present experimental results comparing the DGT with the NTT for CRYSTALS-Dilithium [64], NewHope [9], and qTESLA [30] cryptosystems. These cryptosystems were being considered for standardization during the second round of NIST's project, time of development of our experiments. We considered the DGT for polynomial multiplication as introduced in the literature by Badawi et al. [22] and also our new formulation, named *DGT-T*, which combines the DGT transform with its twisting procedure. Lastly, experimental results indicate that the DGT does not perform better for polynomial multiplication in comparison with the NTT, considering portable-C and AVX2 implementations targeting the x64 architecture.

Chapter 7 presents experimentals results indicating the suitability of the Recursive DGT (RDGT) in GPUs. RDGT is a formulation of the DGT based on the Recursive FFT [71, 23] that considers architectures with a fast but small first-level memory. First, we compare the RDGT with state-of-the-art CUDA-enabled implementations of DGT for polynomial multiplication in the BFV homomorphic encryption scheme [68] in NVIDIA Tesla GPUs. Then, we implement both RDGT and Recursive NTT for polynomial multiplication in the CKKS homomorphic encryption scheme [45].

Chapter 6

DGT Evaluation on Round-Two NIST's PQC Candidates

At the end of 2016, NIST initiated the Post-Quantum Cryptography project [101], a process for standardizing quantum-resistant public-key algorithms. Fifteen candidates are still under analysis at the end of the third round. Seven submissions are based on lattices, the most prominent class of problems, comprehending five of the seven finalists. Most lattice cryptosystems are constructed over algebraic lattices and require efficient polynomial multiplication algorithms to be competitive performance-wise.

Polynomial multiplication can be handled efficiently using algorithms such as Karatsuba [81] and Toom-Cook [132, 50], or the Number-Theoretic Transform (NTT). The NTT has quasi-linear complexity, and it is known to be efficient in the case when the polynomial ring is of the form $\mathbb{Z}_p[x]/(x^n+1)$, with n being a power of two and p a prime number satisfying $p \equiv 1 \pmod{2n}$. For this choice of parameters, x^n+1 splits in $\mathbb{Z}_p$ into irreducible polynomials of degree one, and the polynomial multiplication in the NTT domain reduces to the component-wise product of integers modulo p.

Lyubashevsky and Seiler [90] showed how to evaluate the NTT when the polynomial ring is $\mathbb{Z}_{7681}[x]/(x^{768}-x^{384}+1)$ in the NTTRU key encapsulation scheme. The polynomial $x^{768}-x^{384}+1$ initially splits into $(x^{384}+684)(x^{384}-685)$; then, each factor forms a splitting tree in which each root splits into two polynomials of the form $x^i \pm r'$ until the irreducible polynomials $x^3 \pm r$ are reached. This observation allowed the NTT to be instantiated using smaller choices of modulus p, such that $n \mid (p-1)$ or, even, $\frac{n}{2} \mid (p-1)$ [18]. Before that, the smallest prime number such that there exists a primitive $2n$-th root of unity modulo p is 12 289.

In the second version of Kyber documentation [18], the modulus $p = 3329$ is such that $n \mid (p-1)$ but $2n \nmid (p-1)$ and elements in $\mathbb{Z}_p[x]/(x^{256}+1)$ decompose modulo polynomials of the form $x^2 - r'$. Then, component-wise multiplication is performed with degree-one polynomials instead of scalars. The possibility of stopping at an arbitrary level in the NTT splitting tree was further discussed in detail by Alkim, Bilgin, and Cenk [11].

In contrast with the NTT, the Discrete Galois Transform (DGT) computes the polynomial multiplication in the domain of Gaussian integers, denoted $\mathbb{Z}_p[i]$, in which the arithmetic is performed similarly to the complex numbers, by representing elements $\alpha \in \mathbb{Z}_p[i]$ as $\alpha = a + bi$, with $a, b \in \mathbb{Z}_p$ and $i = \sqrt{-1}$. One of the main advantages of the DGT

is computing an $\frac{n}{2}$-length transform by processing two coefficients at a time. Because of that, the DGT, by itself, only requires that $\frac{n}{2} \mid (p-1)$.

The DGT was considered for polynomial multiplication in combination with the Residue Number System (RNS) in the context of homomorphic encryption schemes [2, 13]. Badawi et al. proposed algorithms for polynomial multiplication using the DGT via Gentleman-Sande and Cooley-Tukey butterflies [4]. Alves et al. introduced a recursive form of DGT based on Badawi et al.'s algorithms, obtaining speedups to compute the transform and homomorphic multiplication.

In this scenario, we analyze adopting the DGT for polynomial multiplication in smaller cryptosystems, such as key encapsulation mechanisms and digital signature schemes. Our contributions are two-fold. First, we propose new algorithms for computing the forward and inverse DGT using the Cooley-Tukey and Gentleman-Sande butterflies, respectively (Section 6.5.1). This order of the butterflies allows merging the twisting procedure with both forward and inverse DGT transforms computation. We experimentally evaluate the performance of polynomial multiplication via DGT by providing software implementations for the NewHope key exchange protocol [9], and for the signature schemes CRYSTALS-Dilithium [64] and qTESLA [30]. Also, we evaluate the new algorithms for the DGT computation by comparing software implementations of the cryptosystems mentioned above using the DGT with and without the merge (Section 6.5.2).

We verified that merging the twisting procedure with the computation of the DGT transform does not bring performance gains. For CRYSTALS-Dilithium, NewHope1024CCA, and qTESLA-p-I, we obtained a slowdown of 0.68, 0.65, and 0.85, in this order. Although the merge may simplify the polynomial multiplication via DGT, it increases the overall number of integer operations modulo p. In this sense, we adopted Badawi et al.'s algorithms [4] for polynomial multiplication via DGT. Nonetheless, our experimental evaluation indicates that it may be very hard to explore the x64 architecture capabilities to obtain speedups using the DGT.

6.1 The Discrete Fourier Transform

In general, the product of two polynomials $a(x), b(x) \in \mathbb{Z}[x]$ with degree up to n can be defined as

$$c(x) = a(x) \cdot b(x) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_i b_j x^{i+j}. \tag{6.1}$$

In the cyclotomic case with n a power of two, where the polynomial multiplication is taken modulo $x^n + 1$, the fact that $x^n \equiv -1 \mod x^n + 1$ allow us to rewrite Equation 6.1 as

$$c(x) = a(x) \cdot b(x) \mod x^n + 1 = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (-1)^{\left\lfloor \frac{i+j}{n} \right\rfloor} a_i b_j x^{i+j \bmod n}. \tag{6.2}$$

This strategy of replacing x^n with -1 is known as the *negacyclic convolution*.

In general, a schoolbook algorithm for calculating the polynomial multiplication as in Equation 6.2 has the asymptotic cost of $O(n^2)$ operations. The Fast Fourier Transform (FFT) is a well-known algorithm used to reduce the computational complexity of polyno-

mial multiplication. A vector containing the operand's coefficients is treated as a signal in the time domain, which is transformed to the frequency domain via Discrete Fourier Transform (DFT). Thus, the polynomial multiplication corresponds to the component-wise multiplication in the frequency domain. The FFT computes the DFT of a vector's element with $O(\log n)$ operations; thus, the overall cost of a polynomial multiplication is $O(n \log n)$.

> The Discrete Fourier Transform (DFT) is the Fourier transform over a finite field. The Fourier transform decomposes functions of time, denoted $f(t)$, into functions of frequency $F(\zeta)$, and vice-versa:
>
> $$F(\zeta) = \int_{-\infty}^{+\infty} f(t) e^{-j\zeta t}\, dt \quad \text{and} \quad f(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(\zeta) e^{j\zeta t}\, d\zeta.$$
>
> The underlying idea is that any periodic function can be represented by a series of sines and cosines. For example, given a sound wave recorded over time, the Fourier transform can decompose the amplitudes into its constituent frequencies in Hertz. Jean-Baptiste Joseph Fourier, a French mathematician who lived in the XVIII and XIX centuries, was the first to explore the Fourier transforms.

The DFT takes as input a finite sequence of complex numbers with size n, denoted $\mathbf{f}$, and compute its transform as the complex sequence obtained by computing $\mathbf{F}$ as

$$F_r = \sum_{j=0}^{n-1} f_j \zeta_n^{-jr}, \tag{6.3}$$

where $\zeta_n = \exp(2\pi i/n)$ is a primitive n-th complex root of unity and $r \in [n]$. Similarly, the inverse transform is defined as

$$f_r = n^{-1} \sum_{j=0}^{n-1} F_j \zeta_n^{jr}. \tag{6.4}$$

A naïve implementation of Equations 6.3 and 6.4 incurs in an algorithm with quadratic cost. However, the DFT can be efficiently computed using FFT algorithms, which are presented in the following.

6.2 Fast Fourier Transform Algorithms

Fast Fourier Transform (FFT) algorithms adopt a divide-and-conquer paradigm which allows the DFT to be computed with the quasi-linear cost of $O(n \log n)$ operations [51]. A widely used FFT algorithm is the radix-2 FFT, which divides the original problem (the computation of the summations defined in Equations 6.3 and 6.4) into two sub-problems. Because of that, we assume that the size n is a power of two. Depending on how the sub-problems are defined, the radix-2 algorithms may be classified as Decimation-In-Time (DIT) or Decimation-In-Frequency (DIF). Details can be found in [49].

In the following, for a primitive n-th complex root of unity $\zeta_n = \exp(2\pi i/n)$, we repeatedly use the fact that $\zeta_n^2 = \zeta_{\frac{n}{2}}, \zeta_n^{\frac{n}{2}+r} = \zeta_n^{-r}$, and $\zeta_n^{rk} = \zeta_n^{(r+n)k}$.

6.2.1 Decimation-in-time Radix-2 Algorithms

A Decimation-In-Time (DIT) algorithm splits the summation in Equation 6.3 into two sets, one containing the elements of $\mathbf{f}$ with even indexes and the other with the odd ones. Then, the problem of computing the forward DFT is rewritten as

$$\begin{aligned} F_r &= \sum_{k=0}^{\frac{n}{2}-1} f_{2k}\zeta_n^{-2kr} + \sum_{k=0}^{\frac{n}{2}-1} f_{2k+1}\zeta_n^{-(2k+1)r} \\ &= \sum_{k=0}^{\frac{n}{2}-1} f_{2k}\zeta_{\frac{n}{2}}^{-kr} + \zeta_n^{-r}\sum_{k=0}^{\frac{n}{2}-1} f_{2k+1}\zeta_{\frac{n}{2}}^{-kr}. \end{aligned} \tag{6.5}$$

Notice that, using the fact that $\zeta_{\frac{n}{2}}^{-rk} = \zeta_{\frac{n}{2}}^{-(r+\frac{n}{2})k}$, it only suffices to compute F_r for $r \in \left[\frac{n}{2}\right]$. Since the inverse DFT is the forward transform but taking negative powers of the root of unity ζ_n, this same strategy can be used for evaluating Equation 6.4.

From Equation 6.5, we define $g_k = f_{2k}$ and $h_k = f_{2k+1}$, which leads to the two sub-problems G_r and H_r:

$$G_r = \sum_{k=0}^{\frac{n}{2}-1} g_k\zeta_{\frac{n}{2}}^{-kr} \quad \text{and} \quad H_r = \sum_{k=0}^{\frac{n}{2}-1} h_k\zeta_{\frac{n}{2}}^{-kr}, \tag{6.6}$$

which can be seen as two problems with size $\frac{n}{2}$ similar to the original problem as in Equation 6.3. This process can be repeated until the two sub-problems become trivial; then, the solutions for G_r and H_r can be combined to solve the original problem as $F_r = G_r + \zeta_n^{-r}H_r$. For obtaining the solution for the second half of F_r, which corresponds to the indexes $r \in \left\{\frac{n}{2}, \ldots, n-1\right\}$, it follows that

$$\begin{aligned} F_{r+\frac{n}{2}} &= \sum_{k=0}^{\frac{n}{2}-1} g_k\zeta_{\frac{n}{2}}^{-k\left(r+\frac{n}{2}\right)} + \zeta_n^{-(r+\frac{n}{2})}\sum_{k=0}^{\frac{n}{2}-1} h_k\zeta_{\frac{n}{2}}^{-k\left(r+\frac{n}{2}\right)} \\ &= \sum_{k=0}^{\frac{n}{2}-1} g_k\zeta_{\frac{n}{2}}^{-kr} - \zeta_n^{-r}\sum_{k=0}^{\frac{n}{2}-1} h_k\zeta_{\frac{n}{2}}^{-kr}, \end{aligned} \tag{6.7}$$

which provides the solution for the original problem as $F_r = G_r - \zeta_n^{-r}H_r$. The expressions $F_r = G_r + \zeta_n^{-r}H_r$ for $r \in \left[\frac{n}{2}\right]$ and $F_r = G_r - \zeta_n^{-r}H_r$ for $r \in \left\{\frac{n}{2}, \ldots, n-1\right\}$ are commonly referred to as the *Cooley-Tukey butterfly* [51].

6.2.2 Decimation-in-frequency Radix-2 Algorithms

Decimation-In-Frequency (DIF) radix-2 algorithms divide the original problem, defined in Equation 6.3, into two halves. Then, a different sub-problem is defined for the even and odd indexes. The general idea is the opposite of that adopted in DIT algorithms.

First, the original problem is rewritten as

$$\begin{aligned} F_r &= \sum_{j=0}^{\frac{n}{2}-1} f_j \zeta_n^{-jr} + \sum_{j=\frac{n}{2}}^{n-1} f_j \zeta_n^{-jr} \\ &= \sum_{j=0}^{\frac{n}{2}-1} f_j \zeta_n^{-jr} + \sum_{j=0}^{\frac{n}{2}-1} f_{j+\frac{n}{2}} \zeta_n^{-\left(j+\frac{n}{2}\right)r} \\ &= \sum_{j=0}^{\frac{n}{2}-1} \left(f_j + f_{j+\frac{n}{2}} \zeta_n^{-r\frac{n}{2}} \right) \zeta_n^{-jr}. \end{aligned} \tag{6.8}$$

Then, the sub-problems are derived by taking the odd and even indexes of F_r separately. When r is even, the original problem is $F_r = F_{2k}$:

$$\begin{aligned} F_{2k} &= \sum_{j=0}^{\frac{n}{2}-1} \left(f_j + f_{j+\frac{n}{2}} \zeta_n^{-kn} \right) \zeta_n^{-2kj} \\ &= \sum_{j=0}^{\frac{n}{2}-1} \left(f_j + f_{j+\frac{n}{2}} \right) \zeta_{\frac{n}{2}}^{-kj}, \end{aligned} \tag{6.9}$$

for $k \in \left[\frac{n}{2}\right]$. From the above summation, we define $G_k = F_{2k}$ and $g_j = f_j + f_{j+\frac{n}{2}}$, which let us define the sub-problem G_k as

$$G_k = \sum_{j=0}^{\frac{n}{2}-1} g_j \zeta_{\frac{n}{2}}^{-kj}, \tag{6.10}$$

for $k \in \left[\frac{n}{2}\right]$. On the other hand, when r is odd, $F_r = F_{2k+1}$ and then

$$\begin{aligned} F_{2k+1} &= \sum_{j=0}^{\frac{n}{2}-1} \left(f_j + f_{j+\frac{n}{2}} \zeta_n^{-(2k+1)\frac{n}{2}} \right) \zeta_n^{-j(2k+1)} \\ &= \sum_{j=0}^{\frac{n}{2}-1} \left(\left(f_j - f_{j+\frac{n}{2}} \right) \zeta_n^{-j} \right) \zeta_{\frac{n}{2}}^{-kj}. \end{aligned} \tag{6.11}$$

Similarly, by defining $H_k = F_{2k+1}$ and $h_j = \left(\left(f_j - f_{j+\frac{n}{2}} \right) \zeta_n^{-j} \right)$, the second sub-problem is

$$H_k = \sum_{j=0}^{\frac{n}{2}-1} h_j \zeta_{\frac{n}{2}}^{-kj}, \tag{6.12}$$

for $k \in \left[\frac{n}{2}\right]$. The computation of $g_j = f_j + f_{j+\frac{n}{2}}$ and $h_j = \left(\left(f_j - f_{j+\frac{n}{2}} \right) \zeta_n^{-j} \right)$ defines the *Gentleman-Sande butterfly* [71]. Notice that all sub-problems defined in both Cooley-Tukey and Gentleman-Sande butterflies are identical to the original problem in Equation 6.3. Because of that, the above formulas can be repeatedly used between themselves to obtain small enough sub-problems.

In summary, any radix-2 algorithm for computing the FFT can be used for obtaining polynomial multiplication with quasi-linear computational cost by using properties of

the primitive n-th root of unity and a divide-and-conquer paradigm. Nonetheless, the FFT is defined in the set of complex numbers, which may incur precision errors from floating-point arithmetic use. Thus, in lattice-based cryptography, the Number-Theoretic Transform (NTT) is used instead, in which the base field is $\mathbb{Z}_p$, for some prime number p.

6.3 The Number-Theoretic Transform

The Number-Theoretic Transform (NTT) is a variation of the DFT that replaces the primitive n-th complex root of unity by a primitive n-th root of unity in a ring $\mathbb{Z}_q$ [119]. For existing a primitive n-th root of unity in $\mathbb{Z}_q$ when n is a power of two, the NTT requires $q = p$ to be a prime number and that $n \mid (p-1)$. In this case, Pollard [119] proved that there exists a primitive n-th root of unity in $\mathbb{Z}_p$ that can be computed as $r^{(p-1)/n}$, where r is the primitive root modulo p. Because of that, some cryptosystems based on polynomial rings adopt non-NTT algorithms for efficient polynomial multiplication, such as Karatsuba [81] and Tom-Cook [132, 50] multiplications, or an integer modulus q which allows fast modular reduction.

For a polynomial $a(x) = \sum_{j=0}^{n-1} a_j x^j \in \mathbb{Z}[x]$, the n-point NTT computes

$$\mathrm{NTT}(a(x), \zeta_n) = \left(a(\zeta_n^0), a(\zeta_n^1), \ldots, a(\zeta_n^{n-1})\right).$$

In its turn, the inverse transform computes

$$\mathrm{INTT}(a(x), \zeta_n^{-1}) = n^{-1} \cdot \mathrm{NTT}(a(x), \zeta_n^{-1})$$

in which n^{-1} is the multiplicative inverse of n modulo p.

Notice that the NTT can be used to compute the cyclic convolution using the fact that ζ_n^j are the roots of the polynomial $x^n - 1$. Thus, the polynomial multiplication $c(x) = a(x) \cdot b(x) \mod x^n - 1$, which corresponds to the cyclic convolution, can be obtained by computing three n-point NTTs as

$$c(x) = \mathrm{INTT}(\mathrm{NTT}(a(x), \zeta_n) \odot \mathrm{NTT}(b(x), \zeta_n), \zeta_n^{-1}),$$

where $\odot$ denotes the component-wise multiplication. Considering the butterflies presented in Section 6.2, polynomial multiplication modulo $x^n - 1$ can be performed with asymptotic cost $O(n \log n)$ by means of radix-2 FFT algorithms based on the Cooley-Tukey and Gentleman-Sande butterflies.

An iterative radix-2 NTT algorithm based on the Cooley-Tukey butterfly is presented in Algorithm 5 [124]. Notice that, in the first step, Algorithm 5 changes the order of coefficients using the bit-reversal operation[1], denoted BitReverse, producing an output in standard ordering.

[1]The bit-reversal operation takes as input an integer number ϵ which is in the integer interval $\{0, \ldots, \mu\}$. Then, it *i*) computes the binary representation of ϵ as $(b_0, b_1, \ldots, b_{\log_2(\mu)-1})$, *ii*) reverses the integer representation as $(b_{\log_2(\mu)-1}, \ldots, b_1, b_0)$, and *iii*) returns the corresponding integer obtained from converting from the base-2 to base-10 representation. For example, when $\mu = 8$, the bit-reversed

Algorithm 5: In-place radix-2 NTT algorithm based on the Cooley-Tukey butterfly

Input: A polynomial $a(x) \in \mathbb{Z}_p[x]$ of degree at most $n-1$, a prime number p, and ζ_n a primitive n-th root of unity modulo p
Output: $\mathbf{a} = \mathsf{NTT}(\mathbf{a}, p, \zeta_n)$

$\mathbf{a} = \mathsf{BitReverse}(\mathbf{a})$
for $m = 2; m < n; m = 2m$ **do**
  $\zeta_m = \zeta_n^{\frac{n}{m}}$
  $\zeta = 1$
  for $j = 0; j < \frac{m}{2}; j = j + 1$ **do**
    for $k = 0; k < n; k = k + m$ **do**
      $t = \zeta \cdot a_{k+j+\frac{m}{2}}$
      $u = a_{k+j}$
      $a_{k+j} = (u + t) \pmod p$
      $a_{k+j+\frac{m}{2}} = (u - t) \pmod p$
    $\zeta = \zeta \cdot \zeta_m$
return a

For obtaining the polynomial multiplication $c(x) = a(x) \cdot b(x) \in R_p$, for $R = \mathbb{Z}[x]/(x^n + 1)$, one can use the fact that $x^n + 1 \mid x^{2n} - 1$ to adopt the $2n$-point NTT by zero-padding the input polynomials with zeros in the n highest coefficients. By doing so, performing the reduction modulo $x^n + 1$ on the output of the $2n$-point INTT produces the expected result.

As done for the polynomial $x^n - 1$, it would be more efficient if the NTT could be evaluated directly in the roots of $x^n + 1$, which are given by ζ_{2n}^{2j+1}. By rewriting ζ_{2n}^{2j+1} as $\zeta_{2n} \cdot \zeta_n^j$, it evidences that the n-point NTT can be used for multiplication modulo $x^n + 1$ by scaling the operands as $\hat{a}(x) = \zeta_{2n} \cdot a(x)$. Thus, removing the scaling factor as $c(x) = \zeta_{2n}^{-1} \cdot \hat{c}(x)$ after computing $\hat{c}(x) = \mathrm{INTT}(\mathrm{NTT}(\hat{a}(x), \zeta_n) \odot \mathrm{NTT}(\hat{b}(x), \zeta_n), \zeta_n^{-1})$ produces the expected result in R_p. This technique of scaling the input polynomials by ζ_{2n} is referred to as the *negative wrapped convolution*.

For avoiding pre and post-computation steps, the powers of ζ_{2n} were incorporated in the NTT via Cooley-Tukey by Roy et al. [124]. Similarly, the powers of ζ_{2n}^{-1} were merged with the INTT via Gentleman-Sande by Pöppelmann et al. [120]. As a result, the state-of-the-art NTT allows polynomial multiplication avoiding pre and post-processing phases and also several calls to the bit-reversal procedure [85].

6.4 The Discrete Galois Transform

The Discrete Galois Transform (DGT) is an alternative to the NTT over the finite field $\mathbb{F}_{p^2}$ for a prime number p. The DGT was introduced as a method for integer convolution by Crandall [55] and was further considered for polynomial multiplication by Badawi et al. [22].

counterparts of the elements $\{0, 1, 2, 3, 4, 5, 6, 7\}$ are $\{0, 4, 2, 6, 1, 5, 3, 7\}$.

Let p be a prime number, ζ_n a primitive n-th root of unity in $\mathbb{Z}_p$, and $\mathbf{f}$ a vector such that $f_j \in \mathbb{F}_{p^2}$. The DGT and its inverse transform are defined in Equations 6.13 and 6.14, respectively.

$$F_k = \sum_{j=0}^{n-1} f_j \zeta_n^{jk} \tag{6.13}$$

$$f_k = n^{-1} \sum_{j=0}^{n-1} F_j \zeta_n^{-jk} \tag{6.14}$$

In Equation 6.14, n^{-1} is taken as the multiplicative inverse of n modulo p. Here, the elements in $\mathbb{F}_{p^2}$ are represented using the set of Gaussian integers modulo p, which is defined as $\mathbb{Z}_p[i] = \{a + ib \mid a, b \in \mathbb{Z}_p\}$, for $i = \sqrt{-1}$. The arithmetic in $\mathbb{Z}_p[i]$ is identical to the complex but the real and imaginary parts are taken modulo p.

Consider $a, b \in \mathbb{F}_{p^2}$ represented as in $\mathbb{Z}_p[i]$, with p a prime number. The operations of addition, subtraction, and multiplication in $\mathbb{Z}_p[i]$ are defined as follows.

$$\begin{aligned}(a \pm b) &= (\Re(a) \pm \Re(b)) + i\,(\Im(a) \pm \Im(b)) \\ (a \cdot b) &= (\Re(a)\Re(b) - \Im(a)\Im(b)) + i\,(\Re(a)\Im(b) + \Im(a)\Re(b))\end{aligned}$$

Notice that the multiplications in Equations 6.13 and 6.14 are done between a Gaussian integer, either f_j or F_j, and an integer, which is some power of ζ_n. In this case, since $\Im(b) = 0$, the above formula for multiplication in $\mathbb{Z}_p[i]$ simplifies to

$$\Re(f_j)\Re(\zeta_N^{jk}) + i\left(\Im(f_j)\Re(\zeta_N^{jk})\right),$$

requiring only two integer multiplications. This fact will be used to estimate the total number of operations required for computing a DGT transform using the algorithms of Badawi et al. [22] and the new algorithms proposed in Section 6.5.1.

6.4.1 The Negacyclic Case

Crandall [55] defined the *folding* and *twisting* procedures that allow the computation of the negacyclic convolution (6.2) via DGT. The folding procedure maps an input operand $\mathbf{f} \in \mathbb{Z}_p^n$ to $\mathbf{F} \in \mathbb{Z}_p[i]^{\frac{n}{2}}$, halving its length:

$$F_j = f_j + i f_{j+\frac{n}{2}} \in \mathbb{Z}_p[i], \quad j \in \left[\frac{n}{2}\right].$$

The inverse transform from $\mathbb{Z}_p[i]^{\frac{n}{2}}$ to $\mathbb{Z}_p^n$ is denoted as *unfolding* and it is given by

$$f_j = \Re(F_j) \quad \text{and} \quad f_{j+\frac{n}{2}} = \Im(F_j), \quad j \in \left[\frac{n}{2}\right].$$

Also, the twisting procedure multiplies the folded vector $\mathbf{F}$ by powers of $\omega_{\frac{n}{2}}$, a primitive $\frac{n}{2}$-th root of i modulo p:

$$F_j = F_j \cdot \omega_{\frac{n}{2}}^j.$$

The corresponding inverse convolution multiplies a vector $\mathbf{F} \in \mathbb{Z}_p[i]^{\frac{n}{2}}$ by powers of the

inverse primitive $\frac{n}{2}$-th root of i modulo p used in the forward step:

$$F_j = F_j \cdot \omega_{\frac{n}{2}}^{-j}.$$

Hence, the polynomial multiplication $a(x) \cdot b(x) \mod x^n + 1$ via DGT is computed by applying the folding procedure to $\mathbf{a} = (a_0, \ldots, a_{n-1})$ and $\mathbf{b} = (b_0, \ldots, b_{n-1})$, the coefficient vectors of the operands, resulting in the vectors $\mathbf{a}', \mathbf{b}' \in \mathbb{Z}_p[i]^{\frac{n}{2}}$. Then, each vector $\mathbf{a}'$ and $\mathbf{b}'$ is twisted by $\omega_{\frac{n}{2}}$ and transformed via DGT. After computing the point-wise multiplication in the DGT domain, the result of the polynomial multiplication is given by computing the IDGT, the removal of the twisting factors, and the unfolding procedure, in this order.

In comparison with the NTT, the main advantage of using the DGT is the negacyclic convolution being computed with half-sized transforms. Because of that, storing the powers of ζ_n requires only half of the space in memory. However, the DGT imposes some additional constraints:

- The component-wise multiplication is performed in $\mathbb{Z}_p[i]$ instead of $\mathbb{Z}_p$;
- The input polynomials have to be folded and twisted beforehand;
- A procedure is required for finding a primitive $\frac{n}{2}$-th root of i modulo p;
- The aforementioned procedure requires that $4n \mid (p-1)$ instead of $2n \mid (p-1)$, in comparison with the NTT.

Notice that, for a pair of values for n and p, a primitive $\frac{n}{2}$-th root of i modulo p can be precomputed only once. At first, the DGT was proposed for prime numbers p that are Gaussian integers, that is, prime numbers satisfying $p \equiv 3 \pmod 4$ [55]. Then, Badawi et al. showed that the DGT transform can be applied using primes $p \equiv 1 \pmod 4$. Recall that any prime number either satisfies $p \equiv 3 \pmod 4$ or $p \equiv 1 \pmod 4$. Additionally, Badawi et al. proposed an algorithm for finding primitive $\frac{n}{2}$-th roots of i modulo p when $p \equiv 1 \pmod 4$.

The next section revisits such an algorithm, extending the discussion started by the authors [4, Section V.B] and showing that only primes $p \equiv 1 \pmod 4$ are suitable for polynomial multiplication using the DGT when the polynomial degree is a power of two.

Generating Primitive Roots Modulo p

For polynomial multiplication with reduction via negacyclic convolution, the DGT requires the existence of a primitive $\frac{n}{2}$-th root of unity modulo p and a primitive $\frac{n}{2}$-th root of i modulo p. Assume that n is always a power of two and, for readability, let $N = \frac{n}{2}$. First, we analyze the requirement of a primitive N-th root of unity modulo p.

Let p be a prime number and $a \in \mathbb{N}$. By Fermat's Little Theorem, it follows that

$$a^p \equiv a \pmod p.$$

When a and p are relative primes, then

$$a^{p-1} \equiv 1 \pmod p. \tag{6.15}$$

By definition, a primitive root of a prime p is an integer r such that $r \pmod p$ has multiplicative order $p-1$, that is, $r^{p-1} \equiv 1 \pmod p$. Then, in Equation 6.15, a is a primitive root of p. From Equation 6.15, it is possible to determine primitive k-th roots of unity by computing $a^{\frac{p-1}{k}} \pmod p$, for every k which satisfies $k \mid (p-1)$. In our case, it is required that $N \mid (p-1)$, with N being a power of two.

If $p \geq 3$ is a prime number, then either $p \equiv 3 \pmod 4$ or $p \equiv 1 \pmod 4$. In the first case, we have that p is a number of the form $p = 4k+3$, with $k \in \mathbb{Z}$. Thus, $(p-1) = 4k+2 = 2(2k+1)$. Since $k \in \mathbb{Z}$, then $2k+1$ is an odd number. In this case, $(p-1)$ is only divisible by a power of two 2^ℓ when ℓ is zero or one. Considering that the dimension of the polynomial ring is always much bigger than two, prime numbers such that $p \equiv 3 \pmod 4$ are not suitable for cryptographic purposes. On the other hand, when $p \equiv 1 \pmod 4$, it follows that p is a prime number of the form $p = 4k+1$, with $k \in \mathbb{Z}$. Thus, $(p-1) = 4k$ and k can be either an odd or even number. In this case, $(p-1)$ can be a power of two divisible by N.

Because of that, from hereon, we focus on the case when p is a prime number such that $p \equiv 1 \pmod 4$, presenting the algorithm of Badawi et al. for finding primitive N-th roots of i modulo p.

Generating primitive N-th roots of i modulo p. Badawi et al. proposed an efficient algorithm for determining a primitive N-th root of i modulo a prime number $p \equiv 1 \mod 4$ [4]. For describing the algorithm, we define relevant properties of Gaussian integers and prove a result on the factorization of a prime number $p \equiv 1 \pmod 4$ in $\mathbb{Z}_p[i]$.

Let $z = a+ib$ be a Gaussian integer. The norm of z is $\mathrm{N}(z) = \mathrm{N}(a+ib) = (a+ib)(a-ib) = a^2+b^2$, that is, $\mathrm{N}(z) = z \cdot \overline{z}$ An element $z \in \mathbb{Z}[i]$ is a *Gaussian prime* if one of the following properties is satisfied:

- If both a and b are nonzero, then $a+ib$ is a Gaussian prime if and only if a^2+b^2 is a prime number.
- If $a = 0$, then ib is a Gaussian prime if and only if $|b|$ is a prime number and $|b| \equiv 3 \pmod 4$.
- If $b = 0$, then a is a Gaussian prime if and only if $|a|$ is a prime number and $|a| \equiv 3 \pmod 4$.

Lemma 6.1 ([135]). *Let p be a prime number such that $p \equiv 1 \pmod 4$. Then, there is a Gaussian prime z such that $p = z \cdot \overline{z}$.*

Proposition 6.2 ([135]). *For each prime number $p \equiv 1 \pmod 4$, there are exactly two Gaussian primes z and $\overline{z}$ with norm p.*

Badawi et al.'s algorithm is based on Lemma 6.1 and Proposition 6.2 and comprehends three steps: *i*) finding the factorization of p into two Gaussian primes, which are denoted as z_0 and z_1; *ii*) finding a generator g_i for the cyclic groups defined by each z_j, with $j \in \{0,1\}$; and *iii*) computing a primitive N-th root of i using the Chinese Remainder Theorem (CRT).

1. Finding the factorization of p. Algorithm 6 gives a procedure for factorization of p in the set of Gaussian integers.

Algorithm 6: FactorIntoGaussianPrimes: Find elements z_0 and z_1 such that $p = z_0 \cdot z_1$

Input: A prime p such that $p \equiv 1 \pmod 4$
Output: Gaussian primes z_0 and z_1 such that $p = z_0 \cdot z_1$

do
  $a \xleftarrow{\$} \mathbb{Z}_p$
while $a^{(p-1)/2} \not\equiv -1 \pmod p$
$k = a^{(p-1)/4} \mod p$
$z = \gcd(p, k + i)$
return $(z_0, z_1) = (z, \overline{z})$

From Equation 6.15, Algorithm 6 assumes that if p is a prime, then $a^{p-1} \equiv 1 \pmod p$ for all $a \in \mathbb{Z}_p$. Hence, the square root of a^{p-1} must be equivalent to either 1 or -1. In the latter case, we can find a number k^2 such that $k \equiv a^{(p-1)/4} \equiv i \pmod p$. In other words, if $k^2 \equiv -1 \pmod p$ then $k^2 + 1 \equiv 0 \pmod p$ and p divides $k^2 + 1$. Since $k^2 + 1$ factors into $(k + i) \cdot (k - i)$, we may have found the factorization of p in $\mathbb{Z}_p[i]$. In order to determine the exact factors of p, we introduce Lemma 6.4, which requires the result of Lemma 6.3.

Lemma 6.3 ([135]). *If $z \in \mathbb{Z}[i]$ is such that $\mathrm{N}(z)$ is a prime number, then z is a Gaussian prime.*

Lemma 6.4. *Let p be an odd prime such that $p \equiv 1 \pmod 4$ and let $k \in \mathbb{Z}_p$. The greatest common divisor of p and $k + i$ is either $k + i$ or a Gaussian prime z such that $z \mid p$ and $z \mid k + i$.*

Proof. By Fermat's theorem on sums of two squares[2], we have that $p = a^2 + b^2$, with $a, b \in \mathbb{Z}$. Since $a^2 + b^2 = (a + ib)(a - ib)$ and $\mathrm{N}(a + ib) = \mathrm{N}(a - ib) = p$, then $a + ib$ and $a - ib$ are Gaussian primes and $p = (a + ib)(a - ib)$ is the unique factorization of p in $\mathbb{Z}_p[i]$, disregarding the order of factors[3]. On the other hand, we have that $(k + i)(k - i) \equiv p \pmod p$, by construction. Combining the two facts, we obtain that $p = (a + ib)(a - ib) \equiv (k + i)(k - i)$, which is equivalent to $(k + i)(k - i) = \ell(a + ib)(a - ib)$ for some $\ell \in \mathbb{Z}$. When $\ell = 1$, we have an equality and we find that $(k + i)$ and $(k - i)$ are indeed the factors of p. When $\ell \neq 1$, $(k + i)$ is not a Gaussian prime and still can be factored in $\mathbb{Z}_p[i]$; otherwise, it would be a factor of p. We know that p divides $(k + i)(k - i)$ but not $k + i$, or its conjugate, since $k < p$ and $(k + i)/p$ is not a Gaussian integer. Then, $k + i$ and p must share a common factor z that can be found as the greatest common divisor. Since the two factors of p are $a + ib$ and $a + ib$, z must be one of them. Finally, the factors of p can be found by computing the greatest common divisor of p and $k + i$ and its conjugate.

[2]Fermat's theorem on sums of two squares states that an odd prime p can be expressed as $p = a^2 + b^2$, with $x, y \in \mathbb{Z}$, if and only if $p \equiv 1 \pmod 4$.

[3]Wuthrich proves in Theorem 5.8 that any $0 \neq z \in \mathbb{Z}[i]$ has a unique factorization [135].

Since $p = a^2 + b^2$ and $\mathrm{N}(a + ib) = \mathrm{N}(a - ib) = a^2 + b^2$, by Lemma 6.3, the factors are Gaussian primes. □

Still, there is no guarantee that $k + i$ is a Gaussian prime. By Lemma 6.4, we find that the greatest common divisor of p and $k + i$ is $k + i$ or there exists some z such that $z \mid p$ and $z \mid k + i$. Thus, since $z = \gcd(p, k + i)$ results in a Gaussian prime, we take it as the first factor of p. From Lemma 6.1, $\overline{z}$ is the second factor.

2. Finding a generator. From the first step, we have two Gaussian primes defining a cyclic group corresponding to the set of Gaussian integers modulo z_j for $j = \{0, 1\}$. So, the next step consists in using Algorithm 7 to find a generator g_j for each cyclic group.

Algorithm 7: SampleGenerator: Find a generator g for the cyclic group $\mathbb{Z}_p[i] \pmod z$

Input: A prime number p, and $z \in \mathbb{Z}_p[i]$ a factor of p
Output: A generator g for the cyclic group $\mathbb{Z}_p[i] \pmod z$

```
do
    a ←$ Z_p
    b ←$ Z_p
    g = (a + bi) mod z
    for j = 1; j < p; j = j + 1 do
        d = g^j mod z
        if d = 1 and j = p − 1 then
            return g mod p
while true
```

Algorithm 7 describes a brute-force procedure for finding a generator for the cyclic group $\mathbb{Z}_p[i] \pmod{z_i}$. Elements a and b are sampled uniformly at random from $\mathbb{Z}_p$ to construct a Gaussian integer $g_j \in \mathbb{Z}_p[i] \mod z_j$. The algorithm repeats these steps until an element with multiplicative order $p - 1$ is found. Recall that a primitive N-th root of i modulo p is computed only once for a given pair of values for n and p. Because of that, Algorithm 7 is not heavily invoked in this context. However, Gauss's algorithm can replace the above procedure for finding a generator for the cyclic groups.

3. Constructing a primitive N-th root of i modulo p. The method for computing a primitive N-th root of i modulo p, denoted ω_N, uses Algorithm 6 to find the factorization of p as $p = z_0 \cdot z_1 \in \mathbb{Z}_p[i]$. Then, the generators g_j for $\mathbb{Z}_p[i] \pmod{z_j}$ are found using Algorithm 7. The N-th roots of i modulo p in the corresponding cyclic groups are computed as $\omega^{(j)} = g_j^{\frac{(p-1)}{4N}} \mod z_j$. Lastly, a candidate root is computed in the larger ring $\mathbb{Z}_p[i]$ as $\omega = z_1 \cdot \left(z_1^{-1} \cdot \omega^{(0)} \mod z_0\right) + z_0 \cdot \left(z_0^{-1} \cdot \omega^{(1)} \mod z_1\right) \mod p$. If $\omega^N \equiv i \pmod p$, ω is asserted as a primitive N-th root of i modulo p. The complete procedure for generating ω_N is described in Algorithm 8.

Notice that since Algorithm 7 is not deterministic, so it is not Algorithm 8, producing a different output at each execution.

Algorithm 8: Compute a primitive N-th root of i modulo p

Input: An integer $N = \frac{n}{2}$, and a prime number p satisfying $4n \mid (p-1)$ and $p \equiv 1 \pmod 4$

Output: A primitive N-th root of i modulo p

$(z_0, z_1) = \mathsf{FactorIntoGaussianPrimes}(p)$
do
  for $j = 0; j < 2; j = j + 1$ **do**
    $g_j = \mathsf{SampleGenerator}(p, z_j)$
    $\omega^{(j)} = g_j^{\lfloor (p-1)/(4n) \rfloor} \mod z_j$
  $\omega = z_1 \cdot \left(z_1^{-1} \cdot \omega^{(0)} \mod z_0\right) + z_0 \cdot \left(z_0^{-1} \cdot \omega^{(1)} \mod z_1\right) \mod p$
  if $\omega^N \equiv i \pmod p$ **then**
    return ω
while *true*

6.5 Polynomial Multiplication via DGT

Similarly to FFT and NTT, the DGT can be efficiently computed using both Cooley-Tukey and Gentleman-Sande butterflies. Badawi et al. [4] introduced a DGT transform via Gentleman-Sande and an inverse DGT via Cooley-Tukey, which are presented in Algorithms 9 and 10, respectively.

Algorithm 9: In-place forward DGT via Gentleman-Sande

Input: A folded vector $\mathbf{F} \in \mathbb{Z}_p[i]^N$ in standard order, a prime number p, and ζ_N a primitive N-th root of unity in $\mathbb{Z}_p$

Output: $\mathbf{F} \leftarrow \mathsf{DGT}(\mathbf{F}, p, \zeta_N)$ in bit-reversed order

for $m = N/2; m \geq 1; m = m/2$ **do**
  for $j = 0; j < m; j = j + 1$ **do**
    $a = \zeta_N^{jN/2m}$
    for $i = j; i < N; i = i + 2m$ **do**
      $u = F_i$
      $v = F_{i+m}$
      $F_i = (u + v) \pmod p$
      $F_{i+m} = a(u - v) \pmod p$
return $\mathsf{BitReverse}(\mathbf{F})$

Additionally, Badawi et al. [4] define Algorithm 11 for polynomial multiplication via DGT in $R_q = \mathbb{Z}_q[x]/(x^n + 1)$, with n a power of two and q an integer modulus. The integer moduli p and q are distinguished from hereon: q is adopted in a broader context of the cryptosystem, whereas p is a prime number internal to the procedure of polynomial multiplication. Notice that the input polynomials $a(x), b(x)$ are taken in R_q with the restriction that $a_i, b_i \in \left[0, q \leq \left\lfloor \sqrt{\frac{p}{4N}} \right\rfloor\right]$. In this case, the modulus q acts as an upper bound for the coefficient's growth after integer multiplications as follows. Consider the polynomial multiplication $c(x) = a(x) \cdot b(x) \in R_q$. Then,

$$c_j = \sum_{k=0}^{n} a_j b_k x^{j+k}.$$

Algorithm 10: In-place inverse DGT via Cooley-Tukey

Input: A vector $\mathbf{F} \in \mathbb{Z}_p[i]^N$ in bit-reversed order, a prime number p, and ζ_N a primitive N-th root of unity in $\mathbb{Z}_p$

Output: $\mathbf{F} \leftarrow \mathsf{IDGT}(\mathbf{F}, p, \zeta_N)$ in standard order

$\mathbf{F} = \mathsf{BitReverse}(\mathbf{F})$
for $m = 1; m < N; m = 2m$ **do**
  for $j = 0; j < m; j = j + 1$ **do**
    $a = \zeta_N^{-jN/2m}$
    for $i = j; i < N; i = i + 2m$ **do**
      $u = F_i$
      $v = a \cdot F_{i+m}$
      $F_i = (u + v) \pmod p$
      $F_{i+m} = (u - v) \pmod p$
return $\mathbf{F} \cdot N^{-1}$

Algorithm 11: Polynomial multiplication in R_q via DGT

Input: Polynomials $a(x), b(x) \in R_q$, with $a_i, b_i \in \left[0, q \leq \left\lfloor\sqrt{\frac{p}{4N}}\right\rfloor\right]$, a prime number p satisfying $4n \mid (p-1)$, ζ_N a primitive N-th root of unity in $\mathbb{Z}_p$, and ω_N a primitive N-th root of i modulo p

Output: $c(x) = a(x) \cdot b(x) \in R_q = \mathbb{Z}_q[x]/(x^n + 1)$ with n a power of two

for $j = 0; j < N; j = j + 1$ **do**
  $a'_j = a_j + ia_{j+N}$
  $b'_j = b_j + ib_{j+N}$
for $j = 0; j < N; j = j + 1$ **do**
  $a'_j = \omega_N^j \cdot a'_j \pmod p$
  $b'_j = \omega_N^j \cdot b'_j \pmod p$
$a' = \mathsf{DGT}(\mathbf{a}', p, \zeta_N)$
$b' = \mathsf{DGT}(\mathbf{b}', p, \zeta_N)$
for $j = 0; j < N; j = j + 1$ **do**
  $c'_j = a'_j \cdot b'_j \pmod p$
$c' = \mathsf{IDGT}(\mathbf{c}', p, \zeta_N)$
for $j = 0; j < N; j = j + 1$ **do**
  $c'_j = \omega_N^{-j} \cdot c'_j \pmod p$
for $j = 0; j < N; j = j + 1$ **do**
  $c_j = \Re(c'_j)$
  $c_{j+N} = \Im(c'_j)$
return c

Also, consider that each coefficient of the operands $a(x), b(x)$ are represented in the interval $[0, q)$. Since $|a_j| < q$ and $|b_k| < q$, then $|a_j b_k| = |a_j||b_k| \leq q^2$ and $|c_j| \leq \sum_{k=0}^{n} q^2 \leq nq^2$. Assuming that the coefficients from the output of Algorithm 11 are in the balanced representation $(-p/2, p/2]$, it follows the restriction that $nq^2 \leq \frac{p}{2}$, which is equivalent to the condition $q \leq \left\lfloor\sqrt{\frac{p}{2n}}\right\rfloor$ [22]. Algorithm 11 was proposed for homomorphic encryption schemes [22, 21, 20, 4, 2] in which the size of parameters is huge in comparison with schemes for encryption and digital signatures. In this context, the modulus p is taken as

$p = 2^{64} - 2^{32} + 1$, a 64-bit prime number which special form allows an efficient modular reduction algorithm [22]. Thus, since $p \neq q$, the requirements of the transform only apply to the value of p, providing freedom for choosing the system's modulus q focusing on security estimates.

However, for small cryptosystems such as encryption and digital signatures schemes, the estimated bound for q leads to prohibitive values of p. For Kyber [18], which adopts small values for q and n, we have that $q = 3329$ and $n = 256$. These values require $p \geq 5\,674\,107\,392$, translating into a 32-bit prime number. In order to minimize the effect of computing the polynomial multiplication using a much larger prime number, p can be chosen with a special form, such as a Solinas prime or a pseudo/general Mersenne prime, as already suggested in the literature [55, 22]. Nevertheless, no 32-bit special prime is known, and the closest prime is $2^{31} - 1$, which does not surpass the predetermined upper bound and also $n \nmid (p - 1)$, even for small values of n such as $n = 256$. Additionally, in some cryptosystems like NewHope [9], both ciphertext and public key are transmitted in the transform domain. Thus, replacing the original modulus q with p in the polynomial multiplication would enlarge the bandwidth requirement for transmission and storage of public parameters.

Alternatively, the arithmetic operations can be performed directly in $\mathbb{Z}_p[i]$ assuming $p = q$. In this scenario, the cryptosystem modulus q must consider the restrictions imposed by the underlying transform. In particular, the DGT requires that $n \mid (p - 1)$ and $4n \mid (p - 1)$ for the existence of a primitive N-th root of unity and a primitive N-th root of i modulo p, respectively.

6.5.1 Merging Twisting with DGT

Inspired by the independent work of Roy et al. [124] and Pöppelmann et al. [120], we introduce new algorithms for computing the DGT transform merging the twisting procedure with the DGT via Cooley-Tukey. Similarly, we merge the inverse twisting with the IDGT via Gentleman-Sande.

Merging Twisting with DGT via Cooley-Tukey

As discussed by Crandall [55], the folding and twisting procedures provide the following N-point DGT transform:

$$F'_r = \sum_{j=0}^{N-1} \omega_N^j F_j \zeta_N^{jr}, \tag{6.16}$$

which can be used for computing polynomial reduction modulo x^n+1 for free via negacyclic convolution. We refer to the summation in Equation 6.16 as the problem to be solved using the divide-and-conquer strategies presented in Section 6.2 for the DFT.

For a Decimation-In-Time algorithm, the coefficients F_j in Equation 6.16 are grouped

into even- and odd-indexed sets as follows:

$$\begin{aligned} F'_r &= \sum_{k=0}^{\frac{N}{2}-1} \omega_N^{2k}\ F_{2k}\ \zeta_N^{2kr} + \sum_{k=0}^{\frac{N}{2}-1} \omega_N^{2k+1}\ F_{2k+1}\ \zeta_N^{(2k+1)r} \\ &= \sum_{k=0}^{\frac{N}{2}-1} \omega_{\frac{N}{2}}^{k}\ F_{2k}\ \zeta_{\frac{N}{2}}^{kr} + \omega_N \zeta_N^r \sum_{k=0}^{\frac{N}{2}-1} \omega_{\frac{N}{2}}^{k}\ F_{2k+1}\ \zeta_{\frac{N}{2}}^{kr}. \end{aligned} \tag{6.17}$$

Then, the coefficients are renamed as $G_k = F_{2k}$ and $H_k = F_{2k+1}$ and the sub-problems G'_r and H'_r are defined as

$$G'_r = \sum_{k=0}^{\frac{N}{2}-1} \omega_{\frac{N}{2}}^{k} G_k \zeta_{\frac{N}{2}}^{kr} \quad \text{and} \quad H'_r = \sum_{k=0}^{\frac{N}{2}-1} \omega_{\frac{N}{2}}^{k} H_k \zeta_{\frac{N}{2}}^{kr}.$$

From Equation 6.17, the first $\frac{N}{2}$ coefficients of the solution for the original problem F'_r are obtained by computing $F'_r = G'_r + \omega_N \zeta_N^r H'_r$ for $r \in \left[\frac{N}{2}\right]$. Similarly, the second half of the original problem is derived as

$$\begin{aligned} F'_{r+\frac{N}{2}} &= \sum_{k=0}^{\frac{N}{2}-1} \omega_{\frac{N}{2}}^{k}\ G_k\ \zeta_{\frac{N}{2}}^{k\left(r+\frac{N}{2}\right)} + \omega_N\ \zeta_N^{r+\frac{N}{2}} \sum_{k=0}^{\frac{N}{2}-1} \omega_{\frac{N}{2}}^{k}\ H_k\ \zeta_{\frac{N}{2}}^{k\left(r+\frac{N}{2}\right)} \\ &= \sum_{k=0}^{\frac{N}{2}-1} \omega_{\frac{N}{2}}^{k}\ G_k\ \zeta_{\frac{N}{2}}^{kr} - \omega_N\ \zeta_N^r \sum_{k=0}^{\frac{N}{2}-1} \omega_{\frac{N}{2}}^{k}\ H_k\ \zeta_{\frac{N}{2}}^{kr}, \end{aligned} \tag{6.18}$$

and the solution to the second half of F'_r is obtained from Equation 6.18 by combining the solutions from the sub-problems G'_r and H'_r as $F'_{r+\frac{N}{2}} = G'_r - \omega_N \zeta_N^r H'_r$.

Notice that the coefficients in F'_r, G'_r, and H'_r are multiplied by the primitive root of i modulo p corresponding to the problem size. Specially, considering that the sub-problems are divided until a small enough problem is reached, the next iteration replaces $\omega_{\frac{N}{2^j}}$ with $\omega_{\frac{N}{2^{j+1}}}$, for $j \geq 0$. In summary, the powers of ζ_N at the j-th level are multiplied by $\omega_N^{2^j}$, with $j \in \{\log_2 N - 1, \ldots, 0\}$.

As a result, we introduce Algorithm 12 for computing the forward DGT using the Cooley-Tukey butterfly without requiring bit-reversal and precomputation steps. We refer to this new algorithm as *DGT-T* in which the letter 'T' stands for the twisting procedure. Given the powers of ζ_N in bit-reversed order, the algorithm takes as input a folded vector $\mathbf{F} \in \mathbb{Z}_p[i]^N$ in standard order and outputs $\mathsf{DGT}(\mathbf{F})$ in bit-reversed order [49]. Notice that all arithmetic operations in Algorithm 12 are performed in $\mathbb{Z}_p[i]$.

Merging Inverse Twisting with IDGT via Gentleman-Sande

Given the component-wise multiplication of two polynomials $a(x)$ and $b(x)$ evaluated by Equation 6.16, the result of the polynomial multiplication modulo $x^n + 1$ is recovered by first computing

$$f'_r = \omega_N^{-r} \sum_{\ell=0}^{N-1} F_\ell\ \zeta_N^{-\ell r}, \quad r \in [N], \tag{6.19}$$

Algorithm 12: DGT-T: in-place forward DGT via Cooley-Tukey

Input: A vector $\mathbf{F} \in \mathbb{Z}_p[i]^N$ in standard order, a prime number p, ζ_N a primitive N-th root of unity in $\mathbb{Z}_p$, and ω_N a primitive N-th root of i modulo p

Output: $\mathbf{F} \leftarrow \mathsf{DGT\text{-}T}(\mathbf{F}, p, \zeta_N, \omega_N)$ in bit-reversed order

```
1  d = N/2
2  for m = 1; m < N; m = 2m do
3      for k = 0; k < m; k = k + 1 do
4          j1 = 2kd
5          j2 = j1 + d − 1
6          a = ζ_N^{BitReverse(k)} · ω_N^d
7          for j = j1; j ≤ j2; j = j + 1 do
8              u = a · F_{j+d}
9              F_{j+d} = (F_j − u) (mod p)
10             F_j = (F_j + u) (mod p)
11     d = d/2
12 return F
```

and unfolding the vector containing the coefficients f'_r. In Equation 6.19, the scaling by N^{-1} was intentionally omitted.

Similarly to the forward DGT, the computation of the inverse DGT can be seen as a problem $\mathbf{f}'$ of size $N = \frac{n}{2}$ that can be recursively divided into two sub-problems $\mathbf{g}'$ and $\mathbf{h}'$ with half of the size. The definition of the sub-problems derives from Equation 6.19 as follows:

$$\begin{aligned} f'_r &= \omega_N^{-r} \left(\sum_{j=0}^{\frac{N}{2}-1} F_j \ \zeta_N^{-jr} + \sum_{j=\frac{N}{2}}^{N-1} F_j \ \zeta_N^{-jr} \right) \\ &= \omega_N^{-r} \left(\sum_{j=0}^{\frac{N}{2}-1} F_j \ \zeta_N^{-jr} + \sum_{j=0}^{\frac{N}{2}-1} F_{j+\frac{N}{2}} \ \zeta_N^{-\left(j+\frac{N}{2}\right)r} \right) \\ &= \omega_N^{-r} \sum_{j=0}^{\frac{N}{2}-1} \left(F_j + F_{j+\frac{N}{2}} \ \zeta_N^{-r\frac{N}{2}} \right) \zeta_N^{-rj}, \quad r \in [N]\,. \end{aligned}$$

Thus, when r is even, $f'_r = f'_{2k}$:

$$\begin{aligned} f'_{2k} &= \omega_N^{-2k} \sum_{j=0}^{\frac{N}{2}-1} \left(F_j + F_{j+\frac{N}{2}} \ \zeta_N^{-2k\frac{N}{2}} \right) \zeta_N^{-2kj} \\ &= \omega_N^{-2k} \sum_{j=0}^{\frac{N}{2}-1} \left(F_j + F_{j+\frac{N}{2}} \right) \zeta_N^{-2kj}, \quad k \in \left[\frac{N}{2} \right]. \end{aligned}$$

Consider that the sub-problem $\mathbf{g}'$ is defined as $g'_k = f'_{2k}$ and the coefficients are renamed as $g_j = (F_j + F_{j+\frac{N}{2}})$. Then,

$$g'_k = \omega_{\frac{N}{2}}^{-k} \sum_{j=0}^{\frac{N}{2}-1} g_j \zeta_{\frac{N}{2}}^{-kj}. \tag{6.20}$$

Similarly, when r is odd, the coefficients f'_r are computed as

$$\begin{aligned} f'_{2k+1} &= \omega_N^{-(2k+1)} \sum_{j=0}^{\frac{N}{2}-1} \left(F_j + F_{j+\frac{N}{2}}\ \zeta_N^{-(2k+1)\frac{N}{2}} \right) \zeta_N^{-(2k+1)j} \\ &= \omega_N^{-2k}\ \omega_N^{-1} \sum_{j=0}^{\frac{N}{2}-1} \left(F_j - F_{j+\frac{N}{2}} \right) \zeta_N^{-2kj}\ \zeta_N^{-j} \\ &= \omega_N^{-2k} \sum_{j=0}^{\frac{N}{2}-1} \left(\left(F_j - F_{j+\frac{N}{2}} \right) \zeta_N^{-j}\ \omega_N^{-1} \right) \zeta_N^{-2kj}, \quad k \in \left[\frac{N}{2} \right]. \end{aligned}$$

Then, the coefficients of $\mathbf{h}'$ are $h'_k = f'_{2k+1}$ and $h_j = (F_j - F_{j+\frac{N}{2}})\zeta_N^{-j}\ \omega_N^{-1}$. It follows that

$$h'_k = \omega_{\frac{N}{2}}^{-k} \sum_{j=0}^{\frac{N}{2}-1} h_j \zeta_{\frac{N}{2}}^{-kj}, \tag{6.21}$$

and both sub-problems $\mathbf{g}'$ and $\mathbf{h}'$ are solvable using the same algorithm for the original problem $\mathbf{f}'$. From the above formulas, we introduce Algorithm 13, a Decimation-In-Frequency algorithm for computing the IDGT, which is referred to as *IDGT-T*. In this algorithm, no bit-reversal step is required in the output because the input vector $\mathbf{F}$ and the powers of ζ_N are assumed to be in bit-reversed order.

Algorithm 13: IDGT-T: in-place inverse DGT via Gentleman-Sande

Input: A vector $\mathbf{F} \in \mathbb{Z}_p[i]^N$ in bit-reversed order, a prime number p, ζ_N a primitive N-th root in $\mathbb{Z}_p$, and ω_N a primitive N-th root of i modulo p

Output: $\mathbf{F} \leftarrow \mathsf{IDGT\text{-}T}(\mathbf{F}, p, \zeta_N, \omega_N)$ in standard order

```
1  d = 1
2  for m = N; m > 1; m = m/2 do
3      for j1 = 0; j1 < d; j1 = j1 + 1 do
4          r = 0
5          for i = j1; i < N − 1; i = i + 2d do
6              u = F_i
7              v = F_{i+d}
8              F_i = (u + v) (mod p)
9              F_{i+d} = ζ_N^{−BitReverse(r)} ω_N^{−d} (u − v) (mod p)
10             r = r + 1
11     d = 2d
12 return F · N^{−1} (mod p)
```

Comparing Equation 6.19 with Equations 6.20 and 6.21 shows that ω_N^{-1} is replaced by $\omega_{\frac{N}{2}}^{-1}$ in the corresponding sub-problems. Considering that the problem is recursively divided until a small problem is reached, we have that $\omega_{\frac{N}{2^j}}$ is replaced by $\omega_{\frac{N}{2^{j+1}}}$ in the next iteration. Then, in Algorithm 13, it suffices to multiply the powers of ζ_N by $\omega_N^{-2^j}$ for $j \in \{0, \ldots, \log_2 N - 1\}$.

6.5.2 Experimental Results on x64 Architecture

In this section, we report experimental results for the implementation of DGT in the x64 architecture. We considered round-two submitters of NIST's Post-Quantum Cryptography standardization project [101] that adopt the NTT for polynomial multiplication. In particular, we selected submitters that could benefit from DGT considering the requirement on the modulus p being a prime number satisfying $4n \mid (p-1)$. Thus, we replaced the portable-C implementation of NTT in the submitter's reference code with constant-time implementations for DGT. First, the DGT-T performance is compared with the DGT formulation of Badawi et al. [4]. The experiments indicated that Badawi et al.'s formulation provides a faster algorithm for polynomial multiplication in the x64 architecture. Because of that, we adopted their procedures in further experiments. Finally, we compared AVX2-optimized implementations of both transforms in the qTESLA digital signature scheme [30].

Among the round-two submitters of NIST's Post-Quantum Cryptography standardization project, the cryptosystems adopting the NTT for polynomial multiplication are CRYSTALS-Kyber [18], CRYSTALS-Dilithium [64], NewHope [9], FALCON [69], and qTESLA [30]. For CRYSTALS-Kyber, the ring parameters are $p = 3329$ and $n = 256$. Since $1024 \nmid 3328$, Kyber was not considered for this experiment. For both NewHope and FALCON, it follows that $p = 12\,289$ and $n = 1024$. Thus, we opted by NewHope, since FALCON only uses the NTT in public-key operations. Shortly, considering the objective of selecting distinct pairs of values for the ring parameters (p, n), the experiments were executed for CRYSTALS-Dilithium, NewHope and qTESLA.

The experimental results were obtained by collecting the median clock cycles for 10 000 executions on an Intel Skylake with processor Intel Core i7-6700K running at the constant clock frequency of 4000 MHz with both Turbo Boost and Hyperthreading technologies disabled. The source code was compiled using the GCC compiler version 11.2.1. Appendix A presents additional experimental results on an Intel Haswell with processor Intel Core i7-4770 running at the constant clock frequency of 3400 MHz. The parameter sets considered for the experiments are in Table 6.1. The parameters k and ℓ determine the module rank in the scope of CRYSTALS-Dilithium; especially, a public matrix $\mathbf{A}$ is uniformly sampled in $(R_q)^{k \times \ell}$.

Table 6.1: Parameter sets considered for comparing the NTT and DGT transforms.

Cryptosystem	p	n	(k, ℓ)
Dilithium II	8 380 417	256	(4,3)
Dilithium III	8 380 417	256	(5,4)
Dilithium IV	8 380 417	256	(6,5)
NewHope512CCA	12 289	512	(1,1)
NewHope1024CCA	12 289	1024	(1,1)
qTESLA-p-I	343 576 577	1024	(1,1)
qTESLA-p-III	856 145 921	2048	(1,1)

Performing Polynomial Multiplication with DGT-T

In this section, we argue that Badawi et al.'s formulation of DGT better suits small cryptosystems such as CRYSTALS-Dilithium [64], NewHope [9], and qTESLA [30] in comparison with DGT-T. In general, the arithmetic operations in DGT are performed over Gaussian integers. However, the multiplications in Algorithms 9 and 10 are done between Gaussian integers and powers of ζ_N, a primitive N-th root of unity in $\mathbb{Z}_p$. In other words, the multiplications are performed between Gaussian integers and integers. In turn, the multiplications in our algorithms rely entirely on $\mathbb{Z}_p[i]$ because ζ_N is multiplied by $\omega_N \in \mathbb{Z}_p[i]$. Thus, even though Badawi et al.'s and our algorithms have the asymptotic cost $O(n \log n)$, the overall number of integer operations are quite different. In Algorithm 14, we describe our algorithm for polynomial multiplication in a power-of-two cyclotomic ring using DGT-T algorithms, for which we analyze the overall number of operations.

Algorithm 14: Polynomial multiplication in R_p via DGT-T

Input: Polynomials $a(x), b(x) \in R_p$, and a prime number p satisfying $4n \mid (p-1)$
Output: $c(x) = a(x) \cdot b(x) \in R_p = \mathbb{Z}_p[x]/(x^n + 1)$ with n a power of two

for $j = 0; j < N; j = j + 1$ **do**
    $a'_j = a_j + i a_{j+N}$
    $b'_j = b_j + i b_{j+N}$
$a' = \mathsf{DGT\text{-}T}(\mathbf{a}', p, \zeta_N, \omega_N)$
$b' = \mathsf{DGT\text{-}T}(\mathbf{b}', p, \zeta_N, \omega_N)$
for $j = 0; j < N; j = j + 1$ **do**
    $c'_j = a'_j \cdot b'_j \pmod{p}$
$c' = \mathsf{IDGT\text{-}T}(\mathbf{c}', p, \zeta_N, \omega_N)$
for $j = 0; j < N; j = j + 1$ **do**
    $c_j = \Re(c'_j)$
    $c_{j+N} = \Im(c'_j)$
return c

Consider $a, b \in \mathbb{Z}_p[i]$. The multiplication $a \cdot b \in \mathbb{Z}_p[i]$ is defined as

$$(a \cdot b) = (\Re(a)\Re(b) - \Im(a)\Im(b)) + i\,(\Re(a)\Im(b) + \Im(a)\Re(b))\,. \tag{6.22}$$

Thus, a multiplication in $\mathbb{Z}_p[i]$ requires four multiplications (M) and two additions (A) in $\mathbb{Z}_p$. On the other hand, a multiplication between a Gaussian integer and an integer consumes 2M plus 2A, which is equivalent to assuming that $\Im(b) = 0$ in Equation 6.22. In this sense, computing a polynomial multiplication using Algorithm 11 has the overall cost of

$$C_{\mathsf{DGT}} = 2\left(\frac{n}{2}\,(4\mathrm{M} + 2\mathrm{A})\right) + 2\left(\frac{n}{2}\log\left(\frac{n}{4}\right)(2\mathrm{M} + 4\mathrm{A})\right).$$

Moreover, a polynomial multiplication using Algorithm 14 invoking the DGT-T expends $C_{\mathsf{DGT\text{-}T}}$ integer operations:

$$C_{\mathsf{DGT\text{-}T}} = 2\left(\frac{n}{2}\log\left(\frac{n}{4}\right)(4\mathrm{M} + 6\mathrm{A})\right).$$

If one tries to determine whether $C_{\mathsf{DGT\text{-}T}} < C_{\mathsf{DGT}}$, a contradiction is found:

$$\log\left(\frac{n}{4}\right)(2\mathrm{M} + 2\mathrm{A}) \leq 4\mathrm{M} + 2\mathrm{A}.$$

For the Dilithium cryptosystem, for which $n = 256$ is the smallest ring degree considered in our analysis, the above inequality leads to $6\mathrm{M} + 6\mathrm{A} > 2\mathrm{M} + \mathrm{A}$, showing that our algorithm consumes four extra integer multiplications. Besides, each multiplication usually requires a Montgomery reduction [97] and additions may require a Barrett reduction [24] when lazy reduction is not possible. Then, the additional cost of Algorithm 14 in comparison with Algorithm 11 makes the proposed algorithm not attractive for polynomial multiplication in power-of-two cyclotomic rings in the cryptographic context.

Notice that using Karatsuba [81] for computing the multiplication in Equation 6.22 would reduce the number of integer multiplications at the expense of increasing the number of additions, which have a similar cost for small integers in modern x64 architectures. Nonetheless, Algorithm 14 might be of independent interest for another application.

For completeness, we present next experimental results obtained by using both DGT's formulations in selected round-two submissions of NIST's Post-Quantum Cryptography standardization project [102].

For experimentally evaluating the DGT's variants, we ran a polynomial multiplication procedure using the parameter sets corresponding to Dilithium II, NewHope1024CCA, and qTESLA-p-I. The results are summarized in Table 6.2. Notice that the ring parameters of CRYSTALS-Dilithium are identical in all parameter sets. Thus, in a standalone evaluation of polynomial multiplication, any parameter set would have the same performance.

Table 6.2: Skylake cycle counts for the median of 10 000 executions of CRYSTALS-Dilithium, NewHope, and qTESLA reference code using both versions of the DGT: Badawi et al.'s version presented in Section 6.4, denoted DGT, and our formulation combining the transform computation with the twisting procedure denoted as DGT-T.

	Dilithium II			**NewHope1024CCA**		
	DGT	DGT-T	DGT/DGT-T	DGT	DGT-T	DGT/DGT-T
Poly. Mul.	26 334	38 846	0.68	142 814	218 672	0.65

	qTESLA-p-I		
	DGT	DGT-T	DGT/DGT-T
Poly. Mul.	97 434	114 435	0.85

As a result, the DGT-T increases the cost of polynomial multiplication via DGT up to 53%. For a fair comparison, the polynomial multiplication procedure was executed in the context of the corresponding cryptosystems, adopting optimization techniques used in the reference codes for the NTT.

Because of that, the experiments in the following sections consider Badawi et al.'s formulation of DGT, applying the twisting procedure outside the scope of the transforms.

A Comparison of NTT and DGT Transforms

In this section, we implement the polynomial multiplication via DGT as in Algorithm 11 proposed by Badawi et al. [4], which invokes Algorithm 9 for the forward transform and Algorithm 10 for its inverse. The DGT implementations ran in constant time and are written in portable-C programming language. For each cryptosystem, namely CRYSTALS-Dilithium [64], NewHope [9], and qTESLA [30], the polynomial multiplication module using the NTT was replaced with our implementation based on the DGT. We used the same code-level optimization techniques as in the reference code to allow a fair comparison between both implementations.

Next, we present experimental results for our implementations, comparing their performance with the reference codes. The results for CRYSTALS-Dilithium, NewHope and qTESLA are summarized in Tables 6.3, 6.4, and 6.5, respectively.

Table 6.3: Skylake cycle counts for the median of 10 000 executions of CRYSTALS-Dilithium reference code using either the NTT or DGT transform.

	Dilithium II			**Dilithium III**		
	Reference	Ours	NTT/DGT	Reference	Ours	NTT/DGT
Keypair	284 048	291 066	0.98	404 813	422 813	0.96
Sign	1 156 893	1 194 617	0.97	1 586 314	1 692 240	0.94
Verify	287 561	302 921	0.95	394 411	419 900	0.94
Transform	7011	7784	0.90	6540	7252	0.90
Point-wise Mul.	1231	2085	0.59	1010	1698	0.59
Inv. Transform	9725	8441	1.15	7736	7965	0.97

	Dilithium IV		
	Reference	Ours	NTT/DGT
Keypair	539 496	561 041	0.96
Sign	1 472 362	1 575 270	0.93
Verify	546 682	574 469	0.95
Forward Transform	6547	6972	0.94
Point-wise Mul.	1224	1842	0.66
Inverse Transform	7734	7795	0.99

Table 6.4: Skylake cycle counts for the median of 10 000 executions of the NewHope reference code using either the NTT or DGT transform.

	NewHope512CCA			**NewHope1024CCA**		
	Reference	Ours	NTT/DGT	Reference	Ours	NTT/DGT
Keypair	117 593	124 872	0.94	222 445	240 229	0.93
Encaps	163 558	169 052	0.97	320 449	336 735	0.95
Decaps	180 173	184 762	0.98	360 227	374 513	0.96
Forward Transform	19 998	22 513	0.89	41 555	45 713	0.91
Point-wise Mul.	4694	7731	0.61	9808	15 463	0.63
Inverse Transform	21 342	16 554	1.29	44 203	35 388	1.25

In our implementations, the powers of a primitive $\frac{n}{2}$-th root of unity and the powers of a primitive $\frac{n}{2}$-th root of i modulo p are precomputed. In particular, the $\frac{n}{2}$-th root of i

Table 6.5: Skylake cycle counts for the median of 10 000 executions of qTESLA reference code using either the NTT or DGT transform.

	qTESLA-p-I			**qTESLA-p-III**		
	Reference	Ours	NTT/DGT	Reference	Ours	NTT/DGT
Keygen	2 336 010	2 321 057	1.01	13 234 216	13 213 465	1.00
Sign	2 059 141	2 044 792	1.01	4 754 599	4 768 281	1.00
Verify	759 582	745 238	1.02	1 982 184	1 944 773	1.02
Transform	30 374	33 328	0.91	83 043	82 168	1.01
Inv. Transform	32 112	30 720	1.05	73 675	67 575	1.09

modulo p is obtained through a script implementing Badawi et al.'s procedure [4, Section 5.2]. A primitive $\frac{n}{2}$-th root of unity in $\mathbb{Z}_p$ is obtained by calculating $r^{(p-1)/\left(\frac{n}{2}\right)} \pmod p$, where r is the primitive root of p. The primitive root of a prime p can be computed in the Wolfram Language using PrimitiveRoot[p] [133] via WolframAlpha [134].

Now, we strictly analyze the numbers of integer operations modulo p for computing a polynomial multiplication. In the NTT, a Gentlemen-Sande or Cooley-Tukey butterfly consumes one multiplication and two additions/subtractions, taking as input two coefficients. In turn, the same butterfly in the DGT consumes 2M plus 4A by taking as input four coefficients at a time. Since the size of the DGT transform is $\frac{n}{2}$ instead of n, the overall cost is roughly $\frac{n}{2}\log\frac{n}{2}(2\mathrm{M}+4\mathrm{A})$. For the NTT, the estimated cost is $n\log n(1\mathrm{M}+2\mathrm{A})$. Thus, the computation of a DGT transform shows a reduction of $n\log 2(2\mathrm{M}+4\mathrm{A})$ integer operations modulo p.

The polynomial multiplication via DGT requires applying the twisting factors before the forward DGT and removing them after the computation of the inverse transform. Each of these tasks adds a linear cost of $\frac{n}{2}(4\mathrm{M}+2\mathrm{A})$ integer operations modulo p, which is amortized by the reduced cost of computing a half-sized transform. The third aspect of performing polynomial multiplication via DGT is point-wise multiplication, which is computed in $\mathbb{Z}_p[i]^{\frac{n}{2}}$. Thus, it consumes $n(2\mathrm{M}+1\mathrm{A})$ integer operations modulo p, in comparison with only nM operations for NTT.

In summary, the DGT and IDGT are expected to show an improvement for larger values of n and p, as indicated in our experimental results in Tables 6.3, 6.4, and 6.5. The improvement is reinforced by the fact that the DGT has been efficiently used in lattice-based homomorphic encryption schemes [2], which require much larger ring parameters than schemes for digital signatures and public-key encryption.

Additionally, the IDGT is computed using the Gentleman-Sande butterfly, which is expected to provide better performance than the DGT using the Cooley-Tukey butterfly. Usually, Gentleman-Sande is the most efficient butterfly because the additions and subtractions are performed before the multiplication, which may avoid a complete reduction modulo p. However, notice that optimization techniques particular to the reference implementations also influence the final results. In general, since the DGT/IDGT by itself only requires half of the powers of a root of unity, it may consume fewer clock cycles due to cache misses. Nevertheless, it is expected for the point-wise multiplication to have a less efficient behavior than in the NTT.

For CRYSTALS-Dilithium, the original reference code outperforms the DGT by at

most 7% for signature generation in Dilithium IV. In this case, the number of function calls related to polynomial multiplication depends on the parameter k and the number of iterations of the rejection sampling procedure. The point-wise multiplication, which consumes up to 69% more clock cycles, is the most invoked function, being called a quadratic number of times concerning the parameter k.

For NewHope, the performance of polynomial multiplication via NTT and DGT is similar. Although the IDGT is more capable than the INTT of exploring the loop unrolling technique, achieving 1.29 and 1.25 of speedup in our experiments, this advantage is mostly hampered by the additional cost imposed by the point-wise multiplication. Additionally, the forward transform and point-wise multiplication are invoked twice as many times as the IDGT.

Finally, for qTESLA, our implementation is even with the reference source code, attaining 1.02 of speedup for signature verification in both parameter sets. In this context, the cycle count for the inverse transform in Table 6.5 refers to the added cost of executing the point-wise multiplication and the inverse IDGT. In general, this wrapped function is called k times the number of calls to the forward procedure, which is either 4 or 5. In the signature generation, this ratio is multiplied by the number of iterations of the rejection sampling algorithm.

AVX2 Implementation of qTESLA

Considering the slight speedup attained for qTESLA-p-I and qTESLA-p-III in Table 6.5, we present experimental results for polynomial multiplication via DGT in a highly-optimized implementation using AVX2. In Table 6.6, we compare the performance of our AVX2-optimized implementation with the AVX2-optimized code provided by qTESLA's authors.

Table 6.6: Skylake cycle counts for the median of 10 000 executions of qTESLA digital signature scheme using an AVX2-optimized implementation of either NTT or DGT.

	qTESLA-p-I		
	Reference	Ours	NTT/DGT
Keygen	2 218 689	2 212 576	1.00
Sign	1 276 929	1 375 358	0.93
Verify	635 918	638 032	1.00
Transform	10 018	8714	1.15
Inv. Transform	7747	9791	0.79

Unfortunately, in our implementations, the DGT was unable to explore the hardware capabilities offered by the vector instructions. As a result, the performance is highly influenced by the misalignment inherent to the folding procedure and the overhead induced by the multiplication in $\mathbb{Z}_p[i]$, as depicted in Figure 6.1. Recall that Gaussian integer multiplications require the integer multiplication modulo p of the real and imaginary parts of the operands.

Nonetheless, in the next chapter, we present a recursive formulation for the DGT that indicates its good adaptability in the GPU architecture.

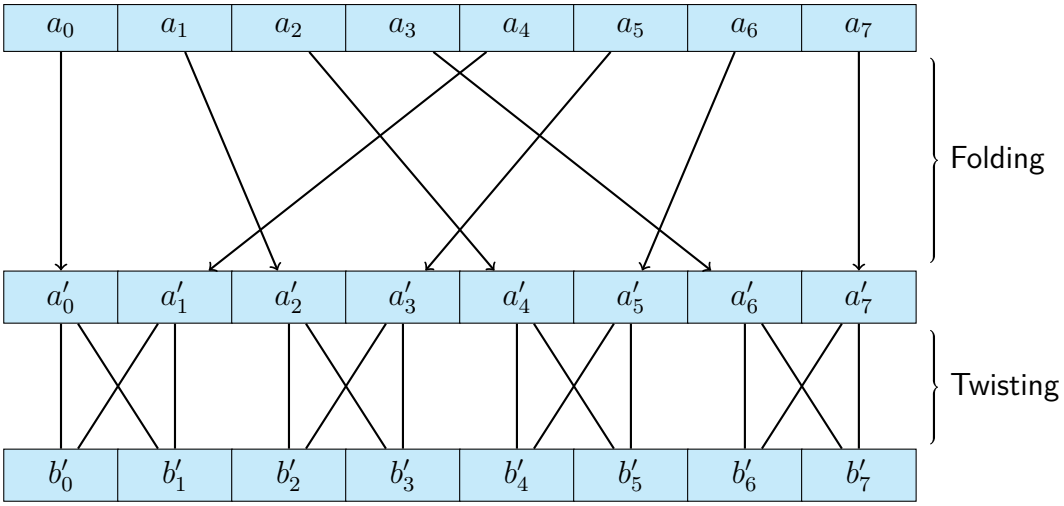

Figure 6.1: Memory accesses during folding and twisting procedures.

Chapter 7

Evaluation of Recursive Algorithms on GPUs

Gentleman and Sande [71] introduced the Recursive FFT (RFFT) and Bailey [23] recaptured its formulation as the four-step FFT algorithm. More recently, the Recursive NTT (RNTT) was introduced in GPU implementations [58, 80] of algorithms for multiplying large integer numbers and polynomials in rings of the form $\mathbb{Z}[x]/(x^n + 1)$, with n a power of two. The RNTT is a simply adaptation of the four-step FFT algorithm, replacing the set of complex numbers by the finite field $\mathbb{Z}_p$. Concurrently, Badawi et al. [4] explored using the DGT for polynomial multiplication in CUDA-implementations of homomorphic encryption schemes.

In this context, we introduce the Recursive DGT (RDGT), which we use for polynomial multiplication in the quotient ring $\mathbb{Z}_p[x]/(x^n+1)$, with n a power of two and p a prime number (Section 7.2). First, we compare a CUDA-implementation of the BFV homomorphic encryption scheme [68] adopting the RDGT with state-of-the-art implementations [22, 2] (Section 7.3). Then, we present experimental results for CUDA-implementations of both RNTT and RDGT in the CKKS homomorphic encryption scheme [45]. Apart from a straightforward comparison of both recursive transforms, RNTT and RDGT are evaluated in the inference phase of Logistic Regression (Section 7.4). As a result, the experimental results indicate that RDGT naturally performs better in the GPU platform. However, the RNTT presents similar performance when the arithmetic density is improved. All experiments were executed in NVIDIA Tesla GPUs running on Google Cloud instances.

Notations. For defining the recursive transforms, consider that a polynomial $a(x) = \sum_{j=0}^{n-1} a_j x^j \in \mathbb{Z}[x]$, for $n = N_r N_c$ a power of two, is conveniently represented as a matrix $\mathbf{A}_{N_r \times N_c}$ containing the coefficients $a_j \in \mathbb{Z}$. The matrix $\mathbf{A}$ can also be indexed as an one-dimensional array by concatenation of rows. The j-th row and the k-th column of $\mathbf{A}_{N_r \times N_c}$ are denoted by $\mathbf{A}_{j,\square} \in \mathbb{Z}^{N_c}$ and $\mathbf{A}_{\square,k} \in \mathbb{Z}^{N_r}$, respectively.

7.1 The Recursive Number-Theoretic Transform

The Recursive NTT (RNTT) is an adaptation of Bailey's four-step FFT [23] that replaces FFT calls by the NTT and takes the twiddle factor ζ_n as a primitive n-th root of unity in $\mathbb{Z}_p$.

In 1966, the Recursive FFT (RFFT) was proposed by Gentleman and Sande [71] and remained forgotten until 1989 when Bailey [23] recaptured it as the four-step FFT algorithm. The required amount of local memory was insufficient to store all the operands for large FFTs. Thus, many data had to be stored in devices such as disks or tapes, in which memory accesses are costly. In this context, the RFFT replaces the computation of a large transform with many minor FFT instances. Consider that the size of the transform is $n = N_r N_c$. The forward RFFT transform on a coefficient vector arranged as a matrix $\mathbf{A}_{N_r \times N_c}$ is as follows.

1. Perform N_c FFTs of size N_r along the columns;
2. Multiply each element $a_{j,k}$ of the matrix $\mathbf{A}_{N_r \times N_c}$ by the corresponding twiddle factor ζ_n^{jk}, for $\zeta_n = \exp(2\pi i/n)$;
3. Perform N_r FFTs of size N_c along the rows; and
4. Transpose $\mathbf{A}$ from $N_r \times N_c$ to $N_c \times N_r$.

Similarly, the inverse RFFT consists of these four steps executed in the reversed order, but IFFT replaces the calls to FFT and the twiddle factors are ζ_n^{-jk}.

Further, in 2018, Dai et al. [58, Section VIII.B] introduced the RNTT in a lattice-based key-policy attribute-based encryption. They recursively apply the Cooley-Tukey butterfly, following an approach similar to the one used by Emmart and Weems [67] for the FFT, which is, in turn, the four-step FFT of Bailey [23]. The forward and inverse RNTTs are given in Algorithm 15 and 16, respectively.

Algorithm 15: RNTT: Recursive NTT

Input: A polynomial $a(x) \in R_p$ treated as a matrix $\mathbf{A}_{N_r \times N_c}$, a prime number p, integer numbers N_r and N_c such that $n = N_r N_c$, and ζ_n a primitive n-th root of unity in $\mathbb{Z}_p$

Output: $\mathbf{A} = \mathsf{RNTT}(\mathbf{A}, p, N_r, N_c, \zeta_n)$

for $k = 0; k < N_c; k = k + 1$ **do**
  $\mathbf{A}_{\square,k} = \mathsf{NTT}(\mathbf{A}_{\square,k}, p, \zeta_n)$
for $j = 0; j < N_r; j = j + 1$ **do**
  for $k = 0; k < N_c; k = k + 1$ **do**
    $a_{j,k} = a_{j,k} \cdot \zeta_n^{jk} \pmod{p}$
for $j = 0; j < N_r; j = j + 1$ **do**
  $\mathbf{A}_{j,\square} = \mathsf{NTT}(\mathbf{A}_{j,\square}, p, \zeta_n)$
return A

Algorithm 16: RINTT: Recursive INTT

Input: A matrix $\mathbf{A}_{N_r \times N_c}$ in the RNTT domain, a prime number p, integer numbers N_r and N_c such that $n = N_r N_c$, and ζ_n a primitive n-th root of unity in $\mathbb{Z}_p$

Output: $a = \mathsf{RINTT}(\mathbf{A}, p, N_r, N_c, \zeta_n)$

for $j = 0; j < N_r; j = j + 1$ **do**
  $\mathbf{A}_{j,\square} = \mathsf{INTT}(\mathbf{A}_{j,\square})$
for $j = 0; j < N_r; j = j + 1$ **do**
  for $k = 0; k < N_c; k = k + 1$ **do**
    $a_{j,k} = a_{j,k} \cdot \zeta_n^{-jk} \pmod{p}$
for $k = 0; k < N_c; k = k + 1$ **do**
  $\mathbf{A}_{\square,k} = \mathsf{INTT}(\mathbf{A}_{\square,k})$
return A

Recall from Section 6.3 that multiplication in $R_p = \mathbb{Z}_p[x]/(x^n + 1)$, with n a power of two and p a prime number, can be efficiently performed via NTT using the negative wrapped convolution. In this sense, Dai et al. adopt the NTT for arithmetic in R_p for $p = 2^{64} - 2^{32} + 1$, implementing the forward and inverse NTT transforms using recursive algorithms. The polynomial multiplication in R_p via RNTT is presented in Algorithm 17, using the fact that $\zeta_{2n}^2 = \zeta_n$.

Algorithm 17: Polynomial multiplication in R_p via RNTT

Input: Polynomials $a(x), b(x) \in R_p$, a prime number p, integer numbers N_r and N_c such that $n = N_r N_c$, and ζ_{2n} a primitive $2n$-th root of unity in $\mathbb{Z}_p$

Output: $c(x) = a(x) \cdot b(x) \in R_p = \mathbb{Z}_p[x]/(x^n + 1)$ with n a power of two

for $j = 0; j < n; j = j + 1$ **do**
  $a_j = a_j \cdot \zeta_{2n}^j \pmod{p}$
  $b_j = b_j \cdot \zeta_{2n}^j \pmod{p}$
$\mathbf{A} = \mathsf{RNTT}(\mathbf{A}, p, N_r, N_c, \zeta_{2n}^2)$
$\mathbf{B} = \mathsf{RNTT}(\mathbf{B}, p, N_r, N_c, \zeta_{2n}^2)$
for $j = 0; j < n; j = j + 1$ **do**
  $c_j = a_j \cdot b_j \pmod{p}$
$\mathbf{C} = \mathsf{RINTT}(\mathbf{C}, p, N_r, N_c, \zeta_{2n}^2)$
for $j = 0; j < n; j = j + 1$ **do**
  $c_j = c_j \cdot \zeta_{2n}^{-j} \pmod{p}$
return C

The forward and inverse recursive transforms invoked in Algorithm 17 are given in Algorithm 15 and 16, respectively. In particular, the negative wrapped convolution is computed in Lines 1 and 9 of Algorithm 17. Thus, the NTT in Algorithm 15 can be computed using any algorithm that does *not* merge the multiplication by ζ_{2n} with the transform.

7.2 The Recursive Discrete Galois Transform

In this section, we propose the Recursive DGT (RDGT) that splits the DGT transform into smaller blocks to avoids the synchronization of large sets of threads in GPUs. Similarly to the RNTT, the RDGT is an adaptation of the four-step FFT algorithm [23] in the finite field $\mathbb{F}_{p^2}$. In this context, the RDGT is used for polynomial multiplication in the ring $R_p = \mathbb{Z}_p[x]/(x^n + 1)$ for n a power of two and p a prime number satisfying $p \equiv 1 \pmod 4$ and $4n \mid (p-1)$. From hereon, consider that $N = \frac{n}{2}$.

The steps for polynomial multiplication via RDGT are the same described in Algorithm 11 (Chapter 6). Shortly, the input polynomials are mapped into vectors of Gaussian integers through the folding procedure. Then, the folded vectors are twisted by ω_N, a primitive N-th root of i in $\mathbb{Z}_p$. The point-wise multiplication in the RDGT domain is done as $\mathbf{c} = \mathsf{RDGT}(\mathsf{twist}(\mathsf{fold}(a(x)))) \odot \mathsf{RDGT}(\mathsf{twist}(\mathsf{fold}(b(x))))$, and the result of the polynomial multiplication is obtained by computing $\mathsf{unfold}(\mathsf{untwist}(\mathsf{RIDGT}(\mathbf{c})))$.

The Recursive DGT is presented in Algorithm 18, in which an N-length vector of Gaussian integers is treated as a matrix $\tilde{\mathbf{A}}$ with dimensions $N_r \times N_c$. In this case, one may choose the values for N_r and N_c to be as close as possible since equality allows reusing precomputed values for the primitive roots, minimizing memory requirements, and achieving workload balance.

Algorithm 18: RDGT: Recursive DGT

Input: A vector of Gaussian integers represented as a matrix $\tilde{\mathbf{A}}_{N_r \times N_c}$, a prime number p, integer numbers N_r and N_c such that $N = N_r N_c$, and ζ_N a primitive N-th root of unity in $\mathbb{Z}_p$

Output: $\tilde{\mathbf{A}} = \mathsf{RDGT}(\tilde{\mathbf{A}}, p, N_r, N_c, \zeta_N)$

1 **for** $k = 0; k < N_c; k = k + 1$ **do**
2 $\tilde{\mathbf{A}}_{\square,k} = \mathsf{DGT}(\tilde{\mathbf{A}}_{\square,k}, p, \zeta_N)$
3 **for** $j = 0; j < N_r; j = j + 1$ **do**
4 **for** $k = 0; k < N_c; k = k + 1$ **do**
5 $\tilde{a}_{j,k} = \tilde{a}_{j,k} \cdot \zeta_N^{\mathsf{BitReverse}(j)\cdot k} \pmod p$
6 **for** $j = 0; j < N_r; j = j + 1$ **do**
7 $\tilde{\mathbf{A}}_{j,\square} = \mathsf{DGT}(\tilde{\mathbf{A}}_{j,\square}, p, \zeta_N)$
8 **return** $\tilde{\mathbf{A}}$

In Algorithm 18, the DGT routine refers to the forward transform via Gentleman-Sande (Algorithm 9), avoiding the bit-reverse procedure at the output. In Line 5, the twiddle factors are $\zeta_N^{\mathsf{BitReverse}(j)\cdot k}$ instead of ζ_N^{jk} because the DGT's output is in bit-reversed order. Thus, the indexes coincide with the position of the corresponding element $\tilde{a}_{j,k}$.

Algorithm 19: RIDGT: Recursive IDGT

Input: A vector $\tilde{\mathbf{A}} \in \mathbb{Z}_p[i]^N$ in the RDGT domain, a prime number p, integer numbers N_r and N_c such that $N = N_r N_c$, and ζ_N a primitive N-th root of unity in $\mathbb{Z}_p$

Output: $\tilde{\mathbf{A}} = \mathsf{RIDGT}(\tilde{\mathbf{A}}, p, N_r, N_c, \zeta_N)$

1 **for** $j = 0; j < N_r; j = j + 1$ **do**
2 $\tilde{\mathbf{A}}_{j,\square} = \mathsf{IDGT}(\tilde{\mathbf{A}}_{j,\square}, p, \zeta_N)$
3 **for** $j = 0; j < N_r; j = j + 1$ **do**
4 **for** $k = 0; k < N_c; k = k + 1$ **do**
5 $\tilde{a}_{j,k} = \tilde{a}_{j,k} \cdot \zeta_N^{-j \cdot \mathsf{BitReverse}(k)} \cdot N_c^{-1} \pmod{p}$
6 **for** $k = 0; k < N_c; k = k + 1$ **do**
7 $\tilde{\mathbf{A}}_{\square,k} = \mathsf{IDGT}(\tilde{\mathbf{A}}_{\square,k}, p, \zeta_N)$
8 **return** $\tilde{\mathbf{A}}$

The inverse counterpart of the RDGT, denoted RIDGT, is described in Algorithm 19. It adopts the IDGT transform via Cooley-Tukey (Algorithm 10) without bit-reversing the input vector. Notice Gentleman-Sande butterflies transform standard order vectors into bit-reversed ones, whereas Cooley-Tukey butterflies perform the opposite transformation. Thus, the bit-reverse procedure can be avoided in both DGT and IDGT.

The RIDGT starts by applying the IDGT over the rows of $\tilde{\mathbf{A}}$. The twiddle factors from the forward transform are removed by multiplying the element $\tilde{a}_{j,k}$ by $\zeta_N^{-j \cdot \mathsf{BitReverse}(k)}$, since the column indexes of the output of the previous step are still in bit-reversed order. Notice that precomputing the values of $\zeta_N^{-j \cdot \mathsf{BitReverse}(k)}$ allows the twiddle factors to be stored multiplied by the scaling factor $N_c^{-1} \pmod{p}$. By doing so, the multiplication by N^{-1} modulo p can be avoided at the end of each call to the IDGT over the rows of $\tilde{\mathbf{A}}$. Finally, after executing the IDGT over the columns of $\tilde{\mathbf{A}}$, the matrix indexes are in standard ordering. Similarly to the scaling factor for the rows of $\tilde{\mathbf{A}}$, the powers of ζ_N^{-1} can be precomputed and multiplied by the scalar $N_r^{-1} \pmod{p}$.

7.3 Comparing RDGT and DGT on GPUs

The Recursive DGT (RDGT) first appeared in our implementation [13] of the BFV[1] homomorphic encryption scheme [68]. Further, we directly compared the RDGT and RNTT transforms [15] in the CKKS homomorphic encryption scheme [45]. The target architecture in both works [13, 15] is NVIDIA GPUs.

Graphics Processing Units (GPUs) were originally designed as a programmable processing unit dedicated to graphics manipulation. A GPU is composed of many small and specialized cores that together delivers a massive performance through a Single Instruction Multiple Data (SIMD) paradigm. Shortly, a GPU processes a strenuous workload by demanding each core to execute concurrently the same task in a distinct piece of data.

[1]BFV refers to the initials of Brakerski, Fan, and Vercauteren. Although authored by Fan and Vercauteren, the BFV is an adaptation of the fully homomorphic encryption scheme of Brakerski [34] to the Ring-LWE setting.

GPUs can be used for general-purpose computing using a programming model such as the Compute Unified Device Architecture (CUDA) platform [104] maintained by NVIDIA.

The computation of FFTs in GPUs faces a memory constraint similar to the one encountered in the 90's decade. Although GPUs provide massively parallel processing, the amount of fast memory is constrained and may become a bottleneck on bigger instances. In this context, Govindaraju et al. [74], and Emmart and Weems [67] independently adopted RFFTs for the computation of DFTs and the multiplication of large integer numbers on GPUs.

Some implementations of homomorphic encryption schemes such as BFV [68] and CKKS [46, 45] adopt polynomial rings with dimensions ranging from 2048 to 65536 [2, 22, 80]. In particular, the DGT has been considered as a replacement to the NTT for polynomial multiplication to reduce the computational complexity of expensive routines such as homomorphic multiplication. In this context, the remaining of Section 7.3 presents experimental results of our BFV implementation [13] and compares the state-of-the-art DGT employed by Badawi et al. in two distinct works [2, 22] with our recursive algorithm for computing the DGT. Then, in Section 7.4 we present our experimental evaluation of CKKS using either RDGT or RNTT [15].

7.3.1 BFV Homomorphic Encryption Scheme

The hardness of the BFV homomorphic encryption scheme is based on the decision version of the Ring-LWE Problem [88]. The plaintext space is taken as R_t for $R = \mathbb{Z}[x]/(x^n + 1)$, with n a power of two, and some integer $t > 1$. Similarly, the ciphertext space is R_q for a modulus q much bigger than t. Details on BFV can be found in [68].

The parameter sets adopted in this experiment are given in Table 7.1, replicating the values in the related works [2, 22]. Notice that the plaintext modulus is fixed at $t = 256$.

Table 7.1: Parameter sets considered for comparing distinct BFV implementation on GPUs.

Related Work	$(\log n, \log q)$				
BPAVR [2]	–	(12,60)	(13,120)	(14,360)	(15,600)
BVMA [22]	(11,62)	(12,186)	(13,372)	(14,744)	–

BFV is based on the encryption scheme of Lyubashevsky, Peikert, and Regev [88], requiring as additional parameters a security parameter λ, a decomposition base w, and a one-dimensional discrete Gaussian distribution $D_{\mathbb{Z},\sigma}$ centered at zero. Polynomials in R_q can be expressed in terms of $\ell + 1$ polynomials in basis w in which $\ell = \lfloor \log_w q \rfloor$. Also, Δ denotes the scaling factor $\Delta = \lfloor q/t \rfloor$ and the notation $[a]_q$ refers to the unique integer in $\mathbb{Z}_q$ with $[a]_q \coloneqq a \mod q$. This notation is coefficient-wise extended to polynomials. The BFV homomorphic encryption scheme defines algorithms for key generation, encryption, decryption, homomorphic addition, relinearization, and homomorphic multiplication, as follows.

$\mathsf{BFV.KGen}(\lambda, w)$: The secret key is a ternary polynomial $\mathsf{sk} \leftarrow R_2$ where each coefficient is taken from the set $\{-1, 0, 1\}$. The public key pk is a pair of polynomials $(b, a) =$

$([-(a \cdot \mathsf{sk} + e)]_q, a)$, where $a \xleftarrow{\$} R_q$ is sampled from a uniformly random distribution and $e \leftarrow D_{\mathbb{Z},\sigma}$. The evaluation key evk is a set of $\ell+1$ pairs of polynomials computed as follows. For $i \in [\ell]$, sample $a_i \xleftarrow{\$} R_q$ uniformly at random and $e_i \leftarrow D_{\mathbb{Z},\sigma}$. Then, evk is a vector in which $\mathsf{evk}_i = \left([w^i \cdot \mathsf{sk}^2 - (a_i \cdot \mathsf{sk} + e_i)]_q, a_i\right)$. Output $(\mathsf{sk}, \mathsf{pk}, \mathsf{evk})$.

$\mathsf{BFV.Enc}(m, \mathsf{pk})$: For a message $m \in R_t$, the ciphertext ct is computed under the public key $\mathsf{pk} = (b, a)$ as $\mathsf{ct} = \left([b \cdot u + \Delta \cdot m + e_1]_q, [a \cdot u + e_2]_q\right)$, where $u \xleftarrow{\$} R_2$ is sampled from a uniformly random distribution and $e_1, e_2 \leftarrow D_{\mathbb{Z},\sigma}$.

$\mathsf{BFV.Dec}(\mathsf{ct}, \mathsf{sk})$: Define $c_0 = \mathsf{ct}_0$, $c_1 = \mathsf{ct}_1$, and $\mathbf{s} = \mathsf{sk} = (b, a)$. The message is recovered as plaintext by computing $m = \left[\left\lfloor \frac{t}{q} [c_0 + c_1 \cdot \mathbf{s}]_q \right\rceil\right]_t$.

$\mathsf{BFV.Add}(\mathsf{ct}^{(0)}, \mathsf{ct}^{(1)})$: For two ciphertexts $\mathsf{ct}^{(0)} = (\mathsf{ct}_0^{(0)}, \mathsf{ct}_1^{(0)})$ and $\mathsf{ct}^{(1)} = (\mathsf{ct}_0^{(1)}, \mathsf{ct}_1^{(1)})$, the homomorphic addition is computed as $\left(\left[\mathsf{ct}_0^{(0)} + \mathsf{ct}_0^{(1)}\right]_q, \left[\mathsf{ct}_1^{(0)} + \mathsf{ct}_1^{(1)}\right]_q\right)$.

$\mathsf{BFV.Relin}(\mathbf{c}, \mathsf{evk})$: For $\mathbf{c} = (c_0, c_1, c_2) \in (R_q)^3$, an evaluation key $\mathsf{evk}_i = (b_i, a_i)$ for $i \in [\ell]$, and a decomposition of c_2 in basis w such that $c_2 = \sum_{i=0}^{\ell} \hat{c}_i w^i$, the relinearization procedure returns

$$\left(\left[c_0 + \sum_{i=0}^{\ell} b_i \cdot \hat{c}_i\right]_q, \left[c_1 + \sum_{i=0}^{\ell} a_i \cdot \hat{c}_i\right]_q\right).$$

$\mathsf{BFV.Mul}(\mathsf{ct}^{(0)}, \mathsf{ct}^{(1)}, \mathsf{evk})$: For two ciphertexts $\mathsf{ct}^{(0)} = (\mathsf{ct}_0^{(0)}, \mathsf{ct}_1^{(0)})$ and $\mathsf{ct}^{(1)} = (\mathsf{ct}_0^{(1)}, \mathsf{ct}_1^{(1)})$, the homomorphic multiplication is given by computing

$$\mathbf{c} = \left(\left[\left\lfloor \frac{t}{q} \cdot \mathsf{ct}_0^{(0)} \cdot \mathsf{ct}_0^{(1)} \right\rceil\right]_q, \left[\left\lfloor \frac{t}{q} \cdot \left(\mathsf{ct}_0^{(0)} \cdot \mathsf{ct}_1^{(1)} + \mathsf{ct}_1^{(0)} \cdot \mathsf{ct}_0^{(1)}\right) \right\rceil\right]_q, \left[\left\lfloor \frac{t}{q} \cdot \mathsf{ct}_1^{(0)} \cdot \mathsf{ct}_1^{(1)} \right\rceil\right]_q\right),$$

and returning the output of the relinearization $\mathsf{BFV.Relin}(\mathbf{c}, \mathsf{evk})$.

7.3.2 Experimental Results

We implemented the BFV homomorphic encryption scheme [68] in a library named SPOG[2] [14] using CUDA platform. The experimental environment is inherited from the literature, which comprehends the NVIDIA Tesla V100 and NVIDIA Tesla K80 GPUs.

Table 7.2 contains the raw average running time, in milliseconds, for the average of 100 independent executions of BFV in a Tesla K80 GPU. Notice that, in this context, the authors' source code was not made publicly available to the community. Thus, our results are compared to the values provided by Badawi et al. in their paper [22].

The speedups obtained in the Tesla K80 GPU for encryption, decryption, addition, and multiplication are depicted in Figure 7.1. Consider that, in real-world applications, the use of HE is motivated by their ability to perform additions and multiplications homomorphically. In this sense, these arithmetic operations are expected to be the most

[2]Acronym to Secure Processing on GPGPUs. GPGPU is an abbreviation for General-Purpose Graphics Processing Unit.

Table 7.2: Tesla K80 running time, in milliseconds, for the average of 100 independent executions of SPOG and Badawi et al.'s implementation of BFV [22].

	$(\log n, \log q)$							
	(11,62)		(12,186)		(12,186)		(12,186)	
	Ours	**[22]**	**Ours**	**[22]**	**Ours**	**[22]**	**Ours**	**[22]**
Encryption	0.303	0.541	0.309	1.440	0.575	2.645	1.630	6.657
Decryption	0.089	0.151	0.098	0.194	0.191	0.252	0.542	0.610
Addition	0.009	0.037	0.010	0.052	0.021	0.068	0.066	0.127
Multiplication	0.926	3.343	1.214	3.873	3.061	7.700	13.914	28.953

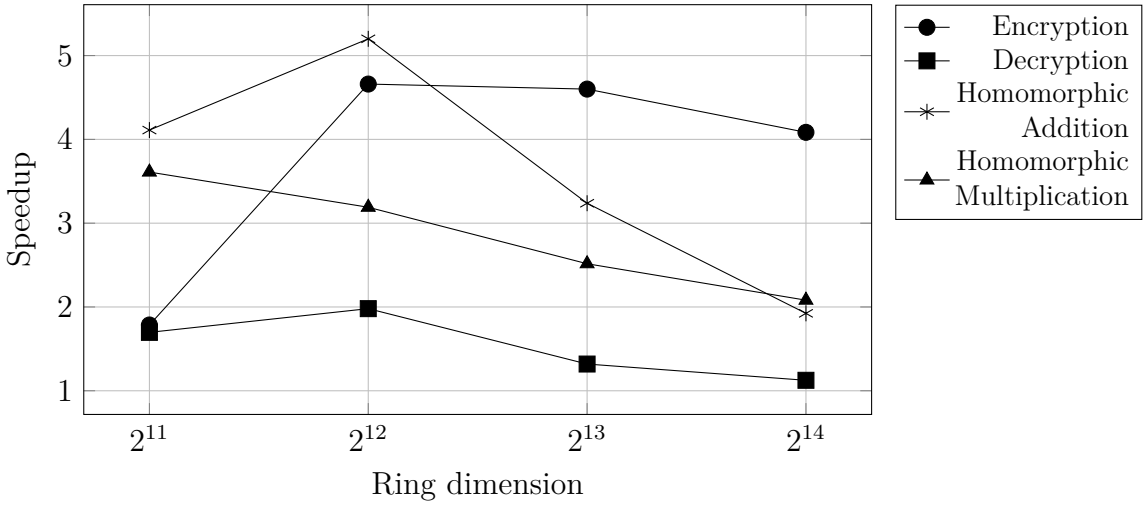

Figure 7.1: SPOG speedups in a Tesla K80 GPU in comparison with Badawi et al.'s implementation of BFV [22].

invoked functions on the GPU. Furthermore, due to the key management policy, the key generation procedure only has to be executed at application setup or in case of key expiration. In turn, encryption is used for creating a private database in a third-party service, and decryption is executed by the data owner when retrieving the computation's output. The speedups for homomorphic addition range from 1.9 to 5.2, but these results are due to complementary optimizations since DGT is not executed during addition. However, the speedups are expressive for homomorphic multiplication, which highly depends on polynomial multiplication, comprehending 3.6, 3.2, 2.5, and 2.1.

Table 7.3: Tesla V100 running time, in milliseconds, for the average of 100 independent executions of SPOG and Badawi et al.'s implementation of BFV [2].

	$(\log n, \log q)$							
	(12,60)		(13,120)		(14,360)		(15,600)	
	Ours	**[2]**	**Ours**	**[2]**	**Ours**	**[2]**	**Ours**	**[2]**
Decryption	0.029	0.054	0.031	0.059	0.049	0.087	0.099	0.111
Multiplication	0.423	0.859	0.472	1.012	0.823	2.010	2.325	4.826

In Table 7.3, experimental results for decryption and homomorphic multiplication on a Tesla V100 GPU are reported in comparison with Badawi et al.'s implementation [2],

whose source code was not made publicly available to the community. In this case, the authors only provided latencies for these two BFV procedures. Similarly, the speedups are depicted in Figure 7.2. For homomorphic multiplication, the main optimization focus, the speedups are consistent in the range of 2.0 and 2.4. For decryption, the results degrade from 1.9 to 1.1. However, the client may execute the decryption in less powerful hardware, as discussed before.

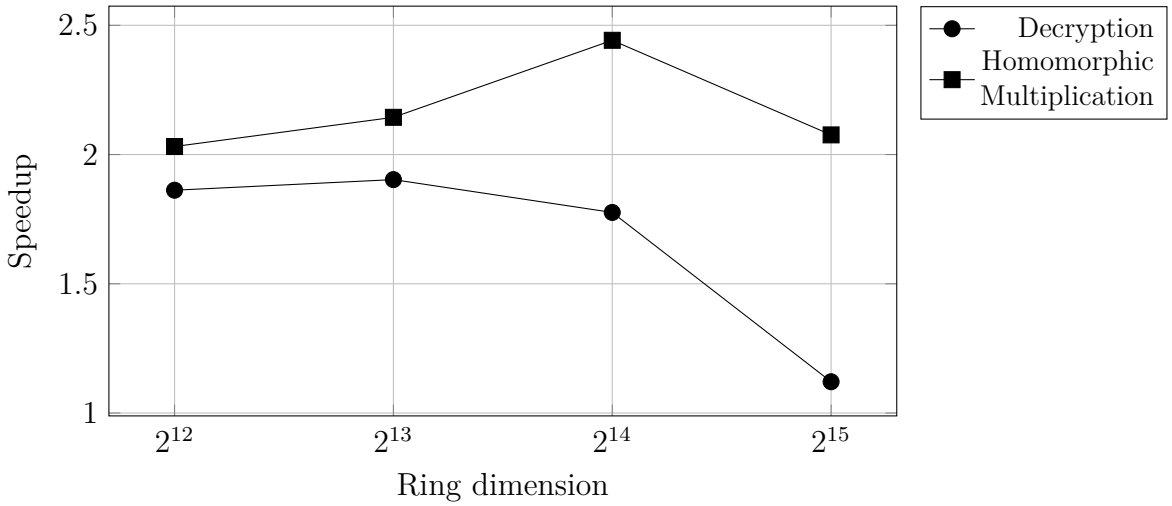

Figure 7.2: SPOG speedups in a Tesla V100 GPU in comparison with Badawi et al.'s implementation of BFV [2].

7.4 Comparing RDGT and RNTT on GPUs

In a further experiment [15], we developed CUDA implementations of the CKKS homomorphic encryption scheme [45] adopting either the RNTT or RDGT transform. Our implementations were evaluated in the NVIDIA Tesla V100 and Tesla A100 GPUs.

Cheon-Kim-Kim-Song proposed a leveled homomorphic encryption scheme known as CKKS [45] in which the plaintext domain is composed of complex numbers. Similar to BFV, CKKS defines algorithms for key generation, encryption, decryption, homomorphic addition, and homomorphic multiplication. However, the homomorphic multiplication requires extending the ciphertext representation from an element in a RNS basis $\mathcal{C}$ to an element in the composed basis $\mathcal{C} + \mathcal{D}$, for a secondary basis $\mathcal{D}$. Then, a multiplication by an evaluation key is performed on this extended basis, and the result is compressed to $\mathcal{C}$. These basis conversions are done through approximate modulus switching functions referred to as ModUp and ModDown. Details on CKKS can be found in [45].

In this context, the Residue Number System (RNS) is used to represent polynomials with large integer coefficientes. Consider that $\mathcal{C} = \{q_0, \ldots, q_\ell\}$ is a set of coprime integers and $q = \Pi_{i=0}^{\ell} q_i$. If $s \in R_{\mathcal{C}}$, there exists $s' \in R_q$ such that $s := \{[s']_{q_0}, \ldots, [s']_{q_\ell}\}$. For example, a CKKS ciphertext, denoted $\mathsf{ct} = (c_0, c_1)$, is a pair of elements in $R_{\mathcal{C}}$, for $\mathcal{C}$ a RNS basis defined as above. In other words, $\mathsf{ct} = \{(c_{0,i}, c_{1,i}\}_{0 \leq i \leq \ell}$ such that $(c_{0,i}, c_{1,i}) \in R_{q_i} \times R_{q_i}$.

Consider that GPUs contains several Streaming Multiprocessors (SM) and modern SMs execute groups of 32 threads at a time, called warps, which are the primary processing

unit in a GPU. Also, recall from Section 7.1 and 7.2 that recursive algorithms compute transforms of size N_r or N_c such that $n = N_r \cdot N_c$, where the input polynomial has degree at most $n - 1$. Thus, N_c blocks of $\lceil N_r/2 \rceil$ threads are needed to compute the transform over the columns. Similarly, the computation of the corresponding transform over the rows requires N_r blocks of $\lceil N_c/2 \rceil$ threads.

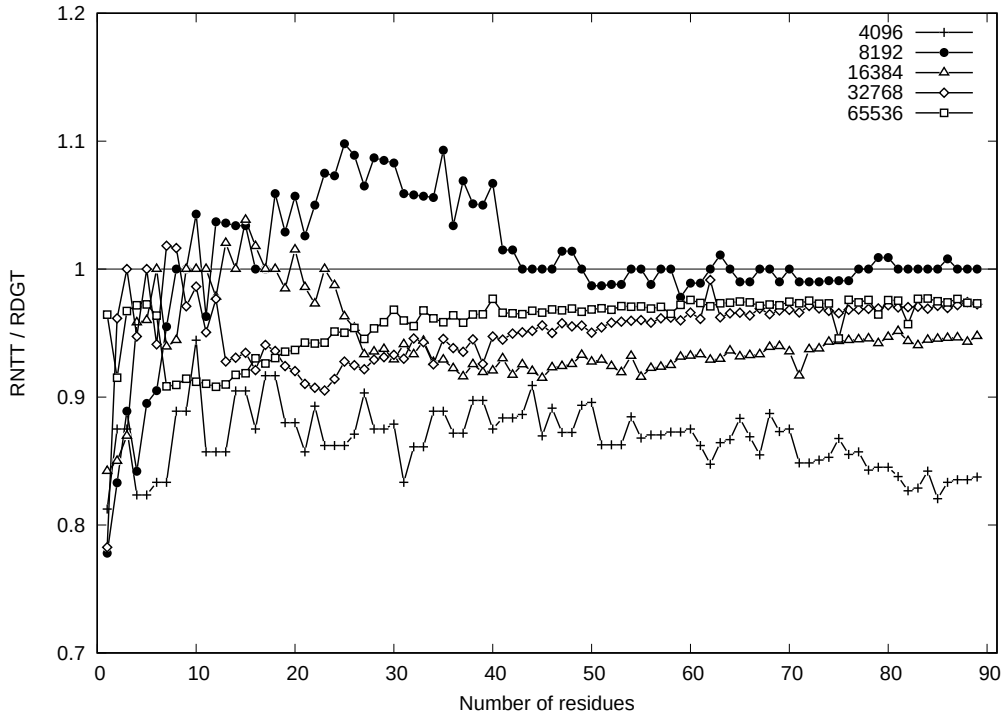

Figure 7.3: Ratio (RNTT/RDGT) of Tesla V100 GPU running time, in milliseconds, for the average of 100 independent executions of RNTT and RDGT on varying polynomial degrees and RNS basis sizes.

In Figure 7.3, we present the ratio RNTT/RDGT for the average running time of RNTT and RDGT transforms on an NVIDIA Tesla V100 GPU. The behavior of the recursive transforms is mostly evidenced in 4096- and 8192-degree polynomial rings. In RDGT, 4096-degree polynomials are folded into 2048-length vectors of Gaussian integers. Since $2048 = 64 \cdot 32$, there will be blocks with $32/2 = 16$ threads running in the GPU, which do not reach the CUDA warp size. Since warps are only composed of threads contained in the same block, blocks smaller than 32 imply wasting SM resources. In RNTT, the operands are processed as 4096-degree polynomials. Thus, $4096 = 64 \cdot 64$, and all blocks are set with 32 threads, fitting in a warp perfectly. On the other hand, the RDGT benefits from the SM processing when $n = 8192$. In this case, some RNTT thread blocks have 64 elements, but the RDGT enters in its optimal setup with 32-thread blocks. The RNTT starts to deviate from its optimal configuration with $n = 8192$. At the same time, RDGT achieves its best performance, showing a speedup for RNS bases containing 10 to 45 residues. For large polynomial rings, the performance of both RDGT

and RNTT suffer from the increased shared memory consumption, which raises bank conflicts. Moreover, thread block synchronizations become costly since the thread blocks need to be split among several warps.

We extended the straightforward comparison of RNTT and RDGT to homomorphic operations. The homomorphic addition consists in the point-wise addition of the corresponding residues represented in a given RNS basis. Thus, we focus on homomorphic multiplication which arithmetic density is much higher, as depicted in Algorithm 20. In Algorithm 20, Transform and InverseTransform refer to the forward and inverse transforms of either RNTT or RDGT. In turn, BasisExtension, ModUp, and ModDown are algorithms for basis manipulation.

Algorithm 20: CKKS homomorphic multiplication

Input: Ciphertexts $\mathsf{ct}_0 = \mathsf{CKKS.Enc}(m_0)$ and $\mathsf{ct}_1 = \mathsf{CKKS.Enc}(m_1)$, and evk an evaluation key

Output: A ciphertext ct_2 such that $\mathsf{ct}_2 = \mathsf{CKKS.Enc}(m_0 \cdot m_1)$

1 $\widehat{\mathsf{ct}}_0 = \mathsf{Transform}(\mathsf{ct}_0)$
2 $\widehat{\mathsf{ct}}_1 = \mathsf{Transform}(\mathsf{ct}_1)$
3 $\hat{d} = \mathsf{BasisExtension}(\widehat{\mathsf{ct}}_0, \widehat{\mathsf{ct}}_1)$
4 $d = \mathsf{InverseTransform}(\hat{d})$
5 $e_2 \coloneqq \mathsf{ModUp}_{\mathcal{C}_\ell \leftarrow \mathcal{D}_\ell}(d_2)$
6 $\hat{e}_2 = \mathsf{Transform}(e_2)$
7 $\widehat{\mathsf{ct}} \coloneqq \hat{e}_2 \cdot \mathsf{evk}$
8 $\mathsf{ct} = \mathsf{InverseTransform}(\widehat{\mathsf{ct}})$
9 $a \coloneqq \mathsf{ModDown}_{\mathcal{D}_\ell \leftarrow \mathcal{C}_\ell}(\mathsf{ct})$
10 $\mathsf{ct}_2 \coloneqq (a_0 + d_0, a_1 + d_1)$
11 **return** ct_2

In Table 7.4, we provide experimental results for each component of the homomorphic multiplication on an instance with $n = 2^{16}$, $\log q = 1831$, and RNS bases of 53 and 54 residues. NVIDIA's profiler showed that the basis extension procedures are critical in homomorphic multiplication. For RNTT, the ModUp and ModDown functions consumes 61% and 23% of the homomorphic multiplication overall running time. Also, these same procedures consumes 43% and 30% of RDGT execution time, respectively.

In particular, the profiling tool indicated that ModUp is roughly two times slower in RNTT in comparison with RDGT. We concluded that the processor scheduler is being less efficient since our implementation of ModUp for RNTT issues 2.2 times more instructions than in RDGT. Because of that, our implementation of basis extension was refactored for the RNTT to induce the processor dual-issue by increasing arithmetic density. As a result, we emulated the processor behavior when executing the RDGT, improving the ModUp procedure and reducing the previous slowdown of 0.48 to 0.92. In Table 7.4, RNTT_{opt} refers to the RNTT using the optimized basis conversion algorithm. Notice that the folding procedure of RDGT naturally increases the arithmetic density of the transform, indicating its best suitability for the GPU platform.

Table 7.4: NVIDIA Tesla V100 GPU running time for subroutines of homomorphic multiplication using either RDGT or RNTT for polynomial multiplication.

	RDGT	**RNTT**	**RNTT$_{\text{opt}}$**	**RDGT/RNTT**	**RDGT/RNTT$_{\text{opt}}$**
ModUp	2377.0	4928.9	2592.8	0.48	0.92
ModDown	1659.2	1854.7	1896.6	0.89	0.87
Transform	1131.2	1054.8	1062.8	1.07	1.06
Integer Op.	354.3	227.3	203.2	1.56	1.74
Total	5521.7	8065.7	5755.5	0.68	0.96

7.4.1 Case Study: Homomorphic Logistic Regression

In this section, we present experimental results for the RDGT and RNTT in a CUDA implementation of Logistic Regression (LR). Our implementations were evaluated in the NVIDIA Tesla V100 and Tesla A100 GPUs.

Logistic Regression (LR) is a machine learning method for predicting a binary outcome based on continuous variables. In particular, LR takes as input datasets composed by n records of the form $(\mathbf{x}_i, y_i)$, with $y_i \in \{0, 1\}$ and $\mathbf{x}_i \in \mathbb{R}^d$. The probability, denoted y, is modeled as a function of the input variables $\mathbf{x}$ as

$$\Pr[y = 1 \mid \mathbf{x}, \mathbf{w}] = \frac{1}{1 + \exp(-\langle (1 \mid \mathbf{x}), \mathbf{w} \rangle)}. \tag{7.1}$$

The training phase aims at learning the weight vector $\mathbf{w}$, which is done by computing a $\mathbf{w}_{\text{approx}}$ that sufficiently approximates $\mathbf{w}$. Thus, the value of y for a new sample $\mathbf{x}$ can be inferred with a predictable accuracy by evaluating the probability $\Pr[y = 1 \mid \mathbf{x}, \mathbf{w}_{\text{approx}}]$ using Equation 7.1. In fact, the probability y cannot be computed homomorphically. In this sense, similar works that also implement LR inference on homomorphic encryption schemes avoid its computation by assuming $\langle (1 \mid \mathbf{x}), \mathbf{w} \rangle$ as the classification result [28]. An alternative approach is to approximate the probability computation using the related Taylor series expansion [76].

Notice that the training phase is done much less frequently than inference. Also, approximating $\mathbf{w}_{\text{approx}}$ is computationally costly and a delicate procedure that may require thousands of multiplications to achieve a suitable $\mathbf{w}_{\text{approx}}$. Consequently, our evaluation of recursive transforms in LR is restricted to the online inference phase. The inference phase is composed of an online and an offline phase. The computation during the online phase is intensive and can be executed homomorphically on a powerful device such as a GPU. The dataset is kept encrypted during this time, and the decryption key is not required. In the offline phase, the probability matrix is decrypted, and the prediction is made by selecting the index of the maximum element.

In this experiment, we adopt the MNIST dataset [84], a data collection of handwritten digits that are represented by images, each with 784 pixels. The digit recognition problem involves classifying m images among $d = 10$ classes of digits, each image having its pixels serialized as an array. The MNIST images are split into a training and a test dataset containing 60 000 and 10 000 records, respectively. The training and test datasets are used for generating the weight vector $\mathbf{w}_{\text{approx}}$ and to determine the accuracy of the generated

model, which is 0.9167, in our case. Shortly, the model is a matrix of real numbers containing $d = 10$ weight vectors $\mathbf{w}_i$ of length $n = 784$. In this context, the LR inference can be executed on an encrypted model or in plaintext. Plaintext models contemplate cases where the model owner is a server contracted to evaluate third-party user data. In turn, when encrypted, the third party will perform computations on the user's model.

For homomorphically evaluating the inner product in Equation 7.1, the model can be treated in both traditional and transposed forms. The same is done for the dataset containing the m images of length n for the inference phase. Each model row is regularly encrypted in a ciphertext having their columns distributed through n slots. For encrypting the images, a set of m images becomes a set of m ciphertexts, each using n slots. Although the traditional approach optimizes memory consumption, it requires a sequence of slot rotations. Conversely, in the transposed form, the image dataset becomes a matrix of n rows and m columns, and each row is encrypted to a single ciphertext. However, the inner product requires each model element to be encrypted in a single ciphertext, implying a considerable increase in memory requirement. Shortly, the traditional and transposed ciphertext designs require $d + m = 10 + 10\,000 = 10\,010$ and $n \cdot (d + 1) = 784 \cdot 11 = 8624$ ciphertexts, respectively. For comparing the two designs, consider that the traditional method performs two homomorphic multiplications for computing the inner product, whereas the transposed one requires one multiplication.

Table 7.5 presents NVIDIA Tesla V100 GPU running times, in microseconds, for computing the online phase of the inference on one record in the MNIST test dataset. The results refer to the minimum parameter set that provides a 128-bit security level and the required multiplicative depth for each approach. The traditional approach runs with $n = 2^{13}$ and $\log q = 167$ while the transposed version runs with $n = 2^{15}$ and $\log q = 115$, offering 172-bit and 1343-bit security level, respectively, according to Albrecht's estimator [8].

Table 7.5: Tesla V100 GPU running time, in microseconds, for computing the online inference of LR per record. The classification result is obtained as the inner product of the input sample and the weights.

	Plaintext		**Encrypted**	
	Traditional	**Transposed**	**Traditional**	**Transposed**
RNTT_{opt}	12 827.6	15.2	14 396.8	238.5
RDGT	13 885.9	15.6	15 461.2	226.8
RNTT_{opt}/RDGT	0.92	0.97	0.93	1.05

In our experiments, the procedure for accumulating the homomorphic multiplication during inner product impacted the RDGT latency on the traditional approach. Nonetheless, the transpose approach avoids the summation of ciphertext slots; thus, RDGT and RNTT show similar execution times. Also, the optimized basis extension for RNTT did not produce a significant impact on the overall result.

Recall that so far the classification result is assumed to be $\langle (1 \mid \mathbf{x}), \mathbf{w} \rangle$ instead of computing the probability as in Equation 7.1. In the following experiment, we approximate the probability function using a polynomial obtained by the truncation at the eighth

term of its Taylor series approximation [1]. This method is evaluated through Horner's rule, increasing by eight the number of homomorphic multiplications needed for inference, affecting performance and memory consumption. Because of that, we evaluated the LR inference on an NVIDIA Tesla A100 GPU, which offers more CUDA cores and bigger memory size. Table 7.6 presents our experimental results, in microseconds, for computing the inference per record in the MNIST test dataset. The traditional design runs with $n = 2^{14}$ and $\log q = 583$, whereas the transposed version runs with $n = 2^{15}$ and $\log q = 531$, offering 94-bit and 225-bit security level, respectively [8].

Table 7.6: Tesla A100 GPU running time, in microseconds, for computing the online inference of LR per record. The classification result is obtained as an approximation of the probability function.

	Plaintext		**Encrypted**	
	Traditional	**Transposed**	**Traditional**	**Transposed**
RNTT	60 968.6	33.4	65 998.5	562.6
RNTT_{opt}	56 053.7	32.5	58 200.0	517.6
RDGT	56 013.2	33.1	60 279.7	475.7
RNTT/RDGT	1.09	1.01	1.09	1.18
RNTT_{opt}/RDGT	1.00	0.98	0.97	1.09
$\text{RNTT}/\text{RNTT}_{\text{opt}}$	1.09	1.03	1.13	1.09

In major instances, like those considered in Table 7.6, the higher arithmetic density of RDGT implied in speedups when compared to the non-optimized RNTT. However, the RNTT with optimized basis extension, denoted NTT_{opt}, is capable of reversing that scenario. In the plaintext transposed design, both implementations present a consistent similarity. Notice that homomorphic addition and multiplication are taken coefficient-wise when the model is in plaintext.

Although the RDGT's folding procedure halves the transform length, it does not imply in reduction of the required number of operations. However, the scalability of a DGT-based implementation seems promising, reducing the execution time of complex methods as the rotation of ciphertext slots within basis extension procedures. This behavior agrees with the conclusion in Section 7.4, which suggests that the RDGT implementation better explore CUDA's paradigm. Also, the results corroborate the fact that the RNTT can perform at least as well as the RDGT when increasing its arithmetic density using the optimized ModUp procedure.

Part III

Final Remarks

Chapter 8

Conclusions

Apart from quantum resistance, lattice-based cryptography offers fast and energy-saving public-key primitives, as evidenced by Saarinen's [105] results presented in Tables 1.1 and 1.2. Furthermore, advances on lattice cryptography in recent years allowed the practical construction of functional encryption schemes, specially Homomorphic Encryption (HE) schemes such as BGV [35], BFV [68], and CKKS [45], which stimulated the development of several general-purpose libraries. Examples include SEAL [127], HElib [75], cuHE [59], among others.

In this context, the objectives of this thesis are threefold: *i*) to provide alternative instantiations for Ring-LWE cryptography, aiming at number fields other than cyclotomic; *ii*) to contribute with the cryptographic engineering of lattice-based cryptosystems submitted to NIST's Post-Quantum Cryptography standardization process; and *iii*) support the cryptographic engineering of privacy-preserving algorithms. We now elaborate on each of these objectives.

Alternative instantiations for Ring-LWE cryptography. In particular, we aimed at extending the Ring-LWE Problem to embrace algebraic constructions via twisted embeddings. As a result, we redefined the Ring-LWE Problem considering a twisting factor, providing security reductions to substantiate our claim that the hardness of Ring-LWE remains unchanged. Also, we offered a Ring-LWE instance over a maximal real subfield of a cyclotomic number field which is not achievable in the original proposal.

Nevertheless, the scope of new algebraic lattice constructions is mostly limited to cyclotomic number fields and their maximal real subfields. The literature accounts for algebraic lattice constructions that determine the ring of integers and its corresponding twisting factor. Similar constructions arise from coding theory for Rayleigh fading channels. Additionally, it is not straightforward to determine the ring of integers of a number field, in general. The special case is monogenic number fields, for which the ring of integers is of the form $\mathbb{Z}[\theta]$, for a generator θ. However, one may want to avoid monogenic number fields in the light of past efforts to determine weak Ring-LWE instances [65, 66, 44, 38, 37, 43, 40, 41, 42, 130]. As discussed in Chapter 5, for the case of maximal real subfields, an alternative algebraic construction still lacks a good representation for applications. Unfortunately, the corresponding polynomial representation may be impractical, requiring exponential integer modulus. However, as discussed in Section 5.3,

ring elements can be represented by coefficient vectors instead of polynomials in the power basis, but require an additional effort of cryptographic engineering.

Although this research aims at extending the Ring-LWE for building new constructions, the line of work in Section 4.2 seems to be the most promising in the short term, possibly leading to weak instances of Ring-LWE. Also, for efficiently implementing the PKE in Chapter 5, one can use a distinct basis for each algorithmic task in the same vector space, considering the efficiency of the related algorithm. In contrast, in the Lyubashevsky et al.'s toolkit, algorithmic tasks are defined in the ring of integers R and its dual $R^\vee$ using the coefficient vector representation [89]. For working in R, the authors define two distinct bases, the powerful and the CRT. For the dual ring $R^\vee$, they define the powerful basis, the CRT basis, and also the decoding basis of $R^\vee$, which is specialized for decoding noise terms during decryption. Thus, the algorithms operate on the coefficient vectors and convert representations when needed by acknowledging the corresponding basis. Details can be found in [89, 92, 57]. Another line for future work is the exploration of other classes of algebraic lattices for the Ring-LWE, such as D_n-lattices [33, 79, 78, 61].

Cryptographic engineering of lattice-based NIST's candidates. Specifically, our objective in this project was to accelerate the polynomial multiplication of second-round NIST candidates that originally used the NTT. We replaced the NTT by the DGT and experimentally evaluated our implementations on an Intel Skylake processor. Unfortunately, neither our portable-C or AVX2-optimized implementations of polynomial multiplication via DGT improved the reference codes. The DGT in the x64 architecture is mainly impacted by memory misalignments caused by arithmetic in $\mathbb{F}_{p^2}$ and the folding procedure. Also, the number of operations in $\mathbb{Z}_p$ is higher in DGT than in the NTT, requiring the execution of more instructions for computing a polynomial multiplication. The idea of experimenting with the DGT for the x64 architecture arose from recent works implementing this transform on GPUs [22, 2]. Although the results looked promising on the GPU platform, an implementer should be aware of the characteristics and latencies of each platform, and also that SIMD techniques may behave distinctively on each architecture.

In Section 6.5.1, we introduced the T-DGT that increases the arithmetic density of DGT by performing all multiplications within the transform in $\mathbb{F}_{p^2}$. Considering the experimental results in Section 7.4, in which the RNTT benefited from increasing the arithmetic density on GPUs, we conjecture that T-DGT could delivery improved performance in a CUDA-enabled implementation compared to DGT.

Cryptographic engineering of privacy-preserving algorithms. In Chapter 7, our efforts were dedicated to support the engagement of DGT's formulations into HE schemes, providing the design and proof-of-concept implementations of both RNTT and RDGT. An adaptation of the four-step FFT, named RDGT, was used for polynomial multiplication in the BGV [68] and CKKS [45] homomorphic encryption schemes. We demonstrated a performance improvement in homomorphic multiplication obtained by using the RDGT instead of the usual DGT. Also, we verified the natural adaptability of RDGT in GPUs. We also found that the same result can be obtained for RNTT by increasing its arithmetic density.

A limitation of the work presented in Section 7.3 is that it does not isolate each implementation technique from the HE scheme, making it unfeasible to accurately evaluate a possible prevalence of the DGT over the NTT. We overcame this limitation, in the work presented in Section 7.4. In its turn, the RDGT and RNTT comparison is limited to CKKS [45], not extrapolating the evaluation to other homomorphic encryption schemes based on the LWE and Ring-LWE problems. The scope of this work can also be extended to other functional encryption schemes, since the RNTT was originally introduced for the implementation of a lattice-based key-policy attribute-based encryption scheme [58]. Moreover, we left to future work evaluating recursive transforms for TFHE [47].

Finally, I would say that researchers of ideal-lattice cryptography may find themselves with a colorful wardrobe of hats and hundreds of paper sheets. During a day of research, one may see himself/herself wearing the hat of an algebraist, a computer scientist, an engineer, a cryptographer, and a cryptanalyst. Also, one may find it challenging to translate the results from coding theory to cryptography since an algebraist, a coding theorist, and a cryptographer may describe the properties of a mathematical structure using their specific vocabulary. Nevertheless, although becoming an interpreter requires some years of experience, the results are worthwhile.

Bibliography

[1] Milton Abramowitz, Irene A Stegun, and Robert H Romer. Handbook of mathematical functions with formulas, graphs, and mathematical tables, 1988.

[2] A. Ahmad Al Badawi, Y. Polyakov, K. M. M. Aung, B. Veeravalli, and K. Rohloff. Implementation and Performance Evaluation of RNS Variants of the BFV Homomorphic Encryption Scheme. *IEEE Transactions on Emerging Topics in Computing*, pages 1–1, 2019.

[3] M. Ajtai. Generating Hard Instances of Lattice Problems (Extended Abstract). In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 99–108, New York, NY, USA, 1996. ACM.

[4] Ahmad Al Badawi, Bharadwaj Veeravalli, and Khin Mi Mi Aung. Efficient Polynomial Multiplication via Modified Discrete Galois Transform and Negacyclic Convolution. In Kohei Arai, Supriya Kapoor, and Rahul Bhatia, editors, *Advances in Information and Communication Networks*, pages 666–682, Cham, 2019. Springer International Publishing.

[5] Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan. Homomorphic Encryption Security Standard. Technical report, HomomorphicEncryption.org, Toronto, Canada, November 2018.

[6] Martin R. Albrecht, Benjamin R. Curtis, Amit Deo, Alex Davidson, Rachel Player, Eamonn W. Postlethwaite, Fernando Virdia, and Thomas Wunderer. Estimate all the LWE, NTRU schemes! Cryptology ePrint Archive, Report 2018/331, 2018. `https://eprint.iacr.org/2018/331`.

[7] Martin R. Albrecht and Amit Deo. Large modulus Ring-LWE $\geq$ Module-LWE. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 267–296. Springer, 2017.

[8] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of Learning with Errors. *J. Math. Cryptol.*, 9(3):169–203, 2015.

[9] Erdem Alkim, Roberto Avanzi, Joppe Bos, Léo Ducas, Antonio de la Piedra, Thomas Pöppelmann, Peter Schwabe, and Douglas Stebila. NewHope: Algorithm Specifications and Supporting Documentation. NIST Post-Quantum Cryptography Standardization Process, 2019. `https://newhopecrypto.org/`.

[10] Erdem Alkim, Paulo S. L. M. Barreto, Nina Bindel, Patrick Longa, and Jefferson E. Ricardini. The Lattice-Based Digital Signature Scheme qTESLA. Cryptology ePrint Archive, Report 2019/085, 2019. `https://eprint.iacr.org/2019/085`.

[11] Erdem Alkim, Yusuf Bilgin, and Murat Cenk. *Compact and Simple RLWE Based Key Encapsulation Mechanism*, pages 237–256. Springer Nature Switzerland, 09 2019.

[12] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum Key Exchange—A New Hope. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 327–343, Austin, TX, 2016. USENIX Association.

[13] Pedro Geraldo M. R. Alves, Jheyne N. Ortiz, and Diego F. Aranha. Faster Homomorphic Encryption over GPGPUs via Hierarchical DGT. In Nikita Borisov and Claudia Diaz, editors, *Financial Cryptography and Data Security*, pages 520–540, Berlin, Heidelberg, 2021. Springer Berlin Heidelberg.

[14] Pedro G.M.R. Alves and Jheyne N. Ortiz. SPOG: Secure Processing on GPGPUs. `https://github.com/spog-library`, 2021.

[15] Pedro G.M.R. Alves, Jheyne N. Ortiz, and Diego F. Aranha. Performance of Hierarchical Transforms in Homomorphic Encryption: A case study on Logistic Regression inference. Cryptology ePrint Archive, Report 2022/099, 2022. `http://eprint.iacr.org/2022/099`.

[16] A. A. Andrade and J. C. Interlando. Rotated $\mathbb{Z}^n$-Lattices via Real Subfields of $\mathbb{Q}(\zeta_{2^r})$. *TEMA (São Carlos)*, 20:445 – 456, 12 2019.

[17] M. Aranés and A. Arenas. On the defining polynomials of maximal real cyclotomic extensions. *Revista de la Real Academia de Ciencias Exactas, Físicas y Naturales. Serie A. Matemáticas*, 101(2):187–203, 2008.

[18] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS – Kyber: Algorithm Specifications And Supporting Documentation. NIST Post-Quantum Cryptography Standardization Process, 2019. `https://pq-crystals.org/kyber/`.

[19] L. Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, Mar 1986.

[20] A. Al Badawi, Y. Polyakov, K. Aung, B. Veeravalli, and K. Rohloff. Implementation and Performance Evaluation of RNS Variants of the BFV Homomorphic Encryption

Scheme. *IEEE Transactions on Emerging Topics in Computing*, 9(02):941–956, apr 2021.

[21] Ahmad Al Badawi, Yuriy Polyakov, Khin Mi Mi Aung, Bharadwaj Veeravalli, and Kurt Rohloff. Implementation and Performance Evaluation of RNS Variants of the BFV Homomorphic Encryption Scheme. *IACR Cryptol. ePrint Arch.*, 2018:589, 2018.

[22] Ahmad Al Badawi, Bharadwaj Veeravalli, Chan Fook Mun, and Khin Mi Mi Aung. High-Performance FV Somewhat Homomorphic Encryption on GPUs: An Implementation using CUDA. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(2):70–95, May 2018.

[23] David H. Bailey. FFTs in external or hierarchical memory. *The Journal of Supercomputing*, 4(1):23–35, Mar 1990.

[24] Paul Barrett. Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor. In Andrew M. Odlyzko, editor, *Advances in Cryptology — CRYPTO' 86*, pages 311–323, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg.

[25] Andrea Basso, Jose Maria Bermudo Mera, Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, Michiel Van Beirendonck, and Frederik Vercauteren. Saber: Mod-LWR based KEM (round 3 submission). Submission to the NIST Post-Quantum Cryptography Standardization Project, 2020. `https://www.esat.kuleuven.be/cosic/pqcrypto/saber/index.html`.

[26] E. Bayer-Fluckiger, F. Oggier, and E. Viterbo. New algebraic constructions of rotated Zn-lattice constellations for the Rayleigh fading channel. *IEEE Transactions on Information Theory*, 50(4):702–714, April 2004.

[27] Eva Bayer-Fluckiger. Lattices and Number Fields. *Algebraic Geometry: Hirzebruch 70*, 241, 1999.

[28] Ayoub Benaissa, Bilal Retiat, Bogdan Cebere, and Alaa Eddine Belfedhal. TenSEAL: A Library for Encrypted Tensor Operations using Homomorphic Encryption, 2021.

[29] Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. NTRU Prime: reducing attack surface at low cost. Cryptology ePrint Archive, Report 2016/461, 2016. `http://eprint.iacr.org/2016/461`.

[30] Nina Bindel, Erdem Alkim, Paulo S. L. M. Barreto, ohannes Buchmann, Edward Eaton, Gus Gutoski, Juliane Krämer, Patrick Longa, Harun Polat, Jefferson E. Ricardini, and Gustavo Zanon. Submission to NIST's post-quantum project (2nd round): lattice-based digital signature scheme qTESLA. NIST Post-Quantum Cryptography Standardization Process, 2019. `https://qtesla.org/`.

[31] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, and Damien Stehlé. CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM. Cryptology ePrint Archive, Report 2017/634, 2017. http://eprint.iacr.org/2017/634.

[32] Joppe W. Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme. In Martijn Stam, editor, *Cryptography and Coding*, pages 45–64, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[33] J. Boutros, E. Viterbo, C. Rastello, and J. C. Belfiore. Good lattice constellations for both Rayleigh fading and Gaussian channels. *IEEE Transactions on Information Theory*, 42(2):502–518, Mar 1996.

[34] Zvika Brakerski. Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, pages 868–886, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[35] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) Fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, page 309–325, New York, NY, USA, 2012. Association for Computing Machinery.

[36] Peter Campbell, Michael Groves, and Dan Shepherd. SOLILOQUY: A Cautionary Tale. *Docbox.Etsi.Org*, pages 1–9, 2013.

[37] Wouter Castryck, Ilia Iliashenko, and Frederik Vercauteren. On error distributions in ring-based LWE. *LMS Journal of Computation and Mathematics*, 19(A):130–145, 2016.

[38] Wouter Castryck, Ilia Iliashenko, and Frederik Vercauteren. Provably Weak Instances of Ring-LWE Revisited. In *Proceedings of the 35th Annual International Conference on Advances in Cryptology — EUROCRYPT 2016 - Volume 9665*, pages 147–167, New York, NY, USA, 2016. Springer-Verlag New York, Inc.

[39] Cong Chen, Oussama Danba, Jeffrey Hoffstein, Andreas Hülsing, Joost Rijneveld, John M. Schanck, Tsunekazu Saito, Peter Schwabe, William Whyte, Keita Xagawa, Takashi Yamakawa, and Zhenfei Zhang. NTRU algorithm specifications and supporting documentation. Submission to the NIST Post-Quantum Cryptography Standardization Project, 2020. https://ntru.org/resources.shtml.

[40] Hao Chen. Solving Ring-LWE over Algebraic Integer Rings. Cryptology ePrint Archive, Report 2019/791, 2019. https://ia.cr/2019/791.

[41] Hao Chen. Subset Attacks on Ring-LWE with Wide Error Distributions I. Cryptology ePrint Archive, Report 2020/440, 2020. https://ia.cr/2020/440.

[42] Hao Chen. Ring-LWE over two-to-power cyclotomics is not hard. Cryptology ePrint Archive, Report 2021/418, 2021. https://ia.cr/2021/418.

[43] Hao Chen, Kristin Lauter, and Katherine E. Stange. Security Considerations for Galois Non-dual RLWE Families. In Roberto Avanzi and Howard Heys, editors, *Selected Areas in Cryptography – SAC 2016*, pages 443–462, Cham, 2017. Springer International Publishing.

[44] Hao Chen, Kristin E. Lauter, and Katherine E. Stange. Attacks on the Search-RLWE problem with small error. Cryptology ePrint Archive, Report 2015/971, 2015. https://eprint.iacr.org/2015/971.

[45] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. A full RNS variant of approximate homomorphic encryption. In Carlos Cid and Michael J. Jacobson Jr., editors, *Selected Areas in Cryptography - SAC 2018 - 25th International Conference, Calgary, AB, Canada, August 15-17, 2018, Revised Selected Papers*, volume 11349 of *Lecture Notes in Computer Science*, pages 347–368. Springer, 2018.

[46] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic Encryption for Arithmetic of Approximate Numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 409–437, Cham, 2017. Springer International Publishing.

[47] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster Fully Homomorphic Encryption: Bootstrapping in Less Than 0.1 Seconds. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016*, pages 3–33, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[48] Ilaria Chillotti, Marc Joye, and Pascal Paillier. Programmable Bootstrapping Enables Efficient Homomorphic Inference of Deep Neural Networks. In Shlomi Dolev, Oded Margalit, Benny Pinkas, and Alexander A. Schwarzmann, editors, *Cyber Security Cryptography and Machine Learning - 5th International Symposium, CSCML 2021, Be'er Sheva, Israel, July 8-9, 2021, Proceedings*, volume 12716 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2021.

[49] Eleanor Chu and Alan George. *Inside the FFT Black Box – Serial and Parallel Fast Fourier Transform Algorithms*. CRC Press, Boca Raton, FL, 2000.

[50] Stephen A. Cook. *On the minimum computation time of functions*. PhD thesis, Department of Mathematics, Harvard University, 1966. URL: http://cr.yp.to/bib/entries.html#1966/cook.

[51] James Cooley and John Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.

[52] Sueli I.R. Costa, Frédérique Oggier, Antonio Campello, Jean-Claude Belfiore, and Emanuele Viterbo. *Lattices Applied to Coding for Reliable and Secure Communications*. Springer, Cham, 2017.

[53] Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev. Recovering Short Generators of Principal Ideals in Cyclotomic Rings. In *Proceedings of the 35th Annual International Conference on Advances in Cryptology — EUROCRYPT 2016 - Volume 9666*, pages 559–585, New York, NY, USA, 2016. Springer-Verlag New York, Inc.

[54] Ronald Cramer, Léo Ducas, and Benjamin Wesolowski. Short Stickelberger Class Relations and Application to Ideal-SVP. In *EUROCRYPT (1)*, pages 324–348. Springer, 2017.

[55] Richard E. Crandall. Integer convolution via split-radix fast Galois transform, 1999.

[56] Eric Crockett and Chris Peikert. Challenges for Ring-LWE. Cryptology ePrint Archive, Report 2016/782, 2016. `https://ia.cr/2016/782`.

[57] Eric Crockett and Chris Peikert. $\Lambda o \lambda$: Functional Lattice Cryptography. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 993–1005. ACM, 2016.

[58] Wei Dai, Yarkın Doröz, Yuriy Polyakov, Kurt Rohloff, Hadi Sajjadpour, Erkay Savaş, and Berk Sunar. Implementation and evaluation of a lattice-based key-policy abe scheme. *IEEE Transactions on Information Forensics and Security*, 13(5):1169–1184, 2018.

[59] Wei Dai and Berk Sunar. cuHE: A Homomorphic Encryption Accelerator Library. In Enes Pasalic and Lars R. Knudsen, editors, *Cryptography and Information Security in the Balkans*, pages 169–186, Cham, 2016. Springer International Publishing.

[60] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. Saber: Module-LWR Based Key Exchange, CPA-Secure Encryption and CCA-Secure KEM. In Antoine Joux, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *Progress in Cryptology – AFRICACRYPT 2018*, pages 282–305, Cham, 2018. Springer International Publishing.

[61] Robson R. de Araujo and Grasiele C. Jorge. Constructions of full diversity D_n-lattices for all n. *Rocky Mountain J. Math.*, 50(4):1137–1150, 08 2020.

[62] Léo Ducas and Alain Durmus. *Ring-LWE in Polynomial Rings*, pages 34–51. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[63] Leo Ducas, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehle. CRYSTALS – Dilithium: Digital Signatures from Module Lattices. Cryptology ePrint Archive, Report 2017/633, 2017. `http://eprint.iacr.org/2017/633`.

[64] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS – Dilithium: Algorithm Specifications and Supporting Documentation. NIST Post-Quantum Cryptography Standardization Process, 2019. `https://pq-crystals.org/dilithium/index.shtml`.

[65] Kirsten Eisenträger, Sean Hallgren, and Kristin Lauter. *Weak Instances of PLWE*, pages 183–194. Springer International Publishing, Cham, 2014.

[66] Yara Elias, Kristin E. Lauter, Ekin Ozman, and Katherine E. Stange. *Provably Weak Instances of Ring-LWE*, pages 63–92. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[67] Niall Emmart and Charles C. Weems. HIGH PRECISION INTEGER MULTIPLICATION WITH A GPU USING STRASSEN'S ALGORITHM WITH MULTIPLE FFT SIZES. *Parallel Processing Letters*, 21(03):359–375, September 2011.

[68] Junfeng Fan and Frederik Vercauteren. Somewhat Practical Fully Homomorphic Encryption. Cryptology ePrint Archive, Report 2012/144, 2012. `https://ia.cr/2012/144`.

[69] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhan. Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU. NIST Post-Quantum Cryptography Standardization Process, 2019. `https://falcon-sign.info`.

[70] Taher El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *CRYPTO*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, 1984.

[71] W. M. Gentleman and G. Sande. Fast fourier transforms: For fun and profit. In *Proceedings of the November 7-10, 1966, Fall Joint Computer Conference*, AFIPS '66 (Fall), page 563–578, New York, NY, USA, 1966. Association for Computing Machinery.

[72] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `crypto.stanford.edu/craig`.

[73] Craig Gentry. Fully Homomorphic Encryption Using Ideal Lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, page 169–178, New York, NY, USA, 2009. Association for Computing Machinery.

[74] Naga K. Govindaraju, Brandon Lloyd, Yuri Dotsenko, Burton Smith, and John Manferdelli. High Performance Discrete Fourier Transforms on Graphics Processors. In *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*, SC '08, pages 2:1–2:12, Piscataway, NJ, USA, 2008. IEEE Press.

[75] Shai Halevi and Victor Shoup. HElib. `https://github.com/shaih/HElib`, 2021.

[76] Kyoohyung Han, Seungwan Hong, Jung Hee Cheon, and Daejun Park. Efficient Logistic Regression on Large Encrypted Data. *IACR Cryptol. ePrint Arch.*, page 662, 2018.

[77] Jeffrey Hoffstein. *Lattices and Cryptography*, pages 1–87. Springer New York, New York, NY, 2008.

[78] Grasiele C. Jorge and Sueli I.R. Costa. On rotated D_n-lattices constructed via totally real number fields. *Archiv der Mathematik*, 100(4):323–332, 2013.

[79] Grasiele C. Jorge, Agnaldo J. Ferrari, and Sueli I. R. Costa. Rotated Dn-lattices. *Journal of Number Theory*, 132(11):2397 – 2406, 2012.

[80] Wonkyung Jung, Sangpyo Kim, Jung Ho Ahn, Jung Hee Cheon, and Younho Lee. Over 100x Faster Bootstrapping in Fully Homomorphic Encryption through Memory-centric Optimization with GPUs. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(4):114–148, Aug. 2021.

[81] Anatoly A. Karatsuba and Y. Ofman. Multiplication of multidigit numbers on automata. *Soviet Physics Doklady*, 7:595–596, 1963. URL: `http://cr.yp.to/bib/entries.html#1963/karatsuba`.

[82] Serge Lang. *Algebra*. Springer, New York, NY, 2002.

[83] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, Jun 2015.

[84] Yann LeCun, Corinna Cortes, and CJ Burges. MNIST handwritten digit database, 2010.

[85] Patrick Longa and Michael Naehrig. Speeding up the Number Theoretic Transform for Faster Ideal Lattice-Based Cryptography. In Sara Foresti and Giuseppe Persiano, editors, *Cryptology and Network Security*, pages 124–139, Cham, 2016. Springer International Publishing.

[86] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-Fly Multiparty Computation on the Cloud via Multikey Fully Homomorphic Encryption. In *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*, STOC '12, page 1219–1234, New York, NY, USA, 2012. Association for Computing Machinery.

[87] Vadim Lyubashevsky and Daniele Micciancio. Generalized Compact Knapsacks Are Collision Resistant. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming*, pages 144–155, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[88] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. *On Ideal Lattices and Learning with Errors over Rings*, pages 1–23. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[89] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A Toolkit for Ring-LWE Cryptography. Cryptology ePrint Archive, Report 2013/293, 2013. http://eprint.iacr.org/2013/293.

[90] Vadim Lyubashevsky and Gregor Seiler. NTTRU: Truly Fast NTRU Using NTT. Cryptology ePrint Archive, Report 2019/040, 2019. https://eprint.iacr.org/2019/040.

[91] Arif Mandangan, Hailiza Kamarulhaili, and Muhammad Asyraf Asbullah. Good basis vs bad basis: On the ability of Babai's Round-off Method for solving the Closest Vector Problem. *Journal of Physics: Conference Series*, 1366:012016, 11 2019.

[92] Christoph M. Mayer. Implementing a Toolkit for Ring-LWE Based Cryptography in Arbitrary Cyclotomic Number Fields. Cryptology ePrint Archive, Report 2016/049, 2016. http://eprint.iacr.org/2016/049.

[93] Daniele Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions from worst-case complexity assumptions. In *Proceedings of the 43rd Symposium on Foundations of Computer Science*, FOCS '02, pages 356–365, Washington, DC, USA, 2002. IEEE Computer Society.

[94] Daniele Micciancio. Generalized Compact Knapsacks, Cyclic Lattices, and Efficient One-Way Functions. In *Computational Complexity*, pages 365–411, 2007.

[95] Daniele Micciancio and Chris Peikert. Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, pages 700–718, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[96] Daniele Micciancio and Oded Regev. Worst-Case to Average-Case Reductions Based on Gaussian Measures. *SIAM J. Comput.*, 37(1):267–302, April 2007.

[97] Peter L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44:519–521, 1985.

[98] Dustin Moody, Gorjan Alagic, Daniel Apon, David Cooper, Quynh Dang, John Kelsey, Yi-Kai Liu, Carl Miller, Rene Peralta, Ray Perlner, Angela Robinson, Daniel Smith-Tone, and Jacob Alperin-Sheriff. Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process, 2020-07-22 2020.

[99] Sean Murphy and Rachel Player. δ-subgaussian Random Variables in Cryptography. In Julian Jang-Jaccard and Fuchun Guo, editors, *Information Security and Privacy*, pages 251–268, Cham, 2019. Springer International Publishing.

[100] Sean Murphy and Rachel Player. Discretisation and Product Distributions in Ring-LWE. *Journal of Mathematical Cryptology*, 15(1):45–59, 2021.

[101] National Institute of Standards and Technology - NIST. Post-Quantum Cryptography, 2017. https://csrc.nist.gov/projects/post-quantum-cryptography.

[102] National Institute of Standards and Technology - NIST. Round 2 Submissions - Post-Quantum Cryptography, 2019. https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[103] National Institute of Standards and Technology - NIST. Third PQC Standardization Conference , 2021. https://csrc.nist.gov/Events/2021/third-pqc-standardization-conference.

[104] NVIDIA. CUDA Toolkit Documentation. http://docs.nvidia.com/cuda/, 2022. Last accessed: 2022/01/26.

[105] Markku-Juhani O. Saarinen. Mobile Energy Requirements of the Upcoming NIST Post-Quantum Cryptography Standards. In *2020 8th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, pages 23–30, 2020.

[106] Frédérique Oggier and Emanuele Viterbo. Algebraic Number Theory and Code Design for Rayleigh Fading Channels. *Commun. Inf. Theory*, 1(3):333–416, December 2004.

[107] Frédérique Oggier. Introduction to algebraic number theory, 2010. https://feog.github.io/ANT10.pdf.

[108] Jheyne N. Ortiz. Algebraic construction of lattices for the Ring-LWE problem. 9th International Workshop on Cryptography, Robustness, and Provably Secure Schemes for Female Young Researchers (CrossFyre), 2019. https://www.crossing.tu-darmstadt.de/news_events/conferences_workshops/crossfyre_19/index.en.jsp.

[109] Jheyne N. Ortiz, Robson R. de Araujo, Diego F. Aranha, Sueli I. R. Costa, and Ricardo Dahab. The Ring-LWE Problem in Lattice-Based Cryptography: The Case of Twisted Embeddings. *Entropy*, 23(9), 2021.

[110] Jheyne N. Ortiz, Robson R. de Araujo, Ricardo Dahab, Diego F. Aranha, and Sueli I. R. Costa. In Praise of Twisted Canonical Embedding. Cryptology ePrint Archive, Report 2018/356, 2018. https://eprint.iacr.org/2018/356.

[111] Jheyne N. Ortiz, Robson R. de Araujo, Ricardo Dahab, Diego F. Aranha, and Sueli I. R. Costa. On Lattices for Cryptography. Latin America Week on Coding and Information (LAWCI 2018), 2018. http://www.dev.ime.unicamp.br/lawci/index.php.

[112] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.

[113] Chris Peikert. Limits on the Hardness of Lattice Problems in Lp Norms. *Comput. Complex.*, 17(2):300–351, May 2008.

[114] Chris Peikert. A Decade of Lattice Cryptography. *Found. Trends Theor. Comput. Sci.*, 10(4):283–424, March 2016.

[115] Chris Peikert. *How (Not) to Instantiate Ring-LWE*, pages 411–430. Springer International Publishing, Cham, 2016.

[116] Chris Peikert and Zachary Pepin. Algebraically Structured LWE, Revisited. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography*, pages 1–23, Cham, 2019. Springer International Publishing.

[117] Chris Peikert, Oded Regev, and Noah Stephens-Davidowitz. Pseudorandomness of ring-LWE for Any Ring and Modulus. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 461–473, New York, NY, USA, 2017. ACM.

[118] Chris Peikert, Oded Regev, and Noah Stephens-Davidowitz. Pseudorandomness of Ring-LWE for Any Ring and Modulus (Slides), 2017. `https://web.eecs.umich.edu/~cpeikert/pubs/slides-anyring.pdf`.

[119] John M. Pollard. The fast Fourier transform in a finite field. *Mathematics of Computation*, 25:365–374, 1971. URL: `http://cr.yp.to/bib/entries.html#1971/pollard`.

[120] Thomas Pöppelmann, Tobias Oder, and Tim Güneysu. High-Performance Ideal Lattice-Based Cryptography on 8-Bit ATxmega Microcontrollers. In Kristin Lauter and Francisco Rodríguez-Henríquez, editors, *Progress in Cryptology – LATINCRYPT 2015*, pages 346–365, Cham, 2015. Springer International Publishing.

[121] Oded Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 84–93, New York, NY, USA, 2005. ACM.

[122] Paulo Ribenboim. *Classical Theory of Algebraic Numbers*. Universitext. Springer-Verlag New York, 2001.

[123] Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos. On Data Banks and Privacy Homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.

[124] Sujoy Sinha Roy, Frederik Vercauteren, Nele Mentens, Donald Donglong Chen, and Ingrid Verbauwhede. Compact Ring-LWE Cryptoprocessor. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems – CHES 2014*, pages 371–391, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[125] Nikola Samardzic, Axel Feldmann, Aleksandar Krastev, Srinivas Devadas, Ronald Dreslinski, Christopher Peikert, and Daniel Sanchez. F1: A Fast and Programmable Accelerator for Fully Homomorphic Encryption. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '21, page 238–252, New York, NY, USA, 2021. Association for Computing Machinery.

[126] Pierre Samuel and Allan J. Silberger. *Algebraic Theory of Numbers*. Hermann, Paris, 1970.

[127] Microsoft SEAL (release 3.7). `https://github.com/Microsoft/SEAL`, September 2021. Microsoft Research, Redmond, WA.

[128] Peter W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.*, 26(5):1484–1509, October 1997.

[129] N. P. Smart and F. Vercauteren. *Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes*, pages 420–443. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[130] Katherine E. Stange. Algebraic aspects of solving Ring-LWE, including ring-based improvements in the Blum-Kalai-Wasserman algorithm. Cryptology ePrint Archive, Report 2019/183, 2019. `https://ia.cr/2019/183`.

[131] Ian N. Stewart and David O. Tall. *Algebraic Number Theory and Fermat's Last Theorem: Third Edition*. A K Peters/CRC Press: New York, 2001.

[132] Andrei L. Toom. The complexity of a scheme of functional elements realizing the multiplication of integers. *Soviet Mathematics Doklady*, 3:714–716, 1963.

[133] Eric W. Weisstein. Primitive Root. Last accessed: 2022/01/17.

[134] Wolfram. Wolfram|Alpha: Making the world's knowledge computable. Last accessed: 2022/01/17.

[135] Christian Wuthrich. Further Number Theory. `https://www.maths.nottingham.ac.uk/plp/pmzcw/download/fnt_chap5.pdf`, 2011. Last accessed: 2020/06/18.

Appendix A

Experimental Results on an Intel Haswell Architecture

Table A.1: Haswell cycle counts for the median of 10 000 executions of CRYSTALS-Dilithium reference code using either the NTT or DGT transform.

	Dilithium II			**Dilithium III**		
	Reference	Ours	NTT/DGT	Reference	Ours	NTT/DGT
Keypair	293 260	301 156	0.97	426 152	435 756	0.98
Sign	1 253 880	1 299 164	0.97	1 735 124	1 800 600	0.96
Verify	309 724	323 544	0.96	427 364	440 788	0.97
Transform	8016	8636	0.93	7736	8208	0.94
Point-wise Mul.	1536	2300	0.67	1280	1676	0.76
Inv. Transform	10 184	9272	1.10	9104	8712	1.04

	Dilithium IV		
	Reference	Ours	NTT/DGT
Keypair	560 096	581 004	0.96
Sign	1 608 484	1 703 436	0.94
Verify	579 272	606 068	0.96
Forward Transform	7660	8016	0.96
Point-wise Mul.	1280	1916	0.67
Inverse Transform	8948	8704	1.03

Table A.2: Haswell cycle counts for the median of 10 000 executions of the NewHope reference code using either the NTT or DGT transform.

	NewHope512CCA			**NewHope1024CCA**		
	Reference	Ours	NTT/DGT	Reference	Ours	NTT/DGT
Keypair	124 712	136 360	0.91	252 972	268 258	0.94
Encaps	177 572	186 656	0.95	366 420	374 630	0.98
Decaps	200 184	207 328	0.97	421 056	422 972	1.00
Forward Transform	21 514	25 924	0.83	49 564	54 592	0.91
Point-wise Mul.	5656	8232	0.69	11 288	16 452	0.69
Inverse Transform	23 224	18 308	1.27	52 980	41 448	1.28

Table A.3: Haswell cycle counts for the median of 10 000 executions of qTESLA reference code using either the NTT or DGT transform.

	qTESLA-p-I			**qTESLA-p-III**		
	Reference	Ours	NTT/DGT	Reference	Ours	NTT/DGT
Keygen	3 114 686	3 149 422	0.99	17 587 772	17 679 170	0.99
Sign	2 278 650	2 248 942	1.01	5 382 434	5 317 728	1.01
Verify	812 980	816 596	1.00	2 168 358	2 170 600	1.00
Transform	34 400	37 196	0.92	91 012	93 212	0.98
Inv. Transform	35 376	33 984	1.04	83 072	78 952	1.05

Table A.4: Haswell cycle counts for the median of 10 000 executions of qTESLA digital signature scheme using an AVX2-optimized implementation of either NTT or DGT.

	qTESLA-p-I		
	Reference	Ours	NTT/DGT
Keygen	2 995 658	3 060 310	0.98
Sign	1 445 964	1 593 796	0.91
Verify	692 154	712 036	0.97
Transform	11 132	9932	1.12
Inv. Transform	11 444	13 884	0.82