Error correction and reconciliation techniques for lattice-based key generation protocols

Charbel Saliba

THÈSE

pour obtenir le titre de

Docteur en Sciences

Spéecialité Sciences et Technologies de l'Information et de la Communication

---

Error correction and reconciliation techniques for lattice-based key generation protocols

---

par

Charbel SALIBA

ETIS UMR8051, CY Cergy Paris Université / ENSEA / CNRS
6 avenue du Ponceau, 95014 Cergy-Pontoise Cedex, France

Devant le jury composé de:

Rapporteure	Ghaya REKAYA-BEN OTHMAN	Professeure à Télécom-Paris
Rapporteur	Marco BALDI	Professeur Associé à l'Università Politecnica delle Marche
Examinateur / Président du jury	Nicolas SENDRIER	Directeur de Recherche à l'INRIA de Paris
Examinateur	Weiqiang WEN	Maître de conférences à Télécom-Paris
Directrice de thèse	Inbar FIJALKOW	Professeure des universités à l'ENSEA
Co-encadrante	Laura LUZZI	Maître de conférences à l'ENSEA

Security engineering, especially in
this third wave, requires you to
think differently.
You need to figure out not how
something works, but how
something can be made to not work.
You have to imagine an intelligent
and malicious adversary inside
your system constantly trying new
ways to subvert it.
You have to think like an alien

BRUCE SCHNEIER

Acknowledgment

My most sincere gratitude goes to my supervisor Dr. Laura Luzzi, who, through her intellectual rigor, her always relevant comments, her knowledge of the subject and methodological aspects, as well as her great availability, has largely contributed to my training in educational research. The constant monitoring she provided at all stages of this doctoral research made this project a reality. My admiration for her, both professionally and personally, is immeasurable. Dr. Luzzi also showed a clear interest in the topic, which made our numerous exchanges (both face-to-face and virtual mode) so stimulating! Due to the lack of space, I will declare my silence.

To my thesis director Inbar Fijalkow, a great thanks for her contribution to the realization of this thesis through her remarks which are always very enlightening, her mastery of various fields and her great experience. The critical opinions expressed by Professor Fijalkow allowed me to deepen my reflection and to take a retrospective look at this research.

My thanks also go to Professor Cong Ling of Imperial College who assisted us in this work and provided us with many insights and ideas that were the driving force of this research. Professor Cong was always available for discussion and made me feel like I was one of his students.

My work was initiated thanks to the INEX Paris-Seine AAP 2017 project. I thank the various actors of INEX for their help in the development of this project.

I would like to thank the members of the jury individually for doing me the honor of judging this work. I thank them for their careful reading of my thesis as well as for the comments they will address to me during this defense in order to improve my work. Special thanks to Jean-Pierre Tillich and Iryna Andriyanova who followed me during the monitoring committees of my thesis. It is thanks to their comments that I was able to improve several weak points to which I had not paid attention.

During these years I was part of the ICI team of the ETIS laboratory. I thank all the members of ICI who gave me a warm welcome and with whom I had various discussions. I consider of course also those that I have crossed regularly. I thank the members of ETIS who helped me solve the problems on a daily basis. Tarek Elouaret, Irched Chafaa and Habiba Lahdiri who helped me out when I had problems with the machines and the technical equipment.

I would also like to acknowledge the invaluable support that my family has given me. I would especially like to thank my father Joseph and my mother Leila, who among other things give the unwavering support on the moral, emotional and intellectual levels. I would like to thank them here for their contribution, without which the realization of this research would have been impossible. I must not forget my sisters Antoinette and Rita and my brother Maroun who have always believed in me and have always encouraged me to reach the goal.

Charbel Saliba

Abstract

In the last decade, there has been significant progress in the development of quantum computers, with massive investments by major tech companies. It is assumed that once large-scale quantum computers are built, many commonly used public key cryptosystems will no longer be secure. To prevent quantum attacks, researchers are already working on the design of post-quantum cryptographic protocols, and the US National Institute of Standards and Technology (NIST) is holding an international competition to select new cryptographic standards.

Lattice-based cryptographic constructions are promising candidates because they offer strong theoretical security guarantees and can be implemented efficiently. One of the most widely used cryptographic primitives based on lattices is the Learning With Errors (LWE) problem introduced by Regev. Later works have proposed structured variants of LWE such as Ring-LWE and Module-LWE, which allow for a more compact representation.

In this thesis, we consider two lattice-based NIST candidates for Key Encapsulation Mechanisms (KEMs) and propose new error correction and reconciliation techniques in order to improve their efficiency, their security, as well as their reliability. Unlike some previous works on error correction for lattice-based protocols, we provide rigorous error probability bounds.

We first consider FrodoKEM, a lattice-based cryptosystem based on LWE, and introduce a modified error correction mechanism to improve its performance. Our encoder maps the secret key block-wise into a scaled version of the 8-dimensional Gosset lattice E_8. We propose three sets of parameters for our modified implementation. The first implementation outperforms FrodoKEM in terms of plausible security; the second allows to reduce the bandwidth by halving the modulus, and the third allows to increase key sizes.

The second KEM we are considering is KyberKEM, which is based on Module-LWE. We propose a reconciliation technique using the lattice E_8, and show that our scheme can outperform KyberKEM in terms of security with comparable error probability and similar bandwidth requirements. We also investigate the use of higher dimensional lattices for reconciliation.

Résumé

Au cours de la dernière décennie, le développement des ordinateurs quantiques a considérablement progressé, avec des investissements massifs de la part des grandes entreprises technologiques. On suppose qu'une fois que les ordinateurs quantiques à grande échelle seront construits, de nombreux systèmes cryptographiques à clé publique couramment utilisés ne seront plus sécurisés. Pour prévenir les attaques quantiques, les chercheurs travaillent déjà à la conception de protocoles cryptographiques post-quantiques, et le National Institute of Standards and Technology (NIST) des États-Unis organise un concours international pour sélectionner de nouvelles normes cryptographiques.

Les constructions cryptographiques à base de réseaux euclidiens sont des candidats prometteurs car elles offrent de fortes garanties de sécurité théoriques et peuvent être mises en œuvre efficacement. L'une des primitives cryptographiques les plus largement utilisées basées sur des réseaux est le problème d'apprentissage avec erreurs, ou en anglais *Learning with errors* (LWE) introduit par Regev. Des travaux ultérieurs ont proposé des variantes structurées de LWE telles que Ring-LWE et Module-LWE, qui permettent une représentation plus compacte.

Dans cette thèse, nous considérons deux candidats au défi du NIST basés sur les réseaux euclidiens pour les mécanismes d'encapsulation de clé (KEM) et proposons de nouvelles techniques de correction d'erreur et de réconciliation afin d'améliorer leur efficacité, leur sécurité, ainsi que leur fiabilité. Contrairement à certains travaux antérieurs sur la correction d'erreurs pour les protocoles basés sur les réseaux, nous proposons des bornes de probabilité d'erreur rigoureuses.

Nous considérons d'abord FrodoKEM, un cryptosystème basé sur LWE, et introduisons un mécanisme de correction d'erreur pour améliorer ses performances. Notre encodeur mappe la clé secrète par bloc dans le réseau de Gosset à 8 dimensions E_8. Nous proposons trois ensembles de paramètres pour notre implémentation modifiée. La première implémentation surpasse FrodoKEM en termes de sécurité plausible ; la seconde permet de réduire la bande passante en divisant par deux le module, et la troisième permet d'augmenter la taille des clés.

Le deuxième KEM que nous considérons est KyberKEM, qui est basé sur Module-LWE. Nous proposons une technique de réconciliation utilisant toujours le réseau euclidien E_8, et montrons que notre schéma peut surpasser KyberKEM en termes de sécurité avec une probabilité d'erreur comparable et des exigences de bande passante similaires. Nous étudions également l'utilisation de réseaux de dimension supérieure pour la réconciliation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context and motivation

Nowadays, many of our daily activities are dependent on the web, so various forms of communication, entertainment, finance and work related tasks are handled online. Personal and financial information is highly desirable for cyber attackers. An attacker can passively observe the transmitted data without making any changes so that the victim is not aware of the eavesdropper's presence. Furthermore, an attacker can modify or damage data in order to confuse and harm victims. This is why it is crucial to protect our devices against these malicious activities of attackers.

Cryptography is the basic technology used to ensure the privacy and security of data as it travels over the Internet. It takes advantage of mathematical problems that are thought to be computationally difficult to solve in order to design algorithms that ensure security and reliability. Most of these cryptographic algorithms fit into one of two classifications: *symmetric* and *asymmetric*. Symmetric protocols are mostly used to encrypt and decrypt data using a shared private key, and they are usually much faster than asymmetric algorithms. The most popular example of symmetric cryptography algorithms is the Advanced Encryption Standard (AES). Asymmetric cryptography (or Public-key cryptography) relies on two pairs of keys: the first is a *public key* that can be revealed to the public, and the second key, called a *secret key*, must be kept secret and unknown to others. Applications of asymmetric cryptography involve encryption protocols, digital signatures, key establishment schemes and many other use cases. Examples of asymmetric algorithms include RSA, ElGamal, Diffie-Hellman and ECC (elliptic-curve cryptography). Breaking such schemes involves solving difficult problems like prime factorization and discrete logarithm in polynomial time, which has so far been impractical on classical computers.

Post-quantum Cryptography

In recent years, research on quantum computing has seen significant progress. These machines will be capable of solving mathematical problems that were thought to be intractable for conventional computers, and therefore of breaking many existing public key cryptosystems currently in use. In fact, Shor [Sch87] proposed a polynomial time quantum algorithm for the integer factorization problem which can be adapted into an efficient quantum algorithm to solve the discrete logarithm problem [Joz01]. This poses a real threat to almost all existing public key cryptosystems.

Another important quantum algorithm that needs to be mentioned is Grover's algorithm. It provides a quadratic speedup compared to classical solutions [Gro96], allowing faster brute force attacks on a wide range of unstructured search problems, including searching for a symmetric key and finding pre-images of hash functions. For instance, a 128-bit symmetric private key can be cracked within 2^{64} iterations, and therefore, this kind of attack can be prevented by doubling the size of the private key.

Consequently, there is a pressing need to develop new cryptographic systems that are secure against quantum attacks. Post-quantum cryptography offers solid alternatives for cryptographic systems that ensure security against quantum and classical computers.

The NIST post-quantum cryptography challenge

Over the past few years, many organizations have started researching public-key cryptographic algorithms that might be candidates for a new standardization process. In December 2016, the U.S. National Institute of Standards and Technology (NIST) launched a challenge to collect quantum-resistant algorithms covering public-key encryption schemes, key-establishment protocols and digital signature methods. In July 2020, NIST selected for its third round seven algorithms to be shortlisted for the standardization process, while keeping eight algorithms aside as alternate candidates in case security concerns arise (see Table 1.1).

	Signatures		**KEM/Encryption**		**Overall**	
Lattice-based	Dilithium Falcon		Kyber NTRU SABER	FrodoKEM NTRU Prime	5	2
Code-based			McEliece	BIKE HQC	1	2
Multivariate	Rainbow	GeMSS			1	1
Stateless Hash or Symmetric based		Picnic Sphincs+				2
Isogeny				SIKE		1
Total	3	3	4	5	7	8

Table 1.1: NIST's third round finalist and alternates.

Most of them are built using lattice-based and code-based cryptography, while few others use multivariate equations hash functions, and isogenies of elliptical curves. Seven of these schemes rely on lattice cryptography constructions and are among the leading candidates for public-key post-quantum cryptography.

1.2 Lattice-based cryptography

Lattice-based cryptography is one of the most promising candidates for post-quantum cryptography because it enjoys strong theoretical security guarantees and is effective in practice. It allows the construction of various cryptographic schemes such as public-key encryption schemes, key encapsulation mechanisms, digital signatures, hash functions, fully homomorphic encryption and much more, while ensuring security reductions to computational problems which are considered hard to break even on quantum computers.

Computationally hard lattice problems

The most important problem that is involved in such constructions is the *shortest vector problem* (SVP) for which the γ-approximation version (SVP_γ) is intensively study. A large family of SVP_γ variants follows such as the *decision shortest vector problem* (GapSVP_γ), *unique shortest vector problem* ($\mathsf{unique\text{-}SVP}_\gamma$), the *closest vector problem* (CVP_γ) and the *bounded distance decoding* (BDD_γ) [MG12].

These lattice problems are easy to solve once one chooses a "good" basis, that is, a nearly orthogonal basis with relatively short vectors. *Lattice reduction* algorithms seek to produce such a good basis when the input is an arbitrary basis for a lattice. An early efficient algorithm which can output an almost reduced lattice basis is the *LLL* algorithm presented in [LLL82], adapted by Lenstra, Lenstra, and Lovász in 1982. This algorithm has several applications such as factoring polynomials over the rational numbers [LLL82], integer programming [Kan83] and many other applications in cryptanalysis. Moreover, it runs in polynomial time and approximates the shortest vector inside an n-dimensional lattice within a factor of $2^{O(n)}$. Solving SVP_γ within a constant approximation factor, or even within a factor $n^{c/\log\log n}$ for some $c > 0$ is conjectured to be hard [Mic01, Kho05, HR07], and there's no polynomial time algorithm capable of doing it. The best known algorithm for solving SVP has a provable running time of $2^{O(n)}$ [AKS01], and several improvements are found in [NV08, PS09, MV10, ADRSD15, ASD17] that led to $2^{n+o(n)}$ time and space algorithms. In terms of space-time trade-off, the best current known result was given in [ACKS20] and solves SVP in $2^{1.740n+o(n)}$ time and $2^{0.5n+o(n)}$ space.

A question that arises is whether these families of problems remain difficult for quantum computers. So far, the most plausible quantum algorithm (that is also the fastest classical algorithm for SVP) is given in [ADRSD15] and requires $2^{n+o(n)}$ time and space. An improvement from the quantum side was proposed in [ACKS20] that solves SVP in $2^{0.9533n+o(n)}$ time and

classical $2^{0.5n+o(n)}$ space. Heuristically, the best known result in [Laa15a] can solve SVP within $2^{0.265n+o(n)}$ times and space. Quite recently, there is also a new heuristic quantum algorithm by "random walk" [CL21] which achieves $2^{0.257n+o(n)}$ by relying on $\exp(n)$ many quantum bits (instead of $\text{poly}(n)$ quantum bits in [Laa15a]). This algorithm is far from being realistic due to the need of a large amount of q-bits. That's why it is believed that it is hard to construct a polynomial time quantum algorithm that approximates lattice problems to within polynomial factors. This justifies the use of lattice-based cryptosystems for post-quantum cryptography.

Worst-case to average-case reduction

A significant advantage of lattice-based cryptography compared to other cryptographic techniques is that the security of most lattice-based cryptographic primitives is based on the *worst-case* hardness of lattice problems. In cryptography a problem is considered to be hard only if it is hard in the average-case, i.e. it is hard for all but a negligible fraction of instances, whereas in complexity theory, hardness is described in terms of worst-case hardness, which means that it is difficult to solve the problem for some given instance, even though a large collection of instances might be easy to solve. In 1996, a breakthrough paper by Ajtai [Ajt96] showed that solving SVP on a random lattice on average according to a certain distribution which is easy to sample, involves a solution for the approximate SVP for any lattice within a polynomial approximation factor n^c. This connection illustrates the worst-case to average-case reduction. It shows that if the second mentioned problem is hard in some (worst) cases, then the first problem is also hard on average. Based on this concept, many cryptographic primitives were constructed such as Ajtai's one way function [Ajt96] and collision resistant hash functions [GGH96] illustrating the role of lattices in building cryptographic functions that are as difficult to break as the worst-case scenario of certain lattice problems.

Learning With Errors

A very versatile lattice primitive called *Learning With Errors* (LWE) was introduced by Regev in 2009 [Reg09]. The LWE problem guarantees a worst-case reduction from the shortest independent vector problem (SIVP) and the decision version of the shortest vector problem (GapSVP) for reasonable approximation factors in generic lattices. Therefore, it can be used to build a variety of cryptographic algorithms and provides guarantees in terms of IND-CPA (indistinguishability under chosen-plaintext attack) security [Reg09, KTX07, PVW08] and IND-CCA (indistinguishability under chosen-ciphertext attack) security [PW11, Pei09]. Other applications of the LWE problem include identity-based encryption schemes [GPV08, CHKP12, ABB10], various forms of leakage-resilient encryption [AGV09, DGK+10, GKPV10], homomorphic encryption [Bra12, BV14a, BGV14, SCC19] and much more. In order to fix ideas and to have a clearer vision of our discussion, we give below two formal definitions which present the LWE problem in its *search* and *decision* form.

Definition 1.1 (Search-LWE)**.** *Let $q = poly(n)$ be an integer and χ a Gaussian-like distribution over $\mathbb{Z}$. Let $m = poly(n)$, referred sometimes as the number of samples. For a secret vector $\mathbf{s} \in \mathbb{Z}_q^n$ and an $m \times n$ dimensional matrix $\mathbf{A}$ sampled uniformly from $\mathbb{Z}_q^{m \times n}$, consider an error term $\mathbf{e}$ drawn from the χ^m distribution and denote $\mathbf{b} = \mathbf{As} + \mathbf{e}$. The problem asks to recover the secret $\mathbf{s}$ given the pair $(\mathbf{A}, \mathbf{b})$.*

The second version of the problem is the *decision*-LWE.

Definition 1.2 (Decision-LWE)**.** *Consider the pair $(\mathbf{A}, \mathbf{b})$ with the same distribution as in Definition 1.1. The problem asks to distinguish between the two samples $(\mathbf{A}, \mathbf{b})$ and $(\mathbf{A}', \mathbf{b}')$, where $\mathbf{A}'$ and $\mathbf{b}'$ are uniform samples from $\mathbb{Z}_q^{m \times n}$ and $\mathbb{Z}_q^n$ respectively.*

Ring-LWE and Module-LWE

A structured variant of LWE, the decision *Ring Learning With Errors* (R-LWE) was proposed in [LPR10] by Lyubashevsky *et al.* to allow more compact representations and shorter public and secret keys. It consists in distinguishing polynomially many samples $(\mathbf{a}_i, \mathbf{a}_i \cdot \mathbf{s} + \mathbf{e}_i) \in R_q \times R_q$ from uniform samples $(\mathbf{a}_i, \mathbf{b}_i) \in R_q \times R_q$, where R is taken to be the ring of integers of some cyclotomic number field, i.e., the field $K = \mathbb{Q}(\zeta_N)$ with ζ_N being any primitive N-th complex root of unity. Note that each $\mathbf{a}_i$ is generated uniformly from R_q, whereas $\mathbf{e}_i$ and $\mathbf{s}$ are generated from a well defined distribution on R.

Solving R-LWE was shown to be at least as hard as solving approximate SIVP on ideal lattices, a special class of lattices generated from fractional ideals in the underlying field. Based on this hardness, many cryptographic applications were proposed including efficient signature schemes [Lyu12, MP12], fast encryption [LPR10], fast homomorphic encryption [GHS12, BGV14, BV11a] and pseudo-random functions [BPR12].

The R-LWE problem inspired the authors of [LS15] to introduce the *Module Learning With Errors* (M-LWE) variant. This new generalized form seems to offer better security guarantees and more flexibility in designing cryptographic schemes. Mathematically speaking, the problem consists in distinguishing uniform samples $(\vec{\mathbf{a}}_i, \mathbf{b}_i) \leftarrow R_q^d \times R_q$ from samples $(\vec{\mathbf{a}}_i, \mathbf{b}_i) \leftarrow R_q^d \times R_q$ where $\vec{\mathbf{a}}_i \leftarrow R_q^d$ is uniform, $\mathbf{b}_i = \langle \vec{\mathbf{a}}_i, \vec{\mathbf{s}} \rangle + \mathbf{e}_i$ with a fresh $\mathbf{e}_i$ generated from a well defined distribution Ψ on R_q, and $\vec{\mathbf{s}} \leftarrow \Psi^d$.

Similar to its former counterparts, M-LWE also enjoys worst-case to average-case reductions from lattice problems such as Mod-GapSVP$_\gamma$ and Mod-SIVP$_\gamma$ that are conjectured to be hard to solve on *module-lattices*, i.e., lattices corresponding to modules over the ring R, for γ polynomial in the lattice dimension. Note that Mod-GapSVP$_\gamma$ is easy for module rank equals 1, and conjectured to be hard when the rank is at least 2. The Module-LWE problem is attractive for cryptographic cryptosystems as it offers a trade-off between concrete security and efficiency depending on the dimension d.

Key encapsulation mechanisms (KEMs) based on LWE and its variants

The LWE problem, as well as its variants, have various applications in cryptographic systems and protocols due to their flexibility and efficiency. A great number of schemes have been proposed in the literature and it would be outside the scope of this thesis to illustrate all known applications. We will focus on two schemes that were submitted to the NIST challenge.

FrodoKEM A key encapsulation mechanism called FrodoKEM [N+20], which is based on the LWE problem, has been selected as an alternate candidate for the third round of the challenge. The core of FrodoKEM is a public-key encryption scheme called FrodoPKE which uses the method from Regev's LWE encryption scheme [Reg09] in order to encrypt multiple bits at a time. This scheme benefits from IND-CPA security. FrodoKEM is built from FrodoPKE by applying the Fujisaki-Okamoto transformation [HHK17] in order to obtain IND-CCA security for the resulting key encapsulation mechanism.

KyberKEM Two schemes based on Module-LWE have been proposed for the NIST standardization process: the signature scheme Dilithium [DKL+18] and the KyberKEM key encapsulation mechanism [A+20]. The latter scheme provides an IND-CCA secure KEM, which has first been described in [BDK+18]. Like most key encapsulation protocols in the literature, KyberKEM is derived from an IND-CPA secure public-key encryption scheme via a slightly tweaked Fujisaki-Okamoto transform.

Error-correction and reconciliation for noisy Diffie-Hellman

As mentioned above, many cryptographic schemes are based on LWE and its variants. In order to motivate the sequel of this thesis, we give explicit examples of such constructions. We present a common setting for LWE-based key encapsulation mechanisms (KEM) in Table 1.2.

Parameters: m, n, q and error distribution χ on $\mathbb{Z}_q$		
Alice (Server)		**Bob (Client)**
$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m\times n}$		$\mathbf{S}' \xleftarrow{\$} \chi^{\bar{m}\times m}, \mathbf{E}' \xleftarrow{\$} \chi^{\bar{m}\times n}$
$\mathbf{S} \xleftarrow{\$} \chi^{n\times\bar{n}}$, $\mathbf{E} \xleftarrow{\$} \chi^{m\times\bar{n}}$		$\mathbf{E}'' \xleftarrow{\$} \chi^{\bar{m}\times\bar{n}}$
$\mathbf{B} = \mathbf{AS} + \mathbf{E} \in \mathbb{Z}_q^{m\times\bar{n}}$	$\xrightarrow{(\mathbf{A},\mathbf{B})}$	$\mathbf{m} \xleftarrow{\$} \{0,1\}^\ell$
	$\xleftarrow{\mathbf{U}}$	$\mathbf{U} = \mathbf{S}'\mathbf{A} + \mathbf{E}'$
$\mathbf{V}' = \mathbf{US}$		$\mathbf{V} = \mathbf{S}'\mathbf{B} + \mathbf{E}''$

Table 1.2: Common setting for LWE-based KEMs.

We consider two terminals, Alice and Bob (Server and Client), whose aim is to generate the same private key on both sides. For concreteness we fix the set $\mathbb{Z}_q^{m\times n}$ for integers m, n, together with a modulus integer q. Alice chooses a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$ as well as two

random 'small' error matrices $\mathbf{S}, \mathbf{E}$ such that each component is generated from the Gaussian-like error distribution χ, and sends the LWE sample pair $(\mathbf{A}, \mathbf{B})$ as a public key to Bob. Bob on his side generates $\mathbf{S}', \mathbf{E}'$ and $\mathbf{E}''$ and computes the LWE samples

$$\mathbf{U} = \mathbf{S}'\mathbf{A} + \mathbf{E}' \bmod q, \quad \mathbf{V} = \mathbf{S}'\mathbf{B} + \mathbf{E}'' \bmod q.$$

The term $\mathbf{U}$ is sent back to Alice who uses it to calculate $\mathbf{V}' = \mathbf{US}$ with her secret key $\mathbf{S}$. Note that

$$\mathbf{V} - \mathbf{V}' = \mathbf{S}'\mathbf{E} + \mathbf{E}'' - \mathbf{E}'\mathbf{S} \bmod q. \tag{1.1}$$

When the distribution χ is chosen appropriately, the term $(\mathbf{S}'\mathbf{E} + \mathbf{E}'' - \mathbf{E}'\mathbf{S})$ is small with high probability, and therefore the values $\mathbf{V}$ and $\mathbf{V}'$ are close to $\mathbf{S}'\mathbf{AS}$. This observation is the basis of lattice-based encryption schemes and key encapsulation mechanisms.

The structure of the protocol brings to mind the Diffie-Hellman protocol [DH76] where the public parameter $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ corresponds to the generator g and the noise-free product is analogous to exponentiation. The presence of noise due to error terms leads to what we call *Noisy Diffie-Hellman* protocols. The term "Noisy Diffie-Hellman" was first used in a talk by P. Gaborit at PQCrypto 2010 [AGL+10b, AGL+10a] and also described in [LP11]. In this kind of protocols, if the random noise terms are large, an error will occur during the recovery of the private key, affecting the reliability of the scheme. As a result, it is necessary to choose the error distribution in such a way that the failure probability is guaranteed to be exponentially small.

As mentioned previously, thanks to the Fujisaki-Okamoto transform, one can transform an IND-CPA secure public-key encryption scheme into an IND-CCA secure key encapsulation mechanism. However, and especially when working in lattice-based cryptosystems, having small error probability is important not only for reliability, but also for security when going from IND-CPA to IND-CCA. For instance, an insufficiently small error probability can cause a leakage of information due to decryption failure attacks [DGJ+19] where a failure boosting technique is used to increase the failure rate. To keep the error probability small, one can use either *error correction* or *reconciliation* approaches. These techniques make it possible to agree on an exact shared private key (instead of an approximation) by providing Bob with some additional information.

Encryption-based approach. This first approach is used in many LWE-based encryption schemes, such as [LP11, N+20, A+20]. Table 1.3 illustrates this approach.

Parameters: $m, n, \bar{m}, \bar{n}, q$ and error distribution χ on $\mathbb{Z}_q$		
Alice (Server)		**Bob (Client)**
$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$		$\mathbf{S}' \xleftarrow{\$} \chi^{\bar{m} \times m}, \mathbf{E}' \xleftarrow{\$} \chi^{\bar{m} \times n}$
$\mathbf{S} \xleftarrow{\$} \chi^{n \times \bar{n}}$, $\mathbf{E} \xleftarrow{\$} \chi^{m \times \bar{n}}$		$\mathbf{E}'' \xleftarrow{\$} \chi^{\bar{m} \times \bar{n}}$
$\mathbf{B} = \mathbf{AS} + \mathbf{E} \in \mathbb{Z}_q^{m \times \bar{n}}$	$\xrightarrow{(\mathbf{A}, \mathbf{B})}$	$\mathbf{m} \xleftarrow{\$} \{0,1\}^{\ell}$
		$\mathbf{U} = \mathbf{S}'\mathbf{A} + \mathbf{E}'$
		$\mathbf{V} = \mathbf{S}'\mathbf{B} + \mathbf{E}''$
$\mathbf{V}' = \mathbf{US}$	$\xleftarrow{(\mathbf{U}, \mathbf{C})}$	$\mathbf{C} = \mathbf{V} + \textsc{Encode}(\mathbf{m})$
$\mathbf{m}' = \textsc{Decode}(\mathbf{C} - \mathbf{V}')$		

Table 1.3: Encryption-based KEM.

As shown, Bob unilaterally generates a uniform message $\mathbf{m} \in \{0,1\}^{\ell}$ and encodes it using some well defined encoding function Encode that maps $\{0,1\}^{\ell}$ into $\mathbb{Z}_q^{\bar{m} \times \bar{n}}$. He then sends the cyphertext $\mathbf{C} = \mathbf{V} + \textsc{Encode}(\mathbf{m})$ to Alice so that she can recover $\mathbf{m}'$ by applying the decoding function $\textsc{Decode}(\mathbf{C} - \mathbf{US})$. Note that in the context of public-key encryption, the encoding function Encode has to be injective so that Alice can recover the original message using Decode.

Reconciliation-based approach. This approach was originally described in [DXL12]. The reconciliation method allows two parties who have obtained noisy observations to come to an exact agreement about the value of the key, and consists in sending an auxiliary message from Bob to Alice in order to help her recover the private key from her noisy observation. The reconciliation approach in [Pei14], with priority to keep a low bandwidth, uses the one dimensional lattice $\mathbb{Z}$ in order to agree on one bit of private key. Other papers proposed reconciliation steps using higher-dimensional lattices, for instance [ADPS16b] considers the four dimensional lattice $\tilde{D}_4$.

Parameters: $m, n, \bar{m}, \bar{n}, q$ and error distribution χ on $\mathbb{Z}_q$		
Alice (Server)		**Bob (Client)**
$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$		$\mathbf{S}' \xleftarrow{\$} \chi^{\bar{m} \times m}, \mathbf{E}' \xleftarrow{\$} \chi^{\bar{m} \times n}$
$\mathbf{S} \xleftarrow{\$} \chi^{n \times \bar{n}}$, $\mathbf{E} \xleftarrow{\$} \chi^{m \times \bar{n}}$		$\mathbf{E}'' \xleftarrow{\$} \chi^{\bar{m} \times \bar{n}}$
$\mathbf{B} = \mathbf{AS} + \mathbf{E} \in \mathbb{Z}_q^{m \times \bar{n}}$	$\xrightarrow{(\mathbf{A}, \mathbf{B})}$	
		$\mathbf{U} = \mathbf{S}'\mathbf{A} + \mathbf{E}'$
		$\mathbf{V} = \mathbf{S}'\mathbf{B} + \mathbf{E}''$
$\mathbf{V}' = \mathbf{US}$	$\xleftarrow{(\mathbf{U}, \mathbf{R})}$	$\mathbf{R} = \textsc{HelpRec}(\mathbf{V})$
$\boldsymbol{\mu}' = \textsc{Rec}(\mathbf{V}', \mathbf{R})$		$\boldsymbol{\mu} = \textsc{Rec}(\mathbf{V}, \mathbf{R})$

Table 1.4: Reconciliation-based KEM.

As shown in Table 1.4, Bob produces a reconciliation message $\mathbf{R}$ using the function HelpRec and uses it to generate the private key $\boldsymbol{\mu}$ by applying the function Rec that computes a private

key $\boldsymbol{\mu}$ given a noisy observation and the reconciliation information. He then sends the message $\mathbf{R}$ to Alice. This information aims to correct the bias that exists between $\mathbf{V}$ and $\mathbf{V}'$. Alice can now use the function Rec to compute the private key $\boldsymbol{\mu}'$.

Relation to source coding with side information and physical layer security. The reconciliation problem is closely related to the problem of source coding with side information, i.e. the well-known Wyner-Ziv problem in information theory [WZ76,Kra08] illustrated in Figure 1.1.

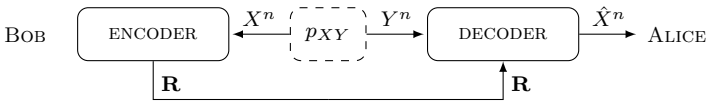

Figure 1.1: The Wyner-Ziv problem.

In this setting, Bob and Alice observe two correlated sources X^n and Y^n respectively. After receiving X^n, Bob sends a codeword $\mathbf{R}$ to Alice, who uses it together with her observation Y^n in order to reconstruct an estimate $\hat{X}^n$ of X^n with respect to a chosen distortion measure.

We note that in Table 1.4, $\mathbf{V}$ plays the role of X^n and $\mathbf{V}'$ plays the role of Y^n, but these observations do not have an i.i.d. distribution.

It is known that for jointly Gaussian sources with mean squared error distortion, the optimal Wyner-Ziv distortion function can be achieved using nested lattice codes [ZSE02].

Wyner-Ziv reconciliation using lattices has already been used in physical layer security, i.e. for key generation from Gaussian sources [LLB13]. In this work, the key generation problem was decoupled into two problems: randomness extraction (in order to extract from Bob's signal a private key which is independent from the eavesdropper's observations) and source coding with side information (sent by Bob to help Alice reconstruct the private key from her observation). It was shown that both objectives can be achieved with lattices. In particular, a lattice partition chain $\Lambda_3 \subseteq \Lambda_2 \subseteq \Lambda_1$ is used, where

- Λ_1 is good for quantization and serves as the "source-code" component of Wyner-Ziv coding;
- Λ_2 serves as the "channel-code" component in Wyner-Ziv coding;
- Λ_3 is used for randomness extraction.

Based on this, Bob computes the reconciliation message $\mathbf{R}$ as

$$\mathbf{R} = Q_{\Lambda_1}(\mathbf{V}) \bmod \Lambda_2$$

and transmits it to Alice. Furthermore, Bob computes the private key $\boldsymbol{\mu}$ as

$$\boldsymbol{\mu} = Q_{\Lambda_2}(\mathbf{V}, \mathbf{R}) \bmod \Lambda_3.$$

Here, the notations Q_Λ and $\bmod \Lambda$ indicate the lattice quantization function and modulo lattice operation corresponding to the lattice Λ.

In order to get the optimal rate-distortion function one needs to consider a sequence of lattices whose dimension tends to infinity [ZSE02]. However, this is not practical for cryptographic applications as the decoding complexity would become too large. Moreover, the result in [LLB13] is based on random lattices, and does not specify how to choose practical lattices in finite dimension.

Impact of error dependencies Developing error correction and reconciliation mechanisms for lattice-based protocols is a difficult challenge since the target error probability is far beyond the range of numerical simulations, and moreover the components of the error distribution are not independent. Some papers which use error correcting codes to improve the performance [LLZ$^+$18,L$^+$19] compute the error probability under an independence assumption which does not hold in practice. However, this has been shown to lead to underestimating the error probability by a very large exponential factor [DVV19]. In this thesis, we choose instead to derive rigorous error probability bounds following the example of [ADPS16b]. Evaluating those bounds still requires extensive numerical simulations which become very complex when the coding lattice Λ_2 is high-dimensional.

1.3 Contributions and main results

In this work, we aim to improve some of the existing key encapsulation mechanisms proposed for the NIST challenge in terms of security, reliability and bandwidth. One possible method of achieving these improvements is to introduce lattice-based error correction mechanisms or reconciliation techniques that improve performance, as has been done in previous works [Pei14, ADPS16b, vP16, SLL21].

Our main inspiration comes from the method of [LLB13], but we focus on reconciliation/error correction using a sublattice Λ_2 of small dimension in order to keep the complexity of the protocol low. This can be seen as an extension of the reconciliation/error correction mechanisms in NewHope [ADPS16b] and its successor NewHopeSimple [ADPS16a]. In particular, we look for a lattice that admits low-complexity quantization, and so we choose the Gosset lattice, as well as the small dimensional Barnes-Wall lattices. To go further with our study, we also consider higher dimensional Barnes-Wall lattices which admit an efficient bounded distance decoding algorithm up to half of the minimum distance [MN08].

Main results

- We first consider the alternative NIST candidate FrodoKEM [N+20], which is an LWE-based key encapsulation mechanism, and propose a modification at the level of the encoding function. We define a new encoder which maps the private key block-wise into the 8-dimensional Gosset lattice E_8. We propose three sets of parameters for our modified implementation. Thanks to the improved error correction, the first implementation allows to reduce the bandwidth by 7% by halving the modulus q; the second outperforms FrodoKEM in terms of plausible security by 10 to 13 bits by increasing the error variance, and the third one aims to increase the key size by approximately 50%. In all cases, our scheme can ensure a smaller decryption failure probability compared to the original FrodoKEM.

- Next, we focus on the KyberKEM protocol [A+20]. We propose a modification of KyberKEM featuring a reconciliation mechanism based on E_8. Similarly to KyberKEM, our scheme generates 256 bits of key and requires 5 or 6 bits of reconciliation per dimension. We show that it can outperform KyberKEM in terms of the modulus q with comparable error probability and similar requirements in terms of bandwidth. For instance, our construction guarantees a smaller error probability than KyberKEM-768's, i.e. $P_e \leq 2^{-174} < 2^{-164}$, with a smaller modulus $q = 2^{11} < 3329$, using 5 bits of reconciliation per dimension. For this choice of q, our scheme achieves 176 bits of post-quantum security compared to 164 bits. Such improvement can also be applied to KyberKEM-512, but it failed to extend to KyberKEM-1024. Note that unlike KyberKEM, where the modulus q is prime and the Number Theoretic Transform [CT65, LN16, Sei18] is used for fast polynomial multiplication, we choose q to be a power-of-two. In this case, efficient polynomial multiplication is still possible using Karatsuba / Toom-Cook algorithms such as [BCLV17, DWZ18, DKRV18]. Moreover, unlike [Pei14, ADPS16b], we don't need dithering anymore to obtain a uniform key thanks to the fact that q is even.

  In Appendix D, we investigate the use of Barnes-Wall lattices for reconciliation, but we are not able to improve KyberKEM's performance with this construction.

1.4 Organization of the thesis

A brief summary of each chapter is presented below where we highlight the main contributions and the tools used to achieve our results.

- **Chapter 2 - Preliminaries.** We start by giving the required notation in Section 2.1. We provide mathematical background about lattices and their properties in Section 2.2 and study some particular lattices for later use. The essential cryptographic definitions are organized in Section 2.3. We end the chapter by defining some Gaussian-like error distributions that are used in lattice-based cryptography (Section 2.4). Appendix A contains

proofs of some theorems cited in Chapter 2.

- **Chapter 3 - Error correction for FrodoKEM.** We recall the LWE problem in Section 3.2 with a recent reduction to the DGS-BDD problem derived from [N+20]. A brief introduction to the FrodoKEM protocol is given in Section 3.3 including some modifications proposed in the literature. Subsections 3.3.2 and 3.3.3 review the reliability and the security achievements of the scheme. In Section 3.4 we propose a new error correction mechanism in order to improve the performance of FrodoKEM. In Subsection 3.4.3, we derive an upper bound for the decryption failure probability of our modified protocol, while the concrete security is examined in Subsection 3.4.5. The performance comparison with the original FrodoKEM protocol is presented in Subsection 3.4.4. The details of the numerical simulations performed are presented in Appendix B.
- **Chapter 4 - KyberKEM with reconciliation.** Section 4.2 introduces the Module-LWE problem which is the basis of the KyberKEM protocol presented in Section 4.3. In Subsections 4.3.2 and 4.3.3, we review the decryption failure probability and security bounds for the protocol. In Section 4.4, we describe our proposed modification for the encoding and decoding function. In Subsections 4.4.3 and 4.4.4, we provide an error probability bound of our modified protocol, and we study the concrete security against known attacks. Finally, in Subsection 4.4.5 we compare the performance of the modified protocol with KyberKEM for different choices of parameters. Further details about our numerical simulations, as well as some partial results about higher-dimensional reconciliation using Barnes-Wall lattices are presented in Appendices C and D.
- **Conclusion and Perspectives.** This chapter concludes the thesis and provides some perspectives for future improvements.

1.5 Publications

- **International Conferences with proceedings**

  [1] C. Saliba, L. Luzzi, C. Ling, "A reconciliation approach to key generation based on Module-LWE", *IEEE International Symposium on Information Theory* (ISIT), Melbourne, Australia, July 2021.

  [2] C. Saliba, L. Luzzi, C. Ling, "Error Correction for FrodoKEM Using the Gosset Lattice", *International Zurich Seminar on Communications* (IZS), Zurich, Switzerland, March 2022.

- **National Conferences**

  [3] C. Saliba, L. Luzzi, C. Ling, "Key exchange based on Ring-LWE using Barnes-Wall lattices", *Journées Codage et Cryptographie (JC2)*, November 2020.

Chapter 2

Preliminaries

This chapter covers the preliminaries made use of in the next chapters. First, we give some guideline for the notation used in the manuscript. Next we present general background about lattices and study their properties. We give examples that are relevant in our work, along with some computationally hard problems on which we can define. After that, we cover cryptographic definitions around some basic protocols and provide security description. Finally, we recall some Gaussian-like distributions related to our work.

2.1 Notation

Throughout this thesis all logarithms are base 2 unless otherwise indicated. Vectors and matrices are written in boldface and $\mathcal{M}_{m,n}(\mathbb{A})$ denotes the set of $m \times n$ matrices with coefficients in the ring $\mathbb{A}$. The symbol i indicates the complex number $\sqrt{-1}$. A constant vector $(\alpha, \alpha, \ldots, \alpha)$ is simplified to $\boldsymbol{\alpha}$. The Euclidean norm of $\mathbf{x} \in \mathbb{R}^n$ is $\|\mathbf{x}\|$. The open n-dimensional ball of center $\mathbf{x}_0$ and radius r is $\mathcal{B}_n(\mathbf{x}_0, r) = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x} - \mathbf{x}_0\| < r\}$. The same way we define the closed ball $\bar{\mathcal{B}}_n(\mathbf{x}_0, r) = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x} - \mathbf{x}_0\| \leq r\}$. Rounding a real number $x \in \mathbb{R}$ to the nearest integer is denoted by $\lfloor x \rceil$, for which $\lfloor \pm a \pm 1/2 \rceil = \pm a$, $\forall\, a \in \mathbb{N}$. The rounding function can be extended to vector notation as $\lfloor \mathbf{x} \rceil = (\lfloor x_1 \rceil, \ldots, \lfloor x_n \rceil)$. The symbol $||$ is used for concatenation. Given $x \in \mathbb{R}$, we assign $\mathrm{sign}(x)$ to be 1 if $x \geq 0$, and -1 otherwise. The integer interval $\{a, a+1, \ldots, b\}$ is also denoted as $[[a, b]]$ for any $b > a$ in $\mathbb{Z}$, and $]]a, b]]$ when the integer a is excluded. For an even (resp. odd) positive integer m, the operation $r' = r \bmod^{\pm} m$ is defined to be the unique element r' in the range $]]-\frac{m}{2}, \frac{m}{2}]]$ (resp. $\left]\right]-\frac{m-1}{2}, \frac{m-1}{2}\left]\right]$). We write $X \sim Y$ to denote that two random variables X and Y have the same distribution. For a discrete distribution P, $\mathrm{Supp}(P)$ denotes the support of P, i.e., the set of points where the distribution is not zero. When an element a is taken uniformly at random from a set A we write $a \xleftarrow{\$} A$, and when it is an output of some function f we write $a \leftarrow f$.

A function $\epsilon(n)$ is called *negligible* if for every positive polynomial P, there exist a range N such that for all $n \geq N$ we have $\epsilon(n) < \frac{1}{P(n)}$.

We present some common asymptotic notation in Table 2.1 which can be used to describe the running time of algorithms.

Main Notation	Asymptotic Behaviour	Limit Definition
$f \in O(g)$	$f \leq g$	$\lim_{n\to\infty} f(n)/g(n) < \infty$
$f \in o(g)$	$f < g$	$\lim_{n\to\infty} f(n)/g(n) = 0$
$f \in \Omega(g)$	$f \geq g$	$\lim_{n\to\infty} f(n)/g(n) > 0$
$f \in \omega(g)$	$f > g$	$\lim_{n\to\infty} f(n)/g(n) = \infty$
$f \in \Theta(g)$	$f = g$	$\lim_{n\to\infty} f(n)/g(n) \in (0;\infty)$
$f \in \tilde{O}(g) \iff f \in O\left(g \cdot \log^k(g)\right)$ for some integer k		

Table 2.1: Asymptotic Notation.

2.2 Lattices

In this section we introduce the main definitions about lattices that will be needed in the sequel, and define some well known related computational problems.

2.2.1 Definitions and properties

Before we can proceed to the definitions of a lattice and its properties, it is first necessary to formalize the concept of a *discrete subset*.

Definition 2.1 (Discrete subset)**.** Let $S \subseteq \mathbb{R}^n$. We say that S is discrete if

$$\forall \mathbf{x}_0 \in S,\, \exists\, \varepsilon > 0 \,:\, \mathcal{B}_n(\mathbf{x}_0, \varepsilon) \cap S = \{\mathbf{x}_0\}.$$

An n-dimensional lattice is a discrete subset of $\mathbb{R}^n$ that it is closed under reflection and real addition, namely, it is a discrete subgroup of $\mathbb{R}^n$. Mathematically speaking, it can be defined using a set of linearly independent vectors as follows:

Definition 2.2 (Lattice)**.** Let $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m \in \mathbb{R}^n$ be a sequence of m linearly independent row [1] vectors ($m \leq n$). The lattice Λ is the set of all integral combinations of these vectors, i.e.,

$$\Lambda = \left\{ \sum_{i=1}^{m} x_i \cdot \mathbf{b}_i \,:\, x_i \in \mathbb{Z} \right\}.$$

The set $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m\}$ is called a *basis* for Λ, and it is not unique. The integer m is its rank and n is its dimension. The lattice is called *full-rank* if $n = m$. Throughout this thesis we will only consider full-rank lattices unless otherwise indicated. The norm of the shortest non-zero vector in Λ is denoted by $\lambda_1(\Lambda)$. Similarly, we define the *i-th successive minimum* for

[1] Throughout this work, we use the convention that lattice points are always represented as row vectors.

$i \geq 2$ as follows (see Figure 2.1):

$$\lambda_i(\Lambda) = \inf\left\{r \,:\, \bar{\mathcal{B}}_n(\mathbf{0}, r) \text{ contains at least } i \text{ linearly independent lattice vectors}\right\}.$$

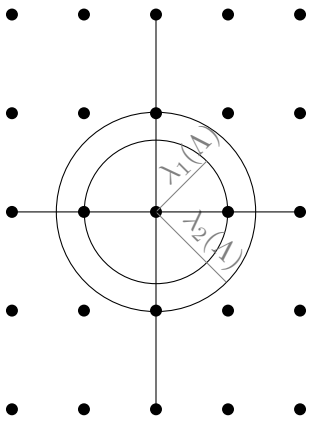

Figure 2.1: The first two successive minima $\lambda_1(\Lambda)$ and $\lambda_2(\Lambda)$.

Definition 2.3 (Generator matrix)**.** A generator matrix $\mathbf{G}$ of $\Lambda \subseteq \mathbb{R}^n$ is an $n \times n$ matrix whose rows form a basis of Λ, i.e.,

$$\mathbf{G} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_n \end{bmatrix},$$

where $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$ is a basis for Λ. In this case, the lattice is given by

$$\Lambda = \{\mathbf{x} \cdot \mathbf{G} \,:\, \mathbf{x} \in \mathbb{Z}^n\}.$$

and will be denoted $\Lambda(G)$.

Since a basis for a given lattice Λ is not unique, the generator matrix is also not unique. If $\mathbf{G}_1$ and $\mathbf{G}_2$ are two generator matrices for Λ, they only differ by a unimodular matrix $\mathbf{U}$, i.e., $\mathbf{G}_2 = \mathbf{U} \cdot \mathbf{G}_1$ with $\mathbf{U} \in GL(n, \mathbb{Z}) = \{\mathbf{A} \in \mathcal{M}_{n,n}(\mathbb{Z}) \,:\, \det(\mathbf{A}) = \pm 1\}$. This allows us to define a lattice invariant called *lattice determinant* and formalized as follows:

Definition 2.4 (Lattice determinant)**.** The lattice *determinant* $\det(\Lambda)$ is defined as the absolute determinant of its generator matrix $|\det(\mathbf{G})|$.

The determinant of a lattice is independent of the choice of the generator matrix since the determinant of any unimodular lattice is ± 1. This invariant quantity is also called *volume* of the lattice and denoted $\mathrm{Vol}(\Lambda)$.

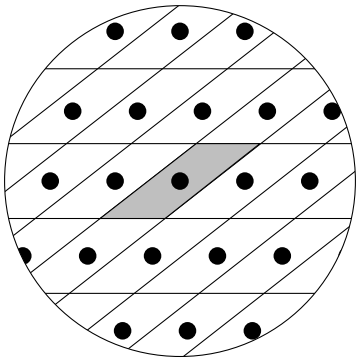

Figure 2.2: The union of translates of the shaded region covers $\mathbb{R}^n$ in a disjoint way.

Furthermore, one can obtain a transformed version of a given lattice which retains its structure, simply by applying a rotation and/or by multiplying by a constant. This leads to the following definition:

Definition 2.5 (Similar lattices)**.** Two lattices $\Lambda_1(\mathbf{G}_1)$ and $\Lambda_2(\mathbf{G}_2)$ are *similar* if there exist an orthogonal matrix $\mathbf{O} \in O(n)$, a unimodular matrix $\mathbf{Z} \in \mathrm{GL}(n, \mathbb{Z})$ and a scalar α such that

$$\mathbf{G}_2 = \alpha \cdot \mathbf{O} \cdot \mathbf{G}_1 \cdot \mathbf{Z}.$$

Fundamental region. Given a lattice, one can tile the entire space $\mathbb{R}^n$ using its *fundamental region* as a building block. This can be formalized as follows:

Definition 2.6 (Fundamental region)**.** A bounded set $\mathcal{F} \subseteq \mathbb{R}^n$ is a fundamental region of $\Lambda \subseteq \mathbb{R}^n$ if, when shifted by the lattice points, it generates a partition of $\mathbb{R}^n$; that is:

- $\bigcup_{\boldsymbol{\lambda} \in \Lambda} (\boldsymbol{\lambda} + \mathcal{F}) = \mathbb{R}^n$.
- For any distinct lattice points $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$, $(\boldsymbol{\lambda}_1 + \mathcal{F}) \cap (\boldsymbol{\lambda}_2 + \mathcal{F}) = \varnothing$.

Figure 2.2 illustrates an example of a fundamental region of a lattice. Note that for any fundamental region $\mathcal{F}$, a point $\mathbf{x} \in \mathbb{R}^n$ can be uniquely expressed as $\mathbf{x} = \boldsymbol{\lambda} + \mathbf{e}_Q$, where $\boldsymbol{\lambda} \in \Lambda$ and $\mathbf{e}_Q \in \mathcal{F}$. We write $\boldsymbol{\lambda} = Q_{\mathcal{F}(\Lambda)}(\mathbf{x})$ to be the *quantization function* applied to $\mathbf{x}$, and $\mathbf{e}_Q = \mathbf{x} \bmod \mathcal{F}(\Lambda) = \mathbf{x} - Q_{\mathcal{F}(\Lambda)}(\mathbf{x})$ to be the *quantization error*. Some properties that will be used implicitly in our proofs can be found in [Zam14] and are listed in the following lemma:

Lemma 2.1. Let Λ be an n-dimensional lattice with some predefined fundamental region $\mathcal{F}$. For all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and $\forall \boldsymbol{\lambda} \in \Lambda$ we have:

(P1) *Distributive law*: $(\mathbf{x} \bmod \mathcal{F}(\Lambda) + \mathbf{y}) \bmod \mathcal{F}(\Lambda) = (\mathbf{x} + \mathbf{y}) \bmod \mathcal{F}(\Lambda)$

(P2) *Shift-invariance*: $(\mathbf{x} + \boldsymbol{\lambda}) \bmod \mathcal{F}(\Lambda) = \mathbf{x} \bmod \mathcal{F}(\Lambda)$

(P3) $Q_{\mathcal{F}(\Lambda)}(\boldsymbol{\lambda} + \mathbf{x}) = \boldsymbol{\lambda} + Q_{\mathcal{F}(\Lambda)}(\mathbf{x})$

(P4) $\forall\, \alpha \in \mathbb{R},\, Q_{\alpha\mathcal{F}(\Lambda)}(\mathbf{x}) = \alpha \cdot Q_{\mathcal{F}(\Lambda)}\left(\frac{1}{\alpha}\mathbf{x}\right)$

Fundamental regions allow us to represent any point in space, in a lattice dependent way. Interestingly, their volume is a lattice invariant.

Theorem 2.1. All the fundamental regions of a lattice Λ have the same volume, namely $\det(\Lambda)$.

Voronoi region and Voronoi relevant vectors. Here we will introduce one of the most important fundamental regions called the *Voronoi region*.

Definition 2.7 (Voronoi Region)**.** For any lattice $\Lambda \in \mathbb{R}^n$, the Voronoi region $\mathcal{V}_{\boldsymbol{\lambda}_0}(\Lambda)$ associated with a lattice point $\boldsymbol{\lambda}_0$ contains all points in $\mathbb{R}^n$ that are closest to $\boldsymbol{\lambda}_0$ than any other lattice point. More formally,

$$\mathcal{V}_{\boldsymbol{\lambda}_0}(\Lambda) = \{\mathbf{x} \in \mathbb{R}^n \,:\, \|\mathbf{x} - \boldsymbol{\lambda}_0\| \leq \|\mathbf{x} - \boldsymbol{\lambda}\|,\, \forall \boldsymbol{\lambda} \in \Lambda \setminus \{\boldsymbol{\lambda}_0\}\}.$$

When $\boldsymbol{\lambda}_0 = \mathbf{0}$, we use the notation $\mathcal{V}(\Lambda)$. The corresponding quantization function denoted simply $Q_\Lambda(\mathbf{x})$ finds the *closest vector point* to $\mathbf{x}$.

Named after mathematician Georgy Voronoi, the Voronoi region has various applications in many fields of science and engineering. Several works in the literature have considered the problem of computing the Voronoi region for a given lattice such as [CS82c, CS84, BK18], and a recent result shows that it is computationally difficult to calculate the exact number of facets of the Voronoi cell for general lattices [DSSV09]. The *diamond-cutting* algorithm [VB96] computes a complete geometrical description of the Voronoi region of any lattice in exponential time and space. The memory requirements for this algorithm increase very rapidly with dimension, which limits its use to small dimensions. A deterministic algorithm presented in [MV13] allows to compute the Voronoi cell of a lattice in time $\tilde{O}\left(2^{2n}\right)$.

The Voronoi region is surrounded by finitely many lattice points. These points are called *Voronoi relevant vectors*. Figure 2.3 illustrates the Voronoi region in gray together with the Voronoi relevant vectors. To formalize this definition we need to recall what a bisecting normal hyperplane is.

Definition 2.8 (Bisecting normal hyperplane)**.** Let $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$. A *bisecting normal hyperplane* $\mathcal{B}(\mathbf{a}, \mathbf{b})$ between $\mathbf{a}$ and $\mathbf{b}$ is the set of points in space that are equidistant from $\mathbf{a}$ and $\mathbf{b}$ in Euclidean norm. Namely,

$$\mathcal{B}(\mathbf{a}, \mathbf{b}) = \{\mathbf{x} \in \mathbb{R}^n \,:\, \|\mathbf{x} - \mathbf{a}\| = \|\mathbf{x} - \mathbf{b}\|\}.$$

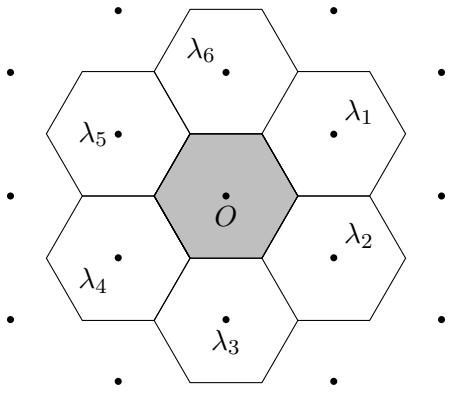

Figure 2.3: Voronoi region (in gray) surrounded by Voronoi relevant vectors.

Note that $\|\mathbf{x}-\mathbf{a}\| = \|\mathbf{x}-\mathbf{b}\| \iff \langle \mathbf{x}-\mathbf{a}, \mathbf{x}-\mathbf{a}\rangle = \langle \mathbf{x}-\mathbf{b}, \mathbf{x}-\mathbf{b}\rangle$, which is equivalent to $\|\mathbf{x}\|^2 - 2\langle \mathbf{a}, \mathbf{x}\rangle + \|\mathbf{a}\|^2 = \|\mathbf{x}\|^2 - 2\langle \mathbf{b}, \mathbf{x}\rangle + \|\mathbf{b}\|^2$. This leads to the following equivalent formula for the bisector normal hyperplane:

$$\mathcal{B}(\mathbf{a}, \mathbf{b}) = \left\{ \mathbf{x} \in \mathbb{R}^n \,:\, \langle \mathbf{x}, \mathbf{b}-\mathbf{a}\rangle = \frac{\|\mathbf{b}\|^2 - \|\mathbf{a}\|^2}{2} \right\} \tag{2.1}$$

Definition 2.9 (Voronoi relevant vectors)**.** Let Λ be an n-dimensional lattice and $\mathcal{V}(\Lambda)$ its Voronoi region. A point $\boldsymbol{\lambda} \in \Lambda$ is a *Voronoi relevant vector*, and we write $\boldsymbol{\lambda} \in \mathrm{VR}_\Lambda$, if the intersection of the bisecting normal hyperplane between $\mathbf{0}$ and $\boldsymbol{\lambda}$ with $\mathcal{V}$ is an $(n-1)$-dimensional face of $\mathcal{V}$. So if Γ is the set of all such faces of $\mathcal{V}$, the definition can be reformulated as

$$\boldsymbol{\lambda} \in \mathrm{VR}_\Lambda \iff \{\mathbf{x} \in \mathbb{R}^n \,:\, \langle \mathbf{x}, \boldsymbol{\lambda}\rangle = \|\boldsymbol{\lambda}\|^2/2\} \cap \mathcal{V} \in \Gamma.$$

Note that from Definition 2.9 we deduce that

$$\forall\, \mathbf{x} \in \mathbb{R}^n,\, \mathbf{x} \in \mathcal{V} \iff \langle \mathbf{x}, \boldsymbol{\lambda}\rangle \leq \frac{\|\boldsymbol{\lambda}\|^2}{2} \text{ for all Voronoi relevant vectors } \boldsymbol{\lambda} \in \Lambda. \tag{2.2}$$

For an n-dimensional Euclidean lattice, the number of Voronoi relevant vectors is at most $2(2^n-1)$ in the worst-case [Min05]. Calculating the Voronoi relevant vectors is not an easy problem and no polynomial algorithm is currently known. The diamond-cutting algorithm [VB96] can be used to determine a complete description of the Voronoi cell, which is unfavorable if used only to describe the Voronoi relevant vectors. In [MV13] the authors propose a deterministic $\tilde{O}\left(2^{2n}\right)$-time and $\tilde{O}\left(2^n\right)$-space algorithm that calculates the Voronoi relevant vectors. For small dimensional lattices one can use the Magma software [2] in order to calculate our Voronoi relevant vectors. A concrete example will be given in Subsection 2.2.3 while reviewing the Gosset lattice.

[2] This software is available online at `http://magma.maths.usyd.edu.au/calc/` with limited computing power.

Sublattice, quotient group and dual lattice. Defining a lattice Λ as a discrete group in $\mathbb{R}^n$ allows us to define a *sublattice* $\Lambda' \subseteq \Lambda$ as a subgroup of Λ.

Definition 2.10 (Sublattice)**.** Given an n-dimensional lattice $\Lambda \subseteq \mathbb{R}^n$, a sublattice Λ' is a subset of Λ satisfying:

- $0 \in \Lambda'$
- $\forall\, \alpha \in \mathbb{Z}$, $\forall\, \mathbf{x}, \mathbf{y} \in \Lambda'$, the vector $\alpha\mathbf{x} + \mathbf{y} \in \Lambda'$.

A criterion which allows us to know when a lattice is a sublattice of another is given in the proposition below.

Proposition 2.1. Let $\Lambda'(\mathbf{G}')$ and $\Lambda(\mathbf{G})$ be two n-dimensional lattices with generator matrices $\mathbf{G}'$ and $\mathbf{G}$ respectively. If $\mathbf{G}' \cdot \mathbf{G}^{-1}$ is an integer matrix, then $\Lambda' \subseteq \Lambda$.

Proof. Suppose $\mathbf{G}' \cdot \mathbf{G}^{-1}$ is an integer matrix and let $\mathbf{x} \in \Lambda'$. Then $\exists \mathbf{y} \in \mathbb{Z}^n$ such that $\mathbf{x} = \mathbf{y} \cdot \mathbf{G}'$. Therefore,

$$\mathbf{x} = \underbrace{\mathbf{y} \cdot \mathbf{G}' \cdot \mathbf{G}^{-1}}_{\mathbf{z} \in \mathbb{Z}^n} \cdot \mathbf{G} = \mathbf{z} \cdot \mathbf{G} \in \Lambda. \qquad \square$$

If $\Lambda' \subseteq \Lambda$ is a sublattice, then one can define the concept of the *quotient group*. For our interest, the quotient group between two lattices is defined with respect to the Voronoi region.

Definition 2.11 (Quotient group)**.** Let $\Lambda' \subseteq \Lambda$ be a sublattice of $\Lambda \subseteq \mathbb{R}^n$. The quotient Λ/Λ' is the set of cosets $(\boldsymbol{\lambda} + \Lambda')$ with $\boldsymbol{\lambda} \in \Lambda$. Note that it can be identified with the set of *coset leaders*, i.e. the points in $\Lambda \cap \mathcal{V}(\Lambda')$.

Since $\mathcal{V}(\Lambda')$ is bounded, the set Λ/Λ' is finite and the number of points inside this quotient can be determined by using the volume measure. In fact, $|\Lambda/\Lambda'| = \mathrm{Vol}(\Lambda')/\mathrm{Vol}(\Lambda)$. A useful lemma for our purposes is given below.

Lemma 2.2. Let $\Lambda' \subset \Lambda$ and $\boldsymbol{\lambda} \in \Lambda$ a fixed vector. The map $\pi : \Lambda/\Lambda' \to \Lambda/\Lambda'$ given by

$$\pi : \mathbf{v} \mapsto \pi(\mathbf{v}) = (\mathbf{v} + \boldsymbol{\lambda}) \bmod \Lambda'$$

is a permutation of Λ/Λ'.

Proof. Let $\mathbf{v}, \mathbf{v}' \in \Lambda/\Lambda'$ such that $\pi(\mathbf{v}) = \pi(\mathbf{v}')$. Then $(\mathbf{v} + \boldsymbol{\lambda}) \bmod \Lambda' = (\mathbf{v}' + \boldsymbol{\lambda}) \bmod \Lambda'$. This by definition can be written as $\mathbf{v} + \boldsymbol{\lambda} - Q_{\Lambda'}(\mathbf{v} + \boldsymbol{\lambda}) = \mathbf{v}' + \boldsymbol{\lambda} - Q_{\Lambda'}(\mathbf{v}' + \boldsymbol{\lambda})$, which means that $\mathbf{v} - \mathbf{v}' = \boldsymbol{\lambda}' \in \Lambda'$. Using Lemma 2.1 we deduce that $\mathbf{v} \bmod \Lambda' = \mathbf{v}' \bmod \Lambda'$. This proves injectivity. Since Λ/Λ' is a finite set, we obtain bijectivity. $\square$

We end up by giving the notion of the *dual lattice*, which is the set of points in space whose inner products with all the vectors in the original lattice are all integers.

Definition 2.12 (Dual lattice)**.** The *dual lattice* of a lattice Λ is defined as

$$\Lambda^* = \{\mathbf{x} \in \mathbb{R}^n \,:\, \langle \mathbf{x}, \Lambda \rangle\} \subseteq \mathbb{Z}.$$

In the next section, we study lattices from a computational perspective. We introduce some fundamental hard problems that are considered in lattice-based cryptography.

2.2.2 Computational problems on lattices

Several problems on lattices have fascinated scientists for many years, and so far, many of them are difficult to attack even on quantum computers. That's why the conjectured intractability of such problems is central to the construction of secure lattice-based cryptosystems. This vast area cannot be covered throughout this thesis and we only present the lattice problems that are most relevant to our study.

The most famous and widely studied computational problem on lattices is the *shortest vector problem*:

Definition 2.13 (Shortest vector problem SVP)**.** Given a lattice Λ in $\mathbb{R}^n$, the shortest vector problem asks to find a non-zero vector $\mathbf{v}$ such that $\|\mathbf{v}\| = \lambda_1(\Lambda)$.

Note that we have used the Euclidean norm here which is the most common, but it is important to mention that the problem can be posed with respect to any norm. This problem was extensively studied by mathematicians in the 19^{th} century and to date there is no polynomial time algorithm (with respect to the lattice dimension n) capable of solving it. For the Euclidean norm, the SVP problem has been conjectured to be NP-hard by van Emde Boas in 1981 [vEB81] before appearing in Ajtai's breakthrough where he showed the NP-hardness of the problem under randomized reductions [Ajt98].

An exact computation of SVP that runs in $2^{O(n \log n)}$ time was proposed in [Kan87] via the *enumeration method*, and requires polynomial space in n. Enumeration is simply an exhaustive search for an integer combination of the basis vectors such that the lattice vector is the shortest, inside some given region. Later in 2001, Ajtai *et al.* [AKS01] proposed a *sieving* probabilistic algorithm that solves SVP in time $2^{O(n)}$. The sieve algorithms perform some kind of randomized enumeration on a smaller set. These algorithms have a better asymptotic running time, but they all require exponential space. Another algorithm based on constructing the *Voronoi cell* was presented in 2001 by Micciancio and Voulgaris [Mic01] with a time complexity of $2^{2n+o(n)}$ and a space complexity of $2^{n+o(n)}$. The fastest known algorithm [ADRSD15] runs in provable $2^{n+o(n)}$ time and is based on *discrete Gaussian sampling*. Note that for most existing algorithms, the constants hidden in the exponents are relatively large, and the enumeration methods are in fact faster than other methods and commonly used in practice to find the shortest vectors in high dimensions [MW14]. Regarding the sieving methods, recent heuristic analyses show that one can solve SVP in time $2^{0.298n+o(n)}$ and space $2^{0.208n+o(n)}$ [LdW15].

The result by [ADRSD15] was known to be the best in the classical and quantum perspective, but more recently, [ACKS20] give a quantum algorithm that solves SVP in $2^{0.9533n+o(n)}$ time and classical $2^{0.5n+o(n)}$ space. By applying a quantum search algorithm, the authors of [LMVDP15] make asymptotic improvements regarding the resolution of the shortest vector problem. They improve the classical result of [Mic01] from $2^{2n+o(n)}$ to $2^{1.799n+o(n)}$, while heuristically they obtain $2^{0.268n+o(n)}$ time complexity rather than the classical result of $2^{0.298n+o(n)}$ time in [LdW15].

In practice, finding a reasonably short vector instead of a shortest vector is also sufficient for many applications, and the question of whether we can find a vector that approximates the shortest vector gives rise to a variant of the SVP problem called the γ-approximation version (SVP_γ).

Definition 2.14 (γ-approximation Shortest Vector Problem SVP_γ)**.** Given a lattice Λ in $\mathbb{R}^n$ and an approximation factor $\gamma \geq 1$, find a non-zero lattice vector of length at most $\gamma\lambda_1(\Lambda)$.

The approximation factor γ is usually a function of the lattice dimension n and may be exponentially large value in n. The famous work by [LLL82] gave a polynomial time algorithm that solves SVP_γ for $\gamma = 2^{n/2}$. Later works have improved this result; it is currently known that there exists a polynomial time algorithm achieving an approximation factor of $2^{n \log\log n/\log n}$ in [AKS01], and several heuristic variants of it are found in [MV10, BLS16, HK17].

Solving SVP_γ for constant values of γ may be a hard problem. In fact, it was shown that for approximation factors up to $\gamma = 2^{(\log n)^{1-\varepsilon}}$ (not polynomial in n, but very close to it), the problem is NP-hard [Mic01, Kho05, HR07]. Further, the SVP_γ problem is unlikely to be NP-hard for $\gamma = \Omega(\sqrt{n})$, as this leads to the equality NP=coNP [Has88, LLS90, Ban93, AR05]. Moreover, showing the NP-hardness up to factor $\sqrt{n/O(\log n)}$ would imply the collapse of the polynomial-time hierarchy [GG00]. When γ is taken to be poly(n), the fastest known algorithms to solve SVP_γ rely on (a variant of) the BKZ lattice basis reduction algorithm [Sch87, SE94a, AKS01, GN08, HPS11, ALNSD20].

The shortest vector problem can be extended to a more generalized form that we call the *Shortest Independent Vector Problem*.

Definition 2.15 (Shortest Independent Vector Problem SIVP_γ)**.** Given a lattice Λ in $\mathbb{R}^n$, find n linearly independent lattice vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ where $\|\mathbf{v}_i\| \leq \gamma\lambda_n(\Lambda)$ for all $i = 1, \ldots, n$.

This problem is involved in many cryptographic primitives [Ajt96, Reg09, GPV08, Pei10, Pei15]. Blömer and Seifert [BS99] showed that SIVP_γ is NP-hard for any constant approximation factor γ with respect to the Euclidean norm. Micciancio and Voulgaris [MV13] showed that exact SIVP can be solved deterministically with $\tilde{O}\left(2^{2n+o(n)}\right)$ operations. This problem is believed to be hard to approximate up to polynomial factors in n, and the best known algorithms within a polynomial approximation factors run in time exponential in n [ADRSD15]. Till now, there are no known quantum algorithms for solving SIVP_γ that outperform classical ones.

Another important problem associated with lattices is the *Closest Vector Problem*.

Definition 2.16 (Closest Vector Problem CVP_γ)**.** Given a lattice Λ in $\mathbb{R}^n$, a point $\mathbf{t}$ in $\mathbb{R}^n$ and an approximation factor $\gamma(n) \geq 1$, find a lattice vector at distance at most $\gamma \cdot \underset{\boldsymbol{\lambda} \in \Lambda}{\operatorname{argmin}} \|\boldsymbol{\lambda} - \mathbf{t}\|$.

A reduction from SVP_γ to CVP_γ was given in [GMSS99] showing that approximating the closest lattice vector is at least as hard as approximating the shortest lattice vector. A reduction in the other direction that preserves the dimension is still unknown. CVP has been proven in [vEB81] to be NP-hard, just as CVP_γ for $\gamma = n^{c/\log\log n}$ and $c < 1/2$ [DKS98]. A polynomial time algorithm that solves CVP_γ is the Babai nearest plane algorithm for $\gamma = 2(2/\sqrt{3})^n$ [Bab86]. An exact solution for CVP is given in [Kan87] using an enumeration method that runs in time $n^{O(n)}$, and many other improvements with better running times can be found in [Hel85, HS07, MW14]. Another algorithm that solves exact CVP in $2^{n+o(n)}$-time is given in [ADSD15].

We end up our discussion with one more problem connected to lattices which is considered a special version of the closest vector problem, namely the *Bounded Distance Decoding* problem.

Definition 2.17 (Bounded Distance Decoding α- BDD)**.** Given a lattice Λ in $\mathbb{R}^n$ and a point $\mathbf{t}$ such that its distance from the lattice is at most $\alpha \cdot \lambda_1(\Lambda)$, find the closest lattice vector to $\mathbf{t}$.

Klein [Kle00] solves the problem for $\alpha = O(1/n)$ in polynomial time, and [LLM06] improve the factor α to $O\left(\sqrt{\log n/n}\right)$ using pre-processing. The α- BDD problem is known to be NP-hard for $\alpha > 1/\sqrt{2}$ [LLM06].

Many other lattice problems exist in the literature and it is infeasible to include them all in one context. SVP_γ and CVP_γ admit various variations such as:

- $\mathsf{unique\text{-}SVP}_\gamma$: Given a lattice Λ such that $\lambda_2(\Lambda) > \gamma\lambda_1(\Lambda)$, find the shortest non-zero lattice vector in Λ.
- GapSVP_γ: Given a lattice Λ, the problem is to decide whether $\lambda_1(\Lambda) < 1$ or $\lambda_1(\Lambda) > \gamma$.

2.2.3 The Gosset lattice E_8

In this subsection, we are going to introduce the most relevant lattice for our study which will be used in the sequel. The *Gosset lattice* E_8 is the densest 8-dimensional lattice [Via17] defined as

$$E_8 = \left\{ (x_0, \ldots, x_7) \in \mathbb{Z}^8 \cup \left(\mathbb{Z} + \tfrac{1}{2}\right)^8 \ : \ \sum_{i=0}^{7} x_i = 0 \bmod 2 \right\}.$$

In particular, E_8 is generated by the following matrix:

$$\mathbf{G}_{E_8} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 \end{bmatrix} \tag{2.3}$$

As a result of how E_8 is defined, it is important to note that $2\mathbb{Z}^8 \subseteq E_8 \subseteq \frac{1}{2}\mathbb{Z}^8$. The volume of E_8 is $|\det(\mathbf{G}_{E_8})| = 1$ and its minimal norm is $\lambda_1(E_8) = \sqrt{2}$. The Voronoi relevant vectors can be divided into two sets. The first set $\mathrm{VR}_{E_8}^{(1)}$ contains 112 vectors of the form $(\pm 1^2, 0^6)$, while the second set $\mathrm{VR}_{E_8}^{(2)}$ consists of 128 vectors of the form $(\pm 0.5^8)$, giving a total of 240 vectors [CS13]. The list of Voronoi relevant vectors can be obtained via a simple Magma code which can be found in Appendix A.3.

Finding the closest point of E_8 We present here an efficient algorithm to find the closest point in E_8 to a vector $\mathbf{x} \in \mathbb{R}^8$, which was proposed in [CS82a]. We first define $f(\mathbf{x}) = \lfloor \mathbf{x} \rceil$, and $g(\mathbf{x}) = \rfloor \mathbf{x} \lceil$ to be the same as $\lfloor \mathbf{x} \rceil$ except that the worst component - that furthest from an integer - is rounded the wrong way. More formally, if $i_0 = \underset{0 \le i \le 7}{\operatorname{argmax}} |x_i - \lfloor x_i \rceil|$ then for $i = 0, \ldots, 7$ we have

$$\rfloor \mathbf{x} \lceil_i = \begin{cases} \lfloor x_i \rceil & \text{if } i \neq i_0 \\ \lfloor x_i \rceil + \operatorname{sign}(x_i) \cdot \operatorname{sign}(|x_i| - \lfloor |x_i| \rceil) & \text{if } i = i_0 \end{cases}$$

The algorithm computes $f(\mathbf{x})$ and $g(\mathbf{x})$, and selects the one that has an even sum of components. Call that vector $\mathbf{y}$. Then it computes $f(\mathbf{x} - \mathbf{1/2})$ and $g(\mathbf{x} - \mathbf{1/2})$, and selects whichever has an even sum of components; adds $\mathbf{1/2}$ and calls the result $\mathbf{y}'$. At the end, the output vector is the closest one to $\mathbf{x}$ among $\mathbf{y}$ and $\mathbf{y}'$. A pseudo-code for the algorithm is presented in Algorithm 1.

Algorithm 1 Closest Vector Point in E_8

function $\mathsf{CVP}_{E_8}(\mathbf{x} \in \mathbb{R}^8)$
  $\mathbf{f} = \lfloor \mathbf{x} \rceil$; $\mathbf{g} = \rfloor \mathbf{x} \lceil$
  $\mathbf{y} = (1 \oplus \sum f_i)\, \mathbf{f} + (1 \oplus \sum g_i)\, \mathbf{g}$
  $\mathbf{f}' = \lfloor \mathbf{x} - \frac{\mathbf{1}}{\mathbf{2}} \rceil$; $\mathbf{g}' = \left\rfloor \mathbf{x} - \frac{\mathbf{1}}{\mathbf{2}} \right\lceil$
  $\mathbf{y}' = (1 \oplus \sum f'_i)\, \mathbf{f}' + (1 \oplus \sum g'_i)\, \mathbf{g}' + \frac{\mathbf{1}}{\mathbf{2}}$
  return $\underset{\boldsymbol{\lambda} \in \{\mathbf{y}, \mathbf{y}'\}}{\operatorname{argmin}} \|\mathbf{x} - \boldsymbol{\lambda}\| \in E_8$

An infinite family of lattices with good (although not optimal) packing properties which generalizes the E_8 lattice was introduced by Barnes and Wall [BW59] and is discussed in the next section.

2.2.4 Barnes-Wall lattices

Barnes-Wall lattices BW^n form a sequence of full-rank lattices whose dimension n is a power-of-two. These lattices are attractive due to their simplicity and relevance for practical applications [AV00, BB98, For88, FV96].

Definition 2.18 (Barnes-Wall lattice)**.** For a power-of-two integer $n \geq 2$, a generator matrix of BW^n can be defined recursively as:

- $\mathbf{G}_{BW^2} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
- For $n \geq 4$, $\mathbf{G}_{BW^n} = \left[\begin{array}{c|c} \mathbf{G}_{BW^{\frac{n}{2}}} & \mathbf{G}_{BW^{\frac{n}{2}}} \\ \hline 0 & \Phi_{\frac{n}{2}} \cdot \mathbf{G}_{BW^{\frac{n}{2}}} \end{array}\right]$, with $\Phi_n = I_{\frac{n}{2}} \otimes \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$.

Remark 2.1. We note that this definition is equivalent to the definition of Barnes-Wall lattices as $\frac{n}{2}$-dimensional complex lattices over the Gaussian integers [MN08], by converting complex numbers into matrices as

$$\begin{aligned} \mathbb{C} &\to \mathbb{R}^{2\times 2} \\ a + \mathrm{i}b &\mapsto \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \end{aligned}$$

The Barnes-Wall lattice BW^n has volume $\mathrm{Vol}(BW^n) = \sqrt{(n/2)^{n/2}}$ and minimal distance $d_{\min}(BW^n) = \sqrt{n/2}$.

Micciancio and Nicolosi [MN08] give a polynomial time algorithm to solve the bounded distance decoding ($\frac{1}{2}$- BDD) for Barnes-Wall lattices: given a vector $\mathbf{s} \in \mathbb{R}^n$ within distance $d_{\min}(BW^n)/2 = \sqrt{n/8}$ from some lattice point $\boldsymbol{\lambda}$ in BW^n, find $\boldsymbol{\lambda}$. This algorithm called ParBW has complexity $\mathcal{O}\left(n \log^2 n\right)$ and induces a partition of the space $\mathbb{R}^n$ as a result of the following theorem:

Theorem 2.2. Let $\mathbf{s} \in \mathbb{R}^n$. For a fixed $\boldsymbol{\lambda} \in BW^n$, we have

$$\textsc{ParBW}(\boldsymbol{\lambda} + \mathbf{s}) = \boldsymbol{\lambda} + \textsc{ParBW}(\mathbf{s}).$$

In other words, ParBW induces a partition of $\mathbb{R}^n$ into fundamental cells.

A proof of this theorem can be found in Appendix A.5. Another important property that characterizes the scaling of integer lattices inside BW^n is given below, with a proof in Appendix A.6.

Proposition 2.2. *For $n \geq 2$ and $k \geq \left\lfloor \frac{\log n}{2} \right\rfloor$, $2^k\mathbb{Z}^n \subseteq BW^n \subseteq \mathbb{Z}^n$.*

The lattice BW^{16}

The densest known lattice in $\mathbb{R}^{16}$ is the 16-dimensional Barnes-Wall lattice BW^{16} that was first introduced in [BW59], and appears later in [CS82b, LS71, CS84]. This lattice is obtained through different constructions such as applying Construction B to the first-order Reed-Muller code $[16, 5, 8]$, as well as through Construction C (or D) and many other methods [CS13].

The generator matrix of this lattice is given in [Slo81, page 336] by the following triangular matrix

$$\mathbf{G}_{BW^{16}} = \frac{1}{2}\begin{bmatrix}
4 & & & & & & & & & & & & & & & \\
2 & 2 & & & & & & & & & & & & & & \\
2 & 0 & 2 & & & & & & & & & & & & & \\
2 & 0 & 0 & 2 & & & & & & & & & & & & \\
2 & 0 & 0 & 0 & 2 & & & & & & & & & & & \\
2 & 0 & 0 & 0 & 0 & 2 & & & & & & & & & & \\
2 & 0 & 0 & 0 & 0 & 0 & 2 & & & & & & & & & \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & & & & & & & & \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & & & & & & & \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & & & & & & \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & & & & & \\
1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & & & & \\
0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & & & \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & & \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}. \tag{2.4}$$

The volume for the corresponding scaling is $1/16$ and the minimal distance is $\sqrt{2}$. The Voronoi relevant vectors can be divided into three sub-types:

$$(\pm 1^2, 0^{14}) \in \mathrm{VR}^{(1)}_{BW^{16}},\ (\pm 0.5^8, 0^8) \in \mathrm{VR}^{(2)}_{BW^{16}},\ \text{and } (\pm 0.5^8, \pm 1, 0^7) \in \mathrm{VR}^{(3)}_{BW^{16}},$$

with cardinality

$$\left|\mathrm{VR}^{(1)}_{BW^{16}}\right| = 480, \left|\mathrm{VR}^{(2)}_{BW^{16}}\right| = 3839 \text{ and } \left|\mathrm{VR}^{(3)}_{BW^{16}}\right| = 61441.$$

The Barnes-Wall lattice BW^{16} contains the sublattice $2D_{16}$ for which $\left|BW^{16}/2D_{16}\right| = 32$ [CS84]. Recall that the lattice D_n is defined by

$$D_n = \left\{ (x_1, \ldots, x_n) \,:\, \sum_{i=1}^{n} x_i = 0 \bmod 2 \right\}.$$

A low-complexity algorithm to find the closest lattice point in D_n to a point $\mathbf{x} \in \mathbb{R}^n$ is given in [CS82a]. Using the decomposition of BW^{16} into cosets of D_{16}, it is possible to obtain a CVP algorithm for BW^{16} from the CVP algorithm for D_{16}. In fact, if $\Lambda = \bigcup_{i=0}^{d-1} (r_i + \Lambda')$ is a union of cosets of Λ' and Φ is an algorithm that calculates the CVP on Λ', then the closest point to $\mathbf{x}$ in Λ is obtained by comparing each of $\Phi(\mathbf{x} - r_i) + r_i$ with $\mathbf{x}$ and choosing the closest [CS82a]. In

our case, $\Lambda = BW^{16}$, $\Lambda' = 2D_{16}$ and r_i are the codewords of the $[16, 5, 8]$ first-order Reed-Muller code.

2.3 Cryptographic definitions

Security protocols today are based either on private key cryptosystems or on public key cryptosystems. Private key cryptography uses the same shared key to ensure communication between the parties. For the most part, the generation of this shared key is performed through public key protocols.

2.3.1 Public-key encryption scheme (PKE)

Public-key encryption is a procedure that allows to encrypt messages using a pair of keys known as a *public key* (which is available for anyone to use) and a *secret key* (kept secret by the intended receiver). Given a finite message space $\mathcal{M}$ and a ciphertext space $\mathcal{C}$, public-key encryption involves three efficient algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ define as follows:

- $\mathsf{Gen}(pp)$ is a probabilistic algorithm that outputs a secret key sk and a public key pk using a public parameter pp as input.
- $\mathsf{Enc}(pp, pk, m)$ is a probabilistic algorithm that encrypts an input message $m \in \mathcal{M}$ using the public key pk and the parameter pp, in order to return a ciphertext $c \in \mathcal{C}$.
- $\mathsf{Dec}(sk, c)$ is a deterministic decryption algorithm that operates on the secret key sk and ciphertext c and delivers a message $\hat{m} \in \mathcal{M}$ or an error symbol $\perp$ denoting decryption failure.

Figure 2.4 illustrates the public-key encryption scheme. In order for Bob to send a message m to Alice, she produces the public key pk as well as the secret key sk. Upon receiving the public key, Bob encrypts m using pk and sends it to Alice who can now decrypt the message using sk and obtain $\hat{m}$ or an error symbol $\perp$. Note that in some schemes, particularly in lattice-based cryptography, the recovered message $\hat{m}$ may be different from the transmitted one. That's why we define the correctness of the scheme in Definition 2.19 below.

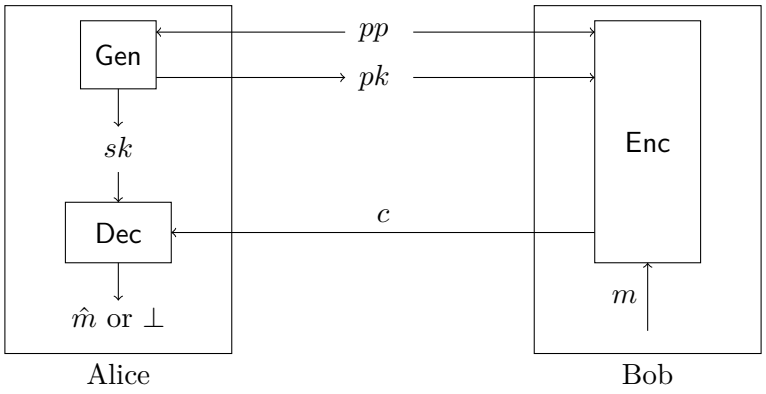

Figure 2.4: Simplified illustration of PKE.

Definition 2.19 (Correctness for PKEs)**.** A public-key encryption PKE is δ-*correct* if

$$\mathbb{P}\left\{\hat{m} \neq m \;\middle|\; \begin{smallmatrix}(pk,sk)\longleftarrow\mathsf{Gen}(pp)\\ c\longleftarrow\mathsf{Enc}(pp,pk,m)\\ \hat{m}\longleftarrow\mathsf{Dec}(sk,c)\end{smallmatrix}\right\} \leq \delta.$$

Remark 2.2. Sometimes we also associate a randomness space $\mathcal{R}$ to the PKE to denote the fact that the randomness used in Enc is picked uniformly from $\mathcal{R}$ on every execution. Note that if $\mathcal{R}$ is an empty set, then the encryption algorithm is purely deterministic. In some cases, the public parameter pp is excluded from $\mathsf{Enc}(pp, pk, m)$ as input, and replaced instead by an element of $\mathcal{R}$ if such a randomness space $\mathcal{R}$ is taken into account.

2.3.2 Key encapsulation mechanism (KEM)

We define the notion of *key encapsulation mechanism* (KEM) as the procedure for transmitting an ephemeral private key to a receiver using the latter's public key. Given a ciphertext space $\mathcal{C}$ and finite key space $\mathcal{K}$, it consists of a tuple of efficient algorithms $(\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$:

- $\mathsf{Gen}(pp)$ is a probabilistic algorithm that takes a public parameter pp as input and returns a secret key sk and a public key pk.
- $\mathsf{Encaps}(pp, pk)$ is a probabilistic algorithm that takes (pp, pk) as input and produces a ciphertext $c \in \mathcal{C}$ and a private key $k \in \mathcal{K}$.
- $\mathsf{Decaps}(sk, c)$ is a deterministic algorithm that takes the secret key sk and ciphertext c to return a key $\hat{k} \in \mathcal{K}$ or a special symbol $\perp$ to indicate that c is not a valid encapsulation.

A visual representation of the scheme is given in Figure 2.5.

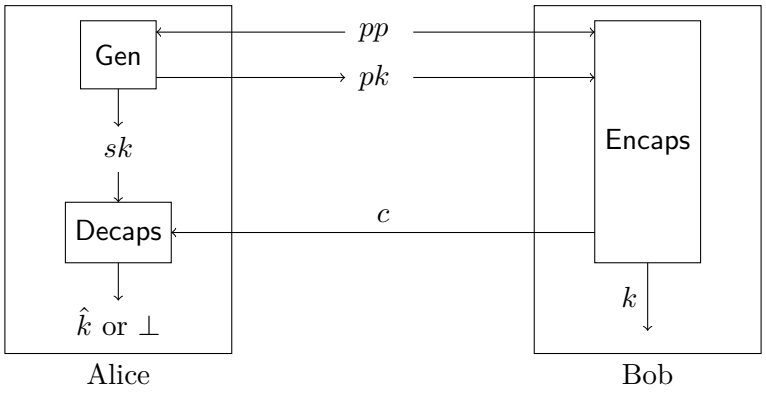

Figure 2.5: Simplified illustration of KEM.

Definition 2.20 (Correctness for KEMs)**.** A key encapsulation mechanism KEM is δ-correct if

$$\mathbb{P}\left\{\hat{k} \neq k \;\middle|\; \begin{array}{c}(pk,sk)\longleftarrow\mathsf{Gen}(pp)\\(c,k)\longleftarrow\mathsf{Encaps}(pp,pk)\\\hat{k}\longleftarrow\mathsf{Decaps}(sk,c)\end{array}\right\} \leq \delta.$$

2.3.3 Cryptographic attacks and security

Cryptographic systems have always been vulnerable to malicious attackers who aim to find out something about the transmitted information or even the secret key. There are different types of attacks that can be applied to different encryption systems, each with a different level of effectiveness. One can analyze security with regard to the most well-known attacks existing in the literature, or through a theoretical approach by measuring the probability that an attacker could guess the encrypted data. The *advantage* of an adversary measures its ability to attack a cryptographic algorithm by comparing the actual algorithm to an idealized version of that type of algorithm. One useful tool for security proofs is the concept of *random oracle model* (ROM). A random oracle is a designed strategy in which a "black box" takes some input data and generates an output, uniformly and randomly, in some conventional space (the space of oracle outputs). Usually the black box is modeled as a hash function. Given an already seen input, the random oracle returns the same output it returned the last time. This model turns out to be very useful for analyzing the security of cryptographic schemes. Referring to [Puc05], a system is secure in the random oracle model if its security can be proven, supposing all parties (including the adversary) have access to the random oracle. Most random oracle systems to date are only proven secure in the context of classical attacks, that is, the attacker can only learn an output value by querying the oracle at the classical state input x. We call this the *classical random oracle model*. It turns out that as we move beyond classical computing, the random oracle model needs to be revisited in the context of quantum attacks. In this model, the attacker may query the random oracle in superposition over multiple x's. We call this the *quantum random oracle model*.

One of the simplest attacks is *brute force attack*. The attacker tries to check all the differ-

ent possibilities in order to get the information he wants. Modern cryptographic systems are secure against these types of attacks because they impose an exponential number of different possibilities that make searching impractical for an attacker. Even quantum computers can only provide a quadratic speedup for black-box queries over a classical brute-force attack using Grover's algorithm, and this can be settled by doubling the size of the underlying security parameters.

Attacks can also target the weakness in physical implementation by analyzing the different run times with different plaintexts or secret keys. This is what we call a *timing attack*. To prevent this kind of attack from happening, a constant-time implementation independent of the input is required.

In public key cryptography, since the public key allows anyone to encrypt any desired message, an attacker could have the ability to obtain ciphertexts corresponding to arbitrary plaintexts. This kind of attack, called *chosen plaintext attack* (CPA), could lead to extracting the secret key used in the underlying cryptosystem. A basic requirement for most modern cryptographic schemes is the property of indistinguishability under chosen-plaintext attack (IND-CPA) for which an attacker is unable to distinguish to which message corresponds a given ciphertext.

A more powerful form of attack called a chosen ciphertext attack (CCA) is a scenario in which the attacker is able to not only encrypt the chosen message, but also collect a piece of ciphertext and obtain the corresponding decrypted plaintext. The security requirement for this context is the indistinguishability under chosen-ciphertext attack (IND-CCA). The concept is the same as for IND-CPA security, with the restriction that the attacker cannot query the decryption of the received ciphertext. The formal definition of IND-CPA and IND-CCA security is discussed in the next Section with respect to the PKE and KEM cryptosystems. Note that IND-CCA security implies IND-CPA security.

Finally, when talking about practical perspectives, a cryptosystem can have different forms of attacks for which we can estimate *concrete security*. It consists in exploring the minimum attack requirements in terms of computational complexity in order to quantify the amount of computations necessary for an adversary to break the system. It provides an upper bound on the advantage of the opponent to break the system according to the existing resources in hand, which are usually *running time* and *memory*.

IND-CPA security for PKE

We define a game to examine whether an attacker is able to identify which message has been encrypted into a given ciphertext with significantly better probability than random guessing. We consider the following game between a challenger and an adversary:

- The challenger derives a public and secret key pk and sk using $\mathsf{Gen}()$.

- The adversary sends the challenger two messages m_0 and m_1 in order to launch the game.
- The challenger chooses a uniformly random bit b bit and encrypts the message m_b message using $\mathsf{Enc}()$ to obtain the ciphertext c^*.
- The attacker receives the decrypted message c^* and tries to guess the original message m_b using $\mathsf{Enc}()$ itself (since it is public). He then outputs its guess b'.
- If $b' = b$, the adversary wins the game. Otherwise, he loses.

Figure 2.6 illustrates the game above.

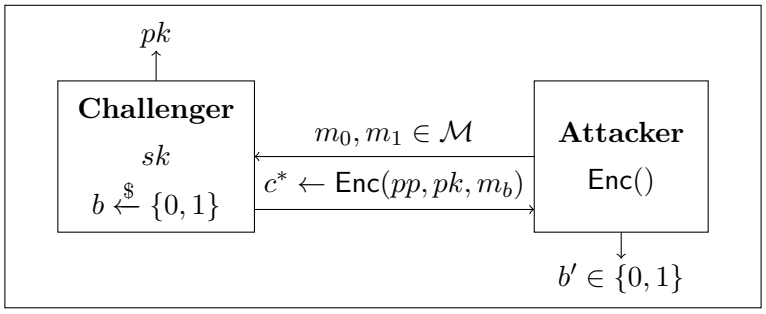

Figure 2.6: Simplified illustration of IND-CPA game for PKE.

The public-key encryption scheme is considered IND-CPA secure if any polynomial time adversary has only a negligible *advantage* over random guessing. This advantage is defined as

$$\mathsf{Adv}_{\mathrm{PKE}}^{\text{IND-CPA}} = \left| \mathbb{P}\left\{b' = b\right\} - \frac{1}{2} \right|.$$

Deterministic encryption algorithms cannot be IND-CPA secure. In fact, if the scheme were deterministic, then the adversary could directly use the encryption algorithm to distinguish which message was encrypted. In contrast, if the underlying encryption scheme is probabilistic, the encrypted message will be only one of many valid ciphertexts, and hence comparing between $\mathsf{Enc}(m_0)$ and $\mathsf{Enc}(m_1)$ does not provide any non-negligible advantage to the adversary.

IND-CPA security for KEM

The notion of IND-CPA security has been extended to also apply to key encapsulation mechanisms. It considers how well an adversary is able to distinguish the real shared private key from a random one, using the public Encaps function. The corresponding indistinguishability game can be illustrated as follows:

- The challenger obtains the public key pk and the secret one sk with the aid of the Gen function.
- The challenger picks a uniform bit b and obtains $(c^*, k_0) = \mathsf{Encaps}(pp, pk)$. If $b = 0$, it outputs (pk, c^*, k_0). If $b = 1$, it outputs (pk, c^*, k_1) where k_1 is generated uniformly at random in the key space $\mathcal{K}$.

- The adversary tries to guess whether k_b corresponds to k_0 or k_1, using $\mathsf{Encaps}()$ and returns a guessing bit b'.
- If $b' = b$, then the attacker was able to distinguish the two games and therefore wins the challenge.

Figure 2.7 illustrates the game above.

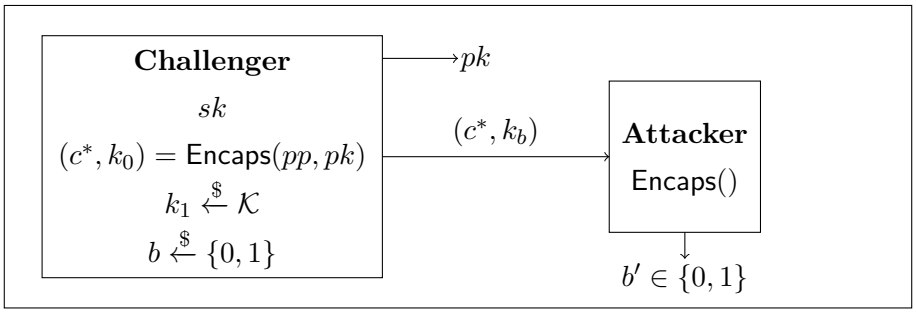

Figure 2.7: Simplified illustration of IND-CPA game for KEM.

A more compact interpretation is given below. A KEM satisfies IND-CPA security if the outputs of the following "real" and "ideal" games are computationally indistinguishable:

Real Game	**Ideal Game**
$(pk, sk) \leftarrow \mathsf{Gen}(pp)$	$(pk, sk) \leftarrow \mathsf{Gen}(pp)$
$(c^*, k_0) \leftarrow \mathsf{Encaps}(pp, pk)$	$(c^*, k_0) \leftarrow \mathsf{Encaps}(pp, pk)$
	$k_1 \xleftarrow{\$} \mathcal{K}$
$\mathrm{Output}(pp, pk, c^*, k_0)$	$\mathrm{Output}(pp, pk, c^*, k_1)$

The advantage of an adversary to distinguish between the two games is defined by:

$$\mathsf{Adv}^{\text{IND-CPA}}_{\text{KEM}} = \left| \mathbb{P}\left\{b' = b\right\} - \frac{1}{2} \right|.$$

IND-CCA security for PKE

In this setting, we consider a classical or quantum algorithm $\mathcal{A}^{\mathsf{Dec}}$ in possession of the adversary that can be used many times, and is capable not only of encrypting messages, but also of decrypting arbitrary ciphertexts at the adversary's demand. Similarly to the IND-CPA definition, a game between the adversary and the challenger is set up to examine whether the adversary is now able to predict with probability better than $1/2$ which message was encrypted given the more powerful algorithm $\mathcal{A}^{\mathsf{Dec}}$:

- The challenger generates the pair (pk, sk) using $\mathsf{Gen}()$.
- The adversary assigns two messages m_0 and m_1 to the challenger.
- The challenger selects $b \xleftarrow{\$} \{0,1\}$ and sends the challenge $c^* = \mathsf{Enc}(pp, pk, m_b)$ back to the adversary.

- The adversary is allowed to call the decryption oracle $\mathcal{A}^{\mathsf{Dec}}$, but cannot request to decrypt c^*, because otherwise $\mathcal{A}^{\mathsf{Dec}}$ outputs a failure symbol $\perp$. He then answers with a guess b' for the value of b.
- The game succeeds if $b' = b$, and it fails otherwise.

Figure 2.8 illustrates the game above.

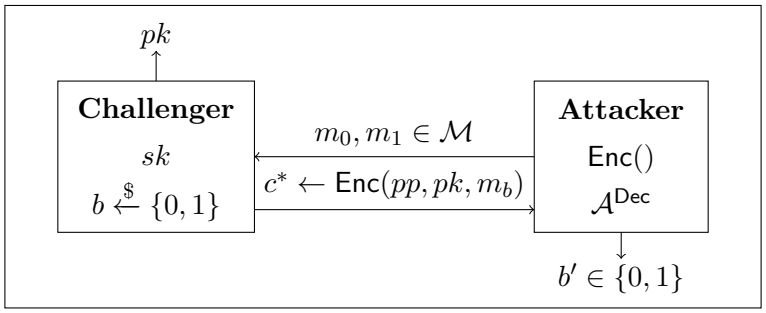

Figure 2.8: Simplified illustration of the IND-CCA game for PKE.

The encryption algorithm is said to be IND-CCA secure if the adversary has a negligible advantage in winning the above game. The advantage can be similarly defined as

$$\mathsf{Adv}_{\mathrm{PKE}}^{\text{IND-CCA}}\left(\mathcal{A}^{\mathsf{Dec}}\right) = \left|\mathbb{P}\left\{b' = b \;\middle|\; \mathcal{A}^{\mathsf{Dec}}\right\} - \frac{1}{2}\right|.$$

IND-CCA security for KEM

In this setup, the adversary is endowed with a decapsulation algorithm $\mathcal{A}^{\mathsf{Decaps}}$ that is able to recover the private key given a specific ciphertext. Once again, we are building a game between the adversary and the challenger to explore the attacker's potential to distinguish the true private key from a uniformly random key. Note that $\mathcal{A}^{\mathsf{Decaps}}$ outputs the failure symbol $\perp$ once it receives the challenge ciphertext. The game works as follows:

- The challenger produces a secret key sk and a public one pk using the Gen function.
- The adversary knows that the challenger will generate (k_0, c^*) from $\mathsf{Encaps}\,(pk)$ and k_1 uniformly at random from the key space $\mathcal{K}$.
- The challenger chooses a uniform b in $\{0,1\}$ and generates $(c^*, k_0) = \mathsf{Encaps}(pp, pk)$. If $b = 0$, he sends (c^*, k_0). If $b = 1$, he sends (c^*, k_1), where k_1 is uniformly random.
- Using $\mathcal{A}^{\mathsf{Decaps}}$ as additional weapon, the adversary tries to guess if the key k_b corresponds to k_0 (the output of Encaps) or to k_1 (uniform key) and outputs a guessing bit b'.
- If $b' = b$, the adversary passes the challenge.

Figure 2.9 illustrates the game above.

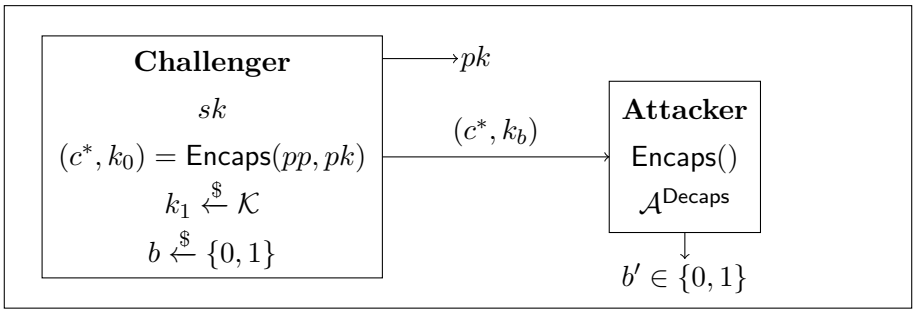

Figure 2.9: Simplified illustration of IND-CCA game for KEM.

The KEM is defined to be IND-CCA secure if the adversary has a negligible advantage in gaining the above game. We give the formal definition of the advantage as:

$$\mathsf{Adv}^{\text{IND-CCA}}_{\text{KEM}}\left(\mathcal{A}^{\mathsf{Decaps}}\right) = \left|\mathbb{P}\left\{b' = b \;\middle|\; \mathcal{A}^{\mathsf{Decaps}}\right\} - \frac{1}{2}\right|.$$

2.3.4 Fujisaki-Okamoto transform

The Fujisaki-Okamoto (FO) transform was first introduced in [FO99] and aims to transform an IND-CPA secure PKE into an IND-CCA KEM in the classical random oracle model (assuming the distribution of ciphertexts for each plaintext is sufficiently close to uniform). This transformation uses some hash functions which are modeled as random oracles in the security proof. Moreover, the encryption scheme is required to be deterministic and bijective, but such restrictions have been removed in [FO13]. Targhi and Unruh (TU) [TU16] deliver a variant of the Fujisaki-Okamoto transform against a quantum adversary in the quantum random oracle model under similar assumptions. However, these two transformations assume that the public-key encryption scheme is perfectly correct, i.e., not having a decryption error, which is not the case for lattice-based schemes.

Hofheinz, Hövelmanns and Kiltz [HHK17] proposed a new version of the FO transform, called $\text{FO}^{\not\perp}$, which allows decryption errors on lattice-based schemes and can always produce an IND-CCA-secure KEM from an IND-CPA PKE scheme. Furthermore, the $\text{FO}^{\not\perp}$ transform achieves a tighter security reduction compared to the original FO. In this thesis, we present a modification of $\text{FO}^{\not\perp}$ denoted as $\text{FO}^{\not\perp'}$ which was used in several lattice cryptosystems such as [AAB$^+$, A$^+$20, N$^+$20] and is summarized in the formal definition given below.

Definition 2.21 ($\text{FO}^{\not\perp'}$ transform)**.** Let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme with message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$, where the randomness space of Enc is $\mathcal{R}$. Three hash functions $G_1 : \{0,1\}^* \to \{0,1\}^{\ell_{\mathbf{pkh}}}$, $G_2 : \{0,1\}^* \to \mathcal{R} \times \{0,1\}^{\ell_{\mathbf{mh}}}$ and $F : \{0,1\}^* \to \{0,1\}^{\ell_{\mathbf{k}}}$ are set with $\ell_{\mathbf{pkh}}, \ell_{\mathbf{mh}}$ and $\ell_{\mathbf{k}}$ fixed parameters, as well as an additional parameter $\ell_{\mathbf{s}}$. The key encapsulation mechanism $\mathsf{KEM}^{\not\perp'} = \text{FO}^{\not\perp'}(\mathsf{PKE}, G_1, G_2, F)$ is defined as follows:

$\mathsf{KEM}^{\not\perp'}.\mathsf{Gen}()$

1. $(pk, sk) \xleftarrow{\$} \mathsf{PKE.Gen}()$
2. $\mathbf{s} \xleftarrow{\$} \{0,1\}^{\ell_{\mathbf{s}}}$
3. $\mathbf{pkh} \leftarrow G_1(pk)$
4. $sk' \leftarrow (sk, \mathbf{s}, pk, \mathbf{pkh})$
5. **return** (pk, sk')

$\mathsf{KEM}^{\not\perp'}.\mathsf{Encaps}(pk)$

1. $\mu \xleftarrow{\$} \mathcal{M}$
2. $(\mathbf{r}, \mathbf{mh}) \leftarrow G_2(\mathbf{pkh}||\mu)$
3. $c \leftarrow \mathsf{PKE.Enc}(\mathbf{r}, pk, \mu)$
4. $\mathbf{k} \leftarrow F(c||\mathbf{mh})$
5. **return** $(c, \mathbf{k})$

$\mathsf{KEM}^{\not\perp'}.\mathsf{Decaps}(c, sk')$

1. $\mu' \leftarrow \mathsf{PKE.Dec}(c, sk)$
2. $(\mathbf{r}', \mathbf{mh}') \leftarrow G_2(\mathbf{pkh}||\mu')$
3. $\mathbf{k}_0 \leftarrow F(c||\mathbf{mh}')$
4. $\mathbf{k}_1 \leftarrow F(c||\mathbf{s})$
5. If $c = \mathsf{PKE.Enc}(\mathbf{r}', pk, \mu')$: $\mathbf{k}' \leftarrow \mathbf{k}_0$. Else: $\mathbf{k}' \leftarrow \mathbf{k}_1$
6. **return** $\mathbf{k}'$

The outputs of the hash functions are marked in bold. Note that $\mathbf{pkh}, \mathbf{mh}$ are "hashed" versions of the public key and intermediate shared secret. Moreover,

- The $\mathrm{FO}^{\not\perp'}$ transform outputs a shared private key $\mathbf{k}$ of length $\ell_{\mathbf{k}}$;
- A hashed public key $\mathbf{pkh}$ of length $\ell_{\mathbf{pkh}}$ is generated from pk;
- A binary vector $\mathbf{s}$ of length $\ell_{\mathbf{s}}$ (not smaller than $\ell_{\mathbf{k}}$) is used for pseudo-random shared secret generation in the event of decapsulation failure in the $\mathrm{FO}^{\not\perp'}$ transform ;
- An intermediate shared secret $\mathbf{mh}$ of length $\ell_{\mathbf{mh}}$ is generated from $\mathbf{pkh}$ and a message μ.

$\mathsf{KEM}^{\not\perp'}$ differs from [HHK17] in the following points:

- A single hash function (with longer output) is used to compute $\mathbf{r}$ and $\mathbf{mh}$, whereas $\mathrm{FO}^{\not\perp}$ uses two separate functions, but these are equivalent when the hash functions are modeled as independent random oracles and have appropriate output lengths.
- The public key pk is used as input for G_1 in order to deliver $\mathbf{r}$ and $\mathbf{mh}$ from $\mathbf{pkh} = G_1(pk)$, whereas $\mathrm{FO}^{\not\perp}$ does not; this change preserves the relevant theorems (with trivial changes to the proofs), and has the potential to provide stronger multi-target security.
- The computation of the shared private $\mathbf{k}$ takes the encapsulation c as input to F.

Remark 2.3. It is important to mention that during decapsulation in $\mathsf{KEM}^{\not\perp'}.\mathsf{Decaps}()$, it is crucial to execute step 5 in constant time, because otherwise, a timing attack enables key recovery. This was observed by Guo, Johansson, and Nilsson in [GJN20].

2.4 Error distributions

In this work, we will deal with error terms that will be included in various definitions and applications. In the following, we present some possible choices for the error distribution that have been introduced in lattice-based cryptography. Note that for practical applications, we will need distributions that can be sampled efficiently.

2.4.1 Discrete and rounded Gaussian

We start by defining the *Gaussian function*.

Definition 2.22 (Gaussian function)**.** For any $s > 0$, the Gaussian function on $\mathbb{R}^n$ centered at $\mathbf{c}$ with parameter s is defined as follows:

$$\forall \mathbf{x} \in \mathbb{R}^n, \quad \rho_{s,\mathbf{c}}(\mathbf{x}) = \exp\left(\frac{-\pi\|\mathbf{x}-\mathbf{c}\|^2}{s^2}\right)$$

When $\mathbf{c} = 0$, the distribution is simply denoted by $\rho_s(\mathbf{x})$. Note that $\int_{\mathbf{x}\in\mathbb{R}^n} \rho_{s,\mathbf{c}}(\mathbf{x})d\mathbf{x} = s^n$. This allows us to derive the *continuous Gaussian distribution* from it.

Definition 2.23 (Continuous Gaussian distribution)**.** The continuous Gaussian distribution centered at $\mathbf{c}$ with standard deviation $\sigma = s/\sqrt{2\pi}$ is given by:

$$\forall \mathbf{x} \in \mathbb{R}^n, \quad D_{s,\mathbf{c}}(\mathbf{x}) = \frac{1}{s^n} \cdot \rho_{s,\mathbf{c}}(\mathbf{x})$$

Discrete Gaussian. Definition (2.23) can be extended to any countable set $A \subseteq \mathbb{R}^n$ in the usual way; e.g.

$$D_{s,\mathbf{c}}(A) = \frac{1}{s^n} \cdot \rho_{s,\mathbf{c}}(A) = \frac{1}{s^n} \cdot \sum_{\mathbf{x}\in A} \rho_{s,\mathbf{c}}(\mathbf{x})$$

Since a lattice is a countable set, we can thus introduce the concept of *discrete Gaussian distribution* defined on each point of the lattice as follows:

Definition 2.24 (Discrete Gaussian distribution)**.** Given a lattice Λ, we define the discrete Gaussian distribution over the lattice to have the following probability density function:

$$\forall \boldsymbol{\lambda} \in \Lambda, \quad D_{\Lambda,s,\mathbf{c}}(\boldsymbol{\lambda}) = \frac{D_{s,\mathbf{c}}(\boldsymbol{\lambda})}{D_{s,\mathbf{c}}(\Lambda)} = \frac{\rho_{s,\mathbf{c}}(\boldsymbol{\lambda})}{\rho_{s,\mathbf{c}}(\boldsymbol{\Lambda})}$$

Rounded Gaussian. Informally, the rounded Gaussian distribution is obtained by rounding the samples of a continuous Gaussian distribution to the nearest integer. We will only consider centered distributions.

Definition 2.25 (Rounded Gaussian distribution)**.** The rounded Gaussian distribution Ψ_s with parameter (or width) $s > 0$ is obtained from D_s as follows:

$$\forall\, \mathbf{a} \in \mathbb{Z}^n, \Psi_s(\mathbf{a}) = \int_{\{\mathbf{x}\,:\,\lfloor\mathbf{x}\rceil=\mathbf{a}\}} D_s(\mathbf{x})d\mathbf{x}^n$$

This distribution is also centered at zero and has standard deviation close to $\sqrt{\sigma^2 + \frac{1}{12}}$.

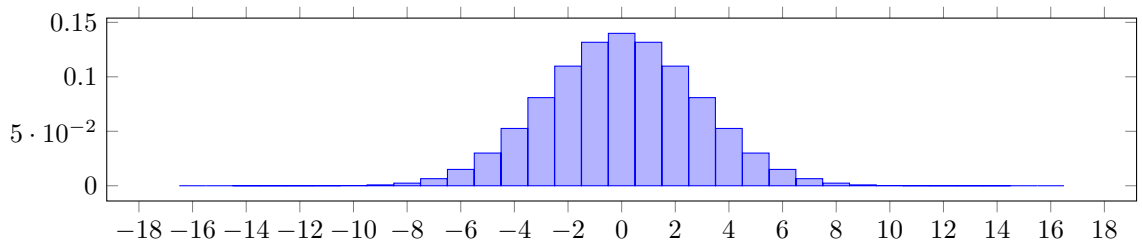

Figure 2.10: The centered binomial distribution ψ_k for $k = 16$.

2.4.2 Centered binomial distribution

Since it is challenging to implement a discrete Gaussian sampler to be efficient and protected against timing attacks, one can replace the former distribution by a discrete distribution that is easier to sample. In particular, [ADPS16b] proposed to use the *centered binomial distribution* ψ_k of standard deviation $\sqrt{k/2}$, which is the distribution of the random variable B_k is defined as

$$B_k = \sum_{i=1}^{k} (b_i - b'_i)$$

where b_i, b'_i are independent and uniformly distributed in $\{0, 1\}$, for $i = 1, \ldots, k$. Note that B_k takes values in $\{-k, \ldots, k\}$ and it is a centered and symmetric random variable. As an example, the distribution ψ_{16} is illustrated in Figure 2.10. The following result allows to calculate the distribution ψ_k:

Proposition 2.3. Let B_k denote the random variable that corresponds to the centered binomial distribution ψ_k. Then,

$$\forall\, t \in \{-k, \ldots, k\},\ \mathbb{P}\{B_k = t\} = \frac{1}{2^{2k}} \cdot \sum_{j=\max(0,t)}^{\min(k+t,k)} \binom{k}{j} \binom{k}{j-t}$$

Proof. Note that $X = \sum_{i=1}^{k} b_i$ has a binomial distribution $\mathcal{B}\left(k, \frac{1}{2}\right)$, as well as $Y = \sum_{i=1}^{k} b'_i$. Combining the cases for positive and negative t into one case we obtain the following:

$$\begin{aligned}
\mathbb{P}\{B_k = t\} &= \sum_{j=\max(0,t)}^{\min(k+t,k)} \mathbb{P}\{X = j\} \cdot \mathbb{P}\{Y = j - t\} \\
&= \sum_{j=\max(0,t)}^{\min(k+t,k)} \binom{k}{j} \left(\frac{1}{2}\right)^k \cdot \binom{k}{j-t} \left(\frac{1}{2}\right)^k \\
&= \frac{1}{2^{2k}} \cdot \sum_{j=\max(0,t)}^{\min(k+t,k)} \binom{k}{j} \binom{k}{j-t}
\end{aligned}$$

□

2.4.3 Subgaussian distribution

Definition 2.26 (Subgaussian distribution in $\mathbb{R}$)**.** For any $\delta \geq 0$, a distribution X over $\mathbb{R}$ is called *subgaussian with parameter* $s > 0$, if $\forall t \in \mathbb{R}$,

$$\mathbb{E}\left[e^{2\pi tX}\right] \leq e^{\pi t^2 s^2}.$$

Remark 2.4. If X is subgaussian with parameter s, then for any $c \in \mathbb{R}$, cX is subgaussian with parameter $|c|s$. Moreover, using the inequality $\cosh(x) \leq \exp(x^2/2)$, it can be shown that any B-bounded centered random variable X (i.e. $\mathbb{E}[X] = 0$ and $|X| \leq B$) is subgaussian with parameter $B\sqrt{2\pi}$ [LPR].

We extend the definition to $\mathbb{R}^n$ as follows:

Definition 2.27 (Subgaussian distribution in $\mathbb{R}^n$)**.** We say that a random vector X^n in $\mathbb{R}^n$ is subgaussian with parameter $s > 0$ if for any unit vector $\mathbf{u} \in \mathbb{R}^n$, the inner product $\langle X^n, \mathbf{u}\rangle$ is subgaussian with parameter $s > 0$, i.e., $\forall t \in \mathbb{R},$ and $\forall \mathbf{u} \in \mathbb{R}^n : \|\mathbf{u}\| = 1$,

$$\mathbb{E}\left[e^{2\pi t\langle X^n, \mathbf{u}\rangle}\right] \leq e^{\pi t^2 s^2}.$$

Proposition 2.4. Let $X_1, \ldots, X_k$ be independent subgaussian random variables over $\mathbb{R}$ with parameters s_i. Then $\sum_{i=1}^k X_i$ is subgaussian with parameter $s = \left(\sum_{i=1}^k s_i^2\right)^{\frac{1}{2}}$.

Proof. It suffices to prove the proposition for $k = 2$.

$$\mathbb{E}\left[e^{2\pi t(X_1+X_2)}\right] = \mathbb{E}\left[e^{2\pi tX_1} \cdot e^{2\pi tX_2}\right] = \mathbb{E}\left[e^{2\pi tX_1}\right] \cdot \mathbb{E}\left[e^{2\pi tX_2}\right] \leq e^{\pi t^2(s_1^2+s_2^2)} \qquad \square$$

Proposition 2.5. Let X^n be a centered random variable in $\mathbb{R}^n$ such that each component is s-subgaussian. Then X^n is subgaussian with parameter $s\sqrt{n}$.

Proof. We need to show that for every unit vector $\mathbf{u}$, $\langle X^n, \mathbf{u}\rangle$ is subgaussian with parameter $s\sqrt{n}$. In fact,

$$\begin{aligned}
\mathbb{E}\left[e^{2\pi t\langle X^n, \mathbf{u}\rangle}\right] &= \mathbb{E}\left[e^{2\pi t(X_1u_1+\cdots+X_nu_n)}\right] \\
&= \mathbb{E}\left[e^{2\pi tX_1u_1} \cdot \ldots \cdot e^{2\pi tX_nu_n}\right] \\
&\leq \left(\mathbb{E}\left[e^{2\pi tnX_1u_1}\right] \cdot \ldots \cdot \mathbb{E}\left[e^{2\pi tnX_nu_n}\right]\right)^{1/n} \quad \text{(using Hölder's inequality)} \\
&\leq \left(e^{\pi u_1^2 t^2 n^2 s^2} \cdot \ldots \cdot e^{\pi u_n^2 t^2 n^2 s^2}\right)^{1/n} \\
&= e^{\pi t^2 s^2 n s^2 (u_1^2+\cdots+u_n^2)} \\
&= e^{\pi t^2 s^2 n}.
\end{aligned}$$

This shows that X^n is subgaussian with parameter $\sqrt{s^2 n} = s\sqrt{n}$. $\square$

The following theorem inspired from [HKZ12] establishes a tail inequality for subgaussian random variables.

Theorem 2.3. Let X^n be a subgaussian vector in $\mathbb{R}^n$ with parameter s. Then $\forall \epsilon > 0$:

$$\mathbb{P}\left\{\|X^n\| > \frac{s}{\sqrt{2\pi}}\sqrt{n} \cdot \sqrt{1 + 2\epsilon + 2\epsilon^2}\right\} \leq e^{-n\epsilon^2}$$

The proof of this theorem can be found in Appendix A.7.

Theorem 2.4. The centered binomial distribution ψ_k is subgaussian with parameter $\sqrt{k\pi}$.

Proof. Remark that the distribution ψ_1 is subgaussian with parameter $s = \sqrt{\pi}$. In fact, B_1 takes values in $\{-1, 0, 1\}$ with probabilities $\frac{1}{4}, \frac{1}{2}, \frac{1}{4}$ respectively. So

$$\begin{aligned}
\mathbb{E}\left[e^{2\pi t B_1}\right] &= \sum_{k\in\{-1,0,1\}} e^{2\pi t k} \cdot \mathbb{P}\{B_1 = k\} \\
&= \frac{1}{4}e^{-2\pi t} + \frac{1}{2}e^{-2\pi\cdot 0} + \frac{1}{4}e^{2\pi t} \\
&= \frac{1}{2} + \frac{1}{2}\left(\frac{e^{2\pi t} + e^{-2\pi t}}{2}\right) \\
&= \frac{1 + \cosh(2\pi t)}{2} \leq e^{\frac{4\pi^2 t^2}{4}} = e^{\pi t^2 \cdot (\pi)}
\end{aligned}$$

Hence using Proposition 2.4 we obtain the desired result. □

2.4.4 Rényi divergence

Definition 2.28 (Rényi Divergence)**.** Given two discrete probability distributions P and Q such that $\mathrm{Supp}(P)$ is finite and $\alpha > 1$, we define the Rényi divergence of order α between P and Q to be

$$D_\alpha(P||Q) = \frac{1}{1-\alpha}\ln\left(\sum_{x\in\mathrm{Supp}(P)} P(x)\left(\frac{P(x)}{Q(x)}\right)^{\alpha-1}\right)^{3}.$$

The Rényi divergence evaluates the closeness between two probability distributions. It can be seen as an alternative to the statistical distance and it was shown in [BLRL$^+$18] that using the Rényi divergence rather than the statistical distance yields tighter security reductions.

Although many security proofs in lattice cryptography assume that the error has a rounded Gaussian distribution, many lattice cryptosystems use a different error distribution which is more suitable for practical implementation yet is close to a rounded Gaussian in Rényi divergence. This shift does not significantly decrease the security compared to a rounded Gaussian distribution. For instance, it was shown in [ADPS16b, Theorem 4.1] that $D_9\left(\psi_{16}||\Psi_{\sqrt{8}\sqrt{2\pi}}\right) \approx 0.00063$. More generally, Figure 2.11 shows the Rényi divergence between the centered binomial distribution ψ_k and the rounded Gaussian distribution $\Psi_{\sqrt{k\pi}}$ [SLS$^+$20, Section 5.3].

[3] This definition differs from that of [LSS14] in which the logarithm of the sum is used.

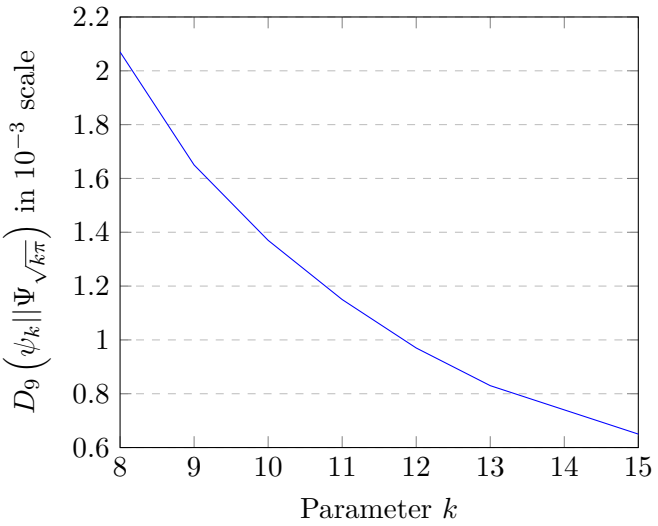

Figure 2.11: Rényi divergence of the centered binomial distribution ψ_k and the rounded Gaussian distribution $\Psi_{\sqrt{k\pi}}$ according to k ($\alpha = 9$).

We give in Appendix A.2 an example of calculating the Rényi divergence between the rounded distribution $\Psi_{\sqrt{16\pi}}$ and the centered binomial ψ_{16}.

Chapter 3

Error correction for FrodoKEM

3.1 Introduction

In this chapter, we will present an error correction mechanism for FrodoKEM, which is an IND-CCA secure key encapsulation mechanism that was selected to pass the third round of the NIST Post-Quantum Cryptography project and is now one of the alternative candidates among key establishment algorithms. The security of FrodoKEM is based on the hardness of what is called the *Learning With Errors* problem (LWE) introduced by Regev [Reg09] and discussed in the next Section.

3.2 Learning With Errors (LWE)

Definition 3.1 (Search-LWE problem: $\mathsf{LWE}_{q,\chi}$)**.** For $n \geq 1$, let $q = q(n) \leq \text{poly}(n)$ be an integer. Choose a secret $\mathbf{s} \in \mathbb{Z}_q^n$. Choose a polynomial number of samples $\mathbf{a}_1, \ldots, \mathbf{a}_m$ from $\mathbb{Z}_q^n$ independently and uniformly random. Fix an "error" probability distribution $\chi : \mathbb{Z}_q \mapsto \mathbb{R}^+$. For $i = 1, \ldots, m$, set $b_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i \bmod q$, where each $e_i \in \mathbb{Z}_q$ is chosen independently according to χ. The goal is to recover the secret $\mathbf{s}$ given these m equations.

It is important to mention that the instances of the LWE problem can be given in matrix form $\mathbf{b} = \mathbf{As} + \mathbf{e}$ where $\mathbf{A}$ is an $m \times n$ matrix. More precisely,

$$\begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} = \begin{bmatrix} \ldots \mathbf{a}_1 \ldots \\ \vdots \\ \ldots \mathbf{a}_m \ldots \end{bmatrix} \cdot \begin{bmatrix} s_1 \\ \vdots \\ s_n \end{bmatrix} + \begin{bmatrix} e_1 \\ \vdots \\ e_m \end{bmatrix} \bmod q.$$

One of the possible choices for the error distribution χ is the rounded Gaussian distribution with parameter αq for $\alpha \in (0, 1)$, and reduced modulo q. For a suitable choice of q and α, a solution to $\mathsf{LWE}_{q,\chi}$ implies a quantum solution to worst-case lattice problems as the theorem below indicates.

Theorem 3.1 ([Reg09])**.** Let n, q be integers and $\alpha \in (0,1)$ be such that $\alpha q > 2\sqrt{n}$. Let Ψ_s be a rounded Gaussian with parameter $s = \alpha q$. Solving LWE_{q,Ψ_s} efficiently implies an efficient quantum algorithm that approximates the decision version of the shortest vector problem $\mathsf{GapSVP}_{\tilde{O}(n/\alpha)}$ and the shortest independent vectors problem $\mathsf{SIVP}_{\tilde{O}(n/\alpha)}$ in the worst case.

The value of α is usually taken to be $1/\mathrm{poly}(n)$ due to the approximations factors of the underlying lattice problem. The modulus q cannot be taken to be greater than polynomial since it may affect the efficiency for the underlying cryptographic applications as it increases the size of the input $\mathbf{As} + \mathbf{e}$. The ratio between the modulus q and the noise width, called *modulus-to-noise ratio*, is simply $1/\alpha$, and the security is enhanced once this ratio becomes smaller [1].

The second form of LWE is the decision version which provides a simpler security analysis related to indistinguishability proofs.

Definition 3.2 (Decision-LWE)**.** Consider the same settings as in Definition 3.1. The goal is to distinguish with some non-negligible advantage between polynomially many pairs of the form:

$$(\mathbf{a}_i, \mathbf{b}_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i \bmod q) \text{ and } (\mathbf{a}_i, \mathbf{b}_i^*)\,,$$

with $\mathbf{b}_i^*$ uniformly random in from $\mathbb{Z}_q^m$.

Note that search-LWE and decision-LWE are equivalent problems [Reg09, Ste14, Mah15].

Definition 3.3 (LWE advantage)**.** For an attacker $\mathcal{A}$, we define the advantage $\mathsf{Adv}^{\mathsf{LWE}}_{n,m,q,\chi}$ as

$$\left| \mathbb{P}\left\{ b' = 1 \;\middle|\; \begin{array}{c} \mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m\times n} \\ (\mathbf{s},\mathbf{e}) \leftarrow \chi^n \times \chi^m \\ \mathbf{b} = \mathbf{As}+\mathbf{e} \\ b' \leftarrow \mathcal{A}(\mathbf{A},\mathbf{b}) \end{array} \right\} - \mathbb{P}\left\{ b' = 1 \;\middle|\; \begin{array}{c} \mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m\times n} \\ \mathbf{b} \xleftarrow{\$} \mathbb{Z}_q^{m} \\ b' \leftarrow \mathcal{A}(\mathbf{A},\mathbf{b}) \end{array} \right\} \right|$$

where $b' = 1$ is the output of the distinguisher $\mathcal{A}$.

Solving LWE. Most of the literature considers the hardness of the LWE problem asymptotically. Herold *et al.* show that when taking a polynomial number of samples, all known algorithms that solve LWE in dimension n run in time $2^{O(n)}$ [HKM18]. [KF15] considers the case when the LWE error term is binary in $\{0,1\}^m$, and shows that in this case, the LWE problem can be solved in subexponential time, indicating that LWE with binary error is weaker than general LWE. However, the asymptotic regime can mask some logarithmic and constant factors which are relevant when choosing parameters in finite dimension. When constructing an LWE-based cryptosystem, the choice of parameters should take into account the complexity of the fastest known algorithm that resolves LWE, and ensure that the attack requires a prohibitively large number of operations in order to provide a well-defined security bound.

[1] This is due to the fact that the the approximation factor $\tilde{O}(n/\alpha)$ of the underlying lattice problems depends on the modulus-to-noise ratio, which ensures higher security once it becomes larger.

There are many algorithms to consider in general such as BKW attacks [KF15, HKM18, BGJ+20] and the Arora-Ge algorithm (linearization attack) [AG11] that proposed a new algebraic technique for solving LWE. The latter algorithm has a total complexity (time and space) of $2^{\tilde{O}(\sigma^2)}$. This implies that taking a small modulus q in LWE-based cryptosystems would trigger an LWE attack. In fact, choosing a small modulus requires taking small values of σ in order to guarantee a correct decryption. These attacks are generally exhaustive search attacks; there exist other types of attacks via lattice reduction. The exhaustive search strategies solve LWE by directly finding a vector $\mathbf{s}$ for which $\mathbf{As}$ is close to $\mathbf{b}$ in terms of Euclidean distance, and require an exponential number of LWE samples. Lattice reduction attacks require a polynomial number of samples and can be divided in two categories: *Primal attack* and *Dual attack*. These two attacks use efficient lattice reduction algorithms such as the BKZ algorithm [SE94b, CN11] with block-size b, which requires up to polynomially many calls to an SVP oracle in dimension b, but some heuristics allow to decrease the number of calls to be essentially linear [Che13]. A brief description of these attacks is given below.

- The *primal attack* [AAB+] solves the search version of the LWE problem by constructing a unique-SVP, instance from the LWE problem and solving it using BKZ. The width of the block dimension b for BKZ is examined in order to find the unique solution. Given the matrix LWE instance $(\mathbf{A}, \mathbf{b} = \mathbf{As} + \mathbf{e})$, one builds the lattice

$$\Lambda = \left\{ \mathbf{x} \in \mathbb{Z}^{m+n+1} \ : \ (\mathbf{A}|\mathbf{I}_m| - \mathbf{b})\,\mathbf{x} = \mathbf{0} \bmod q \right\}.$$

  This lattice is of rank $n+1$ and dimension $d = m+n+1$. In fact, if we set $\mathbf{A}' = (\mathbf{A}|\mathbf{I}_m| - \mathbf{b})$, then by the rank theorem we have:

$$\operatorname{rank}(\mathbf{A}') + \dim(\operatorname{Ker}(\mathbf{A}')) = m + n + 1.$$

  That's why $\dim(\operatorname{Ker}(\mathbf{A}')) = n + 1$, which means that there are q^{n+1} integer solutions in the cube of side q. Hence, the volume of Λ is q^m. Moreover, Λ has a unique-SVP solution $\mathbf{v} = (\mathbf{s}, \mathbf{e}, 1)$ of norm $\approx \sigma\sqrt{n+m}$, where σ is the standard deviation of the noise. The integer m represents the number of used LWE samples that is usually bounded by $\operatorname{poly}(n)$ and can be optimized numerically.

  Using the typical models of BKZ (geometric series assumption, Gaussian heuristic [Che13, APS15]), one can derive the success condition for the primal attack, which is is satisfied if and only if

$$\sigma\sqrt{b} \leq \delta^{2b-d-1} \cdot q^{m/d}, \ \text{ where } \delta = \left((\pi b)^{1/b} \cdot b/(2\pi e)\right)^{1/(2b-2)}.$$

  The BKZ algorithm can now search for the unique solution using the resulting value b of the block dimension.

- The *dual attack* [AAB+] consists in finding a short vector inside the dual lattice

$$L^* = \left\{(\mathbf{v}, \mathbf{w}) \in \mathbb{Z}^m \times \mathbb{Z}^n \,:\, \mathbf{v}^T\mathbf{A} = \mathbf{w}^T \bmod q\right\}$$

in order to solve the decision version of the problem. This lattice is of dimension $d = n+m$, volume q^n and generated by the rows of the basis matrix

$$\mathbf{B} = \begin{bmatrix} q\mathbf{I}_n & 0 \\ \mathbf{A} \bmod q & \mathbf{I}_m \end{bmatrix}.$$

As above, the BKZ algorithm with block size b will output such a short vector of length $\delta^{d-1}q^{n/d}$. The dual attack then uses this vector as a distinguisher for LWE. In fact, given a short pair $(\mathbf{v}, \mathbf{w})$ one can compute

$$\mathbf{z} = \mathbf{v}^T\mathbf{b} \bmod q = \mathbf{v}^T(\mathbf{A}\mathbf{s} + \mathbf{e}) \bmod q = \mathbf{w}^T\mathbf{s} + \mathbf{v}^T\mathbf{e} \bmod q.$$

If $(\mathbf{A}, \mathbf{b})$ is an LWE sample, this is distributed as a discrete Gaussian of standard deviation $\sigma||(\mathbf{v}, \mathbf{w})||$ and often returns small samples. If $\mathbf{b}$ is uniform modulo q, then $\mathbf{z}$ is also uniform modulo q.

As mentioned above, the BKZ algorithm calls SVP several number of time. This number of calls is hard to estimate but it is polynomial, so one can give a conservative estimate of the cost of the primal and dual attacks by considering a single call to SVP, which is called the *core* SVP *hardness* [ADPS16b].

Enumeration and sieve algorithms discussed in Subsection 2.2.2 are used to implement the SVP oracle used by BKZ. The main idea of lattice enumeration is to systematically enumerate all lattice points in a bounded region of space. On the other hand, the sieve algorithms perform some kind of randomized enumeration of a smaller set. Asymptotically, enumeration is super-exponential, while sieve algorithms are known to run in exponential time. Consequently, we choose the latter to predict the cost of solving SVP and will argue that for the targeted dimension, enumerations are expected to be greatly slower than sieving.

Classical lattice sieve algorithms [MV10, NV08] have been improved in terms of complexity from $2^{0.415b}$ down to $2^{0.292b}$ [BDGL16, Laa15b]. With a quantum computer, it would be possible to reduce the complexity of lattice sieve algorithms to $2^{0.265b}$ [Laa15a, LMVDP15]. Moreover, all of them depend upon classically building lists of size $2^{0.2075b}$, which makes it uncertain whether one could build a quantum SVP algorithm that performs in time better than $2^{0.2075b}$. This running time gives a *plausible* estimate of the cost of the best attack.

Reduction to BDDwDGS. The choice of the error distribution for LWE will be reduced to the choice of its "width". As we saw in Theorem 3.1, when the standard deviation σ of the discrete

Gaussian is greater than $\sqrt{2/\pi}\sqrt{n}$, then an appropriate quantum reduction from GapSVP_γ and SIVP_γ to LWE is established in the worst-case, with $\gamma = \tilde{O}(n/\alpha)$. The value of σ can be improved to any $\sigma > \frac{1}{2\pi}\sqrt{n}$ as [N+20] mentioned out.

The security of FrodoKEM is based on a reduction from a variant of the worst-case bounded-distance decoding (BDD) problem to LWE that supports a smaller error width (i.e. $\sigma < \frac{1}{2\pi}\sqrt{n}$). This variant called *bounded-distance decoding with discrete Gaussian samples* (BDDwDGS) requires a discrete Gaussian sampler over the dual lattice. A formal definition is given below.

Definition 3.4 (BDD with discrete Gaussian samples ($\mathsf{BDDwDGS}_{\Lambda,d,r}$) [N+20])**.** Let Λ be an n-dimensional lattice with $d < \lambda_1(\Lambda)/2$ and let $r > 0$. The BDD with discrete Gaussian samples is defined under two hypotheses:

- A target point $\mathbf{t} \in \mathbb{R}^n$ such that $\mathrm{dist}(\mathbf{t}, \Lambda) \leq d$
- Having access to an oracle that samples from $D_{\Lambda^*,s}$ for any adaptively queried $s \geq r$.

The goal is to output the closest lattice point to $\mathbf{t}$.

Note that in this case the parameter s for $D_{\Lambda^*,s}$ has the ability to vary, while the existing BDDwDGS algorithms [AR05, LLM06, DRSD14] use discrete Gaussian samples that all have equal width parameter s. However, for certain constraints on r, the oracle in Definition 3.4 can be replaced by an oracle with fixed-width r that samples from $D_{\mathbf{w}+\Lambda^*,r}$ for any queried coset $\mathbf{w} + \Lambda^*$.

Theorem 5.12 in [N+20] relates the hardness of BDDwDGS to the hardness of the decision-LWE problem:

Theorem 3.2 (BDDwDGS hard $\Longrightarrow$ decision-LWE hard)**.** Let $\epsilon = \epsilon(n)$ be a negligible function and let $m = \mathrm{poly}(n)$ and $C = C(n) > 1$ be arbitrary. There is a probabilistic polynomial-time (classical) algorithm that, given access to an oracle that solves $\mathsf{DLWE}_{n,m,q,\alpha}$ with non-negligible advantage and input a number $\alpha \in (0, 1)$, an integer $q \geq 2$, a lattice $\Lambda \in \mathbb{R}^n$, and a parameter $r \geq Cq \cdot \eta_\epsilon(\Lambda^*)$, solves $\mathsf{BDDwDGS}_{\Lambda,d,r}$, using $N = m \cdot \mathrm{poly}(n)$ samples, where $d = \sqrt{1 - 1/C^2} \cdot \alpha q/r$.

We note here that the BDDwDGS problem hasn't been as extensively studied as SIVP, and the known approaches to solving BDDwDGS are limited when $d > \frac{1}{r}\sqrt{\ln N/(2\pi)}$ [AR05, LLM06, DRSD14]. In particular, if

$$\alpha q \gg \sqrt{\ln N/2\pi}, \tag{3.1}$$

then the BDDwDGS problem is plausibly hard in the worst case.

3.3 FrodoKEM

FrodoKEM [N+20] is a family of key encapsulations mechanisms that has been selected by the NIST project for Post Quantum Cryptography Standardization and chosen as an alternative

candidate for the third round. The security of the protocol is based on the hardness of the LWE problem, offering a relatively easy to implement algorithm. Moreover, the scheme is designed to ensure IND-CCA security at three different levels, corresponding to the brute-force security of AES-128, AES-192 and AES-256. The IND-CCA secure KEM is induced from an IND-CPA secure PKE scheme called FrodoPKE using the Fujisaki-Okamoto (FO) transform.

In terms of efficiency, the main operations in the FrodoKEM protocol require a matrix-vector product modulo a power-of-two integer q yielding a straightforward modular arithmetic implementation that is resistant to timing attacks.

3.3.1 Presentation of the algorithm

Our discussion in this section will mainly focus on FrodoPKE that will be used as a building block for FrodoKEM. FrodoPKE is designed to guarantee IND-CPA security at three levels: Frodo-640, Frodo-976 and Frodo-1344. The corresponding message space is $\mathcal{M} = \{0,1\}^{64B}$ for $B \in \{2,3,4\}$ which depends on the chosen level. Each level is parameterized by an integer dimension n such that $n \equiv 0 \bmod 8$, a standard deviation σ and a discrete error distribution χ_{Frodo} which is close to the rounded Gaussian $\Psi_{\sigma\sqrt{2\pi}}$ in Rényi divergence (see Table 3.2). The LWE modulus q is a power-of-two integer such that $q \leq 2^{16}$. A sketch of the FrodoPKE protocol is given in Table 3.1.

Parameters: q; $n \in \{640, 976, 1344\}$; $\bar{n} = 8$; $B \in \{2,3,4\}$ FrodoKEM's distribution χ_{Frodo}		
Alice (server)		**Bob (Client)**
$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times n}$		
$\mathbf{S}, \mathbf{E} \leftarrow \chi_{\text{Frodo}}^{n\times\bar{n}}$		$\mathbf{S}', \mathbf{E}' \leftarrow \chi_{\text{Frodo}}^{\bar{n}\times n}$,
$\mathbf{B} := \mathbf{AS} + \mathbf{E} \in \mathbb{Z}_q^{n\times\bar{n}}$	$\xrightarrow{(\mathbf{A},\mathbf{B})}$	$\mathbf{E}'' \leftarrow \chi_{\text{Frodo}}^{\bar{n}\times\bar{n}}$
		$\mathbf{U} := \mathbf{S}'\mathbf{A} + \mathbf{E}' \in \mathbb{Z}_q^{\bar{n}\times n}$
		$\mathbf{V} := \mathbf{S}'\mathbf{B} + \mathbf{E}'' \in \mathbb{Z}_q^{\bar{n}\times\bar{n}}$
		$\mathbf{m} \xleftarrow{\$} \{0,1\}^{64B}$
$\mathbf{V}' := \mathbf{C} - \mathbf{US} \in \mathbb{Z}_q^{\bar{n}\times\bar{n}}$	$\xleftarrow{(\mathbf{U},\mathbf{C})}$	$\mathbf{C} = \mathbf{V} + \text{FRODO.ENCODE}(\mathbf{m})$
$\mathbf{m}' = \text{FRODO.DECODE}(\mathbf{V}')$		

Table 3.1: Simplified description of FrodoPKE.

In this algorithm, Alice generates $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times n}$, samples $\mathbf{S}, \mathbf{E} \leftarrow \chi_{\text{Frodo}}^{n\times\bar{n}}$, then computes the LWE samples $\mathbf{B} = \mathbf{AS} + \mathbf{E}$ and outputs a public key $(\mathbf{A}, \mathbf{B})$. Bob chooses $\mathbf{S}', \mathbf{E}', \mathbf{E}'' \leftarrow \chi_{\text{Frodo}}^{\bar{n}\times n}$, then computes the LWE samples $\mathbf{U} = \mathbf{S}'\mathbf{A} + \mathbf{E}'$ and $\mathbf{V} = \mathbf{S}'\mathbf{B} + \mathbf{E}''$. A message $\mathbf{m}$ in $\{0,1\}^{64B}$ is generated unilaterally on Bob's side and encoded into $\mathbb{Z}_q^{\bar{n}\times\bar{n}}$ using the function

FRODO.ENCODE($\cdot$) presented in Algorithm 2. Bob sends the ciphertext $(\mathbf{U}, \mathbf{C})$ to Alice, where $\mathbf{C} = \mathbf{V} + \text{FRODO.ENCODE}(\mathbf{m})$. Alice recovers $\mathbf{m}'$ using the decoding function in Algorithm 3.

Algorithm 2 FRODO.ENCODE

Input: Bit string $\mathbf{k} \in \{0,1\}^\ell, \ell = B \cdot 64$

Output: Matrix $\mathbf{K} \in \mathbb{Z}_q^{8\times 8}$

1: **for** i=0 to 7 **do**
2: **for** j=0 to 7 **do**
3: $k \leftarrow \sum_{l=0}^{B-1} \mathbf{k}_{(8i+j)B+l} \cdot 2^l$
4: $\mathbf{K}_{i,j} \leftarrow ec(k) = k \cdot q/2^B$
5: **return** $\mathbf{K} = (\mathbf{K}_{i,j})_{\substack{0\le i\le 7\\ 0\le j\le 7}}$

Algorithm 3 FRODO.DECODE

Input: Matrix $\mathbf{K} \in \mathbb{Z}_q^{8\times 8}$

Output: Bit string $\mathbf{k} \in \{0,1\}^\ell, \ell = B \cdot 64$

1: **for** i=0 to 7 **do**
2: **for** j=0 to 7 **do**
3: $k \leftarrow dc(\mathbf{K}_{i,j}) = \lfloor \mathbf{K}_{i,j} \cdot 2^B/q \rceil \bmod 2^B$
4: $k = \sum_{l=0}^{B-1} k_l \cdot 2^l$ where $k_l \in \{0,1\}$
5: **for** $l = 0$ to $B-1$ **do**
6: $\mathbf{k}_{(8i+j)B+l} \leftarrow k_l$
7: **return k**

The algorithms above use the encoding function $ec : \mathbb{Z}_{2^B} \to \mathbb{Z}_q$ and the decoding function $dc : \mathbb{Z}_q \to \mathbb{Z}_{2^B}$ defined as follows:

$$ec\colon \quad k \quad \mapsto k \cdot \frac{q}{2^B} \qquad \text{and} \qquad dc\colon \quad c \quad \mapsto \left\lfloor c \cdot \frac{2^B}{q} \right\rceil \bmod 2^B$$

One can easily verify that $dc(ec(k)) = k$.

3.3.2 Reliability and bandwidth

The decryption error probability of FrodoPKE is $P_e = \mathbb{P}\{\mathbf{m}' \neq \mathbf{m}\}$. The value of $\mathbf{V}'$ before decrypting it with the function FRODO.DECODE($\cdot$) can be simplified as follows:

$$\begin{aligned}\mathbf{V}' &= \mathbf{V} + \text{FRODO.ENCODE}(\mathbf{m}) - (\mathbf{S}'\mathbf{A} + \mathbf{E}')\,\mathbf{S} \\ &= (\mathbf{S}'\mathbf{B} + \mathbf{E}'') + \text{FRODO.ENCODE}(\mathbf{m}) - \mathbf{S}'\mathbf{A}\mathbf{S} - \mathbf{E}'\mathbf{S} \\ &= \mathbf{S}'(\mathbf{A}\mathbf{S} + \mathbf{E}) + \mathbf{E}'' + \text{FRODO.ENCODE}(\mathbf{m}) - \mathbf{S}'\mathbf{A}\mathbf{S} - \mathbf{E}'\mathbf{S} \\ &= \text{FRODO.ENCODE}(\mathbf{m}) + \underbrace{\mathbf{S}'\mathbf{E} + \mathbf{E}'' - \mathbf{E}'\mathbf{S}}_{\mathbf{E}'''}\end{aligned} \tag{3.2}$$

Hence,

$$\mathbf{m}' = \text{FRODO.DECODE}(\mathbf{V}') = \text{FRODO.DECODE}(\text{FRODO.ENCODE}(\mathbf{m}) + \mathbf{E}''')\,. \tag{3.3}$$

The following Lemma was proven in [N+20] and shows that whenever the entries of $\mathbf{E}'''$ are small enough, the expression of $\mathbf{m}'$ is simply the original message $\mathbf{m}$.

Lemma 3.1. Let $e \in \left[\!\left[-\frac{q}{2^{B+1}}, \ldots, \frac{q}{2^{B+1}}\right[\!\right[$ and $k \in \mathbb{Z}_{2^B}$. Then, $dc(ec(k) + e) = k$.

Proof.

$$\begin{aligned} dc(ec(k)+e) &= \left\lfloor (ec(k)+e)\cdot\frac{2^B}{q}\right\rceil \bmod 2^B \\ &= \left\lfloor \left(k\frac{q}{2^B}+e\right)\cdot\frac{2^B}{q}\right\rceil \bmod 2^B \\ &= \left\lfloor k+e\frac{2^B}{q}\right\rceil \bmod 2^B \end{aligned}$$

But $-\frac{q}{2^{B+1}} < e < \frac{q}{2^{B+1}}$ implies that $-\frac{1}{2} < e\frac{2^B}{q} < \frac{1}{2}$, so $\left\lfloor k+e\frac{2^B}{q}\right\rceil = k$. □

Corollary 3.1. Let $\mathbf{K} \longleftarrow$ FRODO.ENCODE $(\cdot)$ and $\mathbf{E} \in \mathbb{Z}_q^{8\times 8}$ such that each entry of $\mathbf{E}$ is in the interval $\left[\left[-\frac{q}{2^{B+1}}, \ldots, \frac{q}{2^{B+1}}\right[\right[$, then, FRODO.DECODE$(\mathbf{K}+\mathbf{E})$ = FRODO.DECODE$(\mathbf{K})$.

Proof. This follows from Lemma 3.1. and the fact that FRODO.DECODE uses $dc\,(\mathbf{K}_{i,j}+\mathbf{E}_{i,j})$ in the third line of Algorithm 3, where $\mathbf{K}_{i,j} \longleftarrow ec$ in Algorithm 2. □

From Corollary 3.1 we deduce that FRODO.DECODE$(\mathbf{V}')$ is equal to $\mathbf{m}$ if each entry of $\mathbf{E}'''$ is in $\left[\left[-\frac{q}{2^{B+1}}, \ldots, \frac{q}{2^{B+1}}\right[\right[$. Due to matrix multiplication, each entry of $\mathbf{E}'''$ is the sum of $2n$ products of two independent samples from χ_{Frodo} and one more independent sample from χ_{Frodo}. The support of χ_{Frodo} is rather small (see Table 3.2), and hence the distribution of the product with itself can be efficiently computed. Moreover, the distribution of the sum of such $2n$ terms, as well as the additional error term, is obtained by applying the standard convolution approach. The overall error probability is given by

$$P_e \leq \sum_{0\leq i,j<8} \mathbb{P}\left\{\mathbf{E}'''_{i,j} \notin \left[\left[-\frac{q}{2^{B+1}}, \ldots, \frac{q}{2^{B+1}}\right[\right[\right\}.$$

We note that FrodoKEM admits the same failure probability as the underlying FrodoPKE as computed in Subsection 3.3.2.

In FrodoKEM, the transmitted information between Alice and Bob is compressed using FRODO.PACK $(\cdot)$ [N$^+$20, Algorithm 3] and then decompressed using FRODO.UNPACK $(\cdot)$ [N$^+$20, Algorithm 4]. The compression function transforms the matrices $\mathbf{U}$ and $\mathbf{C}$ in Table 3.1 to a bit string of length $\log(q)\cdot\bar{n}\cdot n$ and $\log(q)\cdot\bar{n}\cdot\bar{n}$ respectively. For example in Frodo-640, the ciphertext size is $15\times 640\times 8 + 15\times 8\times 8 = 82944$ bits $= 9720$ bytes.

Table 3.2 illustrates the failure probability for each level according to the selected parameter sets, together with the corresponding ciphertext size.

	n	q	σ	**Support of** χ	B	$\bar{n}$	ciphertext size (bytes)	P_e	Rényi Divergence α	Rényi Divergence $D_\alpha\left(\chi_{\text{Frodo}}, \Psi_{\sigma\sqrt{2\pi}}\right)$
Frodo-640	640	2^{15}	2.8	$[-12, \ldots, 12]$	2	8	9720	$2^{-138.7}$	200	$0.324 \cdot 10^{-4}$
Frodo-976	976	2^{16}	2.3	$[-10, \ldots, 10]$	3	8	15744	$2^{-199.6}$	500	$0.140 \cdot 10^{-4}$
Frodo-1344	1344	2^{16}	1.4	$[-6, \ldots, 6]$	4	8	21632	$2^{-252.2}$	1000	$0.264 \cdot 10^{-4}$

Table 3.2: Parameter selection for FrodoKEM.

3.3.3 Security

In [N+20] the security of the FrodoKEM protocol is evaluated with respect to three criteria: IND-CPA security for FrodoPKE, IND-CCA security for FrodoKEM as well as concrete security against known attacks. We provide a brief summary of this security analysis.

IND-CPA security for FrodoPKE. The public-key encryption scheme FrodoPKE is considered IND-CPA secure assuming the hardness of decision-LWE. The proof follows the steps in [LP11]. Informally speaking, the idea is to prove that for any encrypted message $\mathbf{m}$, the adversary cannot distinguish a communicated information which appears clearly to the public from uniformly random. First of all by looking at Table 3.1, the public key $(\mathbf{A}, \mathbf{B})$ is computationally indistinguishable from a uniform one $(\mathbf{A}, \mathbf{B}^*)$ assuming the hardness of decision-LWE (see Definition 3.2). Moreover, the matrices $\mathbf{U}$ and $\mathbf{V}$ are likewise indistinguishable from uniformly random matrices $\mathbf{U}^*$ and $\mathbf{V}^*$ for the same reason. This implies that $\mathbf{C} = \mathbf{V} + \text{FRODO.ENCODE}(\mathbf{m})$ is also indistinguishable from uniform using the following lemma:

Lemma 3.2. Let G be a finite group and $b \in G$ a fixed element. Taking uniformly random elements $a \in G$ implies that $a + b$ will also be uniformly distributed in G.

So as a result, the pair $(\mathbf{U}, \mathbf{C})$ is indistinguishable from uniform.
In order to give a concrete bound on the advantage $\mathsf{Adv}^{\text{IND-CPA}}_{\text{FrodoPKE}}$, a combination of both Theorem 5.9 and Theorem 5.10 in FrodoKEM [N+20] leads to the following final result:

Theorem 3.3 (Uniform-secret DLWE $\Longrightarrow$ IND-CPA security of FrodoPKE)**.** Let $n, q, \bar{n}, k$ be positive integers with $q \geq 2$ a power-of-two, and let $\Psi_{s>0}$ be a probability distribution on $\mathbb{Z}$. There exist classical algorithms $\mathcal{B}'_1, \mathcal{B}'_2$ that use any quantum or classical algorithm $\mathcal{A}$ that attacks the IND-CPA security of FrodoPKE as a "black-box" subroutine in order to provide the following advantage:

$$\mathsf{Adv}^{\text{IND-CPA}}_{\text{FrodoPKE}}(\mathcal{A}) \leq \bar{n} \cdot \mathsf{Adv}^{\text{DLWE}}_{n,2n+k,q,\Psi_s}(\mathcal{B}'_1) + \bar{n} \cdot \mathsf{Adv}^{\text{DLWE}}_{n,2n+\bar{n}+k,q,\Psi_s}(\mathcal{B}'_2) + 2 \cdot 2^{-k},$$

where k is some security parameter such that 2^{-k} is negligible. The running times of $\mathcal{B}'_1$ and $\mathcal{B}'_2$ are approximately that of $\mathcal{A}$.

IND-CCA security for FrodoKEM. Having established the IND-CPA security regarding FrodoPKE, we can now explore the FrodoKEM scheme that guarantees IND-CCA security by applying the Fujisaki-Okamoto transform. The security is studied with respect to both the classical and quantum random oracle models discussed in Subsection 2.3.3.

Theorem 5.1 in [N+20] demonstrates the IND-CCA security of a KEM that is derived from an IND-CPA-secure public-key encryption scheme, even if the KEM and PKE use different error distributions, as long as those distributions are close in terms of Rényi divergence (see Subsection 2.4.4). This is only true in the classical ROM.

Theorem 3.4 (Theorem 5.1 in [N+20])**.** Let $\mathrm{PKE}_X = (\mathsf{Gen}; \mathsf{Enc}; \mathsf{Dec})$ be a δ-correct public-key encryption scheme with message space $\mathcal{M}$ that is parameterized by a distribution X, and let t be an upper bound on the total number of samples drawn from X by Gen and Enc combined. Let G_1, G_2 and F be independent random oracles, and let $\mathsf{KEM}_X^{\not\perp'} = \mathrm{FO}^{\not\perp'}(\mathsf{PKE}_X, G_1, G_2, F)$ be the KEM obtained by applying the $\mathrm{FO}^{\not\perp'}$ transform from Definition 2.21 to PKE_X. Let P, Q be any discrete distributions. There exists a classical algorithm (a reduction) $\mathcal{B}$ against the IND-CPA security of PKE_Q, which uses as a "black box" subroutine any $\mathcal{A}$ against the IND-CCA security of $\mathsf{KEM}_P^{\not\perp'}$ that makes at most q_{RO} oracle queries, for which

$$\mathsf{Adv}_{\mathrm{KEM}}^{\text{IND-CCA}}(\mathcal{A}) \leq \frac{q_{\mathrm{RO}}}{|\mathcal{M}|} + \left(\left(\frac{2 \cdot q_{\mathrm{RO}} + 1}{|\mathcal{M}|} + q_{\mathrm{RO}} \cdot \delta + 3 \cdot \mathsf{Adv}_{\mathrm{PKE}}^{\text{IND-CPA}}\right) \cdot e^{t \cdot D_\alpha(P||Q)}\right)^{1-\frac{1}{\alpha}} \tag{3.4}$$

The total running time of $\mathcal{B}$ is about that of $\mathcal{A}$ plus the time needed to simulate the random oracles.

The bound in (3.4) is generic and can be applied to any PKE scheme with its transformation KEM.

Corollary 3.2. Using the finite support distribution χ_{Frodo}, FrodoKEM is IND-CCA secure, provided that the FrodoPKE protocol using a rounded Gaussian distribution $\Psi_{\sigma\sqrt{2\pi}}$ is IND-CPA secure.

Regarding the IND-CCA security for a KEM against quantum attackers in the quantum random oracle model, a non-tight reduction is proven in Theorem 5.8 in [N+20]. Unfortunately, the FrodoKEM instantiations do not verify the hypotheses of this theorem, so there is no currently security guarantee against quantum attackers. This theorem can be seen as a support for the security of general constructions of LWE-based KEMs in the style of FrodoKEM against quantum adversaries.

Concrete security for FrodoKEM. The concrete security bounds in FrodoKEM are calculated similarly to [LP11, ADPS16b, AGVW17]. BKW types of attacks [KF15] and linearization attacks [AG11] are not relevant to this case due to the small number m of LWE samples available

to the attacker ($m \approx n$ in FrodoKEM). The best known attacks in this context are BKZ primal and dual attacks (see discussion in Paragraph 3.2).

Regarding the LWE parameters, the reduction shown in Paragraph 3.2 clarifies the choice of the standard deviation values in Table 3.2. The choice of the standard deviation $\sigma = \alpha q/\sqrt{2\pi}$ for the distribution in FrodoKEM takes condition (3.1) into account. FrodoKEM gives concrete examples on the number of instances N which an adversary can reasonably obtain. Taking the very large number of samples $N = 2^{256}$ in condition (3.1), one obtains the lower bound $\alpha q > 5.314$ for the Gaussian parameter, and hence a standard deviation greater than $5.314/\sqrt{2\pi} = 2.12$. Moreover, it is unreasonable to assume that an adversary can obtain more than $N = 2^{111}$ samples from $D_{\Lambda^*,s}$, and hence this value of N remains sufficient to not break the BDDwDGS problem. The corresponding σ is 1.4.

Table 3.3 presents the cost of primal and dual attacks on a single instance of the LWE problem via the FrodoKEM script `pqsec.py`[2] with parameters n, σ, q. The authors prefer to make a conservative assumption and evaluate the core SVP hardness of the protocol (see Paragraph 3.2). As also mentioned in Paragraph 3.2, the sieving algorithms are used to predict the core hardness, where the cost of vector operations is 2^{cb} for a well known c (about $b \cdot 2^{cb}$ CPU cycles).

		m	b	Known Classical (C) $= \log\left(b \cdot 2^{0.292b}\right)$	Known Quantum (Q) $= \log\left(b \cdot 2^{0.265b}\right)$	Best Plausible (P) $= \log\left(b \cdot 2^{0.2075b}\right)$
Frodo-640	Primal	712	482	150	137	109
	Dual	716	478	149	136	108
Frodo-976	Primal	1023	705	216	196	156
	Dual	1040	700	214	195	154
Frodo-1344	Primal	1244	929	281	256	202
	Dual	1253	923	279	254	201

Table 3.3: Security bounds for FrodoKEM in $\log_2$ scale. This illustrates the optimized cost over all possible choices of b - the block dimension of BKZ - and m - the number of used samples.

After a series of reductions, Table 3.4 represents the resulting security bounds that are slightly weaker than the exact values in Table 3.3. The additional column "IND-CCA security bound C" represents the bound (3.4), where $\mathsf{Adv}^{\text{IND-CPA}}_{\text{PKE}}$ is replaced with the best known LWE attack.

[2] `https://github.com/lwe-frodo/parameter-selection/blob/master/pqsec.py`

	LWE security			IND-CCA security bound
	C	Q	P	C
Frodo-640	145	132	104	141
Frodo-976	210	191	150	206
Frodo-1344	275	250	197	268

Table 3.4: Security bounds for FrodoKEM after a series of reductions.

3.3.4 Drawbacks of the FrodoKEM protocol

The plain-LWE scheme FrodoKEM was selected as an alternate candidate for the NIST challenge which may provide better longer-term security guarantees since it has the least amount of structure and thus less susceptible to algebraic attacks. From the NIST's perspective, although FrodoKEM can be used in the event that new cryptanalytic results targeting structured lattices emerge, the first priority for standardization is a KEM that would have acceptable performance across widely used applications. With regards to FrodoKEM, the cost of the high security feature is paid for by a much worse performance on all metrics compared to other lattice schemes (see Figures 3.1 and 3.2), and so the protocol must be improved further before it is suitable for standardization.

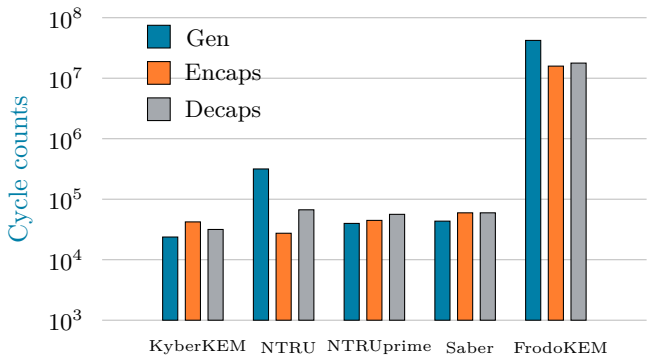

Figure 3.1: Performance of NIST lattice-based KEMs candidates. The cycle count axis has logarithmic scale [Moo21, Slide 26].

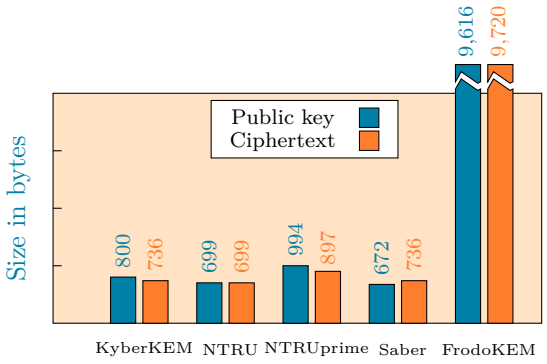

Figure 3.2: Bandwidth cost of NIST Level 1 lattice-based KEMs [Pre21, Slide 13].

In particular, the communication bandwidth required by the protocol is too large and should be reduced. Moreover, increasing its security level against known attacks would give FrodoKEM a better security margin to resist enhanced computing power in the future. In fact, although the current security estimate is already greater than the brute-force security, the plausible security is not and should be improved in order to prevent future attacks.

Error correction for FrodoKEM: state of the art. A modification of FrodoKEM has been proposed in [L+19] using Gray labeling and error correcting codes in order to improve the performance. However, the decryption failure analysis in [L+19] assumes that the coefficients of the error are independent. Unfortunately this assumption does not hold for FrodoKEM, and as shown in [DVV19], it can lead to underestimating the decryption failure by a large exponential factor.

3.4 Error correction for FrodoKEM using the Gosset lattice

In this section, we propose our modified version of FrodoKEM which provides improvements in bandwidth, private key size or plausible security, depending on the chosen parameters. The new encoder maps the message $\mathbf{m}$ into a suitably scaled version of the 64-dimensional lattice $(E_8)^8$, i.e., the product of 8 copies of the Gosset lattice. This lattice has been used before to improve the performance of lattice-based KEMs [ZJGS17, JZ20] We choose E_8 since it gives the densest 8-dimensional packing, resulting in a more efficient decoding, and also admits a low-complexity optimal decoding (see Subsection 2.2.3). Since all integer operations in FrodoKEM are performed modulo q, we identify the lattice points that are equivalent modulo $q\mathbb{Z}^{64}$.

3.4.1 Improving performance using error correction

Our main modification focuses on the FrodoPKE encryption scheme, and the resulting adjustments to FrodoKEM will be discussed later.

Referring to Table 3.1, the main changes in our modified protocol are made in the encryption and decryption algorithms Frodo.Encode($\cdot$) and Frodo.Decode($\cdot$) respectively. Following the approach in [vP16], we search for a suitable scaling parameter β such that

$$q\mathbb{Z}^{64} \subseteq (\beta E_8)^8 \subseteq \mathbb{Z}^{64}. \tag{3.5}$$

The first inclusion is required because we identify points that are equivalent modulo q, while the second one is required because all computations in FrodoKEM are done with integer coefficients. The scalar β is examined in the following. Our aim is to choose the scaling parameter β that verifies condition (3.5) knowing that $2\mathbb{Z}^8 \subseteq E_8 \subseteq \frac{1}{2}\mathbb{Z}^8$, so as to obtain an injective function from $\{0,1\}^\ell$ to $(\beta E_8)^8 / q\mathbb{Z}^{64} \subseteq \mathbb{Z}_q^{8\times 8}$. In order for such an injective function to exist, the number of points in $(\beta E_8)^8 / q\mathbb{Z}^{64}$ must be greater or equal to 2^ℓ. This can be translated into the following volume condition:

$$\frac{\mathrm{Vol}\left(q\mathbb{Z}^{64}\right)}{\mathrm{Vol}\left((\beta E_8)^8\right)} \geq 2^\ell \iff \frac{q^{64}}{\beta^{64}} \geq 2^\ell \iff \beta \leq q/2^{\ell/64}.$$

Clearly, it is better to choose β satisfying the equality $\beta = q/2^{\ell/64}$ since this value yields the best possible minimum distance in the output lattice and thus the best error correction. Note that in this case, each element in $\beta E_8/q\mathbb{Z}^8$ is now identified with the corresponding coset leader

in $E_8/2^{\ell/64}\mathbb{Z}^8$.

If we consider the three security levels of FrodoKEM, ℓ must be in $\{128, 192, 256\}$, which implies $\beta = q/2^{\ell/64} \in \{q/4, q/8, q/16\}$. The construction of the encoder proceeds as follows. First, $\mathbf{m} \in \{0,1\}^\ell$ is partitioned into 8 substrings $\mathbf{m}_i \in \{0,1\}^{\ell/8}$, $i = 0, \ldots, 7$. Each substring is mapped into the set $E_8/2^{\ell/64}\mathbb{Z}^8$ of cardinality $2^{\ell/8}$, so that the whole message is mapped into $\left(E_8/2^{\ell/64}\mathbb{Z}^8\right)^8$. First we define a function $f : \{0,1\}^8 \longrightarrow E_8/2\mathbb{Z}^8$ that maps $\mathbf{b} = [b_1, b_2, \ldots, b_8] \in \{0,1\}^8$ as follows:

$$\begin{cases} f(\mathbf{b}) = [b_1, \ldots, b_7, -1] \cdot \mathbf{G}_{E_8} \bmod 2 & \text{if } b_1 = 0 \;\&\&\; b_8 = 0 \\ f(\mathbf{b}) = [b_1, \ldots, b_7, 0] \cdot \mathbf{G}_{E_8} \bmod 2 & \text{if } b_1 = 0 \;\&\&\; b_8 = 1 \\ f(\mathbf{b}) = [b_1, \ldots, b_7, 1] \cdot \mathbf{G}_{E_8} \bmod 2 & \text{if } b_1 = 1 \;\&\&\; b_8 = 0 \\ f(\mathbf{b}) = [b_1, \ldots, b_7, 2] \cdot \mathbf{G}_{E_8} \bmod 2 & \text{if } b_1 = 1 \;\&\&\; b_8 = 1 \end{cases}$$

A simple `Matlab` simulation shows that f is a bijective function. As an example, for $\ell = 128$, the value of β is $q/4$. Hence mapping 8 bits of information into $E_8/2\mathbb{Z}^8$ allows to map 16 bits into $E_8/4\mathbb{Z}^8$, which is extended to $\frac{q}{4}E_8/q\mathbb{Z}^8$ in the standard way by multiplying by $q/4$. We can map 16 bits into the quotient $E_8/4\mathbb{Z}^8$ as follows: map the first 8 bits into $E_8/2\mathbb{Z}^8$ using f, and the remaining ones into $2\mathbb{Z}^8/4\mathbb{Z}^8$. This last mapping is obtained by simply multiplying the input string by 2. This example can be extended to the cases $\ell = 192$ and $\ell = 256$ by considering the chain $E_8 \supseteq 2\mathbb{Z}^8 \supseteq 4\mathbb{Z}^8 \supseteq 8\mathbb{Z}^8 \supseteq 16\mathbb{Z}^8$. We denote the function that maps the remaining $\ell/8 - 8$ bits by g and can be defined as follows:

$$\begin{cases} g(b_8, \ldots, b_{15}) = 2 \cdot (b_8, \ldots, b_{15}) \in 2\mathbb{Z}^8/4\mathbb{Z}^8 & \text{if } \ell = 128 \\ g(b_8, \ldots, b_{23}) = 2 \cdot (b_8, \ldots, b_{15}) + 4 \cdot (b_{16}, \ldots, b_{23}) \in 2\mathbb{Z}^8/8\mathbb{Z}^8 & \text{if } \ell = 192 \\ g(b_8, \ldots, b_{31}) = 2 \cdot (b_8, \ldots, b_{15}) + 4 \cdot (b_{16}, \ldots, b_{23}) + 8 \cdot (b_{24}, \ldots, b_{31}) \in 2\mathbb{Z}^8/16\mathbb{Z}^8 & \text{if } \ell = 256 \end{cases}$$

The function g is bijective and the inverse g^{-1} can be obtained by

$$\begin{cases} g^{-1}(\mathbf{y}) = \frac{\mathbf{y}}{2} \in \{0,1\}^8 & \text{if } \ell = 128 \\ g^{-1}(\mathbf{y}) = \left(\frac{\mathbf{y} \bmod \mathbf{4}}{\mathbf{2}} || \frac{\mathbf{y} - \mathbf{y} \bmod \mathbf{4}}{\mathbf{4}}\right) \in \{0,1\}^{16} & \text{if} \ell = 192 \\ g^{-1}(\mathbf{y}) = \left(\frac{\mathbf{y} \bmod \mathbf{4}}{\mathbf{2}} || \frac{\mathbf{y} \bmod \mathbf{8} - \mathbf{y} \bmod \mathbf{4}}{\mathbf{4}} || \frac{\mathbf{y} - \mathbf{y} \bmod 8}{8}\right) \in \{0,1\}^{24} & \text{if } \ell = 256 \end{cases}$$

The encoding function FRODO.ENCODE$(\cdot)$ can now be changed to E8.ENCODE$(\cdot)$ as shown in Algorithm 4. Moreover, as a result, each substring $\mathbf{m}_i \in \{0,1\}^{\ell/8}$ is mapped into a vector in $\beta E_8/q\mathbb{Z}^8 \subseteq \mathbb{Z}_q^8$ that corresponds to a diagonal block of the output matrix $\mathbf{O}$. In fact, each diagonal block is equal to the 8-dimensional row-vector $\mathbf{S}_i$ that can be expressed as

$$\text{BLOCK}_i(\mathbf{O}) = (O_{i \bmod 8,0}, O_{i+1 \bmod 8,1}, \ldots, O_{i+7 \bmod 8,7}), \; i = 0, \ldots, 7.$$

Algorithm 4 Encoding ℓ bits into $\left(\beta E_8/q\mathbb{Z}^8\right)^8$

Input: $\mathbf{m} = (m_0, \ldots, m_{\ell-1}) \in \{0,1\}^\ell$
Output: $\mathbf{O} \in \left(\beta E_8/q\mathbb{Z}^8\right)^8 \subseteq \mathbb{Z}_q^{8\times 8}$

function E8.ENCODE($\mathbf{m}$)
  for i=0 to 7 **do**
    $\mathbf{m}_i = (m_{i(\ell/8)}, m_{i(\ell/8)+1}, \ldots, m_{i(\ell/8)+\ell/8-1}) \in \{0,1\}^{\ell/8}$
    $\mathbf{X}_i = f(\mathbf{m}_{i,0}, \ldots, \mathbf{m}_{i,7}) \in E_8/2\mathbb{Z}^8$ ▷ Encoding the first 8 bits of $\mathbf{m}_i$
    $\mathbf{X}'_i = g(\mathbf{m}_{i,8}, \ldots, \mathbf{m}_{i,\ell/8-1}) \in 2\mathbb{Z}^8/2^{\ell/64}\mathbb{Z}^8$ ▷ Encoding the remaining bits of $\mathbf{m}_i$
    $\mathbf{S}_i = \mathbf{X}_i + \mathbf{X}'_i \in E_8/2^{\ell/64}\mathbb{Z}^8 \cong \beta E_8/q\mathbb{Z}^8$ ▷ 8-dimensional row vector
    for j=0 to 7 **do**
      $O_{i,j} = S_{(8-i+j) \bmod 8, j}$ ▷ $S_{i,j}$ is the j-th component of $\mathbf{S}_i$
  return $\mathbf{O} = (O_{i,j})$

Finally, E8.ENCODE is a bijection from $\{0,1\}^\ell$ to $\left(\beta E_8\right)^8/q\mathbb{Z}^{64}$. This allows to define the inverse function E8.ENCODE$^{-1}(\mathbf{Y})$ for any $\mathbf{Y} \in \left(\beta E_8\right)^8/q\mathbb{Z}^{64}$ as described in Algorithm 5.

Algorithm 5 Inverse of Encoding

Input: Matrix $\mathbf{Y} \in \left(\beta E_8/q\mathbb{Z}^8\right)^8$
Output: $\mathbf{m} = (m_0, \ldots, m_{\ell-1}) \in \{0,1\}^\ell$

function E8.ENCODE^{-1}($\mathbf{Y}$)
  for i=0 to 7 **do**
    $\left(m_{i(\ell/8)}, m_{i(\ell/8)+1}, \ldots, m_{i(\ell/8)+7}\right) = f^{-1}\left(\text{BLOCK}_i(\mathbf{Y}) \bmod 2\right)$
    $\left(m_{i(\ell/8)+8}, m_{i(\ell/8)+1}, \ldots, m_{i(\ell/8)+\ell/8-1}\right) = g^{-1}\left(\text{BLOCK}_i(\mathbf{Y}) - \text{BLOCK}_i(\mathbf{Y}) \bmod 2\right)$
  return $\mathbf{m} = (m_0, \ldots, m_{\ell-1})$

Decoding Algorithm. The decoding algorithm E8.DECODE uses the CVP_{E_8} algorithm presented in chapter 2 - Algorithm 1. We describe the decoding protocol in Algorithm 6. It concatenates the outputs of CVP_{E_8} to form an element of $\left(\beta E_8\right)^8/q\mathbb{Z}^{64}$. Since our lattice E_8 is scaled by β, we use the fact that $\mathsf{CVP}_{\beta E_8}(\mathbf{x}) = \beta \cdot \mathsf{CVP}_{E_8}\left(\frac{1}{\beta}\mathbf{x}\right)$ (see Lemma 2.1).

Algorithm 6 Modified decoding algorithm

Input: $\mathbf{N} \in \mathbb{R}_q^{8\times 8}$
Output: $\mathbf{m}' = (m'_0, \ldots, m'_{\ell-1}) \in \{0,1\}^\ell$

function E8.DECODE($\mathbf{N}$)
  for i=0 to 7 **do**
    $\mathbf{Y}_i = \beta \cdot \mathsf{CVP}_{E_8}\left(\frac{1}{\beta}\text{BLOCK}_i(\mathbf{N})\right) \bmod q$ ▷ 8-dimensional row vector in $\beta E_8/q\mathbb{Z}^8$
    for j=0 to 7 **do**
      $O_{i,j} = Y_{(8-i+j) \bmod 8, j}$ ▷ $Y_{i,j}$ is the j-th component of $\mathbf{Y}_i$
  return $\mathbf{m}' = \text{E8.ENCODE}^{-1}(\mathbf{O}) \in \{0,1\}^\ell$

Remark 3.1. A discussion of the efficiency of those algorithms is given in Appendix B.1.

3.4.2 Error distribution

As in the FrodoKEM specifications, we use a discrete and symmetric error distribution χ on $\mathbb{Z}$, centered at zero and with finite support $\{-s, \ldots, s\}$, which approximates a rounded Gaussian distribution with standard deviation σ. In our case, χ is generated for different values of σ, and the support $\{-s, \ldots, s\}$ depends on the chosen σ value. Sometimes the support is taken to be the same as for the rounded Gaussian, and sometimes it is narrowed (to allow fast simulations), keeping a small Rényi divergence with respect to the rounded Gaussian.

Given the target standard deviation σ, we first construct a function $\tilde{\chi}$ on $\{-s, \ldots, s\} \subseteq \mathbb{Z}$ from 2^{16} samples as follows:

$$\forall i \in \{-s, \ldots, s\},\ \tilde{\chi}(i) = \frac{1}{2^{16}} \left\lfloor 2^{16} \cdot \int_{\left[i-\frac{1}{2}, i+\frac{1}{2}\right]} D_{\sigma\sqrt{2\pi}}(x)dx \right\rceil .$$

The distribution χ is obtained from $\tilde{\chi}$ by making small changes in the numerator values of $\tilde{\chi}(i)$ in order to obtain a probability distribution The sampling algorithm for such a distribution is given in [N+20, Algorithm 5]. Note that it is possible to perform this sampling in constant time in order to avoid cache and timing side-channels attacks. We compute the Rényi divergence of χ from the rounded Gaussian $\Psi_{\sigma\sqrt{2\pi}}$ via the script `scripts/Renyi.py` in [ADPS16b]. Explicit examples for our chosen distributions are given in Subsection 3.4.4.

3.4.3 Reliability analysis

In this section we aim to provide an upper bound for the decryption error probability for our algorithm. Clearly, an error occurs whenever the received message $\mathbf{m}'$ differs from the original one $\mathbf{m}$, i.e., $P_e = \mathbb{P}\{\mathbf{m} \neq \mathbf{m}'\}$. From Table 3.1 and following equation (3.2), the expression of $\mathbf{V}'$ can be simplified as

$$\mathbf{V}' = \text{E8.}\textsc{Encode}(\mathbf{m}) + \underbrace{\mathbf{S}'\mathbf{E} + \mathbf{E}'' - \mathbf{E}'\mathbf{S}}_{\mathbf{E}'''}. \tag{3.6}$$

Similarly to equation (3.3), we can express the decoded message $\mathbf{m}'$ as

$$\mathbf{m}' = \mathbf{m} + \text{E8.}\textsc{Decode}\left(\mathbf{E}'''\right). \tag{3.7}$$

Each entry $E'''_{i,j}$ in the matrix $\mathbf{E}'''$ is the sum of $2n$ products of two independent samples from χ, adding to it another independent sample also from χ:

$$\forall\, 0 \leq i, j \leq 7,\ E'''_{i,j} = \sum_{k=0}^{n-1} \left(S'_{i,k} E_{k,j} - E'_{i,k} S_{k,j} \right) + E''_{i,j} \tag{3.8}$$

The distribution of $E'''_{i,j}$, denoted by χ', can be efficiently computed (see Appendix B.2.1) using the product of probability generating functions explicitly defined in Appendix A.1. From

equation (3.8), it is easy to see that two entries of the matrix $\mathbf{E}'''$ which are not on the same row or column are independent, and hence we can extract 8 identically distributed blocks of 8 independent coordinates from this error matrix as represented below.

$$\mathbf{E}''' = \begin{bmatrix} E'''_{0,0} & E'''_{0,1} & E'''_{0,2} & E'''_{0,3} & E'''_{0,4} & E'''_{0,5} & E'''_{0,6} & E'''_{0,7} \\ E'''_{1,0} & E'''_{1,1} & E'''_{1,2} & E'''_{1,3} & E'''_{1,4} & E'''_{1,5} & E'''_{1,6} & E'''_{1,7} \\ E'''_{2,0} & E'''_{2,1} & E'''_{2,2} & E'''_{2,3} & E'''_{2,4} & E'''_{2,5} & E'''_{2,6} & E'''_{2,7} \\ E'''_{3,0} & E'''_{3,1} & E'''_{3,2} & E'''_{3,3} & E'''_{3,4} & E'''_{3,5} & E'''_{3,6} & E'''_{3,7} \\ E'''_{4,0} & E'''_{4,1} & E'''_{4,2} & E'''_{4,3} & E'''_{4,4} & E'''_{4,5} & E'''_{4,6} & E'''_{4,7} \\ E'''_{5,0} & E'''_{5,1} & E'''_{5,2} & E'''_{5,3} & E'''_{5,4} & E'''_{5,5} & E'''_{5,6} & E'''_{5,7} \\ E'''_{6,0} & E'''_{6,1} & E'''_{6,2} & E'''_{6,3} & E'''_{6,4} & E'''_{6,5} & E'''_{6,6} & E'''_{6,7} \\ E'''_{7,0} & E'''_{7,1} & E'''_{7,2} & E'''_{7,3} & E'''_{7,4} & E'''_{7,5} & E'''_{7,6} & E'''_{7,7} \end{bmatrix}$$

Figure 3.3: Each block of 8 independent components is represented by a distinct color.

Decoding is correct whenever $\text{E8.DECODE}(\mathbf{E}''') = 0$. For this it is sufficient to have $\text{BLOCK}_k(\mathbf{E}''') \in \mathcal{V}(\beta E_8)$ for all $k = 0, \ldots, 7$, i.e.,

$$\langle \text{BLOCK}_k(\mathbf{E}'''), \mathbf{v} \rangle < \frac{\|\mathbf{v}\|_2^2}{2}, \forall \mathbf{v} \in \beta \left(\text{VR}_{E_8}^{(1)} \cup \text{VR}_{E_8}^{(2)} \right),$$

where the sets $\text{VR}_{E_8}^{(1)}$ and $\text{VR}_{E_8}^{(2)}$ refer to the two types of Voronoi relevant vectors for E_8 (see Subsection 2.2.3). The error probability can thus be bounded by

$$P_e \leq \sum_{j=1}^{2} \sum_{i=0}^{7} \mathbb{P} \left\{ \exists \mathbf{v}_j \in \text{VR}_{E_8}^{(j)} : \langle \text{BLOCK}_k(\mathbf{E}'''), \mathbf{v}_j \rangle \geq \frac{\beta \|\mathbf{v}_j\|_2^2}{2} \right\} \tag{3.9}$$

Since the error probability is independent of the choice of Voronoi relevant vector for vectors of the same type (because the distribution of each entry of $\mathbf{E}'''$ is symmetric, centered at 0), without loss of generality we can choose $\mathbf{v}_1 = (1, 1, 0, 0, 0, 0, 0, 0)$ and $\mathbf{v}_2 = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$. This reduces the computations to just two cases. Choosing the value of n and the modulus q, we can compute an upper bound for the above expression for different values of σ. Recalling that $\left|\text{VR}_{E_8}^{(1)}\right| = 112$ and $\left|\text{VR}_{E_8}^{(2)}\right| = 128$, equation (3.9) becomes:

$$P_e \leq 8 \cdot 112 \cdot \mathbb{P} \left\{ E'''_{0,0} + E'''_{1,1} \geq \beta \right\} + 8 \cdot 128 \cdot \mathbb{P} \left\{ E'''_{0,0} + \cdots + E'''_{7,7} \geq 2\beta \right\}. \tag{3.10}$$

In order to upper bound P_e, we use the following.

Remark 3.2. We say that a discrete distribution p taking values in $\mathbb{Z}$ is *unimodal* with mode 0 if $p(n+1) \leq p(n)\ \forall n \geq 0$, and $p(n+1) \geq p(n)\ \forall n < 0$.
Note that the convolution of two symmetric discrete unimodal distributions is symmetric unimodal [DJD88, Theorem 4.7].

Since the distribution χ' is symmetric unimodal, so are the distributions χ'_2, χ'_4, χ'_8 of the

sum of two, four and eight independent copies of $E'''_{i,j}$ respectively. While χ'_2 and χ'_4 can be calculated efficiently, the computation of χ'_8 is slow. Thanks to unimodality, we can estimate the term $\mathbb{P}\left\{E'''_{0,0} + \cdots + E'''_{7,7} \geq 2\beta\right\}$ by upper bounding χ'_8 by a piecewise constant function after computing a small number of values. Details are illustrated in Appendix B.2.2 and the error probabilities for the chosen parameters will be presented in Subsection 3.4.4.

3.4.4 Performance comparison

In this section we propose three sets of parameters: the first aims at improving the security level, the second at reducing the bandwidth and the third at increasing the private key size. We show the impact of the proposed modification of FrodoKEM in terms of the performance of the protocol. Note that for all sets of parameters, n and $\bar{n}$ will remain unchanged.

In terms of security against known attacks, our analysis focuses on the dual and primal attacks, which are more relevant for polynomial number of LWE samples as in our case. Table 3.7 represents the concrete security bounds calculated via the FrodoKEM script `pqsec.py` (including parameters $n, \sigma,\ q$) with regard to classical, quantum and plausible attacks for the different parameter sets, compared to the original FrodoKEM.

The communication requirements regarding the protocol are computed using the functions `Frodo.Pack` and `Frodo.Unpack` presented in [N$^+$20, Algorithm 3, Algorithm 4]. In our case we pack both $\mathbf{U} \in \mathbb{Z}^{\bar{n}\times n}$ and $\mathbf{C} \in \mathbb{Z}^{\bar{n}\times\bar{n}}$. Those two vectors, concatenated together, carry about $(\log(q) \times n + \log(q) \times 8)$ bytes.

Parameter set 1 - Improving the security level. For the first parameter set (see Table 3.7), we aim at increasing the plausible security level while keeping the same bandwidth and a similar error probability level as in the original FrodoKEM protocol. The security level of FrodoKEM with respect to primal/dual attacks (see Paragraph 3.2) is already higher than the brute force security level, but this might change due to improvements in the best known attacks. So this choice of parameters represents an even more conservative option for long-term security.

In order to increase the security, we increase the variance σ while keeping q unchanged. Note that we can increase σ because of the higher error correction capability provided by our modified encoder. The error distribution χ corresponding to each level is presented in Table 3.5.

As shown in Table 3.7, compared to the original versions of FrodoKEM, the plausible security level is increased by 7 bits, while the error probability is slightly improved.

Parameter set 2 - Reducing the bandwidth. For the second set of parameters in Table 3.7, we aim at reducing the bandwidth while keeping the same security level. This is achieved by reducing the modulus q by half, which in turn requires a reduction in standard deviation σ in order to preserve a low error probability[3] (see Table 3.6 for explicit distribution values). Overall,

[3]The condition $\sigma \geq 2.12$ is imposed in [N$^+$20] to allow the reduction from the bounded distance decoding with discrete Gaussian sampling (BDDwDGS) to the decision LWE problem. Note that for efficiency reasons, σ

	n	σ	Probability of (in multiples of 2^{-16})																Rényi divergence	
			0	±1	±2	±3	±4	±5	±6	±7	±8	±9	±10	±11	±12	±13	±14	±15	±16	Order α	$D_\alpha\left(\chi, \Psi_{\sigma\sqrt{2\pi}}\right)$
χ	640	3.9	6686	6473	5866	4981	3962	2952	2061	1347	825	473	254	128	60	27	11	4	1	122	0.6926×10^{-4}
χ	976	2.75	9456	8858	7279	5249	3321	1844	898	384	144	47	13	3						554	0.7357×10^{-4}
χ	1344	1.68	15336	12913	7707	3261	977	208	31	3										325	0.2926×10^{-4}

Table 3.5: Error distributions corresponding to parameter set 1.

	n	σ	Probability of (in multiples of 2^{-16})											Rényi divergence	
			0	±1	±2	±3	±4	±5	±6	±7	±8	±9	±10	Order α	$D_\alpha\left(\chi, \Psi_{\sigma\sqrt{2\pi}}\right)$
χ	640	2.3	11278	10277	7774	4882	2545	1101	396	118	29	6	1	674	0.1569×10^{-4}
χ	976	1.8	14340	12339	7857	3704	1292	333	64	9				215	0.4447×10^{-4}
χ	1344	1.14	22220	15490	5241	858	67	2						655	0.3330×10^{-4}

Table 3.6: Error distributions corresponding to parameter set 2.

the modulus to noise ratio of the protocol is decreased. As an example, we compute the new bandwidth requirements for the modified Frodo-640 which is $\log(2^{14}) \times 640 + \log(2^{14}) \times 8 = 9072$, compared to the original FrodoKEM that was $\log(2^{15}) \times 640 + \log(2^{15}) \times 8 = 9720$. This allows to reduce the bandwidth by approximately 7% (see Table 3.7).

	σ	q	Private key-size	Concrete Security			Bandwidth	P_e
				C	Q	P	(bytes)	
Original FrodoKEM								
Frodo-640	2.80	2^{15}	128	145	132	104	9720	2^{-138}
Frodo-976	2.30	2^{16}	192	210	191	150	15744	2^{-199}
Frodo-1344	1.40	2^{16}	256	275	250	197	21632	2^{-252}
Security Improvements - Parameter set 1								
Modified Frodo-640	3.90	2^{15}	128	158	144	113	9720	2^{-149}
Modified Frodo-976	2.75	2^{16}	192	220	200	158	15744	2^{-204}
Modified Frodo-1344	1.68	2^{16}	256	287	261	205	21632	2^{-255}
Reduce Bandwidth - Parameter set 2								
Modified Frodo-640	2.30	2^{14}	128	152	138	109	9072	2^{-152}
Modified Frodo-976	1.80	2^{15}	192	215	197	155	14760	2^{-203}
Modified Frodo-1344	1.14	2^{15}	256	283	257	203	20280	2^{-271}
Increasing the private key size - Parameter set 3								
Modified Frodo-640	2.30	2^{15}	192	139	126	100	9720	2^{-149}
Modified Frodo-976	1.80	2^{16}	256	200	184	144	15744	2^{-203}

Table 3.7: Modified parameters for improving the security level and/or bandwidth of FrodoKEM scheme, as well as increasing the private key size.

is equal to 1.4 in Frodo-1344, while still guaranteeing a large number N of discrete Gaussian samples, namely $N \approx 2^{111}$. For Frodo-1344 we take $\sigma = 1.15$, which still leads to a large number of discrete Gaussian samples, namely $N \approx 2^{75}$.

Parameter set 3 - Increasing the private key size. For the last set of parameters in Table 3.7, we aim to increase the private key size compared to original FrodoKEM with comparable security and error probability. We generate 192 bits from Frodo-640 instead of 128 bits, as well as 256 bits instead of 192 bits in Frodo-976 with lower bandwidth requirements. The modulus q remains unchanged. Regarding Frodo-1344, increasing the private key size above 256 bits is not required by NIST cryptography standards.

3.4.5 IND-CPA / IND-CCA security

IND-CPA security. The proposed modifications in our scheme affect the choice of parameters q and σ, the error distribution and the encoding and decoding functions. As already discussed in Subsection 3.3.3, the IND-CPA security proof of the FrodoPKE protocol relies on the pseudorandomness of the adversary's observation, and the advantage of an attacker is given in Theorem 3.3. We can use the same argument for our modified protocol. This shows that the choice of encoding function has no effect on the security level, which is only affected by the parameters and error distribution. In particular for parameter sets 1 and 2, the modulus-to-noise ratio is reduced compared to the original FrodoKEM (see Table 3.7), and hence $\mathsf{Adv}^{\mathrm{DLWE}}$ in Theorem 3.3 is not increased (see Corollary 3.2 in [BLP+13]). In fact, referring to parameter set 1, the improved security level requires a larger standard deviation σ, and therefore the $\mathsf{Adv}^{\mathrm{DLWE}}$ doesn't increase. Also, regarding parameter set 2, we note that a slightly lower standard deviation is applied while reducing the modulus q by half, which also means that our $\mathsf{Adv}^{\mathrm{DLWE}}$ is decreased compared to FrodoKEM. In contrast, for Parameter set 3, the increase in private key size comes at the cost of an increased modulus-to-noise ratio (since the error width is decreased and q is unchanged). This means that $\mathsf{Adv}^{\mathrm{DLWE}}$ will be increased compared to the original FrodoKEM.

IND-CCA security. Applying the Fujisaki-Okamoto transformation to our IND-CPA secure protocol yields an IND-CCA secure key encapsulation mechanism in the classical random oracle model (see Theorem 3.4). Our KEM protocol uses the finite support distribution χ that is close to $\Psi_{\sigma\sqrt{2\pi}}$ in terms of Rényi divergence. This allows to bound the IND-CCA advantage $\mathsf{Adv}^{\text{IND-CCA}}$ in the same way as for FrodoKEM by using equation (3.4). Our security loss will be minimized by optimizing the Rényi divergence over the order α.

3.5 Conclusion

In this chapter we used the β-scaled version of the 64-dimensional lattice $(E_8)^8$ in order to modify the underlying public-key encryption scheme for FrodoKEM [N+20]. Our encoding function creates first a bijection between $\{0,1\}^8$ and $E_8/2\mathbb{Z}^8$ which can be extended to a bijection from $\{0,1\}^\ell$ to $(\beta E_8)^8/q\mathbb{Z}^{64}$. In addition, we modify the standard deviation σ of the error distribution

and the modulus q depending on the chosen security level and the improvement we aim to make, keeping the error probability unchanged compared to FrodoKEM. Moreover, we preserve the same matrix dimensions n and $\bar{n}$. The main improvements are classified between two sets of parameters and can be summarized as follows:

- The first set of parameters aims to improve plausible security against attacks on LWE. This is done by widening the standard deviation, or error width, keeping the same modulus q. This results in an improvement of about 8% to 9% with respect to classical, quantum and plausible attacks. IND-CCA security remains unchanged, as does the bandwidth and the error probability bound.

- The second set of parameters is intended to improve the communication bandwidth. By halving the modulus q, we obtain a reduction of about 7% in terms of bandwidth, keeping the same bound for the error probability and a slightly enhanced plausible security.

- The last set of parameters increases the number of private key bits sent for both Frodo-640 and Frodo-976 from 128 to 192 bits and 192 to 256 respectively. This is paid for by a slight reduction in concrete security.

Chapter 4

Kyber with reconciliation

4.1 Introduction

In this chapter, we study another NIST candidate, KyberKEM, and propose a reconciliation mechanism to improve its security. KyberKEM is an IND-CCA secure key encapsulation mechanism that derives from an IND-CPA secure public key encryption scheme. Its core security is based on the difficulty of solving the structured variant of the LWE problem called *Module Learning With Errors* (M-LWE). KyberKEM has passed the third round for the NIST Post-Quantum Cryptography Project and is now one of the finalists with regard to key-establishment algorithms. Module-LWE offers a compact structure compared to the standard LWE problem, and therefore enables an efficient KEM scheme with high performance.

In the final part of the chapter we will propose our own modification of KyberKEM by introducing a reconciliation technique.

4.2 Module Learning With Errors (M-LWE)

LWE has shown to be amazingly versatile, serving in the construction of cryptographic protocols and providing long-term high security against both chosen-plaintext and chosen-ciphertext attacks. Unfortunately, these protocols are often inefficient due to their large public key sizes (quadratic in dimension). This inspired the authors of [LPR10] to develop a more efficient variant of LWE called *Ring Learning With Errors* (R-LWE). This variant has a more compact form due to its additional algebraic structure and offers many cryptographic applications including efficient signature schemes [Lyu12, MP12], fast encryption [LPR10], fast homomorphic encryption [GHS12, BGV14, BV11a] and pseudo-random functions [BPR12]. Solving R-LWE is at least as hard as solving approximate SIVP on ideal lattices - which correspond to ideals of the ring of integers R of a number field K.

Later, an extension of R-LWE based on module lattices, called *Module Learning With Errors* (M-LWE), was proposed in [LS15]. In fact, R-LWE can be seen as a special case of M-LWE

where the module dimension is 1. Moreover, from the cryptographic construction viewpoint, most constructions based on R-LWE can be adapted to M-LWE with an efficiency slowdown bounded by a constant factor.

Cyclotomic rings and modules. Before introducing M-LWE, we need to introduce some background notions related to cyclotomic rings and modules. Let $K = \mathbb{Q}(\zeta_N)$ be the N-th cyclotomic number field of degree $n = \varphi(N)$, where ζ_N is any primitive N-th complex root of unity, e.g. $\zeta_N = \exp(2\pi\mathrm{i}/N)$. We consider the ring of integers R of K which can be explicitly expressed as $R = \mathbb{Z}[X]/(X^n + 1)$. For any integer $q \geq 2$, the quotient ring modulo q is defined as $R_q = \mathbb{Z}_q[X]/(X^n + 1)$. We take N to be a power-of-two integer, and hence $n = N/2$ is also a power-of-two. Any element $\mathbf{a} \in R$ can be expressed as $\mathbf{a} = a_0 + a_1 X + \cdots + a_{n-1}X^{n-1}$ for some integers $a_0, \ldots, a_{n-1}$. Using the naive "coefficient embedding" enables to identify $\mathbf{a} \in R$ with the vector $(a_0, \ldots, a_{n-1}) \in \mathbb{Z}^n$.

Given a ring R, an *R-module* is an additive Abelian group M endowed with a multiplication function $R \times M \to M$ such that $\forall \mathbf{r}, \mathbf{r}' \in R$ and $\forall \vec{\mathbf{m}}, \vec{\mathbf{m}'} \in M$, $\mathbf{r}(\vec{\mathbf{m}} + \vec{\mathbf{m}'}) = \mathbf{r}\vec{\mathbf{m}} + \mathbf{r}\vec{\mathbf{m}'}$, $(\mathbf{r} + \mathbf{r}')\vec{\mathbf{m}} = \mathbf{r}\vec{\mathbf{m}} + \mathbf{r}'\vec{\mathbf{m}}$, and $\mathbf{r}(\mathbf{r}'\vec{\mathbf{m}}) = (\mathbf{r}\mathbf{r}')\vec{\mathbf{m}}$. In this thesis, we will consider the product R^d which can be seen as an R-module with the multiplication $\mathbf{r}(\mathbf{r}_0, \ldots, \mathbf{r}_{d-1}) = (\mathbf{r}\mathbf{r}_0, \ldots, \mathbf{r}\mathbf{r}_{d-1})$.

Given a discrete error distribution χ on $\mathbb{Z}$, one can define an error distribution on the ring R as follows: if $\mathbf{a} \in R$, the notation $\mathbf{a} \leftarrow \boldsymbol{\chi}$ means that $\mathbf{a}$ is a ring sample such that the coefficients $a_0, \ldots, a_{n-1}$ are generated independently according to χ. Furthermore, a d-dimensional vector $\vec{\mathbf{a}} = (\mathbf{a}_0, \ldots, \mathbf{a}_{d-1}) \in R^d$ can be generated according to the distribution $\boldsymbol{\chi}^d$ in which the coefficients $\mathbf{a}_0, \ldots, \mathbf{a}_{d-1}$ are generated independently from $\boldsymbol{\chi}$. For $\vec{\mathbf{a}} = (\mathbf{a}_0, \ldots, \mathbf{a}_{d-1})$ and $\vec{\mathbf{b}} = (\mathbf{b}_0, \ldots, \mathbf{b}_{d-1})$ in R^d, the dot product $\langle \vec{\mathbf{a}}, \vec{\mathbf{b}} \rangle$ is defined as

$$\langle \vec{\mathbf{a}}, \vec{\mathbf{b}} \rangle = \sum_{i=0}^{d-1} \mathbf{a}_i \cdot \mathbf{b}_i \in R,$$

where $\mathbf{a}_i \cdot \mathbf{b}_i$ is the multiplication operation on R.

Definition 4.1 (Decision Module Learning With Errors (M-LWE$_{q,\boldsymbol{\chi}}$))**.** Let d be a positive integer parameter and $R_q = \mathbb{Z}_q[X]/(X^n + 1)$ for some prime integer $q \geq 2$. Let $\boldsymbol{\chi}$ be a discrete distribution on R. The problem consists in distinguishing uniform random samples $(\vec{\mathbf{a}}_i^*, \mathbf{b}_i^*) \leftarrow R_q^d \times R_q$ from samples $(\vec{\mathbf{a}}_i, \mathbf{b}_i) \leftarrow R_q^d \times R_q$ where $\vec{\mathbf{a}}_i \leftarrow R_q^d$ is uniform and $\mathbf{b}_i = \langle \vec{\mathbf{a}}_i, \vec{\mathbf{s}} \rangle + \mathbf{e}_i$ with $\vec{\mathbf{s}} \leftarrow \boldsymbol{\chi}^d$ common to all samples and $\mathbf{e}_i \leftarrow \boldsymbol{\chi}$ freshly generated for each sample.

Note that the distribution $\boldsymbol{\chi}$ may be parameterized by some parameter α and thus denoted as $\boldsymbol{\chi}_\alpha$. If the number of samples is m, then the instances of the M-LWE problem can be given in matrix form $\vec{\mathbf{b}} = \mathbf{A}\vec{\mathbf{s}} + \vec{\mathbf{e}}$ where $\mathbf{A}$ is an $m \times n$ matrix. More precisely,

$$\begin{bmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_m \end{bmatrix} = \begin{bmatrix} \ldots\vec{\mathbf{a}_1}\ldots \\ \vdots \\ \ldots\vec{\mathbf{a}_m}\ldots \end{bmatrix} \cdot \begin{bmatrix} \mathbf{s}_1 \\ \vdots \\ \mathbf{s}_n \end{bmatrix} + \begin{bmatrix} \mathbf{e}_1 \\ \vdots \\ \mathbf{e}_m \end{bmatrix} \bmod q.$$

Definition 4.2 (Module-LWE advantage)**.** For an attacker $\mathcal{A}$, we define the advantage $\mathsf{Adv}^{M\text{-}\mathsf{LWE}}_{m,d,\boldsymbol{\chi}_\alpha}$ as

$$\left|\mathbb{P}\left\{b'=1 \;\middle|\; \begin{array}{c}\mathbf{A}\leftarrow R_q^{m\times d}\\ (\vec{\mathbf{s}},\vec{\mathbf{e}})\leftarrow \boldsymbol{\chi}_\alpha^d\times\boldsymbol{\chi}_\alpha^m\\ \vec{\mathbf{b}}=\mathbf{A}\,\vec{\mathbf{s}}+\vec{\mathbf{e}}\\ b'\leftarrow\mathcal{A}(\mathbf{A},\vec{\mathbf{b}})\end{array}\right\}-\mathbb{P}\left\{b'=1 \;\middle|\; \begin{array}{c}\mathbf{A}\leftarrow R_q^{m\times d}\\ \vec{\mathbf{b}}\leftarrow R_q^m\\ b'\leftarrow\mathcal{A}(\mathbf{A},\vec{\mathbf{b}})\end{array}\right\}\right|$$

where $b'=1$ is the output of the distinguisher $\mathcal{A}$.

A reduction from approximate SIVP on the set of so-called *module lattices* to M-LWE has been shown to exist in [LS15, Theorem 4.7] for prime modulus q. Module lattices are a special class of lattices than can be obtained by embedding a module $M\subseteq K^d$ into $(\mathbb{R}^n)^d$, where K is a number field and d is a positive integer. We refer to the SIVP problem over module lattices as Module-SIVP.

Later, C. Peikert and Z. Pepin showed that given a number field extension K'/K of degree r, where K' and K have rings of integers R' and R respectively, there exists a reduction from M'-LWE for $M'=(R')^{d'}$ to M-LWE for $M=R^d$, where $d=rd'$ (see [PP19, Theorem 6.1]). Taking $d'=1$ and by using the results from [PRSD17] on the hardness of Ring-LWE which hold for any q, this implies hardness of Module-LWE for general modulus q. A classical reduction from standard worst-case lattice problems to M-LWE with arbitrary polynomial-sized modulus q was also proven in [BJRLW20] provided that the module rank d is not smaller than the number field degree n.

A restricted variant of the SVP_γ to ideal lattices has been widely studied in recent years, and recent works suggest that this problem may be easier to solve on an ideal lattices than on an arbitrary ones, contrary to what was previously believed. In [PMHS19] the authors introduce the PHS algorithm that solves SVP_γ on ideal lattices with $\gamma=2^{\tilde{O}\left(\log^{\omega+1}|\Delta_K|/n\right)}$, where Δ_K is the discriminant of K and ω is arbitrary in $[0,1/2]$. This algorithm requires a pre-processing phase that runs exponentially in $\log|\Delta|$, and a query phase that runs classically in time and space $\exp\left(\tilde{O}\left((\log|\Delta|)^{\max(2/3,1-2\omega)}\right)\right)$, while it is $\exp\left(\tilde{O}\left((\log|\Delta|)^{1-2\omega}\right)\right)$ in the quantum setting. A twisted version which was proposed in [BRL20] seems to have better approximation factors in terms of practical implementation, while in theory it performs as well as the original PHS algorithm.

A proposition to apply the LLL algorithm on module lattices was given in [FS10] in order to find short vectors in polynomial time. This inspires the authors of [LPMSW19] to construct an algorithm that efficiently finds short vectors in rank-2 modules given access to a CVP oracle for a lattice that is defined only in terms of K. Based on this, they also construct an algorithm that efficiently finds short vectors in rank-n modules when given access to an oracle that finds short vectors in rank-2 modules. This reduction is polynomial and runs in time $\log|\Delta_K|$.

4.3 KyberKEM

4.3.1 Presentation of the algorithm

In this section we present a simplified description of KyberKEM. The protocol admits three parameter sets called KyberKEM-512, KyberKEM-768, and KyberKEM-1024. Our discussion will mainly focus on the underlying KyberPKE encryption scheme that is the basis of the construction of the KyberKEM protocol using the Fujisaki–Okamoto (FO) transform previously discussed in Subsection 2.3.4.

The public-key encryption scheme KyberPKE is similar to the one in [LPR10] with the main difference being the use of Module-LWE instead of Ring-LWE. Fixing $R = \mathbb{Z}[X]/(X^n + 1)$, the encrypted message is of a fixed length of $n = 256$ bits and the corresponding R-module is R_q^d with $d \in \{2, 3, 4\}$. The multiplication in R_q is performed using the Number Theoretic Transform (NTT) (see [CT65, LN16, Sei18] for further details). In order to implement the NTT efficiently, the modulus q should be a prime number that satisfies $q = 1 \bmod n$. For the choice $n = 256$, the smallest primes satisfying the previous condition are $q = 257$, $q = 769$ and $q = 3329$. The first two values are too small to guarantee the negligible failure probability required for CCA security for KyberKEM, and therefore the authors of [A+20] chose $q = 3329$.

A brief illustration of the protocol is given in Table 4.1. The noise vectors in KyberPKE are sampled from the centered binomial distribution ψ_k defined in Subsection 2.4.2 for $k \in \{2, 3\}$ that depends on the exponent of the R-module R_q^d. The extension of ψ_k from $\mathbb{Z}$ to R is denoted by Ψ_k. This choice of error distribution does not significantly decrease security of the scheme compared to a rounded Gaussian distribution, for the reason that those two distributions are close with respect to Rényi divergence (see Subsection 2.4.4). The use of the centered binomial distribution allows a much faster implementation compared to sampling from a discrete Gaussian distribution, which requires a significant algorithmic effort (see [BCNS15, DCRVV15, RVM+14]). The definition below is necessary in the following.

Definition 4.3 (Compression and Decompression functions)**.** Let b, q be positive integers such that $b < \lceil \log q \rceil$. The *compression* function $\texttt{Compress}_q : \mathbb{Z}_q \to \mathbb{Z}_{2^b}$ and the *decompression* function $\texttt{Decompress}_q : \mathbb{Z}_{2^b} \to \mathbb{Z}_q$ are defined as:

$$\texttt{Compress}_q(x, b) = \left\lfloor \frac{2^b}{q} x \right\rceil \bmod 2^b \quad \text{and} \quad \texttt{Decompress}_q(x, b) = \left\lfloor \frac{q}{2^b} x \right\rceil$$

Note that these functions can be extended to vectors by applying them component-wise.

Parameters: $n = 256$; $q = 3329$

$(k_1, k_2, d, b_{\vec{\mathbf{u}}}, b_{\mathbf{v}}) \in \{(3,2,2,10,4),(2,2,3,10,4),(2,2,4,11,5)\}$

Alice (server)		**Bob (Client)**
$\mathbf{A} \xleftarrow{\$} R_q^{d\times d}$		
$\vec{\mathbf{s}}, \vec{\mathbf{e}} \leftarrow \Psi_{k_1}^d$		$\vec{\mathbf{s}'} \leftarrow \Psi_{k_1}^d, \vec{\mathbf{e}'} \leftarrow \Psi_{k_2}^d$
		$\mathbf{e}'' \leftarrow \Psi_{k_2}$
$\vec{\mathbf{b}} := \mathbf{A}\vec{\mathbf{s}} + \vec{\mathbf{e}} \in R_q^d$	$\xrightarrow{(\mathbf{A}, \vec{\mathbf{b}})}$	$\mathbf{m} \xleftarrow{\$} \{0,1\}^{256}$
		$\vec{\mathbf{u}} := \mathbf{A}^T\vec{\mathbf{s}'} + \vec{\mathbf{e}'}$
$\vec{\hat{\mathbf{u}}} = \mathtt{Decompress}_q\left(\mathtt{Compress}_q\left(\vec{\mathbf{u}}, b_{\vec{\mathbf{u}}}\right), b_{\vec{\mathbf{u}}}\right)$	$\xleftarrow[\mathtt{Compress}_q(\mathbf{v}, b_{\mathbf{v}})]{\mathtt{Compress}_q(\vec{\mathbf{u}}, b_{\vec{\mathbf{u}}})}$	$\mathbf{v} := \langle \vec{\mathbf{b}}, \vec{\mathbf{s}'} \rangle + \mathbf{e}'' + \lfloor \frac{q}{2} \rceil \mathbf{m}$
$\hat{\mathbf{v}} = \mathtt{Decompress}_q\left(\mathtt{Compress}_q\left(\mathbf{v}, b_{\mathbf{v}}\right)\right)$		
$\mathbf{v}' := \langle \vec{\mathbf{s}}, \vec{\hat{\mathbf{u}}} \rangle \in R_q$		
$\mathbf{m}' = \mathtt{Compress}_q\left(\hat{\mathbf{v}} - \mathbf{v}', 1\right)$		

Table 4.1: The KyberPKE protocol.

As we can see, additional parameters appear in the algorithm settings. For instance, k_1 and k_2 refer to the centered binomial width, while $b_{\vec{\mathbf{u}}}$ and $b_{\mathbf{v}}$ are inputs to the compression and decompression functions in Definition 4.3 above. Those parameters are chosen to balance between security, ciphertext size, and failure probability, whereas the compression and decompression functions aim to reduce the communication requirements between Alice and Bob, while still guaranteeing a very small decryption failure probability. Using a similar notation as in Subsection 2.3.1, the algorithm proceeds as follows:

- Public parameter: Alice samples a uniform random matrix $\mathbf{A}$ from $R_q^{d\times d}$ which is referred to as the public parameter pp of the PKE.
- $(\vec{\mathbf{s}}, \vec{\mathbf{b}}) \leftarrow \mathsf{Gen}(\mathbf{A})$: Alice chooses $\vec{\mathbf{e}}, \vec{\mathbf{s}} \leftarrow \Psi_{k_1}^d$, computes $\vec{\mathbf{b}} = \mathbf{A}\vec{\mathbf{s}} + \vec{\mathbf{e}}$, and outputs a public key $pk = \vec{\mathbf{b}}$ and a secret key $sk = \vec{\mathbf{s}}$
- $(\vec{\mathbf{u}}, \mathbf{v}) \leftarrow \mathsf{Enc}(\mathbf{A}, \vec{\mathbf{b}}, \mathbf{m})$: Bob chooses independent $\vec{\mathbf{s}'} \leftarrow \Psi_{k_1}^d, \vec{\mathbf{e}'} \leftarrow \Psi_{k_2}^d$ and $\mathbf{e}'' \leftarrow \Psi_{k_2}$, then computes $\vec{\mathbf{u}} = \mathbf{A}^T\vec{\mathbf{s}'} + \vec{\mathbf{e}'} \in R_q^d$ and compresses it using the $\mathtt{Compress}_q$ function described in Definition 4.3. After generating the message $\mathbf{m}$ that needs to be encrypted, Bob computes $\mathbf{v} = \langle \vec{\mathbf{b}}, \vec{\mathbf{s}'} \rangle + \mathbf{e}'' + \lfloor \frac{q}{2} \rceil \mathbf{m}$ and compresses it as well. He outputs $c = (\vec{\mathbf{u}}, \mathbf{v})$ and forwards the compressed terms to Alice.
- $\mathbf{m} \leftarrow \mathsf{Dec}(\vec{\mathbf{s}}, c)$: Alice decompresses these terms into $\vec{\hat{\mathbf{u}}}$ and $\hat{\mathbf{v}}$ respectively, then computes $\mathbf{v}' = \langle \vec{\mathbf{s}}, \vec{\hat{\mathbf{u}}} \rangle$. She recovers the message $\mathbf{m}$ by applying $\mathtt{Compress}_q(\hat{\mathbf{v}} - \mathbf{v}', 1)$.

4.3.2 Reliability

This section provides a proof of the δ-correctness for KyberPKE following [BDK+18], with an exponentially small decryption failure probability δ, depending on the selected parameters. Table 4.2 contains the δ values for the three levels of KyberKEM. An important lemma for the analysis of the proof is given below.

Lemma 4.1 ([N+20])**.** Let $\bmod^{\pm}$ denote the operation defined in Section 2.1, i.e., that produces values in the range that goes from the negative half excluded, to the positive half included. The compression and decompression functions in Definition 4.3 satisfy:

$$\texttt{Decompress}_q\left(\texttt{Compress}_q\left(x,b\right),b\right)=x'$$

with $|x-x' \bmod^{\pm} q| \leq \lfloor q/2^{b+1} \rceil$.

Theorem 4.1. Let d be positive integer and $\vec{\mathbf{s}}, \vec{\mathbf{e}}, \vec{\mathbf{s'}}, \vec{\mathbf{e'}}, \mathbf{e''}$ be distributed as in Table 4.1. Let $\vec{\mathbf{c}_1}$ and $\mathbf{c}_2$ be distributed according to the following distribution:

1. Chose uniformly random $\vec{\mathbf{y}} \leftarrow R^d$ (or $\mathbf{y} \leftarrow R$ in case of $\mathbf{c}_2$),
2. **return** $\left(\vec{\mathbf{y}} - \texttt{Decompress}_q\left(\texttt{Compress}_q\left(\vec{\mathbf{y}},b\right),b\right)\right) \bmod^{\pm} q$.

Then KyberKEM is δ-correct where

$$\delta = \mathbb{P}\left\{\left|\left|\langle \vec{\mathbf{e}}, \vec{\mathbf{s'}}\rangle + \mathbf{e''} + \mathbf{c}_2 - \langle \vec{\mathbf{s}}, \vec{\mathbf{e'}}\rangle - \langle \vec{\mathbf{s}}, \vec{\mathbf{c}_1}\rangle\right|\right|_{\infty} > \lfloor q/4 \rceil\right\}$$

Proof. During the decryption procedure, the vector $\vec{\hat{\mathbf{u}}}$ in Table 4.1 is recovered by applying

$$\texttt{Decompress}_q\left(\texttt{Compress}_q\left(\vec{\mathbf{u}}, b_{\vec{\mathbf{u}}}\right), b_{\vec{\mathbf{u}}}\right)$$

which leads to write

$$\vec{\hat{\mathbf{u}}} = \mathbf{A}^T\vec{\mathbf{s'}} + \vec{\mathbf{e'}} + \vec{\mathbf{c}_1}$$

for some $\vec{\mathbf{c}_1} \in R^d$. Furthermore,

$$\begin{aligned}\hat{\mathbf{v}} &= \texttt{Decompress}_q\left(\texttt{Compress}_q\left(\langle \vec{\mathbf{b}}, \vec{\mathbf{s'}}\rangle + \mathbf{e''} + \lfloor q/2 \rceil \cdot \mathbf{m}, b_{\mathbf{v}}\right), b_{\mathbf{v}}\right)\\ &= \langle \vec{\mathbf{b}}, \vec{\mathbf{s'}}\rangle + \mathbf{e''} + \lfloor q/2 \rceil \cdot \mathbf{m} + \mathbf{c}_2\\ &= \langle \mathbf{A}\,\vec{\mathbf{s}} + \vec{\mathbf{e}}, \vec{\mathbf{s'}}\rangle + \mathbf{e''} + \lfloor q/2 \rceil \cdot \mathbf{m} + \mathbf{c}_2\end{aligned}$$

for some $\mathbf{c}_2 \in R$. Note that the distribution of the terms $\vec{\mathbf{c}_1}$ and $\mathbf{c}_2$ is indistinguishable from the distribution in the theorem statement. This is true because $\vec{\mathbf{c}_1}$ and $\mathbf{c}_2$ are of the form $\left(\mathbf{y} - \texttt{Decompress}_q\left(\texttt{Compress}_q\left(\mathbf{y},b\right),b\right)\right) \bmod^{\pm} q$ where $\mathbf{y}$ is pseudo-random based on the hard-

ness of Module-LWE. Using the above we get:

$$\hat{\mathbf{v}} - \langle \vec{\mathbf{s}}, \vec{\hat{\mathbf{u}}} \rangle = \underbrace{\langle \vec{\mathbf{e}}, \vec{\mathbf{s}'} \rangle + \mathbf{e}'' + \mathbf{c}_2 - \langle \vec{\mathbf{s}}, \vec{\mathbf{e}'} \rangle - \langle \vec{\mathbf{s}}, \vec{\mathbf{c}_1} \rangle}_{\boldsymbol{\omega}} + \lfloor q/2 \rceil \cdot \mathbf{m}$$

On the other hand, $\mathbf{m}' = \texttt{Compress}_q\left(\hat{\mathbf{v}} - \langle \vec{\mathbf{s}}, \vec{\hat{\mathbf{u}}} \rangle, 1\right)$. So decompressing from both sides we obtain

$$\underbrace{\texttt{Decompress}_q\left(\mathbf{m}', 1\right)}_{= \lfloor q/2 \rceil \mathbf{m}'} = \texttt{Decompress}_q\left(\texttt{Compress}_q\left(\hat{\mathbf{v}} - \langle \vec{\mathbf{s}}, \vec{\hat{\mathbf{u}}} \rangle, 1\right), 1\right)$$

which gives using Lemma 4.1

$$\left\| \lfloor q/2 \rceil \mathbf{m}' - \left(\hat{\mathbf{v}} - \langle \vec{\mathbf{s}}, \vec{\hat{\mathbf{u}}} \rangle\right) \right\|_\infty \leq \lfloor q/2^2 \rceil,$$

or equivalently,

$$\|\boldsymbol{\omega} - \lfloor q/2 \rceil (\mathbf{m}' - \mathbf{m})\|_\infty \leq \lfloor q/4 \rceil.$$

Suppose that $\|\boldsymbol{\omega}\|_\infty \leq \lfloor q/4 \rceil$ with probability $1 - \delta$. The triangular inequality yields

$$\|\lfloor q/2 \rceil (\mathbf{m} - \mathbf{m}')\|_\infty < 2 \lfloor q/4 \rceil.$$

Since q is odd, this implies that $\mathbf{m} = \mathbf{m}'$ and hence the δ-correctness of the scheme. □

The value of δ that corresponds to the probability that $\|\boldsymbol{\omega}\|_\infty > \lfloor q/4 \rceil$ is calculated via a Python script as indicated in [BDK+18]. Table 4.2 illustrates the different values of δ for the three different KyberKEM levels.

	n	d	q	k_1	k_2	$(b_{\vec{\mathbf{u}}}, b_{\mathbf{v}})$	δ
KyberKEM-512	256	2	3329	3	2	$(10, 4)$	2^{-139}
KyberKEM-768	256	3	3329	2	2	$(10, 4)$	2^{-164}
KyberKEM-1024	256	4	3329	2	2	$(11, 5)$	2^{-174}

Table 4.2: Error probability for different parameter sets for KyberKEM.

4.3.3 Security

The security of KyberKEM depends on the security of KyberPKE. In order to show IND-CCA security for the former, IND-CPA security must be demonstrated for the latter.

IND-CPA security. IND-CPA security (see Subsection 2.3.3) is established for KyberPKE if a passive adversary is not able to distinguish between the algorithm in Table 4.1 and the algorithm KyberPKE' in which the vector $\vec{\mathbf{b}}$ together with $\vec{\mathbf{u}}$ and $\mathbf{v}$ are uniformly random. By observing that the terms $\vec{\mathbf{b}}$, $\vec{\mathbf{u}}$ and $\mathbf{v}$ of the original algorithm are in fact M-LWE samples,

they are indistinguishable from uniform random variables thanks to the hardness of the decision Module-LWE problem.

The corresponding advantage is given in [BDK+18, Theorem 2] and reformulated as follows: For any IND-CPA adversary $\mathcal{A}$, there exists a M-LWE adversary $\mathcal{B}$ such that

$$\mathsf{Adv}^{\text{IND-CPA}}_{\text{KyberPKE}}(\mathcal{A}) \leq \mathsf{Adv}^{M\text{-LWE}}_{d+1,d,\Psi_{k_1}}(\mathcal{B}) + \mathsf{Adv}^{M\text{-LWE}}_{d+1,d,\Psi_{k_2}}(\mathcal{B})$$

where the notation $\mathsf{Adv}^{M\text{-LWE}}_{d+1,d,\Psi_k}$ indicates the advantage in Definition 4.2 with respect to the centered binomial distribution Ψ_k.

IND-CCA security. Applying the Fujisaki-Okamoto transform from Subsection 2.3.4 to KyberPKE allows to obtain an IND-CCA secure key encapsulation mechanism called KyberKEM. By using three hash functions as random oracles, one can derive bounds regarding classical and quantum advantage. Let $G_1 : \{0,1\}^* \mapsto \{0,1\}^{256}$, $G_2 : \{0,1\}^* \mapsto \{0,1\}^{2\times 256}$ and $F : \{0,1\}^* \mapsto \{0,1\}^*$ initiated with SHAKE-256, be those three hash functions. Theorem 4.2 below bounds the classical advantage.

Theorem 4.2 ([BDK+18])**.** For any classical adversary $\mathcal{A}$ that makes at most q_{RO} many queries to G_1 and G_2, there exists an adversary $\mathcal{B}$ such that

$$\mathsf{Adv}^{\text{IND-CCA}}_{\text{KyberKEM}}(\mathcal{A}) \leq 3\mathsf{Adv}^{\text{IND-CPA}}_{\text{KyberPKE}}(\mathcal{B}) + q_{\text{RO}}\delta + \frac{3q_{\text{RO}}}{2^{256}} \tag{4.1}$$

where δ is the decryption failure probability of KyberPKE.

A security reduction can also be derived in the quantum random oracle model (see [A+20, Subsection 4.3.2]) but it's not tight and therefore can only serve as an asymptotic indication of KyberKEM's CCA-security in the QROM.

Concrete security. The concrete security of KyberKEM is analysed similarly to [ADPS16b] by treating the M-LWE problem as a general LWE problem without taking the algebraic structure into account, since at present no attacks that exploit this structure are known. The best known attacks are the primal and dual attacks discussed in Section 3.2. Table 4.3 shows the concrete security bounds in $\log_2$ scale as an example for the most well-known classical, quantum and plausible attacks.

4.4 Kyber with reconciliation

In this section we consider a key encapsulation mechanism based on Module-LWE using the reconciliation technique discussed in Section 1.2. Our protocol mainly focuses on modifying the three levels of KyberKEM, namely, KyberKEM-512, KyberKEM-768 and KyberKEM-1024, with the aim of improving their security level thanks to a better error correction.

Attack	Known Classical	Known Quantum	Best Plausible
KyberKEM-512: $q = 3329$, $n = 256$, $k_1 = 3, k_2 = 2$, $d = 2$			
Primal	118	107	84
Dual	118	107	83
KyberKEM-768: $q = 3329$, $n = 256$, $k = 2$, $d = 3$			
Primal	182	165	129
Dual	181	164	128
KyberKEM-1024: $q = 3329$, $n = 256$, $k = 2$, $d = 4$			
Primal	256	232	181
Dual	253	230	180

Table 4.3: Security bounds against know primal and dual attacks in $\log_2$ scale [A+20].

We consider the cyclotomic ring of degree $n = 256$ as in KyberKEM. We use an 8-dimensional lattice for reconciliation in order to generate one bit of private key per dimension. As in Chapter 3, the chosen lattice is the Gosset lattice E_8 due to its optimal density and low-complexity quantization. We will show that our scheme can guarantee smaller error probability, smaller modulus and better security compared to the original first two levels at the price of a slightly increased bandwidth.

In our modified version, we choose the modulus to be a power-of-two. In fact, by choosing q to be a prime number as in KyberKEM, one can use the NTT to speed up polynomial multiplication [PG13, Pei14, ADPS16b]. However, prime q is not required for security since the hardness of Module-LWE has been established for general modulus q (see discussion in Section 4.2), and power-of-two moduli have been used in the literature in [DKRV18, LLZ+18, N+20]. These works use other methods for efficient polynomial multiplication, e.g. Karatsuba/Toom-Cook algorithms [BCLV17, DWZ18] and index-based multiplication [BBGM+17]. An advantage of choosing an even q is that a dither is not required to obtain a uniform key with the reconciliation method, unlike [Pei14, ADPS16b].

4.4.1 Polynomial splitting

Before presenting the details of our reconciliation mechanism, we need to consider what is called *polynomial splitting* as in [ADPS16b, Section C]. This will be used to define our protocol as well as to analyze our error probability bound in later section.

Given an integer n_0 such that $n = n_0 L$, we take $S = \mathbb{Z}[Y]/(Y^{n_0} + 1)$. Given a polynomial

$\mathbf{a}(X) \in R$, we define for $\kappa = 0, 1, \ldots, L-1$ the vectors $\mathbf{a}^{(\kappa)}$ as:

$$\begin{aligned}
\mathbf{a}^{(0)} &= (a_0, a_L, a_{2L}, \ldots, a_{n-L}) \\
\mathbf{a}^{(1)} &= (a_1, a_{1+L}, a_{1+2L}, \ldots, a_{1+n-L}) \\
\mathbf{a}^{(2)} &= (a_2, a_{2+L}, a_{2+2L}, \ldots, a_{2+n-L}) \\
&\vdots \\
\mathbf{a}^{(L-1)} &= (a_{L-1}, a_{2L-1}, a_{3L-1}, \ldots, a_{n-1})
\end{aligned}$$

Taking $Y = X^L$ allows to write $\mathbf{a}(X) = \sum_{\kappa=0}^{L-1} \mathbf{a}^{(\kappa)}(Y) X^\kappa$. Accordingly, given two polynomials $\mathbf{a}(X)$ and $\mathbf{b}(X)$, one can express the multiplication $\mathbf{a}(X)\mathbf{b}(X)$ as $\mathbf{p}(X) := \mathbf{a}(X)\mathbf{b}(X) = \sum_{\kappa=0}^{L-1} \mathbf{p}^{(\kappa)}(Y) X^\kappa$ in which

$$\mathbf{p}^{(\kappa)}(Y) = \sum_{i=0}^{L-1} Y^{\delta_{i,\kappa}}\, \mathbf{a}^{(i)}(Y) \cdot \mathbf{b}^{(\kappa - i \bmod L)}(Y), \tag{4.2}$$

where $\delta_{i,\kappa}$ is either 0 or 1. In the sequel we omit the $\bmod L$ operation to simplify notations.

4.4.2 Reconciliation mechanism for key generation

We present here our modified key encapsulation mechanism by adding a reconciliation step in order to improve the reliability of key agreement between Alice and Bob (see discussion in Section 1.2).

We show that our scheme can guarantee a smaller error probability than both KyberKEM-512 and KyberKEM-768, with a smaller modulus $q = 2^{11}$, using 4 bits of reconciliation per dimension. For this choice of q, the post-quantum security is enhanced.

Parameters: $q = 2^{11}$; $n = 256$; $k_1, k_2 \in \{2, 3\}$; $d \in \{2, 3, 4\}$		
Alice (server)		**Bob (Client)**
$\mathbf{A} \xleftarrow{\$} R_q^{d \times d}$		
$\vec{\mathbf{s}}, \vec{\mathbf{e}} \leftarrow \Psi_{k_1}^d$		$\vec{\mathbf{s}'} \leftarrow \Psi_{k_1}^d, \vec{\mathbf{e}'} \leftarrow \Psi_{k_2}^d, \mathbf{e}'' \leftarrow \Psi_{k_2}$
$\vec{\mathbf{b}} := \mathbf{A}\vec{\mathbf{s}} + \vec{\mathbf{e}} \in R_q^d$	$\xrightarrow{(\mathbf{A}, \vec{\mathbf{b}})}$	
		$\vec{\mathbf{u}} := \mathbf{A}^T \vec{\mathbf{s}'} + \vec{\mathbf{e}'} \in R_q^d$
		$\mathbf{v} := \langle \vec{\mathbf{b}}, \vec{\mathbf{s}'} \rangle + \mathbf{e}'' \in R_q$
	$\xleftarrow{(\vec{\mathbf{u}}, \mathbf{r})}$	$\mathbf{r} = \textsc{HelpRec}(\mathbf{v}) = Q_{\Lambda_1}(\mathbf{v}) \bmod \Lambda_2$
$\mathbf{v}' := \langle \vec{\mathbf{u}}, \vec{\mathbf{s}} \rangle \in R_q$		
$\hat{\mathbf{k}} = \textsc{Rec}(\mathbf{v}', \mathbf{r}) = Q_{\Lambda_2}(\mathbf{v}' - \mathbf{r}) \bmod \Lambda_3$		$\mathbf{k} = \textsc{Rec}(\mathbf{v}, \mathbf{r}) = Q_{\Lambda_2}(\mathbf{v} - \mathbf{r}) \bmod \Lambda_3$

Table 4.4: Proposed key encapsulation mechanism with reconciliation, based on Module-LWE.

Our protocol makes use of the following lattices of dimension $n = 256$: the quantization

lattice $\Lambda_1 = \left(\frac{q}{2^p}E_8\right)^{32}$ for some integer $p \geq 1$, the coding lattice $\Lambda_2 = \left(\frac{q}{2}E_8\right)^{32}$ and the shaping lattice $\Lambda_3 = q\left(\mathbb{Z}^8\right)^{32}$. The purpose of the lattice Λ_1 is to quantize the reconciliation message in order to reduce the transmission requirements from Bob to Alice. The lattice Λ_2 is used for error correction, and the choice of the lattice Λ_3 corresponds to the fact that all operations in the protocol are performed modulo q. Since $2\mathbb{Z}^8 \subseteq E_8 \subseteq \frac{1}{2^{p-1}}E_8$, this choice implies that $\Lambda_3 \subseteq \Lambda_2 \subseteq \Lambda_1$.

Referring to Table 4.4, our protocol is the same as KyberPKE (Table 4.1) with regard to the generation of the public and secret keys, the generation of the error terms, as well as the calculation of $\mathbf{v}$ and the vector $\vec{\mathbf{u}}$. Since encryption is replaced by reconciliation, the message $\mathbf{m}$ (which in Table 4.1 was generated unilaterally at Bob's side) is now replaced by a private key $\mathbf{k}$. Note that we eliminate the use of the compression and decompression functions. We could have included a compression step, but decided against it because computing the decryption failure probability would have been too complicated.

Bob sends to Alice the pair $c = (\vec{\mathbf{u}}, \mathbf{r}) \in R_q^d \times \Lambda_1/\Lambda_2$ with

$$\mathbf{r} = \textsc{HelpRec}(\mathbf{v}) := Q_{\Lambda_1}(\mathbf{v}) \bmod \Lambda_2$$

being the reconciliation message. The private key $\mathbf{k}$ in Λ_2/Λ_3 is such that

$$\mathbf{k} = \textsc{Rec}(\mathbf{v}, \mathbf{r}) := Q_{\Lambda_2}(\mathbf{v} - \mathbf{r}) \bmod \Lambda_3.$$

The pair $((\vec{\mathbf{u}}, \mathbf{r}), \mathbf{k})$ defines the output of the encapsulation function $\mathsf{Encaps}(\mathbf{A}, \vec{\mathbf{b}})$. Alice performs the decapsulation procedure $\mathsf{Decaps}(\vec{\mathbf{s}}, (\vec{\mathbf{u}}, \mathbf{r}))$ by computing $\mathbf{v}' = \langle \vec{\mathbf{u}}, \vec{\mathbf{s}} \rangle$ and returning $\hat{\mathbf{k}} = \textsc{Rec}(\mathbf{v}', \mathbf{r})$.

Remark 4.1. When $\mathbf{v} \in R_q$, the notation $Q_{\Lambda_1}(\mathbf{v})$ means that we perform $Q_{\frac{q}{2^p}E_8}$ on each component $\mathbf{v}^{(\kappa)}$ (see Subsection 4.4.3) where $n_0 = 8$ and $L = 32$. Similarly for $Q_{\Lambda_2}(\cdot)$, $\bmod \Lambda_2$ and $\bmod \Lambda_3$ operations.

The number of bits of the private key is calculated as

$$\log|\Lambda_2/\Lambda_3| = \log\left(\frac{\mathrm{Vol}(\Lambda_3)}{\mathrm{Vol}(\Lambda_2)}\right) \overset{\mathrm{Vol}(E_8)=1}{=} \log\left(\frac{q^{256}}{(q/2)^{256}}\right) = 256,$$

so that the protocol provides 256 bits of key. Furthermore, the number of reconciliation bits transmitted from Bob to Alice is

$$\log|\Lambda_1/\Lambda_2| = \log\left(\frac{\mathrm{Vol}(\Lambda_2)}{\mathrm{Vol}(\Lambda_1)}\right) = \log\left(\frac{(q/2)^{256}}{(q/2^p)^{256}}\right) = 256(p-1).$$

Remark 4.2 (Comparison with NewHope [ADPS16b])**.** We remark that our protocol has a similar form to the reconciliation-based protocols in [Pei14] and the first version of the NewHope

protocol [ADPS16b]. For instance, in [ADPS16b] the functions HelpRec and Rec can be written as the above form by considering the product lattices $\Lambda_1 = (q\tilde{D}_4/2^p)^{256}$, $\Lambda_2 = (q\tilde{D}_4)^{256}$ and $\Lambda_3 = q\mathbb{Z}^{1024}$. We point out that a dither is not required in our algorithm unlike in [Pei14, ADPS16b] since the modulus q is an even number. Unlike NewHope, the proposed protocol is based on Module-LWE and uses the same parameters n, d, k as in KyberKEM. In order to obtain 256 bits of key with $n = 256$, a higher-dimensional lattice than $\tilde{D}_4$ is needed, which is why we choose the E_8 lattice.

Bandwidth requirements. Focusing only on the data sent by Bob to Alice in KyberKEM-512, our protocol for $n = 256, d = 2, q = 2^{11}$ and $p = 5$ requires to send a pair $(\vec{\mathbf{u}}, \mathbf{r})$ where $\vec{\mathbf{u}}$ is an element of R_q^d (requiring $11 \times 256 \times 2 = 5632$ bits) and $\mathbf{r} \in \Lambda_1/\Lambda_2$ requires $(p-1) \times 256 = 1024$ bits. In total, 6656 bits are needed.

Besides, the two terms $\texttt{Compress}_q(\vec{\mathbf{u}}, b_{\vec{\mathbf{u}}})$ and $\texttt{Compress}_q(\mathbf{v}, b_{\mathbf{v}})$ are sent in the original KyberKEM-512, where $b_{\vec{\mathbf{u}}} = 10$ for which $\texttt{Compress}_q(\vec{\mathbf{u}}, b_{\vec{\mathbf{u}}})$ requires 512×10 bits, and $b_{\mathbf{v}} = 4$ for which $\texttt{Compress}_q(\mathbf{v}, b_{\mathbf{v}})$ requires 256×4 bits. In total, 6144 bits are needed. Consequently, our protocol requires slightly more bandwidth (a 8.3% increase) than KyberKEM-512. The same calculations for $d = 3$ and $d = 4$ lead to the following table:

	n	d	p	$(b_{\vec{\mathbf{u}}}, b_{\mathbf{v}})$	Original Bandwidth	Modified Bandwidth	Increment
KyberKEM-512	256	2	5	$(10, 4)$	6144	6656	8.3%
KyberKEM-512	256	2	6	$(10, 4)$	6144	6912	12.5%
KyberKEM-768	256	3	5	$(10, 4)$	8704	9472	8.8%
KyberKEM-1024	256	4	7	$(11, 5)$	12544	12800	2%

Table 4.5: Bandwidth requirements (in bits) for our modification of the three levels of KyberKEM. The choice of p is affected by the decryption failure probability that will be discussed in the next section.

Cost of multiplication in the ring R. Our protocols relies heavily on polynomial arithmetic in the ring R_q with modulus $q = 2^{13}$, and polynomial multiplication is a costly operation. As mentioned, since q is not prime, we are replacing the NTT algorithm with Karatsuba/Toom-Cook methods. Asymptotically, NTT polynomial multiplication has complexity of $O(n \log n)$, while Karatsuba polynomial multiplication has a higher asymptotic complexity of $O(n^{1.58})$. Though we lose in asymptotic time complexity, we gain in modular arithmetic since modular reduction comes for free. The complexity comparison of NTT versus Toom-Cook is heavily dependent on hardware implementation, see for example Table 2 in [DKRV18]. For instance, when using `C` language, the costs of Toom-Cook and NTT are comparable, while with an `optimized AVX2` implementation, NTT is much more efficient.

4.4.3 Reliability

In this section, we provide technical details on estimating the error probability $P_e = \mathbb{P}\{\mathbf{k} \neq \hat{\mathbf{k}}\}$. We will prove that in our described protocol, the parameter set we recommend in Table 4.4 yields $P_e < 2^{-141}$ for KyberKEM-512 (with $p = 6$), $P_e < 2^{-174}$ for KyberKEM-768 (with $p = 5$) and $P_e < 2^{-168}$ for KyberKEM-1024 (with a very large p).

Reliability condition According to Table 4.4, the two private keys $\mathbf{k}$ and $\hat{\mathbf{k}}$ are identical whenever

$$Q_{\Lambda_2}(\mathbf{v} - \mathbf{r}) \bmod \Lambda_3 = Q_{\Lambda_2}(\mathbf{v}' - \mathbf{r}) \bmod \Lambda_3. \tag{4.3}$$

Setting $\mathbf{e}_Q = \mathbf{v} - Q_{\Lambda_1}(\mathbf{v})$ we can show that a sufficient condition to achieve equation (4.3) is $Q_{\Lambda_2}((\mathbf{v}' - \mathbf{v}) + \mathbf{e}_Q) = 0$. In fact, using properties (P1), (P2), (P3) or (P4) from Lemma 2.1, we can write

$$\begin{aligned} Q_{\Lambda_2}(\mathbf{v} - \mathbf{r}) &= Q_{\Lambda_2}(\mathbf{e}_Q + Q_{\Lambda_1}(\mathbf{v}) - Q_{\Lambda_1}(\mathbf{v}) + Q_{\Lambda_2}(Q_{\Lambda_1}(\mathbf{v}))) \\ &\overset{(P3)}{=} Q_{\Lambda_2}(\mathbf{e}_Q) + Q_{\Lambda_2}(Q_{\Lambda_1}(\mathbf{v})) \\ &= Q_{\Lambda_2}(Q_{\Lambda_1}(\mathbf{v})), \text{ because } \mathbf{e}_Q \in \mathcal{V}(\Lambda_1) \subseteq \mathcal{V}(\Lambda_2); \end{aligned}$$

and

$$\begin{aligned} Q_{\Lambda_2}(\mathbf{v}' - \mathbf{r}) &= Q_{\Lambda_2}(\mathbf{v}' - Q_{\Lambda_1}(\mathbf{v}) + Q_{\Lambda_2}(Q_{\Lambda_1}(\mathbf{v}))) \\ &\overset{(P3)}{=} Q_{\Lambda_2}(\mathbf{v}' + \mathbf{e}_Q - \mathbf{v}) + Q_{\Lambda_2}(Q_{\Lambda_1}(\mathbf{v})). \end{aligned}$$

We recall that the operation $Q_{\Lambda_1}(\cdot)$ when applied on $\mathbf{x} \in \mathbb{R}^{256}$ is actually $Q_{\frac{q}{2^p}E_8}(\cdot)$ applied on each component $\mathbf{x}^{(\kappa)}$ (Remark 4.1). Similarly, the $Q_{\Lambda_2}(\cdot)$, $\bmod \Lambda_2$ and $\bmod \Lambda_3$ operations can be applied component-wise.

The condition $Q_{\Lambda_2}((\mathbf{v}' - \mathbf{v}) + \mathbf{e}_Q) = 0$ is realized whenever

$$Q_{\frac{q}{2}\left(1 - \frac{1}{2^{p-1}}\right)E_8}\left((\mathbf{v} - \mathbf{v}')^{(\kappa)}\right) = 0 \text{ for } \kappa = 0, \ldots, L-1. \tag{4.4}$$

In fact,

$$\begin{aligned}
&\forall \kappa = 0, \ldots, L-1,\ Q_{\frac{q}{2}\left(1-\frac{1}{2^{p-1}}\right)E_8}\left((\mathbf{v}-\mathbf{v}')^{(\kappa)}\right) = 0 \\
\implies &\forall \kappa = 0, \ldots, L-1,\ (\mathbf{v}-\mathbf{v}')^{(\kappa)} \in \mathcal{V}\left(\tfrac{q}{2}\left(1-\tfrac{1}{2^{p-1}}\right)E_8\right) \\
\implies &\forall \kappa = 0, \ldots, L-1,\ (\mathbf{v}-\mathbf{v}')^{(\kappa)} - \mathbf{e}_Q^{(\kappa)} \in \mathcal{V}\left(\tfrac{q}{2}\left(1-\tfrac{1}{2^{p-1}}\right)E_8\right) + \mathcal{V}\left(\tfrac{q}{2^p}E_8\right) \subseteq \tfrac{q}{2}\mathcal{V}(E_8) \\
\implies &\forall \kappa = 0, \ldots, L-1,\ Q_{\frac{q}{2}E_8}\left((\mathbf{v}-\mathbf{v}')^{(\kappa)} - \mathbf{e}_Q^{(\kappa)}\right) = 0 \\
\iff &\forall \kappa = 0, \ldots, L-1,\ Q_{\frac{q}{2}E_8}\left((\mathbf{v}'-\mathbf{v})^{(\kappa)} + \mathbf{e}_Q^{(\kappa)}\right) = 0, \text{ because } \mathcal{V}\left(\frac{q}{2}E_8\right) \text{ has central symmetry} \\
\implies &Q_{\Lambda_2}\left((\mathbf{v}'-\mathbf{v}) + \mathbf{e}_Q\right) = 0.
\end{aligned}$$

For clearer notations, we assign $C := \frac{q}{2}\left(1-\frac{1}{2^{p-1}}\right)$. Let $\boldsymbol{\omega}$ denote the error difference between $\mathbf{v}$ and $\mathbf{v}'$ such that

$$\begin{aligned}
\boldsymbol{\omega} &= \mathbf{v} - \mathbf{v}' \\
&= \langle \vec{\mathbf{b}}, \vec{\mathbf{s}'} \rangle + \mathbf{e}'' - \langle \vec{\mathbf{u}}, \vec{\mathbf{s}} \rangle \\
&= \langle \mathbf{A}\vec{\mathbf{s}} + \vec{\mathbf{e}}, \vec{\mathbf{s}'} \rangle + \mathbf{e}'' - \langle \mathbf{A}^T\vec{\mathbf{s}'} + \vec{\mathbf{e}'}, \vec{\mathbf{s}} \rangle \\
&= \langle \vec{\mathbf{s}}, \mathbf{A}^T\vec{\mathbf{s}'} \rangle + \langle \vec{\mathbf{e}}, \vec{\mathbf{s}'} \rangle - \langle \mathbf{A}^T\vec{\mathbf{s}'}, \vec{\mathbf{s}} \rangle - \langle \vec{\mathbf{e}'}, \vec{\mathbf{s}} \rangle + \mathbf{e}'' \\
&= \langle \vec{\mathbf{e}}, \vec{\mathbf{s}'} \rangle - \langle \vec{\mathbf{e}'}, \vec{\mathbf{s}} \rangle + \mathbf{e}'' \\
&= (\mathbf{e}_1 \cdot \mathbf{s}'_1 + \cdots + \mathbf{e}_d \cdot \mathbf{s}'_d) - (\mathbf{e}'_1 \cdot \mathbf{s}_1 + \cdots + \mathbf{e}'_d \cdot \mathbf{s}_d) + \mathbf{e}'' \\
&= \boldsymbol{\omega}_1 + \boldsymbol{\omega}_2 + \cdots + \boldsymbol{\omega}_d + \mathbf{e}''
\end{aligned}$$

where $\boldsymbol{\omega}_i = \mathbf{e}_i \cdot \mathbf{s}'_i - \mathbf{e}'_i \cdot \mathbf{s}_i$ for $i = 1, \ldots, d$. This can be written in polynomial form as:

$$\boldsymbol{\omega}(X) = \boldsymbol{\omega}_1(X) + \boldsymbol{\omega}_2(X) + \cdots + \boldsymbol{\omega}_d(X) + \mathbf{e}''(X).$$

So for $\kappa = 0, \ldots, L-1$ and using equation (4.2), the expression of $\boldsymbol{\omega}^{(\kappa)}(Y)$ will be:

$$\boldsymbol{\omega}^{(\kappa)}(Y) = \sum_{j=1}^{d}\sum_{i=0}^{L-1} Y^{\delta_{i,\kappa}}\left(\mathbf{e}_j{}^{(i)} \cdot \mathbf{s}'^{(\kappa-i)}_j(Y) - \mathbf{e}'^{(i)}_j \cdot \mathbf{s}^{(\kappa-i)}_j(Y)\right) + \mathbf{e}''^{(\kappa)}(Y). \tag{4.5}$$

As in [ADPS16b, vP16], we will consider a union bound over all Voronoi relevant vectors. Note that $\boldsymbol{\omega}^{(\kappa)}(Y)$ is still a polynomial with 8 coefficients. From condition (4.4), decoding will be correct if $\boldsymbol{\omega}^{(\kappa)} \in C \cdot \mathcal{V}(E_8)$ for all $\kappa = 0, 1, 2, \ldots, L-1$. Referring to condition (2.2), $\boldsymbol{\omega}^{(\kappa)} \in C \cdot \mathcal{V}(E_8) \iff \langle \boldsymbol{\omega}^{(\kappa)}, \boldsymbol{\lambda} \rangle \leq \frac{\|\boldsymbol{\lambda}\|_2^2}{2}, \forall \boldsymbol{\lambda} \in C\left(\mathrm{VR}^{(1)}_{E_8} \cup \mathrm{VR}^{(2)}_{E_8}\right)$, where $\mathrm{VR}^{(1)}_{E_8}$ and $\mathrm{VR}^{(2)}_{E_8}$ are defined in Subsection 2.2.3.

We mention that multiplying a vector $\mathbf{a}^{(i)}(Y)$ by Y is equivalent to a right shift so that the

last term becomes the first one with a minus sign on it. In other words we can write

$$Y \cdot \mathbf{a}^{(i)}(Y) = Y \cdot \left(a_i + a_{i+L}Y + \cdots + a_{i+n-L}Y^{n_0-1}\right) = -a_{i+n-L} + a_iY + \cdots + a_{i+n-2L}Y^{n_0-1}.$$

Another thing to be mentioned is the conjugation function.

Definition 4.4 (Conjugation)**.** Let $\mathbf{a}(Y) = a_0 + \cdots + a_{n_0-1}Y^{n_0-1} \in S = \mathbb{Z}[Y]/(Y^{n_0}+1)$. We define the polynomial $\mathrm{conj}\,(\mathbf{a})\,(Y)$ as

$$\mathrm{conj}\,(\mathbf{a})\,(Y) = \mathbf{a}(\mathrm{conj}(Y)),$$

where $\mathrm{conj}(Y) = \mathrm{conj}\left(X^L\right) := \mathrm{conj}\left(\zeta^L\right)$ is the complex conjugation of the L-th power of the N-th primitive root of unity.

Consequently, one can express the dot product of $\boldsymbol{\omega}^{(\kappa)}$ with $\boldsymbol{\lambda} \in C\left(\mathrm{VR}_{E_8}^{(1)} \cup \mathrm{VR}_{E_8}^{(2)}\right)$ using the above definition and equation (4.5) as:

$$\begin{aligned}
\langle \boldsymbol{\omega}^{(\kappa)}, \boldsymbol{\lambda}\rangle &= \left\langle \sum_{j=1}^{d}\sum_{i=0}^{L-1} Y^{\delta_{i,\kappa}}\left(\mathbf{e}_j^{(i)}\mathbf{s}'^{(\kappa-i)}_j - \mathbf{e}'^{(i)}_j\mathbf{s}_j^{(\kappa-i)}\right), \boldsymbol{\lambda}\right\rangle + \langle \mathbf{e}''^{(\kappa)}, \boldsymbol{\lambda}\rangle \\
&= \sum_{j=1}^{d}\sum_{i=0}^{L-1}\left(\langle Y^{\delta_{i,\kappa}}\mathbf{e}_j^{(i)} \cdot \mathbf{s}'^{(\kappa-i)}_j, \boldsymbol{\lambda}\rangle - \langle Y^{\delta_{i,\kappa}}\mathbf{e}'^{(i)}_j \cdot \mathbf{s}_j^{(\kappa-i)}, \boldsymbol{\lambda}\rangle\right) + \langle \mathbf{e}''^{(\kappa)}, \boldsymbol{\lambda}\rangle \\
&\overset{(*)}{=} \sum_{j=1}^{d}\sum_{i=0}^{L-1}\left(\langle Y^{\delta_{i,\kappa}}\mathbf{s}'^{(i)}_j \cdot \mathbf{e}_j^{(\kappa-i)}, \boldsymbol{\lambda}\rangle - \langle Y^{\delta_{i,\kappa}}\mathbf{s}_j^{(i)} \cdot \mathbf{e}'^{(\kappa-i)}_j, \boldsymbol{\lambda}\rangle\right) + \langle \mathbf{e}''^{(\kappa)}, \boldsymbol{\lambda}\rangle \\
&\overset{(**)}{=} \sum_{j=1}^{d}\sum_{i=0}^{L-1}\left(\langle Y^{\delta_{i,\kappa}}\mathbf{s}'^{(i)}_j, \mathrm{conj}\left(\mathbf{e}_j^{(\kappa-i)}\right) \cdot \boldsymbol{\lambda}\rangle - \langle Y^{\delta_{i,\kappa}}\mathbf{s}_j^{(i)}, \mathrm{conj}\left(\mathbf{e}'^{(\kappa-i)}_j\right) \cdot \boldsymbol{\lambda}\rangle\right) + \langle \mathbf{e}''^{(\kappa)}, \boldsymbol{\lambda}\rangle
\end{aligned}$$

Equality $(*)$ is true due to $\bmod L$ operation discussed in Subsection 4.4.1, and $(**)$ is true according to Proposition C.1. Using this and the fact that the distributions of $\mathbf{e}_j^{(\kappa-i)}$ and $\mathbf{e}'_j{}^{(\kappa-i)}$ are invariant by $\mathrm{conj}(\cdot)$ (see Lemma C.1) and the distributions of $\mathbf{s}'_j{}^{(i)}$ and $\mathbf{s}_j{}^{(i)}$ are invariant by shifting or by multiplication by Y, we obtain a more compact form of $\langle \boldsymbol{\omega}^{(\kappa)}, \boldsymbol{\lambda}\rangle$:

$$\langle \boldsymbol{\omega}^{(\kappa)}, \boldsymbol{\lambda}\rangle \sim \left\langle (\tilde{\mathbf{s}}'_j, \tilde{\mathbf{s}}_j)_{j=1,\ldots,d}, W_{\boldsymbol{\lambda},\kappa}\right\rangle + \langle \mathbf{e}''^{(\kappa)}, \boldsymbol{\lambda}\rangle$$

where $\tilde{\mathbf{s}}'_j = (\mathbf{s}'^{(0)}_j, \ldots, \mathbf{s}'^{(L-1)}_j)$ as well as $\tilde{\mathbf{s}}_j = (\mathbf{s}_j^{(0)}, \ldots, \mathbf{s}_j^{(L-1)})$ are n-dimensional vectors of independent centered binomial coefficients, and

$$W_{\boldsymbol{\lambda},\kappa} = \left(\mathbf{e}_j^{(\kappa-i)} \cdot \boldsymbol{\lambda}, \ldots, \mathbf{e}'_j{}^{(\kappa-i)} \cdot \boldsymbol{\lambda}\right)_{\substack{i=0,\ldots,L-1,\\ j=1,\ldots,d}} \tag{4.6}$$

can be identified with $W_{\boldsymbol{\lambda},\kappa} \equiv \left(\mathbf{e}^{(0)} \cdot \boldsymbol{\lambda}, \ldots, \mathbf{e}^{(Ld-1)} \cdot \boldsymbol{\lambda}\,,\, \mathbf{e}'^{(0)} \cdot \boldsymbol{\lambda}, \ldots, \mathbf{e}'^{(Ld-1)} \cdot \boldsymbol{\lambda}\right)$, where each component $\mathbf{e}^{(i)} \cdot \boldsymbol{\lambda}$ (and $\mathbf{e}'^{(i)} \cdot \boldsymbol{\lambda}$) is a polynomial multiplication of an 8-dimensional vector $\mathbf{e}^{(i)}$ (and $\mathbf{e}'^{(i)}$) by a Voronoi relevant vector $\boldsymbol{\lambda}$. The multiplication is done modulo (Y^8+1). Moreover,

the $\mathbf{e}^{(i)}$s are independent with centered binomial coefficients, distributed also independently. For instance, if $\boldsymbol{\lambda}_1 \in C \cdot \mathrm{VR}_{E_8}^{(1)}$ is a Voronoi relevant vector of type 1, then $W_{\boldsymbol{\lambda}_1,\kappa}$ is given by the general form

$$C \cdot \left[\left(\pm e_{i_0}^{(0)} \pm e_{i_1}^{(0)}\right), \ldots, \left(\pm e_{i_{2Ld-2}}^{(Ld-1)} \pm e_{i_{2Ld-1}}^{(Ld-1)}\right),\ \left(\pm e'^{(0)}_{i_0} \pm e'^{(0)}_{i_1}\right), \ldots, \left(\pm e'^{(Ld-1)}_{i_{2Ld-2}} \pm e'^{(Ld-1)}_{i_{2Ld-1}}\right)\right].$$

However, if $\boldsymbol{\lambda}_2 \in C \cdot \mathrm{VR}_{E_8}^{(2)}$ is a Voronoi relevant vector of type 2, then $W_{\boldsymbol{\lambda}_2,\kappa}$ is of the form:

$$\frac{C}{2}\left[\left(\pm e_0^{(0)} \pm \cdots \pm e_7^{(0)}\right), \ldots, \left(\pm e_{8Ld-8}^{(Ld-1)} \pm \cdots \pm e_{8Ld-1}^{(Ld-1)}\right),\ \left(\pm e'^{(0)}_0 \pm \cdots \pm e'^{(0)}_7\right), \ldots, \left(\pm e'^{(Ld-1)}_{8Ld-8} \pm \cdots \pm e'^{(Ld-1)}_{8Ld-1}\right)\right].$$

Error probability calculations Recall that an error occurs if $\boldsymbol{\omega}^{(\kappa)} \notin C \cdot \mathcal{V}(E_8)$ for some $\kappa = 0, \ldots, L-1$. So one can bound the decryption failure probability P_e by

$$\mathbb{P}\left\{\exists \kappa \in \{0, \ldots, L-1\},\ \exists \boldsymbol{\lambda} \in C\left(\mathrm{VR}_{E_8}^{(1)} \cup \mathrm{VR}_{E_8}^{(2)}\right) \,:\, \langle \boldsymbol{\omega}^{(\kappa)}, \boldsymbol{\lambda}\rangle > \frac{\|\boldsymbol{\lambda}\|_2^2}{2}\right\}$$

Using the fact that $\langle \boldsymbol{\omega}^{(\kappa)}, \boldsymbol{\lambda}\rangle = \langle (\tilde{\mathbf{s}}', \tilde{\mathbf{s}}), W_{\boldsymbol{\lambda},\kappa}\rangle + \langle \mathbf{e}''^{(\kappa)}, \boldsymbol{\lambda}\rangle$ we obtain:

$$P_e \leq \sum_{\kappa=0}^{L-1} \mathbb{P}\left\{\exists \boldsymbol{\lambda} \,:\, \langle (\tilde{\mathbf{s}}', \tilde{\mathbf{s}}), W_{\boldsymbol{\lambda},\kappa}\rangle > \frac{\|\boldsymbol{\lambda}\|_2^2}{2} - \langle \mathbf{e}''^{(\kappa)}, \boldsymbol{\lambda}\rangle\right\} \tag{4.7}$$

Note that for $\boldsymbol{\lambda} \in C \cdot \mathrm{VR}_{E_8}^{(1)}$ we have $\frac{||\boldsymbol{\lambda}||^2}{2} - \langle \mathbf{e}''^{(\kappa)}, \boldsymbol{\lambda}\rangle \geq C^2 - 2k_2C$. In fact, using the Cauchy–Schwarz inequality and the fact that the centered binomial distribution of parameter k_2 has support in $[-k_2, k_2]$, we obtain:

$$\langle \mathbf{e}''^{(\kappa)}, \boldsymbol{\lambda}\rangle \leq \left|\left\langle \mathbf{e}''^{(\kappa)}, \boldsymbol{\lambda}\right\rangle\right| \leq C \cdot \left|\left\langle \left(e_1''^{(\kappa)}, e_2''^{(\kappa)}\right), (1,1)\right\rangle\right| \leq 2k_2C.$$

The same way we prove that for type 2 vectors $\boldsymbol{\lambda}$, $\frac{||\boldsymbol{\lambda}||^2}{2} - \langle \mathbf{e}''^{(\kappa)}, \boldsymbol{\lambda}\rangle \geq C^2 - 4k_2C$.

It follows that we can bound each term by computing the distribution of $\langle (\tilde{\mathbf{s}}', \tilde{\mathbf{s}}), W_{\boldsymbol{\lambda},\kappa}\rangle$, which is a sum of $2Ld$ i.i.d. random variables of the form $\mathbf{e}^{(i)} \cdot \boldsymbol{\lambda}$. Details are discussed in Appendix C.2.

From our numerical simulations, we obtain the following table:

4.4.4 Security

Similarly to the original KyberKEM scheme, our modified scheme achieves IND-CPA security which is then transformed into IND-CCA security. In addition, we study security against known attacks, that is, concrete security.

Error Probability Bounds						
Values of p / KyberKEM levels and parameters	$p=2$	$p=3$	$p=4$	$p=5$	$p=6$	$p=\infty$
KyberKEM-512: $q=2^{11}$, $k_1=3, k_2=2$	2^{-36}	2^{-87}	2^{-117}	2^{-133}	2^{-141}	
KyberKEM-768: $q=2^{11}$, $k_1=k_2=2$	2^{-48}	2^{-113}	2^{-153}	2^{-174}		
KyberKEM-768: $q=2^{12}$, $k_1=k_2=4$	2^{-47}	2^{-112}	2^{-152}	2^{-172}		
KyberKEM-768: $q=2^{13}$, $k_1=k_2=4$	2^{-193}	2^{-390}	2^{-499}	2^{-557}		
KyberKEM-1024: $q=2^{11}$, $k_1=k_2=2$	2^{-35}	2^{-87}	2^{-120}	2^{-137}		2^{-168}

Table 4.6: Upper bound for error probability for different values of moduli q, noise parameter k and reconciliation rate parameter p. Note that taking p too small leads to a large error probability, whereas if p is large, the size of the ciphertext c will also be large.

IND-CPA security

Recall from Subsection 2.3.3 that a KEM is IND-CPA secure if an attacker cannot distinguish it computationally from an ideal one, in which the shared private key is truly random. We will prove that, with the choice of q in Subsection 4.4.3, our KEM algorithm is IND-CPA secure assuming the hardness of M-LWE given two samples. This proof is generic and holds in the setting of the key generation protocol in Section 4.4 independently of the choice of the lattices Λ_1 and Λ_2 as long as the operations Q_{Λ_i} for $i=1,2$ can be done efficiently. We follow the same argument as in [Pei14, Section 4.2]. We consider the adjacent games below:

Game 1	**Game 2**	**Game 3**	**Game 4**
$\mathbf{A} \xleftarrow{\$} R_q^{d\times d}$	$\mathbf{A} \xleftarrow{\$} R_q^{d\times d}$	$(\mathbf{A}, \vec{\mathbf{b}}) \xleftarrow{\$} R_q^{d\times d} \times R_q^d$	$\mathbf{A} \xleftarrow{\$} R_q^{d\times d}$
$(\vec{\mathbf{b}}, \vec{\mathbf{s}}) \leftarrow \mathsf{Gen}(\mathbf{A})$	$\vec{\mathbf{b}} \xleftarrow{\$} R_q^d$	$(\vec{\mathbf{u}}, \mathbf{v}) \xleftarrow{\$} R_q^d \times R_q$	$(\vec{\mathbf{b}}, \vec{\mathbf{s}}) \leftarrow \mathsf{Gen}(\mathbf{A})$
$((\vec{\mathbf{u}}, \mathbf{r}), \mathbf{k}) \leftarrow \mathsf{Encaps}(\mathbf{A}, \vec{\mathbf{b}})$	$((\vec{\mathbf{u}}, \mathbf{r}), \mathbf{k}) \leftarrow \mathsf{Encaps}(\mathbf{A}, \vec{\mathbf{b}})$	$\mathbf{r} = \textsc{HelpRec}(\mathbf{v})$	$((\vec{\mathbf{u}}, \mathbf{r}), \mathbf{k}) \leftarrow \mathsf{Encaps}(\mathbf{A}, \vec{\mathbf{b}})$
		$\mathbf{k}^* \xleftarrow{\$} \Lambda_2/\Lambda_3$	$\mathbf{k}^* \xleftarrow{\$} \Lambda_2/\Lambda_3$
Output $\left(\mathbf{A}, \vec{\mathbf{b}}, (\vec{\mathbf{u}}, \mathbf{r}), \mathbf{k}\right)$	Output $\left(\mathbf{A}, \vec{\mathbf{b}}, (\vec{\mathbf{u}}, \mathbf{r}), \mathbf{k}\right)$	Output $\left(\mathbf{A}, \vec{\mathbf{b}}, (\vec{\mathbf{u}}, \mathbf{r}), \mathbf{k}^*\right)$	Output $\left(\mathbf{A}, \vec{\mathbf{b}}, (\vec{\mathbf{u}}, \mathbf{r}), \mathbf{k}^*\right)$

Notice that Game 1 is the "real" game corresponding to our modified protocol, and Game 4 is the "ideal" game where the key is uniform and independent from all the random variables observed by the eavesdropper. Game 2 is a modified version of Game 1 in which $\vec{\mathbf{b}}$ is taken to be uniform in R_q^d. The inputs of Game 3 are all uniformly sampled except for the reconciliation message $\mathbf{r}$.

Our aim is to prove that Game 1 and Game 4 are computationally indistinguishable. We'll do so sequentially.

Equivalence of Game 1 and Game 2. Game 1 and Game 2 are computationally indistinguishable under the assumption of hardness of M-LWE.

Equivalence of Game 2 and Game 3. To prove that Game 2 and Game 3 are computationally indistinguishable, we use the following Theorem which is essentially a consequence of the Crypto Lemma [Zam14, Lemma 4.1.1]. It guarantees uniformity of the private key without a dither.

Theorem 4.3. If $\mathbf{v} \in R_q$ is uniformly random, then $\mathbf{k} = \textsc{Rec}(\mathbf{v}, \mathbf{r})$ is uniformly random, given $\mathbf{r} = \textsc{HelpRec}(\mathbf{v})$.

Proof. For fixed $\mathbf{k}, \mathbf{k}' \in \Lambda_2/\Lambda_3$ we define $\forall\, \mathbf{v} \in R_q$

$$\pi_{\mathbf{k},\mathbf{k}'}(\mathbf{v}) = \left(\mathbf{v} - \mathbf{k} + \mathbf{k}'\right) \bmod \Lambda_3.$$

Notice that $\pi_{\mathbf{k},\mathbf{k}'}(\mathbf{v}) \in R_q$ because $(-\mathbf{k} + \mathbf{k}') \in \Lambda_2 \subseteq \mathbb{Z}^n$ and hence $\pi_{\mathbf{k},\mathbf{k}'}(\mathbf{v}) \in \mathbb{Z}_q^n$. So $\pi_{\mathbf{k},\mathbf{k}'}$ is a permutation of R_q by Lemma 2.2. The proof of Theorem 4.3 results from these lemmas:

Lemma 4.2. $\forall\, \mathbf{k}, \mathbf{k}' \in \Lambda_2/\Lambda_3$ and $\forall\, \mathbf{v} \in R_q$ we have $\textsc{HelpRec}(\mathbf{v}) = \textsc{HelpRec}\left(\pi_{\mathbf{k},\mathbf{k}'}(\mathbf{v})\right)$.

Proof. The equalities below follow from the properties (P1), (P2), (P3) and (P4) cited in Lemma 2.1.

$$\begin{aligned}
\textsc{HelpRec}\left(\pi_{\mathbf{k},\mathbf{k}'}(\mathbf{v})\right) &= Q_{\Lambda_1}\left(\pi_{\mathbf{k},\mathbf{k}'}(\mathbf{v})\right) \bmod \Lambda_2 \\
&= Q_{\Lambda_1}\left(\left(\mathbf{v} - \mathbf{k} + \mathbf{k}'\right) \bmod \Lambda_3\right) \bmod \Lambda_2 \\
&= Q_{\Lambda_1}\left(\mathbf{v} - \mathbf{k} + \mathbf{k}' - Q_{\Lambda_3}\left(\mathbf{v} - \mathbf{k} + \mathbf{k}'\right)\right) \bmod \Lambda_2 \\
&\overset{(P3)}{=} \left(Q_{\Lambda_1}(\mathbf{v}) - \mathbf{k} + \mathbf{k}' - Q_{\Lambda_3}\left(\mathbf{v} - \mathbf{k} + \mathbf{k}'\right)\right) \bmod \Lambda_2 \\
&\overset{(P2)}{=} Q_{\Lambda_1}(\mathbf{v}) \bmod \Lambda_2 = \textsc{HelpRec}(\mathbf{v}).
\end{aligned}$$

□

Lemma 4.3. Suppose that $\mathbf{k} = \textsc{Rec}(\mathbf{v}, \mathbf{r}) = Q_{\Lambda_2}\left(\mathbf{v} - \mathbf{r}\right) \bmod \Lambda_3$. Then $\forall\, \mathbf{k}' \in \Lambda_2/\Lambda_3$ we have $\mathbf{k}' = \textsc{Rec}(\pi_{\mathbf{k},\mathbf{k}'}(\mathbf{v}), \mathbf{r})$.

Proof. As before, we will use the properties (P1), (P2), (P3) or (P4) from Lemma 2.1.

$$\begin{aligned}
\textsc{Rec}(\pi_{\mathbf{k},\mathbf{k}'}(\mathbf{v}), \mathbf{r}) &= Q_{\Lambda_2}\left(\mathbf{v} - \mathbf{k} + \mathbf{k}' - Q_{\Lambda_3}\left(\mathbf{v} - \mathbf{k} + \mathbf{k}'\right) - \mathbf{r}\right) \bmod \Lambda_3 \\
&\overset{(P3)}{=} \left(Q_{\Lambda_2}\left(\mathbf{v} - \mathbf{r}\right) - \mathbf{k} + \mathbf{k}' - Q_{\Lambda_3}\left(\mathbf{v} - \mathbf{k} + \mathbf{k}'\right)\right) \bmod \Lambda_3 \\
&\overset{(P2)}{=} \left(Q_{\Lambda_2}\left(\mathbf{v} - \mathbf{r}\right) - \mathbf{k} + \mathbf{k}'\right) \bmod \Lambda_3 \\
&\overset{(P1)}{=} \left(Q_{\Lambda_2}\left(\mathbf{v} - \mathbf{r}\right) \bmod \Lambda_3 - \mathbf{k} + \mathbf{k}'\right) \bmod \Lambda_3 \\
&= \left(\mathbf{k} - \mathbf{k} + \mathbf{k}'\right) \bmod \Lambda_3 \\
&= \mathbf{k}'.
\end{aligned}$$

□

The Corollary below follows immediately from Lemma 4.2 and Lemma 4.3.

Corollary 4.1. $\forall\ \mathbf{k}, \mathbf{k}' \in \Lambda_2/\Lambda_3$ and $\forall\ \mathbf{v} \in R_q$, there exist $\mathbf{v}' = \pi_{\mathbf{k},\mathbf{k}'}(\mathbf{v})$ such that $\textsc{HelpRec}(\mathbf{v}) = \textsc{HelpRec}\left(\pi_{\mathbf{k},\mathbf{k}'}(\mathbf{v})\right)$, and $\mathbf{k} = \textsc{Rec}(\mathbf{v}, \mathbf{r}) \iff \mathbf{k}' = \textsc{Rec}(\pi_{\mathbf{k},\mathbf{k}'}(\mathbf{v}), \mathbf{r})$.

We conclude the proof of Theorem 4.3 by showing that $\mathbf{k}$ is uniform and independent of $\mathbf{r}$ when $\mathbf{v}$ is uniform:

$$\begin{aligned}
\mathbb{P}\{\mathbf{k} \mid \mathbf{r}\} &= \sum_{\mathbf{v} \in R_q} \mathbb{P}\{\mathbf{v}\} \cdot \mathbb{P}\{\mathbf{k} \mid \mathbf{r}, \mathbf{v}\} \\
&= \sum_{\mathbf{v} \in R_q} \mathbb{1}_{\{\mathbf{r} = \textsc{HelpRec}(\mathbf{v})\,,\, \mathbf{k} = \textsc{Rec}(\mathbf{v}, \mathbf{r})\}} \cdot \mathbb{P}\{\mathbf{v}\} \\
&= \sum_{\mathbf{v} \in R_q} \mathbb{1}_{\left\{\mathbf{r} = \textsc{HelpRec}(\pi_{\mathbf{k},\mathbf{k}'}(\mathbf{v}))\,,\, \mathbf{k}' = \textsc{Rec}(\pi_{\mathbf{k},\mathbf{k}'}(\mathbf{v}), \mathbf{r})\right\}} \cdot \mathbb{P}\{\mathbf{v}\} \\
&= \sum_{\mathbf{v}' \in R_q} \mathbb{1}_{\{\mathbf{r} = \textsc{HelpRec}(\mathbf{v}')\,,\, \mathbf{k}' = \textsc{Rec}(\mathbf{v}', \mathbf{r})\}} \cdot \mathbb{P}\{\mathbf{v}'\} \\
&= \sum_{\mathbf{v}' \in R_q} \mathbb{P}\{\mathbf{v}'\} \cdot \mathbb{P}\{\mathbf{k}' \mid \mathbf{r}, \mathbf{v}'\} = \mathbb{P}\{\mathbf{k}' \mid \mathbf{r}\}.
\end{aligned}$$

□

Returning to Game 2 and Game 3, we construct an efficient reduction $\mathcal{S}$ as follows: it takes as input two pairs $(\mathbf{A}, \vec{\mathbf{u}}), (\vec{\mathbf{b}}, \mathbf{v})$, and outputs

$$\left(\mathbf{A}, \vec{\mathbf{b}},\ (\vec{\mathbf{u}},\ \mathbf{r} = \textsc{HelpRec}(\mathbf{v}))\,,\ \mathbf{k} = \textsc{Rec}(\mathbf{v}, \mathbf{r})\right).$$

Note that if two random variables X and Y are computationally indistinguishable and f is a polynomial time function, then $f(X)$ and $f(Y)$ are computationally indistinguishable. Therefore, taking two indistinguishable inputs from $\mathcal{S}$ leads to indistinguishable outputs, due to its efficiency.

First suppose that the inputs are M-LWE instances; i.e. $\vec{\mathbf{u}} = \mathbf{A}^T\vec{\mathbf{s}'} + \vec{\mathbf{e}'}$ and $\mathbf{v} = \langle \vec{\mathbf{b}}, \vec{\mathbf{s}'} \rangle + \mathbf{e}''$, where $\mathbf{A}$ is uniformly random and $\vec{\mathbf{b}}$ is indistinguishable from uniform. Hence, the output of $\mathcal{S}$ will be exactly as in Game 2. Now suppose that the inputs given to $\mathcal{S}$ are uniformly random and independent, then the outputs of $\mathcal{S}$ are exactly as in Game 3. In fact, $\mathbf{A}, \vec{\mathbf{b}}, \vec{\mathbf{u}}, \mathbf{v}$ are uniform, and hence by Theorem 4.3, $\mathbf{k}$ is uniformly random conditioned on $\mathbf{r} = \textsc{HelpRec}(\mathbf{v})$.

Equivalence of Game 3 and Game 4. To show that Game 3 and Game 4 are indistinguishable, we modify Game 1 and Game 2 by choosing $\mathbf{k}^* \xleftarrow{\$} \Lambda_2/\Lambda_3$ and output it instead of $\mathbf{k}$. In this case Game 1 becomes Game 4. Let Game 2' be the modified version of Game 2. As a result we obtain the following:

Game 1' = Game 4	**Game 2'**	**Game 3**	**Game 4**
$\mathbf{A} \xleftarrow{\$} R_q^{d\times d}$	$\mathbf{A} \xleftarrow{\$} R_q^{d\times d}$	$(\mathbf{A}, \vec{\mathbf{b}}) \xleftarrow{\$} R_q^{d\times d} \times R_q^d$	$\mathbf{A} \xleftarrow{\$} R_q^{d\times d}$
$(\vec{\mathbf{b}}, \vec{\mathbf{s}}) \leftarrow \mathsf{Gen}(\mathbf{A})$	$\vec{\mathbf{b}} \xleftarrow{\$} R_q^d$	$(\vec{\mathbf{u}}, \mathbf{v}) \xleftarrow{\$} R_q^d \times R_q$	$(\vec{\mathbf{b}}, \vec{\mathbf{s}}) \leftarrow \mathsf{Gen}(\mathbf{A})$
$((\vec{\mathbf{u}}, \mathbf{r}), \mathbf{k}) \leftarrow \mathsf{Encaps}(\mathbf{A}, \vec{\mathbf{b}})$	$((\vec{\mathbf{u}}, \mathbf{r}), \mathbf{k}) \leftarrow \mathsf{Encaps}(\mathbf{A}, \vec{\mathbf{b}})$	$\mathbf{r} = \textsc{HelpRec}(\mathbf{v})$	$((\vec{\mathbf{u}}, \mathbf{r}), \mathbf{k}) \leftarrow \mathsf{Encaps}(\mathbf{A}, \vec{\mathbf{b}})$
$\mathbf{k}^* \xleftarrow{\$} \Lambda_2/\Lambda_3$	$\mathbf{k}^* \xleftarrow{\$} \Lambda_2/\Lambda_3$	$\mathbf{k}^* \xleftarrow{\$} \Lambda_2/\Lambda_3$	$\mathbf{k}^* \xleftarrow{\$} \Lambda_2/\Lambda_3$
Output $\left(\mathbf{A}, \vec{\mathbf{b}}, (\vec{\mathbf{u}}, \mathbf{r}), \mathbf{k}^*\right)$	Output $\left(\mathbf{A}, \vec{\mathbf{b}}, (\vec{\mathbf{u}}, \mathbf{r}), \mathbf{k}^*\right)$	Output $\left(\mathbf{A}, \vec{\mathbf{b}}, (\vec{\mathbf{u}}, \mathbf{r}), \mathbf{k}^*\right)$	Output $\left(\mathbf{A}, \vec{\mathbf{b}}, (\vec{\mathbf{u}}, \mathbf{r}), \mathbf{k}^*\right)$

By the same reasoning as above, we can prove that Game 1' is computationally indistinguishable from Game 2', and Game 2' is computationally indistinguishable from Game 3. This proves that Game 4 (= Game 1') is computationally indistinguishable from Game 3.

IND-CCA security

Following [Pei14], one can transform an IND-CPA secure KEM into an IND-CPA encryption scheme for message space $\mathcal{M} = \mathcal{K}$ by having the sender use the ephemeral private key $\mathbf{k} \in \mathcal{K}$ as a one-time pad to conceal the message (this assumes that $\mathcal{K}$ has a group structure). Therefore, we can similarly convert our IND-CPA KEM to a passively secure PKE for message space $\mathcal{M} = \Lambda_2/\Lambda_3$ with the same level of reliability and security.
Similarly to KyberKEM, since the decryption failure probability is small, we can obtain an IND-CCA secure KEM using a lightly tweaked Fujisaki-Okamoto transform applied to the IND-CPA secure PKE (see Subsection 2.3.4). A theoretical bound can be given on the CCA advantage in the same way as for KyberKEM as discussed in Subsection 4.3.3 (see also [A+20, Theorem 2 and Theorem 3]). Note that the decryption failure probability bound appears in (4.1).

Concrete security

We study the hardness of Module-LWE by considering it as an LWE problem, since, to date, the best known attacks don't make use of the module structure. There are numerous attacks to consider, however, we essentially deal with two BKZ attacks, namely primal and dual attacks as in KyberKEM. The cost of the primal attack and dual attack are computed using KyberKEM's script [1] and presented in the next section.

4.4.5 Performance comparison

We make a comparison between our protocol and KyberKEM [A+20] in terms of security, decryption error rate and bandwidth. The security against known attacks is increased by 6% to 8% for all KyberKEM levels. In terms of decryption error rate, our scheme guarantees a lower error probability for KyberKEM-512 and KyberKEM-768 compared to the original levels, namely $P_e < 2^{-141}$ for $p = 6$ and $P_e < 2^{-174}$ for $p = 5$ respectively; whereas for KyberKEM-1024 our error probability is not improved for any choice of the parameter p. For KyberKEM-512 and

[1] `https://github.com/pq-crystals/security-estimates/blob/master/MLWE_security.py`

$p = 6$, the bandwidth is increased by 12.5%. Also for KyberKEM-768 and $p = 5$, the bandwidth is increased by 8.8%. Taking smaller p values will not guarantee smaller error probability compared to the original KyberKEM.

A summary of the results is illustrated in Table 4.7.

Attack	m	b	Known Classical	Known Quantum	Best Plausible
KyberKEM-512 Round 3: $q = 3329$, $n = 256$, $d = 2$, $k_1 = 3$, $k_2 = 2$, $P_e \leq 2^{-139}$					
Primal	486	405	118	107	84
Dual	482	404	118	107	83
Our Protocol: $q = 2^{11} = 2048$, $n = 256$, $d = 2$, $k_1 = 3$, $k_2 = 2$, $p = 6$, $P_e \leq 2^{-141}$					
Primal	473	437	127	115	90
Dual	493	435	127	115	90
KyberKEM-768 Round 3: $q = 3329$, $n = 256$, $d = 3$, $k_1 = k_2 = 2$, $P_e \leq 2^{-164}$					
Primal	658	623	182	165	129
Dual	670	620	181	164	128
Our Protocol: $q = 2^{11} = 2048$, $n = 256$, $d = 3$, $k_1 = k_2 = 2$, $p = 5$, $P_e \leq 2^{-174}$					
Primal	658	665	194	176	138
Dual	651	662	194	176	137
KyberKEM-1024 Round 3: $q = 3329$, $n = 256$, $d = 4$, $k_1 = k_2 = 2$, $P_e \leq 2^{-174}$					
Primal	841	873	255	231	181
Dual	862	868	253	230	180
Our Protocol: $q = 2^{11} = 2048$, $n = 256$, $d = 2$, $k_1 = 3$, $k_2 = 2$, $p = \infty$, $P_e \leq 2^{-168}$					
Primal	840	928	271	246	192
Dual	833	923	269	244	191

Table 4.7: Core hardness of our protocol and comparison with the state of the art. b denotes the block dimension of BKZ, and m the number of used samples. The given costs are the smallest ones for all possible choices of m and b.

4.5 Reconciliation using higher-dimensional Barnes-Wall lattices

It is natural to ask whether our technique could be improved by using higher-dimensional lattices for reconciliation. Therefore, we present some preliminary but inconclusive results in Appendix D.1. We replace E_8 by higher-dimensional Barnes-Wall lattices, and analogously to our previous construction, we choose

$$\Lambda_1 = \left(\frac{q}{2^p} BW^{n_0}\right)^L, \ \Lambda_2 = (\beta BW^{n_0})^L \ \text{ and } \Lambda_3 = (q\mathbb{Z}^{n_0})^L \tag{4.8}$$

with $n_0 \times L = 256$. After that we focus on the case $n_0 = 16$ that has been studied in Subsection 2.2.4.

Up to now, higher-dimensional lattices don't appear to offer any benefit. On one hand, due to the scaling conditions imposed by the inclusions $\Lambda_3 \subseteq \Lambda_2$, choosing higher-dimensional lattices doesn't result in a significant gain in terms of minimum distance. Moreover, the corresponding key sizes are not powers of two and thus are ill-suited to cryptographic applications. Finally, with higher-dimensional lattices, it is computationally difficult to obtain a tight bound for the error probability.

4.6 Conclusion

In this chapter, we have proposed a modification of KyberKEM which uses a reconciliation mechanism. Our choice of parameters is slightly modified with respect to the original KyberKEM and results are summarized in Table 4.8.

In terms of the Module-LWE instances, the ring R as well as the degree $n = 256$ and the dimension $d \in \{2, 3, 4\}$ with respect to the chosen level remain unchanged. The modulus q is reduced from $q = 3329$ to $q = 2^{11}$. Since q is a power-of-two, alternative Karatsuba/Toom-Cook algorithms should be used to perform multiplication within R_q instead of using the NTT. The advantage of choosing an even q is that a dither is not required to obtain a uniform key with the reconciliation method. The centered binomial error distribution ψ_k is used with the same parameter $k \in \{2, 3\}$ as in the original KyberKEM for all case levels.

The reconciliation step is done using the 8-dimensional lattice E_8 that allows to generate one bit of key per dimension with a reconciliation rate of $p - 1$. The total size of the shared private key is 256 bits. Our scheme excludes the use of compression and decompression functions which leaves us with a small increase in terms of bandwidth compared to KyberKEM.

In terms of security, our scheme achieves higher security against known attacks, namely an improvement by 6% to 8% as shown in Table 4.8, while IND-CCA security is still guaranteed.

In conclusion, we have shown that the reconciliation technique can provide a real gain in terms of security and reliability. In terms of efficiency, we haven't precisely quantified the complexity of hardware implementation of our protocol, which is left for future work.

Original KyberKEM										
	n	d	(k_1, k_2)	q	p	Concrete Security			Bandwidth	P_e
						C	Q	P	(bits)	
KyberKEM-512	256	2	$(3, 2)$	3329	–	118	107	84	6144	2^{-139}
KyberKEM-768	256	3	$(2, 2)$	3329	–	181	164	128	8704	2^{-164}
KyberKEM-1024	256	4	$(2, 2)$	3329	–	253	230	180	12544	2^{-174}
Modified KyberKEM										
Modified KyberKEM-512	256	2	$(3, 2)$	2^{11}	6	127	115	90	6921	2^{-141}
Modified KyberKEM-768	256	3	$(2, 2)$	2^{11}	5	194	176	137	9472	2^{-174}
Modified KyberKEM-1024	256	4	$(2, 2)$	2^{11}	7	269	244	191	12800	2^{-151}

Table 4.8: Modified parameters for improving the security level of KyberKEM scheme as well as decreasing the error probability.

Conclusions and perspectives

Throughout this thesis, we have devoted our research to examining and evaluating two key encapsulation mechanisms that have been submitted to the NIST Post-Quantum project in order to modify the parameter settings and therefore improve the overall security, decrease the decryption failure probability or reduce the communication requirements. The protocols selected are lattice-based protocols that rely on the Learning With Errors problem and its variants, in favor of robust security. The results obtained show that one can achieve better performance by applying an error correction mechanism as well as a reconciliation technique.

By adopting an error-correction technique based on the 8-dimensional lattice E_8 in both Chapters 3 and 4, we were able to perform the decryption operations with a lower error probability, which leads to an improvement in the IND-CCA advantage. However, one aspect that still needs to be investigated is the efficiency of our technique in terms of practical hardware implementation. The existing decoding algorithms which rely on simple rounding functions suggest that the efficiency will not be too affected.

Since it is well known that the minimum distance and error correction capabilities of lattice codes increase with the lattice dimension, in Appendix D we investigated the use of higher-dimensional lattices, such as Barnes-Wall lattices, for reconciliation. However, our results are inconclusive and we were unable to obtain an error probability bound in the range required for post-quantum cryptography applications. In part, this is due to the fact that it is difficult to obtain rigorous and tight bounds for the error probability for high-dimensional lattices. Moreover, a rather counter-intuitive conclusion is that the use of higher-dimensional lattices does not necessarily bring a gain in terms of minimum distance due to the scaling constraints imposed by the modulo-q integer arithmetic of LWE protocols.

In alternative to lattice coding, we also investigated the use of polar codes and Reed-Muller codes for error correction in LWE-based protocols, but encountered some difficulties and we did not present our partial results in this report. Polar codes [Ari09] are an attractive solution because of the low complexity successive cancellation (SC) decoding, but it is not easy to design and evaluate the performance of polar codes for the "LWE channel" in which the noise is not independent and identically distributed [Wan20]. Another option is to use Reed-Muller codes with the low-complexity decoder by Schnabl and Bossert [SB95]. The advantage is that this decoder is a bounded distance decoder and one can use the sphere bound in order to estimate

the error probability. However, we failed to obtain a good bound for the error probability, which may be due to the fact that BDD decoding is sub-optimal, and that our bound was based on the subgaussian parameter.

Finally, we mention that designing lattice-based protocols which are at the same time efficient and provably secure is still an open problem. In fact, the parameters chosen for the current NIST candidates based on lattices do not satisfy the hypotheses of the worst-case to average-case reduction theorems, although they are heuristically assumed to be secure. Therefore, more work is needed to design protocols which ensure a better decryption failure probability and better modulus for larger error variance. This problem was studied in [Gat18] but the proposed parameters lead to an inefficient cryptosystem. Such a consideration is therefore of significant interest for future work, and error-correction and reconciliation techniques might be a useful tool to achieve this goal. Error correction could also be a valuable tool for fully homomorphic encryption based on LWE or Ring-LWE [BV11b, BV14b], where noise amplification is the main limiting factor for the number of homomorphic operations.

Appendix A

Mathematical implementations, definitions and proofs

A.1 Probability generating function

This section is devoted to review the *probability generating function*.

Definition A.1 (Probability generating function)**.** Let Z be a discrete random variable taking values in a set $A \subseteq \mathbb{Z}$ that is bounded below. The *probability generating function* G_Z of Z is defined as

$$G_Z(x) = \sum_{i \in A} \mathbb{P}\{Z = i\} x^i.$$

This function allows to calculate the distribution of a sum of independent random variables:

Lemma A.1. If $Z_1, Z_2, \ldots, Z_n$ is a sequence of independent (and not necessarily identically distributed) discrete random variables that take values in a lower bounded sets $A_1, A_2, \ldots, A_n \subseteq \mathbb{Z}$ respectively, then

$$G_{Z_1+\cdots+Z_n}(x) = G_{Z_1}(x) \cdot \cdots \cdot G_{Z_n}(x).$$

A.2 Computation of the Rényi divergence

We present here the code written in `Mathematica` to calculate the Rényi divergence for $\alpha = 9$ between the rounded distribution $\Psi_{\sqrt{16\pi}}$ and the centered binomial ψ_{16}, as it was done in [ADPS16b].

```
In[1]:= alpha=9;
        (* Defining psi_16 *)
        psi16[t_] := Module[{sum = 0},
        For[j = Max[0, t], j <= Min[16 + t, 16], j++,
        sum = sum + Binomial[16, j]*Binomial[16, j - t]];
        (1/2^32)*sum];
```

```
(* -- *)
support = 16;
sigma = Sqrt[16/2];
(* Now the rounded gaussian *)
tau = 30;
tail = Ceiling[tau*sigma];
suppChi = tail;
phi[z_] := 0.5*(1 + Erf[z/Sqrt[2]]) // N;
chi[x_] := phi[(x+0.5)/sigma] - phi[(x-0.5)/sigma];
(* Evaluating the Renyi Divergence *)
s = 0;
For[i = -support, i <= support, i++,
s = s + SetPrecision[psi16[i], 10000]^alpha/
SetPrecision[chi[i], 10000]^(alpha-1)];
R = (1/(alpha-1))*Log[s] // N
```

Out[1]= 0.00064

A.3 Voronoi relevant vectors for E_8

```
G:= [2,0,0,0,0,0,0,0,
    -1,1,0,0,0,0,0,0,
    0,-1,1,0,0,0,0,0,
    0,0,-1,1,0,0,0,0,
    0,0,0,-1,1,0,0,0,
    0,0,0,0,-1,1,0,0,
    0,0,0,0,0,-1,1,0,
    1/2,1/2,1/2,1/2,1/2,1/2,1/2,1/2];
E := LatticeWithBasis(8,G);
VoronoiRelevantVectors(E);
```

A.4 Voronoi relevant vectors for BW^{16}

```
G:= [2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
     1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
     1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,
     1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,
     1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,
     1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,
     1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,
     1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,
     1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,
     1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,
     1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,
1/2,1/2,1/2,1/2,0,1/2,0,1/2,1/2,0,0,1/2,0,0,0,0,
```

```
0,1/2,1/2,1/2,1/2,0,1/2,0,1/2,1/2,0,0,1/2,0,0,0,
0,0,1/2,1/2,1/2,1/2,0,1/2,0,1/2,1/2,0,0,1/2,0,0,
0,0,0,1/2,1/2,1/2,1/2,0,1/2,0,1/2,1/2,0,0,1/2,0,
1/2,1/2,1/2,1/2,1/2,1/2,1/2,1/2,1/2,1/2,1/2,1/2,1/2,1/2,1/2,1/2];
E := LatticeWithBasis(16,G);
VoronoiRelevantVectors(E);
```

A.5 Proof of Theorem 2.2

Theorem 2.2 is presented in its real form instead of the complex form since throughout this thesis we are only working with real lattices. To provide concrete and elegant description, we use complex numbers following the notation in [MN08]. For instance, the notation BW^n is now a lattice of dimension $N = 2^n$ in $\mathbb{C}^N$ with generator matrix

$$\mathbf{G}_{BW^n} = \begin{bmatrix} \mathbf{G}_{BW^{n-1}} & \mathbf{G}_{BW^{n-1}} \\ 0 & \phi \cdot \mathbf{G}_{BW^{n-1}} \end{bmatrix}$$

with $\phi = 1 + \mathrm{i}$ and initial condition $BW^0 = [1]$.

Note that everything can be translated from complex space into the real space, and vice-versa, using the following transformation:

$$\mathbb{C} \to \mathbb{R}^{2\times 2}$$
$$a + \mathrm{i}b \mapsto \begin{bmatrix} a & -b \\ b & a \end{bmatrix}$$

In fact,

$$\underbrace{\boldsymbol{\lambda}}_{\in\mathbb{C}^N} = [z_1, \ldots, z_N] \cdot \underbrace{\mathbf{G}_{BW^n}}_{\in\mathbb{C}^N} \iff \underbrace{\boldsymbol{\lambda}}_{\in\mathbb{R}^{2N}} = [\Re(z_1), -\Im(z_1), \ldots, \Re(z_N), -\Im(z_N)] \cdot \underbrace{\mathbf{G}_{BW^n}}_{\in\mathbb{R}^{2N}}$$

In the following we refer to the functions ParBW, SeqBW, RMdec in Algorithms 1,2 and 3 of Micciancio and Nicolosi's paper [MN08].

A.5.1 Modification

We modify Algorithm 3 in [MN08] in the case $r = 0$ as follows: if $\sum_{b_j=0} \rho_j = \sum_{b_j=1} \rho_j$, then return $b_1 \cdot [1, 1, \ldots, 1]$.
It means that we choose the output vector based on the first bit of $\mathbf{b}$. Note that the decoder is still BDD with this modification.

A.5.2 Linearity of ParBW

In this subsection we will prove the following proposition:

Proposition A.1. Let $\boldsymbol{\lambda} \in BW^n$ and $\mathbf{w} \in \mathbb{C}^N$ a target, where $N = 2^n$, then

$$\textsc{ParBW}(p, \underbrace{\boldsymbol{\lambda} + \mathbf{w}}_{\mathbf{s}}) = \boldsymbol{\lambda} + \textsc{ParBW}(p, \mathbf{w}), \quad \forall\, p = 4^k.$$

We will prove this by induction on p and N. The cases $p = 1$ or $N = 1$ will be proven in the next subsection. Now consider the case where $p \geq 4$ and $\mathbf{s} \notin \mathbb{C}^1$. Suppose that Proposition A.1 holds for $N/2$-dimensional vectors and $p/4$ processors; we will show that it also holds for N-dimensional vectors and p processors.
Let $\boldsymbol{\lambda} = [\boldsymbol{\lambda}_0, \boldsymbol{\lambda}_1]$ and $\mathbf{w} = [\mathbf{w}_0, \mathbf{w}_1]$. As defined in Algorithm 1, we have:

$$\begin{aligned}
\mathbf{s} :&= [\mathbf{s}_0, \mathbf{s}_1] \\
&= [\boldsymbol{\lambda}_0 + \mathbf{w}_0, \boldsymbol{\lambda}_1 + \mathbf{w}_1] \\
&= [\boldsymbol{\lambda}_0, \boldsymbol{\lambda}_1] + [\mathbf{w}_0, \mathbf{w}_1],
\end{aligned}$$

and the candidate vectors are

$$\begin{bmatrix} \mathbf{z}_0(\mathbf{s}) \\ \mathbf{z}_1(\mathbf{s}) \\ \mathbf{z}_-(\mathbf{s}) \\ \mathbf{z}_+(\mathbf{s}) \end{bmatrix} := \begin{bmatrix} \textsc{ParBW}(p/4, \boldsymbol{\lambda}_0 + \mathbf{w}_0) \\ \textsc{ParBW}(p/4, \boldsymbol{\lambda}_1 + \mathbf{w}_1) \\ \textsc{ParBW}\left(p/4, \frac{\phi}{2}[\boldsymbol{\lambda}_0 - \boldsymbol{\lambda}_1] + \frac{\phi}{2}[\mathbf{w}_0 - \mathbf{w}_1]\right) \\ \textsc{ParBW}\left(p/4, \frac{\phi}{2}[\boldsymbol{\lambda}_0 + \boldsymbol{\lambda}_1] + \frac{\phi}{2}[\mathbf{w}_0 + \mathbf{w}_1]\right) \end{bmatrix}.$$

Here we use implicitly the fact that $[\boldsymbol{\lambda}_0, \boldsymbol{\lambda}_1] \in BW^n \Longrightarrow \boldsymbol{\lambda}_0, \boldsymbol{\lambda}_1 \in BW^{n-1}$. Note that

$$\begin{aligned}
\mathbf{z}_0^-(\mathbf{s}) &:= [\mathbf{z}_0(\mathbf{s}), \mathbf{z}_0(\mathbf{s}) - 2\phi^{-1}\mathbf{z}_-(\mathbf{s})] \\
&= \Big[\boldsymbol{\lambda}_0 + \textsc{ParBW}(p/4, \mathbf{w}_0), \\
&\qquad \boldsymbol{\lambda}_0 + \textsc{ParBW}(p/4, \mathbf{w}_0) - (\boldsymbol{\lambda}_0 - \boldsymbol{\lambda}_1) - 2\phi^{-1}\textsc{ParBW}\left(p/4, \tfrac{\phi}{2}(\mathbf{w}_0 - \mathbf{w}_1)\right)\Big] \\
&= [\boldsymbol{\lambda}_0, \boldsymbol{\lambda}_1] + \Big[\textsc{ParBW}(p/4, \mathbf{w}_0), \textsc{ParBW}(p/4, \mathbf{w}_0) - 2\phi^{-1}\textsc{ParBW}\left(p/4, \tfrac{\phi}{2}(\mathbf{w}_0 - \mathbf{w}_1)\right)\Big] \\
&= \boldsymbol{\lambda} + [\mathbf{z}_0(\mathbf{w}), \mathbf{z}_0(\mathbf{w}) - 2\phi^{-1}\mathbf{z}_-(\mathbf{w})] \\
&= \boldsymbol{\lambda} + \mathbf{z}_0^-(\mathbf{w}).
\end{aligned}$$

Following the same steps, we can prove that

$$\begin{aligned}\mathbf{z}_0^+(\mathbf{s}) &= \boldsymbol{\lambda} + \mathbf{z}_0^+(\mathbf{w}),\\ \mathbf{z}_1^-(\mathbf{s}) &= \boldsymbol{\lambda} + \mathbf{z}_1^-(\mathbf{w}),\\ \mathbf{z}_1^+(\mathbf{s}) &= \boldsymbol{\lambda} + \mathbf{z}_1^+(\mathbf{w}).\end{aligned}$$

The algorithm will return the value $\mathbf{z}'(\mathbf{s})$ in $\{\mathbf{z}_0^+(\mathbf{s}), \mathbf{z}_0^-(\mathbf{s}), \mathbf{z}_1^+(\mathbf{s}), \mathbf{z}_1^-(\mathbf{s})\}$ such that $\|\mathbf{s} - \mathbf{z}'(\mathbf{s})\|$ is minimal. This is equivalent to saying that $\mathbf{z}'(\mathbf{s}) \in \{\boldsymbol{\lambda} + \mathbf{z}_0^+(\mathbf{w}), \boldsymbol{\lambda} + \mathbf{z}_0^-(\mathbf{s}), \boldsymbol{\lambda} + \mathbf{z}_1^+(\mathbf{s}), \boldsymbol{\lambda} + \mathbf{z}_1^-(\mathbf{s})\}$ such that $\|(\boldsymbol{\lambda} + \mathbf{w}) - (\boldsymbol{\lambda} + \mathbf{z}_*^*(\mathbf{w}))\| = \|\mathbf{w} - \mathbf{z}_*^*(\mathbf{w})\|$ is minimal. In any case, the output is of the form $\boldsymbol{\lambda} + \textsc{ParBW}(p, \mathbf{w})$.

Our next step now is to prove that Proposition A.1 holds for $p = 1$ or $\mathbf{s} \in \mathbb{C}^1$. In this case Algorithm 1 returns $\textsc{SeqBW}(0, \boldsymbol{\lambda} + \mathbf{w})$. This function is presented in Algorithm 2.

A.5.3 Linearity of SeqBW$(r, *)$

In this subsection we refer to the functions $t(\mathbf{s}) = (\mathbf{b}(\mathbf{s}), \rho(\mathbf{s}))$ and $\text{RM}\textsc{Dec}(r, t) = \text{RM}\textsc{Dec}(r, \mathbf{b}, \rho)$ in Algorithms 2 and 3 and to the function $\psi : \mathbb{F}_2^N \mapsto \mathbb{Z}[\mathrm{i}]^N$ defined in [MN08].
Recall that each vector $\boldsymbol{\lambda}$ in BW^n can be written as

$$\boldsymbol{\lambda} = \sum_{r=0}^{n-1} \phi^r \psi(c_r) + \phi^n c_n,$$

where $c_n \in \mathbb{G}^N$ and $c_r \in RM_r^n$ for $r = 0, \ldots, n-1$.

For any $0 \le r \le n$, let

$$BW_r^n = \left\{ \sum_{k=r}^{n-1} \phi^{k-r} \psi(c_k) + \phi^{n-r} c_n \ : \ c_k \in RM_k^n,\, c_n \in \mathbb{G}^N \right\}$$

Proposition A.2. Let $\mathbf{w} \in \mathbb{C}^N$ and $c_r \in RM_r^n$. Then

$$\tilde{\text{RM}}\textsc{Dec}(r, c_r \oplus \mathbf{b}(\mathbf{w}), \rho) = c_r \oplus \tilde{\text{RM}}\textsc{Dec}(r, \mathbf{b}(\mathbf{w}), \rho),$$

where $\tilde{\text{RM}}\textsc{Dec} = \text{RM}\textsc{Dec} \bmod 2$.

Proof. Before we start the proof, observe that for $c_r \in RM_r^n$, $\psi(c_r) \bmod 2 = c_r$.
Let's start with $n = 1$, $r = 0$. In this case it is easy to see from Algorithm 3 that ψ is the identity function and

$$\text{RM}\textsc{Dec}(0, c_0 \oplus \mathbf{b}(\mathbf{w}), \rho) = \psi(c_0) \oplus \text{RM}\textsc{Dec}(0, \mathbf{b}(\mathbf{w}), \rho).$$

So this remains true for $\tilde{\text{RM}}\textsc{Dec}$.

For $n = 1$ and $r = 1$ we have $2^r = N$. Thus

$$\begin{aligned}\mathrm{RM}\textsc{dec}(1, c_1 \oplus \mathbf{b}(\mathbf{w}), \rho) &= c_1 \oplus \mathbf{b}(\mathbf{w}) \\ &= c_1 \oplus \mathrm{RM}\textsc{dec}(1, \mathbf{b}(\mathbf{w}), \rho)\end{aligned}$$

This remains also true for $\tilde{\mathrm{RM}}\textsc{dec}$. Now we continue by induction: suppose Proposition A.2 holds for $(r-1, n-1)$ and $(r, n-1)$. We want to show that it holds for (r, n). Following the notation in Algorithm 3, we find:

$$\begin{aligned}&[t^0, t^1] \longleftarrow t = (c_r \oplus \mathbf{b}(\mathbf{w}), \rho), \\ &c_r \in RM_r^n \Rightarrow c_r = [u', u' \oplus v'],\ u' \in RM_r^{n-1}, v' \in RM_{r-1}^{n-1}, \\ &t_j^+ = \left(v'_j \oplus b(\mathbf{w})_j^+, \min(\rho_j^0, \rho_j^1)\right).\end{aligned}$$

Note that

$$\begin{aligned}v \bmod 2 &= \tilde{\mathrm{RM}}\textsc{dec}(r-1, t^+) \\ &= \tilde{\mathrm{RM}}\textsc{dec}(r-1, v' \oplus \mathbf{b}^+(\mathbf{w}), \rho^+) \\ &= v' \oplus \tilde{\mathrm{RM}}\textsc{dec}(r-1, \mathbf{b}^+(\mathbf{w}), \rho^+) \\ &= v' \oplus \tilde{\mathrm{RM}}\textsc{dec}(r-1, t(\mathbf{w})^+).\end{aligned}$$

Now we compute the vector u. If $v_j = v'_j \oplus b_j^+(\mathbf{w}) \bmod 2$, then

$$t_j^- = \left(u'_j \oplus b_j^0(\mathbf{w}), (\rho_j^0 + \rho_j^1)/2\right).$$

Otherwise,

$$t_j^- = \left(u'_j \oplus b_j^0(\mathbf{w}) \oplus \textsc{Eval}(\rho_j^0 < \rho_j^1), |\rho_j^0 - \rho_j^1|/2\right).$$

Then we have

$$\begin{aligned}u \bmod 2 &= \tilde{\mathrm{RM}}\textsc{dec}(r, t^-) \\ &= \tilde{\mathrm{RM}}\textsc{dec}(r, u' \oplus \mathbf{b}^-(\mathbf{w}), \rho^-) \\ &= u' \oplus \tilde{\mathrm{RM}}\textsc{dec}(r, \mathbf{b}^-(\mathbf{w}), \rho^-) \\ &= u' \oplus \tilde{\mathrm{RM}}\textsc{dec}(r, t(\mathbf{w})^-).\end{aligned}$$

Hence we obtain:

$$\begin{aligned}
&\tilde{\text{RMDEC}}(r, c_r \oplus \mathbf{b}(\mathbf{w}), \rho) \\
&= [u \bmod 2, (u+v) \bmod 2] \\
&= [u' \oplus \tilde{\text{RMDEC}}(r, t(\mathbf{w})^-), u' \oplus \tilde{\text{RMDEC}}(r, t(\mathbf{w})^-) \oplus v' \oplus \tilde{\text{RMDEC}}(r-1, t(\mathbf{w})^+)] \\
&= [u', u' \oplus v'] \oplus [\tilde{\text{RMDEC}}(r, t(\mathbf{w})^-), \tilde{\text{RMDEC}}(r, t(\mathbf{w})^-) \oplus \tilde{\text{RMDEC}}(r-1, t(\mathbf{w})^+)] \\
&= c_r \oplus \tilde{\text{RMDEC}}(r, \mathbf{b}(\mathbf{w}), \rho).
\end{aligned}$$

□

Lemma A.2. If $\mathbf{d} \in BW_r^n$, then $(\mathbf{d}, \mathbf{d}) \in BW_r^{n+1}$.

Proof. Suppose $\mathbf{d} = \psi(c_r) + \phi \cdot \psi(c_{r+1}) + \cdots + \phi^{n-1-r} \cdot \psi(c_{n-1}) + \phi^{n-r} \cdot c_n$, where $c_i \in RM_i^n$. Hence,

$$\begin{aligned}
(\mathbf{d}, \mathbf{d}) &= (\psi(c_r) + \cdots + \phi^{n-r} \cdot c_n, \psi(c_r) + \cdots + \phi^{n-r} \cdot c_n) \\
&= (\psi(c_r), \psi(c_r)) + \cdots + \phi^{n-r} \cdot (c_n, c_n) \\
&= \psi(c_r, c_r) + \cdots + \phi^{n-r} \cdot (c_n, c_n),
\end{aligned}$$

noting that $(\psi(c_i), \psi(c_i)) = \psi(c_i, c_i)$. Also note that if $c_i \in RM_i^n$, then $(c_i, c_i) \in RM_i^{n+1}$. This proves that $(\mathbf{d}, \mathbf{d}) \in BW_r^{n+1}$. □

Lemma A.3. If $\mathbf{d}' \in BW_{r-1}^n$, then $(\mathbf{0}, \mathbf{d}') \in BW_r^{n+1}$.

Proof. Let $\mathbf{d}' = \psi(c'_{r-1}) + \phi \cdot \psi(c'_r) + \cdots + \phi^{n-r} \cdot \psi(c'_{n-1}) + \phi^{n-r+1} \cdot c'_n$, where $c'_i \in RM_i^n$.

$$\begin{aligned}
(\mathbf{0}, \mathbf{d}') &= (\psi(0), \psi(0) + \mathbf{d}') \\
&= (\psi(0), \psi(0) + \psi(c'_{r-1})) + \cdots + \phi^{n-r+1} \cdot (0, c'_n) \\
&= \psi(\underbrace{0, 0 \oplus c'_{r-1}}_{\in RM_r^{n+1}}) + \cdots + \phi^{n-r+1} \cdot (\underbrace{0, 0 + c'_n}_{\in \mathbb{Z}[\mathrm{i}]^{2^{n+1}}})
\end{aligned}$$

This shows that $(\mathbf{0}, \mathbf{d}') \in BW_r^{n+1}$. □

Lemma A.4. For any $a, b \in RM_r^n$ and $r < n$ we have $\psi(a \oplus b) = \psi(a) + \psi(b) + 2\mathbf{d}$ for some $\mathbf{d} \in BW_{r+1}^n$.

Proof. Let's first define the Schur product: for binary vectors $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^N$, we set

$$\mathbf{x} * \mathbf{y} = (x_1 y_1, \ldots, x_N y_N).$$

For $n = 1$, and $r = 0$, we have for $a, b \in RM_0^1 = \{(0,0), (1,1)\}$:

$$\psi(a \oplus b) = a \oplus b = a + b - 2(a * b).$$

So here $\mathbf{d} = -(a * b) \in \mathbb{Z}[\mathrm{i}]^2 = BW_1^1$.
Suppose that the hypothesis holds for n, for all $r < n$. Let $(a, b) \in RM_r^{n+1}$.
- If $r < n$, then write $a = (u, u \oplus v)$ and $b = (u', u' \oplus v')$, where $u, u' \in RM_r^n$ and $v, v' \in RM_{r-1}^n$.
Then

$$\begin{aligned}\psi(a \oplus b) &= \psi(u \oplus u', u \oplus u' \oplus v \oplus v') \\ &= \big(\psi(u \oplus u'), \psi(u \oplus u') + \psi(v \oplus v')\big)\end{aligned}$$

By inductive hypothesis

$$\begin{cases}\psi(u \oplus u') = \psi(u) + \psi(u') + 2\mathbf{d};\ \mathbf{d} \in BW_{r+1}^n \\ \psi(v \oplus v') = \psi(v) + \psi(v') + 2\mathbf{d}';\ \mathbf{d}' \in BW_r^n\end{cases}$$

So we can write

$$\begin{aligned}\psi(a \oplus b) &= (\psi(u) + \psi(u') + 2\mathbf{d}, \psi(u) + \psi(u') + 2\mathbf{d} + \psi(v) + \psi(v') + 2\mathbf{d}') \\ &= (\psi(u), \psi(u) + \psi(v)) + (\psi(u'), \psi(u') + \psi(v')) + 2(\mathbf{d}, \mathbf{d} + \mathbf{d}') \\ &= \psi(a) + \psi(b) + 2(\mathbf{d}, \mathbf{d} + \mathbf{d}')\end{aligned}$$

But $(\mathbf{d}, \mathbf{d}) \in BW_{r+1}^{n+1}$ and $(0, \mathbf{d}') \in BW_{r+1}^{n+1}$, so $(\mathbf{d}, \mathbf{d} + \mathbf{d}') \in BW_{r+1}^{n+1}$.
- If $r = n$, $BW_{n+1}^{n+1} = \mathbb{Z}[\mathrm{i}]^{2N}$, so the statement is trivially true. □

Proposition A.3. For $\boldsymbol{\lambda}_r \in BW_r^n$ and $\mathbf{w} \in \mathbb{C}^N$, we have

$$\textsc{SeqBW}(r, \boldsymbol{\lambda}_r + \mathbf{w}) = \boldsymbol{\lambda}_r + \textsc{SeqBW}(r, \mathbf{w}).$$

Proof. We will proceed by decreasing induction on r.
If $r \geq n$, then SeqBW is nothing more than the rounding function, and the property holds.
Now suppose that the hypothesis remains true for $r + 1$; let's prove it for r:

$$\begin{aligned}&\textsc{SeqBW}(r, \boldsymbol{\lambda}_r + \mathbf{w}) \\ &= \textsc{RMDec}(r, t(\boldsymbol{\lambda}_r + \mathbf{w}))) + \phi\, \textsc{SeqBW}\left(r+1, \frac{\boldsymbol{\lambda}_r + \mathbf{w} - \textsc{RMDec}(r, t(\boldsymbol{\lambda}_r + \mathbf{w}))}{\phi}\right) \\ &= \psi\left(\tilde{\textsc{RMDec}}(r, t(\boldsymbol{\lambda}_r + \mathbf{w}))\right) + \phi\, \textsc{SeqBW}\left(r+1, \frac{\boldsymbol{\lambda}_r + \mathbf{w} - \psi\left(\tilde{\textsc{RMDec}}(r, t(\boldsymbol{\lambda}_r + \mathbf{w}))\right)}{\phi}\right) \\ &\overset{(a)}{=} \psi\left(\tilde{\textsc{RMDec}}(r, c_r \oplus \mathbf{b}(\mathbf{w}), \rho)\right) + \phi\, \textsc{SeqBW}\left(r+1, \boldsymbol{\lambda}_{r+1} + \frac{\psi(c_r) + \mathbf{w} - \psi\left(\tilde{\textsc{RMDec}}(r, c_r \oplus \mathbf{b}(\mathbf{w}), \rho)\right)}{\phi}\right) \\ &= \psi\left(c_r \oplus \tilde{\textsc{RMDec}}(r, \mathbf{b}(\mathbf{w}), \rho)\right) + \phi \cdot \boldsymbol{\lambda}_{r+1} + \phi\, \textsc{SeqBW}\left(r+1, \frac{\psi(c_r) + \mathbf{w} - \psi\left(c_r \oplus \tilde{\textsc{RMDec}}(r, \mathbf{b}(\mathbf{w}), \rho)\right)}{\phi}\right) \\ &\overset{(b)}{=} \psi(c_r) + \psi\left(\tilde{\textsc{RMDec}}(r, \mathbf{b}(\mathbf{w}), \rho)\right) + 2\mathbf{d} + \phi \cdot \boldsymbol{\lambda}_{r+1} + \phi\, \textsc{SeqBW}\left(r+1, \frac{\mathbf{w} - \psi\left(\tilde{\textsc{RMDec}}(r, \mathbf{b}(\mathbf{w}), \rho)\right) - 2\mathbf{d}}{\phi}\right),\end{aligned}$$

where (a) follows from the fact that $\boldsymbol{\lambda}_r = \psi(c_r) + \phi \cdot \boldsymbol{\lambda}_{r+1}$ for $c_r \in RM_r^n$ and $\boldsymbol{\lambda}_{r+1} \in BW_{r+1}^n$, and (b) follows from Lemma A.4. Observe that $\frac{2\mathbf{d}}{\phi} = (1-\mathrm{i})\mathbf{d} \in BW_{r+1}^n$. By inductive hypothesis for $\textsc{SeqBW}(r+1, *)$:

$$
\begin{aligned}
&\textsc{SeqBW}(r, \boldsymbol{\lambda}_r + \mathbf{w}) \\
&= \boldsymbol{\lambda}_r + \psi\left(\tilde{\textsc{RMDec}}(r, \mathbf{b}(\mathbf{w}), \rho)\right) + 2\mathbf{d} - 2\mathbf{d} + \phi \cdot \textsc{SeqBW}\left(r+1, \frac{\mathbf{w} - \psi\left(\tilde{\textsc{RMDec}}(r, \mathbf{b}(\mathbf{w}), \rho)\right)}{\phi}\right) \\
&= \boldsymbol{\lambda}_r + \textsc{RMDec}(r, \mathbf{b}(\mathbf{w}), \rho) + \phi \cdot \textsc{SeqBW}\left(r+1, \frac{\mathbf{w} - \textsc{RMDec}(r, \mathbf{b}(\mathbf{w}), \rho)}{\phi}\right) \\
&= \boldsymbol{\lambda}_r + \textsc{SeqBW}(r, \mathbf{w}).
\end{aligned}
$$

□

A.6 Proof of Proposition 2.2

Lemma A.5. Let $\Phi_n = I_{\frac{n}{2}} \otimes \left[\begin{smallmatrix} 1 & -1 \\ 1 & 1 \end{smallmatrix}\right]$ for $n \geq 4$. Then $2\left(\Phi_n\right)^{-2}$ is an integer matrix.

Proof. It is sufficient to notice that $\Phi_n^{-2} = I_{\frac{n}{2}}^{-2} \otimes \left[\begin{smallmatrix} 1 & -1 \\ 1 & 1 \end{smallmatrix}\right]^{-2} = I_{\frac{n}{2}} \otimes \left[\begin{smallmatrix} 0 & 0.5 \\ -0.5 & 0 \end{smallmatrix}\right]$. □

Lemma A.6. Let $n = 2^{2\alpha}$ with $\alpha \geq 1$ and integer. The two matrices $2^{\alpha}\left(\mathbf{G}_{BW^n}\right)^{-1}\Phi_n^{-1}$ and $2^{\alpha}\left(\mathbf{G}_{BW^{2n}}\right)^{-1}$ are integer matrices.

Proof. The proof follows the induction principle on α. It is easy to verify that property holds for $\alpha = 1$ by direct calculation, i.e., $2 \cdot \left(\mathbf{G}_{BW^4}\right)^{-1} \cdot \Phi_4^{-1}$ and $2 \cdot \left(\mathbf{G}_{BW^8}\right)^{-1}$ are integer matrices. Suppose this property remains true until some integer α, i.e.,

$$
2^{\alpha} \cdot \left(\mathbf{G}_{BW^n}\right)^{-1} \cdot \Phi_n^{-1} \text{ and } 2^{\alpha} \cdot \left(\mathbf{G}_{BW^{2n}}\right)^{-1}
$$

are integer matrices, and let's prove it for $\alpha + 1$. Note that in this case the lemma must be proven for $4n$ and for $8n$. Using the fact that the inverse of a 2×2 block matrix $\left[\begin{smallmatrix} A & X \\ 0 & B \end{smallmatrix}\right]$ is simply $\left[\begin{smallmatrix} A^{-1} & -A^{-1}XB^{-1} \\ 0 & B^{-1} \end{smallmatrix}\right]$ one can write:

$$
2^{\alpha+1}\left(\mathbf{G}_{BW^{4n}}\right)^{-1} = \left[\begin{array}{c|c} 2 \cdot 2^{\alpha}\left(\mathbf{G}_{BW^{2n}}\right)^{-1} & -2^{\alpha}\left(\mathbf{G}_{BW^{2n}}\right)^{-1} \cdot 2\Phi_{2n}^{-1} \\ \hline 0 & 2^{\alpha}\left(\mathbf{G}_{BW^{2n}}\right)^{-1} \cdot 2\Phi_{2n}^{-1} \end{array}\right] \tag{A.1}
$$

Clearly, by hypothesis and using lemma A.5, each block in (A.1) represents an integer matrix. Now let's study the $8n$ case:

$$
2^{\alpha+1}\left(\mathbf{G}_{BW^{8n}}\right)^{-1} = \left[\begin{array}{c|c} 2^{\alpha+1}\left(\mathbf{G}_{BW^{4n}}\right)^{-1} & -2^{\alpha+1}\left(\mathbf{G}_{BW^{4n}}\right)^{-1}\Phi_{4n}^{-1} \\ \hline 0 & 2^{\alpha+1}\left(\mathbf{G}_{BW^{4n}}\right)^{-1}\Phi_{4n}^{-1} \end{array}\right]
$$

Note that $2^{\alpha+1}\left(\mathbf{G}_{BW^{8n}}\right)^{-1}$ an integer matrix if and only if $2^{\alpha+1}\left(\mathbf{G}_{BW^{4n}}\right)^{-1}\Phi_{4n}^{-1}$ is an integer matrix. Accordingly,

$$2^{\alpha+1}\left(\mathbf{G}_{BW^{4n}}\right)^{-1}\Phi_{4n}^{-1}=\left[\begin{array}{c|c}2^{\alpha+1}\left(\mathbf{G}_{BW^{2n}}\right)^{-1}\Phi_{2n}^{-1} & -2^{\alpha+1}\left(\mathbf{G}_{BW^{2n}}\right)^{-1}\Phi_{2n}^{-2}\\ \hline 0 & 2^{\alpha+1}\left(\mathbf{G}_{BW^{2n}}\right)^{-1}\Phi_{2n}^{-2}\end{array}\right]$$

Again, $2^{\alpha+1}\left(\mathbf{G}_{BW^{4n}}\right)^{-1}\Phi_{4n}^{-1}$ is an integer matrix if and only if $2^{\alpha+1}\left(\mathbf{G}_{BW^{2n}}\right)^{-1}\Phi_{2n}^{-2}$ is an integer matrix. As a consequence,

$$2^{\alpha+1}\left(\mathbf{G}_{BW^{2n}}\right)^{-1}\Phi_{2n}^{-2}=\left[\begin{array}{c|c}2^{\alpha}\left(\mathbf{G}_{BW^{n}}\right)^{-1}\Phi_{n}^{-1}\cdot 2\Phi_{n}^{-1} & -2^{\alpha}\left(\mathbf{G}_{BW^{n}}\right)^{-1}\Phi_{n}^{-1}\cdot 2\Phi_{n}^{-2}\\ \hline 0 & 2^{\alpha}\left(\mathbf{G}_{BW^{n}}\right)^{-1}\Phi_{n}^{-1}\cdot 2\Phi_{n}^{-2}\end{array}\right]$$

This proves the result. □

Consequently, we can now state our result as follows:

Proposition A.4. *For $n\geq 2$ and $k\geq\left\lfloor\frac{\log n}{2}\right\rfloor$, $2^k\mathbb{Z}^n\subseteq BW^n\subseteq\mathbb{Z}^n$.*

Proof. The second inclusion is obvious. Following proposition 2.1, the first inclusion is true if $2^k\left(\mathbf{G}_{BW^n}\right)^{-1}$ is an integer matrix. For $n=2$ the proposition is easily verified. Note that for any power-of-two integer $n\geq 4$, it has either the form $n=2^{2\alpha}$ or $n=2^{2\alpha+1}$ for some integer $\alpha\geq 1$. For both cases, $k=\left\lfloor\frac{\log n}{2}\right\rfloor=\alpha$, and that completes the proof using lemma A.6. □

A.7 Proof of Theorem 2.3

We start our proof by the following lemma adopted from [HKZ12]:

Lemma A.7. Let $\mathbf{z}$ be a vector of n independent standard Gaussian random variables. Fix any non-negative vector $\boldsymbol{\rho}\in\mathbb{R}_+^n$ and any vector $\boldsymbol{\beta}\in\mathbb{R}^n$. If $0\leq\gamma<1/(2\|\boldsymbol{\rho}\|_\infty)$ then

$$\mathbb{E}\left[\exp\left(\gamma\sum_{i=1}^{n}\rho_i z_i^2+\sum_{i=1}^{n}\beta_i z_i\right)\right]\leq\exp\left(\|\boldsymbol{\rho}\|_1\gamma+\frac{\|\boldsymbol{\rho}\|^2\gamma^2+\|\boldsymbol{\beta}\|^2/2}{1-2\|\boldsymbol{\rho}\|_\infty\gamma}\right)$$

Corollary A.1. Let X^n be a subgaussian vector in $\mathbb{R}^n$ with parameter s. Then

$$\mathbb{P}\left\{\|X^n\|^2>\frac{s^2}{2\pi}\cdot\left(n+2\sqrt{nt}+2t\right)\right\}\leq e^{-t}$$

Proof. By definition of X^n we have

$$\forall t\in\mathbb{R},\ \text{and}\ \forall\,\mathbf{u}\in\mathbb{R}^n:\|\mathbf{u}\|=1,\qquad\mathbb{E}\left[e^{2\pi t\langle X^n,\mathbf{u}\rangle}\right]\leq e^{\pi t^2 s^2}$$

Let's verify that for all $\boldsymbol{\alpha} \in \mathbb{R}^n$ and a fixed $\sigma > 0$, X^n satisfies the condition:

$$\mathbb{E}\left[e^{\langle \boldsymbol{\alpha}, X^n \rangle}\right] \leq e^{\|\boldsymbol{\alpha}\|^2 \sigma^2 / 2}$$

In fact, for any $\boldsymbol{\alpha} \in \mathbb{R}^n$, there exist $t \in \mathbb{R}$ and a unit vector $\mathbf{u} \in \mathbb{R}^n$ such that $\boldsymbol{\alpha} = 2\pi t \mathbf{u}$. Hence, for this t and $\mathbf{u}$ we have

$$\mathbb{E}\left[e^{\langle \boldsymbol{\alpha}, X^n \rangle}\right] \leq e^{\pi t^2 s^2} = e^{\frac{\|\boldsymbol{\alpha}\|^2 s^2}{4\pi}}$$

Now, take $\sigma = \dfrac{s}{\sqrt{2\pi}}$, (which is fix as s is) we get:

$$\mathbb{E}\left[e^{\langle \boldsymbol{\alpha}, X^n \rangle}\right] \leq e^{\frac{\|\boldsymbol{\alpha}\|^2 \sigma^2}{2}} \tag{A.2}$$

Let $\mathbf{z}$ be a vector of n independent standard Gaussian random variables (sampled independently from X^n). Then for any $\boldsymbol{\alpha} \in \mathbb{R}^n$,

$$\mathbb{E}\left[\exp\left(\mathbf{z}^T \boldsymbol{\alpha}\right)\right] = \exp\left(\|\boldsymbol{\alpha}\|^2 / 2\right).$$

Using the fact that $\mathbb{E}[X] = \sum_i \mathbb{E}[X|Y = A_i] \cdot \mathbb{P}\{Y = A_i\}$, we obtain for any $\lambda \in \mathbb{R}$ and $\epsilon > 0$:

$$\mathbb{E}\left[\exp\left(\lambda \mathbf{z}^T X^n\right)\right] \geq \mathbb{E}\left[\exp\left(\lambda \mathbf{z}^T X^n\right) \;\middle|\; \|X^n\|^2 > \epsilon\right] \cdot \mathbb{P}\left\{\|X^n\|^2 > \epsilon\right\}$$

But $\mathbb{E}\left[\exp\left(\mathbf{z}^T (\lambda X^n)\right)\right] = \exp\left(\|\lambda X^n\|^2 / 2\right)$; so given that $\|X^n\|^2 > \epsilon$ we get

$$\mathbb{E}\left[\exp\left(\lambda \mathbf{z}^T X^n\right)\right] \geq \exp\left(\frac{\lambda^2 \epsilon}{2}\right) \cdot \mathbb{P}\left\{\|X^n\|^2 > \epsilon\right\} \tag{A.3}$$

Moreover, using the law of total expectation, we obtain:

$$\begin{aligned} \mathbb{E}_{X^n}\left[\exp\left(\lambda \mathbf{z}^T X^n\right)\right] &= \mathbb{E}_{\mathbf{z}}\left[\mathbb{E}_{X^n}\left[\exp\left(\lambda \mathbf{z}^T X^n\right)\right] \;\middle|\; \mathbf{z}\right] \\ &\leq \mathbb{E}_{\mathbf{z}}\left[\exp\left(\frac{\lambda^2 \sigma^2}{2} \|\mathbf{z}\|^2\right)\right] \qquad \text{(by (A.2))} \end{aligned} \tag{A.4}$$

Now put $\boldsymbol{\rho} = (1, 1, \ldots, 1)$. By Lemma A.7 with $\boldsymbol{\beta} = \mathbf{0}$ we get:

$$\mathbb{E}\left[\exp\left(\gamma \sum_{i=1}^{n} z_i^2\right)\right] \leq \exp\left(\|\boldsymbol{\rho}\|_1 \gamma + \frac{\|\rho\|^2 \gamma^2}{1 - 2\|\boldsymbol{\rho}\|_\infty \gamma}\right)$$

where $0 \leq \gamma < 1/(2\|\boldsymbol{\rho}\|_\infty) = 1/2$; which is in fact

$$\mathbb{E}\left[\exp\left(\gamma \|\mathbf{z}\|^2\right)\right] \leq \exp\left(n \cdot \gamma + \frac{n \cdot \gamma^2}{1 - 2\gamma}\right) \tag{A.5}$$

Combining (A.3), (A.4) and (A.5):

$$\exp\left(\frac{\lambda^2\epsilon}{2}\right)\cdot\mathbb{P}\left\{\|X^n\|^2>\epsilon\right\}\leq\mathbb{E}_{\mathbf{z}}\left[\exp\left(\frac{\lambda^2\sigma^2}{2}\|\mathbf{z}\|^2\right)\right] \tag{A.6}$$
$$\leq\exp\left(n\cdot\gamma+\frac{n\cdot\gamma^2}{1-2\gamma}\right)$$

where we set $\gamma=\dfrac{\lambda^2\sigma^2}{2}$; and hence

$$\mathbb{P}\left\{\|X^n\|^2>\epsilon\right\}\leq\exp\left(-\epsilon\gamma/\sigma^2+n\cdot\gamma+\frac{n\cdot\gamma^2}{1-2\gamma}\right)$$

(always under the condition that $0\leq\gamma<\frac{1}{2}$). Choose:

$$\epsilon=\sigma^2\cdot(n+\tau)\text{ and }\gamma=\frac{1}{2}\cdot\left(1-\sqrt{\frac{n}{n+2\tau}}\right)<\frac{1}{2}.$$

Consequently, we obtain:

$$\mathbb{P}\left\{\|X^n\|^2>\sigma^2\cdot(n+\tau)\right\}\leq\exp\left(-\frac{n}{2}\left(1+\frac{\tau}{n}-\sqrt{1+\frac{2\tau}{n}}\right)\right) \tag{A.7}$$
$$=\exp\left(-\frac{n}{2}\cdot h\left(\frac{\tau}{n}\right)\right)$$

where $h(x)=1+x-\sqrt{1+2x}$ defined from $\mathbb{R}_+$ to $\mathbb{R}_+$ with inverse $h^{-1}(y)=\sqrt{2y}+y$. Set $\tau=2\sqrt{nt}+2t$ so that

$$\frac{\tau}{n}=2\sqrt{\frac{t}{n}}+2\frac{t}{n}=h^{-1}\left(\frac{2t}{n}\right)$$

which leads to

$$\mathbb{P}\left\{\|X^n\|^2>\sigma^2\cdot(n+2\sqrt{nt}+2t)\right\}\leq e^{-t}$$ □

Corollary A.2. Let X^n be a subgaussian vector in $\mathbb{R}^n$ with parameter s. Then $\forall\epsilon>0$:

$$\mathbb{P}\left\{\|X^n\|>\sigma\sqrt{n}\cdot\sqrt{1+2\epsilon+2\epsilon^2}\right\}\leq e^{-n\epsilon^2}$$

Proof. From Corollary 2.3, we know that for a general n-dimensional subgaussian X^n with parameter s, we have

$$\mathbb{P}\left\{\|X^n\|^2>\frac{s^2}{2\pi}n\cdot\left(1+2\sqrt{\frac{t}{n}}+2\frac{t}{n}\right)\right\}\leq e^{-t}$$

Set $\epsilon=\sqrt{\dfrac{t}{n}}$ and $\sigma=\dfrac{s}{\sqrt{2\pi}}$ to complete the required result. □

Appendix B

FrodoKEM simulations

B.1 Efficiency of E8.Encode and E8.Decode

In order to analyse the efficiency of our Encoding and Decoding algorithms, we count the number of operations needed during calculations. As Table B.1 shows, we can distinguish two cases. The first one is when the function f is calculated on each iteration of Algorithm 4. This is about four times slower than the original Frodo.Encode. In the second case, we can increase the speed of our Encoding function by about two times, by computing f via a lookup table of size 2×256 bytes. Note that this does not imply that the whole algorithm is twice or four times slower than the original FrodoKEM, since the encoding and decoding functions have small costs compared to other building-blocks algorithms.

	$B = 2$	$B = 3$	$B = 4$
Frodo.Encode	$64S + 64P$	$128S + 64P$	$192S + 64P$
Frodo.Decode	$128S + 128P + 64R + 192M$	$192S + 192P + 64R + 256M$	$256S + 256P + 64R + 320M$
TOTAL	$192S + 192P + 64R + 192M$	$320S + 256P + 64R + 256M$	$448S + 320P + 64R + 320M$
E8.Encode	$168S + 64M$	$176S + 64M$	$184S + 64M$
E8.Decode	$928S + 495P + 256R + 192M$	$992S + 495P + 256R + 320M$	$1056S + 495P + 256R + 448M$
TOTAL	$1096S + 495P + 256R + 256M$	$1168S + 495P + 256R + 384M$	$1240S + 495P + 256R + 512M$
When computing f via a lookup table			
E8.Encode	$64S$	$72S$	$80S$
E8.Decode	$640 \cdot S + 256 \cdot P + 256 \cdot R + 192 \cdot M$	$704 \cdot S + 256 \cdot P + 256 \cdot R + 320 \cdot M$	$768 \cdot S + 256 \cdot P + 256 \cdot R + 448 \cdot M$
TOTAL	$704 \cdot S + 544 \cdot P + 256 \cdot R + 256 \cdot M$	$776 \cdot S + 544 \cdot P + 256 \cdot R + 384 \cdot M$	$848 \cdot S + 544 \cdot P + 256 \cdot R + 512 \cdot M$

Table B.1: Comparison of the number of operations for Encoding / Decoding in FrodoKEM vs the modified version. S, P, R and M denote respectively the operations Sum, Product, Rounding and Modulo. Note that the number of operations of Frodo.Encode / Frodo.Decode refers to Algorithms 1 and 2 in [N+20].

B.2 Calculating the error probability

B.2.1 The distribution of one entry of $\mathbf{E}'''$

We consider here our modified FrodoKEM for which the error standard deviation is σ. Recall from Equation (3.8) that the entries of $\mathbf{E}'''$ are a sum of products of elements distributed according to χ:

$$\forall\, 0 \leq i,j \leq 7,\ E'''_{i,j} = \sum_{k=0}^{n-1} \left(S'_{i,k} E_{k,j} - E'_{i,k} S_{k,j} \right) + E''_{i,j}.$$

The distribution of the product of $S'_{i,k}E_{k,j}$ and $E'_{i,k}S_{k,j}$ can be efficiently computed for any i, j and k, by brute-forcing over all the support of χ. We use this result to calculate the probability generating function of the sum of $2n$ products, plus the initial one that corresponds to $E''_{i,j}$. This is done using Lemma A.1.

B.2.2 The distribution of the sum of two, four and eight entries of $\mathbf{E}'''$

After calculating the exact distribution of each entry of $\mathbf{E}'''$, we now need to calculate the distribution of a sum of 2 and 8 entries of $\mathbf{E}'''$, and this is due to Equation (3.10) illustrated below:

$$8 \cdot 112 \cdot \mathbb{P}\left\{ E'''_{0,0} + E'''_{1,1} \geq \beta \right\} + 8 \cdot 128 \cdot \mathbb{P}\left\{ E'''_{0,0} + \cdots + E'''_{7,7} \geq 2\beta \right\}.$$

The distribution of $E'''_{0,0} + E'''_{1,1}$ can be obtained from the probability generating function which is equal to $G_2 = G_{E'''_{0,0}} \cdot G_{E'''_{1,1}}$. This product can be calculated efficiently. Now regarding the distribution of the sum $E'''_{0,0} + \cdots + E'''_{7,7}$, the product $G_8 = \prod_{i=0}^{7} G_{E'''_{i,i}}$ would take a long time to compile (around 4 days). To solve this problem, we first calculate $G_4 = \prod_{i=0}^{3} G_{E'''_{i,i}}$ that represents the probability generating function of a sum of 4 entries of $\mathbf{E}'''$, and then we use the following lemma:

Lemma B.1. Let $f(X) = a_0 + a_1 X + \cdots + a_n X^n$ with $a_i \in]0,1[$ satisfying $a_0 \geq a_1 \geq \cdots \geq a_n$, and let $F(X) = [f(X)]^2$. For an even number $i_0 \in [0, 2n]$ we have

$$\text{Coeff}\left(F, X^{i_0}\right) < \text{Coeff}^2\left(f, X^{i_0/2}\right) + i_0 \cdot \text{Coeff}\left(f, X^{i_0/2+1}\right)$$

Proof.

$$\begin{aligned}
\text{Coeff}\left(f^2, X^{i_0}\right) &= \text{Coeff}^2\left(f, X^{\frac{i_0}{2}}\right) + 2\text{Coeff}\left(f, X^0\right)\text{Coeff}\left(f, X^{i_0}\right) + \cdots + 2\text{Coeff}\left(f, X^{\frac{i_0}{2}-1}\right)\text{Coeff}\left(f, X^{\frac{i_0}{2}+1}\right) \\
&\leq \text{Coeff}^2\left(f, X^{\frac{i_0}{2}}\right) + 2\left[\text{Coeff}\left(f, X^{i_0}\right) + \text{Coeff}\left(f, X^{i_0-1}\right) + \cdots + \text{Coeff}\left(f, X^{\frac{i_0}{2}+1}\right)\right] \\
&\leq \text{Coeff}^2\left(f, X^{\frac{i_0}{2}}\right) + 2\left[\frac{i_0}{2} \cdot \text{Coeff}\left(f, X^{\frac{i_0}{2}+1}\right)\right] \\
&= \text{Coeff}^2\left(f, X^{\frac{i_0}{2}}\right) + i_0 \cdot \text{Coeff}\left(f, X^{\frac{i_0}{2}+1}\right)
\end{aligned}$$

□

Lemma B.1 will help us calculate the coefficients of G_8 given only G_4. It is not hard to

verify that the coefficients of $G_4(X)$ decrease when the powers of X increase. That's why, using Remark 3.2, this property holds also for G_8. Therefore, we can estimate the coefficients of G_8 beyond 2^{13} and hence calculate the required probability $\mathbb{P}\left\{E'''_{0,0} + \cdots + E'''_{7,7} \geq 2\beta\right\}$. In fact,

$$\mathbb{P}\left\{E'''_{0,0} + \cdots + E'''_{7,7} \geq 2\beta\right\} \leq \underbrace{\mathbb{P}\left\{E'''_{0,0} + \cdots + E'''_{7,7} \in \left[\left[2^{13}; 2^{13} + 1024\right]\right]\right\}}_{\text{can be calculated efficiently using Lemma B.1}} + \mathbb{P}\left\{E'''_{0,0} + \cdots + E'''_{7,7} \geq 2^{13} + 1025\right\}$$

The above term $\mathbb{P}\left\{E'''_{0,0} + \cdots + E'''_{7,7} \in \left[\left[2^{13}; 2^{13} + 1024\right]\right]\right\}$ is bounded by a constant cst. Moreover, the number of terms beyond $2^{13} + 1024$ is less than 2^{20}. Thence,

$$\mathbb{P}\left\{E'''_{0,0} + \cdots + E'''_{7,7} \geq 2\beta\right\} \leq cst + 2^{20} \cdot \underbrace{\mathbb{P}\left\{E'''_{0,0} + \cdots + E'''_{7,7} = 2^{13} + 1025\right\}}_{\text{can be calculated efficiently using Lemma B.1}}.$$

Appendix C

KyberKEM simulations and proofs

C.1 Conjugation function's properties

In this section we give some properties of the conjugation function derived from Definition 4.4.

Lemma C.1. Let $\mathbf{a}(Y) \in S = \mathbb{Z}[Y]/(Y^{n_0}+1)$ and recall from Subsection 4.4.1 that $Y = X^L$ for $n = n_0 L$. Then

$$\operatorname{conj}\left(a_0 + a_1 Y + \cdots + a_{n_0-1} Y^{n_0-1}\right) = a_0 - a_{n_0-1} Y - \cdots - a_1 Y^{n_0-1}.$$

Moreover, if the coefficients of $\mathbf{a}(Y)$ are sampled independently from a well defined distribution, then they will be identically distributed as the coefficients of $\operatorname{conj}(\mathbf{a})(Y)$.

Proof. Observe that $\operatorname{conj}\left(\zeta^{iL}\right) = -\zeta^{n-iL}$ for $i = 0, \ldots, n_0 - 1$. □

Proposition C.1. Let $\mathbf{a}(Y), \mathbf{b}(Y) \in S = \mathbb{Z}[Y]/(Y^{n_0}+1)$ and $\boldsymbol{\lambda} \in C\left(\mathrm{VR}_{E_8}^{(1)} \cup \mathrm{VR}_{E_8}^{(2)}\right)$. Then

$$\langle \mathbf{a} \cdot \mathbf{b}, \boldsymbol{\lambda} \rangle = \langle \mathbf{a}, \operatorname{conj}(\mathbf{b}) \cdot \boldsymbol{\lambda} \rangle.$$

Proof. Let's compare $\langle \mathbf{a} \cdot \mathbf{b}, \boldsymbol{\lambda} \rangle$ and $\langle \mathbf{a}, \operatorname{conj}(\mathbf{b}) \cdot \boldsymbol{\lambda} \rangle$ separately. At first,

$$(\mathbf{a} \cdot \mathbf{b})(Y) = \left(\sum_{j=0}^{n_0-1} a_j Y^j\right)\left(\sum_{k=0}^{n_0-1} b_k Y^k\right) = \sum_{j=0}^{n_0-1}\sum_{k=0}^{n_0-1} a_j b_k Y^{j+k}.$$

Let $j + k = i + n_0 \cdot \delta_{i,j}$, where $\delta_{i,j} = 1$ if $i - j < 0$ and 0 otherwise. So $k = i - j \bmod n_0$. Using the identity $Y^{n_0} = -1$, we obtain:

$$\langle \mathbf{a} \cdot \mathbf{b}, \boldsymbol{\lambda} \rangle = \sum_{i=0}^{n_0-1} (\mathbf{a} \cdot \mathbf{b})_i \lambda_i = \sum_{i=0}^{n_0-1}\sum_{j=0}^{n_0-1} a_j b_{i-j \bmod n_0} (-1)^{\delta_{i,j}} \lambda_i.$$

Similarly, we have that

$$\langle \mathbf{a}, \mathrm{conj}(\mathbf{b} \cdot \boldsymbol{\lambda}) \rangle = \sum_{i=0}^{n_0-1} \sum_{j=0}^{n_0-1} a_i \lambda_j \mathrm{conj}(\mathbf{b})_{i-j \bmod n_0} (-1)^{\delta_{i,j}}.$$

Recall that $\mathrm{conj}(\mathbf{b})_0 = b_0$ and $\mathrm{conj}(\mathbf{b})_k = -b_{n_0-k}$ if $k \neq 0$. So,

$$\begin{aligned}\langle \mathbf{a}, \mathrm{conj}(\mathbf{b} \cdot \boldsymbol{\lambda}) \rangle &= \sum_{j=0}^{n_0-1} \sum_{i=0}^{n_0-1} a_j \lambda_i \mathrm{conj}(\mathbf{b})_{j-i \bmod n_0} (-1)^{\delta_{j,i}} \\ &= \sum_{i=0}^{n_0-1} \sum_{j=0}^{n_0-1} a_j \lambda_i \cdot \epsilon_{i,j} \cdot b_{i-j \bmod n_0} (-1)^{\delta_{j,i}},\end{aligned}$$

where $\epsilon_{i,j} = 1$ if $i = j$, and -1 otherwise. Note that, if $i \neq j$, then

$$\epsilon_{i,j}(-1)^{\delta_{j,i}} = -(-1)^{\delta_{j,i}} = (-1)^{\delta_{i,j}};$$

and if $i = j$, then

$$\epsilon_{i,j}(-1)^{\delta_{j,i}} = 1 = (-1)^{\delta_{i,j}}.$$

This concludes the proof. □

C.2 Computing the distribution of $\langle (\tilde{\mathbf{s}}', \tilde{\mathbf{s}}), W_{\boldsymbol{\lambda},\kappa} \rangle$

In this section, we explain how to compute the distribution of $\langle (\tilde{\mathbf{s}}', \tilde{\mathbf{s}}), W_{\boldsymbol{\lambda},k} \rangle$ for KyberKEM, where $W_{\boldsymbol{\lambda},k}$ is defined in equation (4.6).

Recall that $(\tilde{\mathbf{s}}', \tilde{\mathbf{s}}) = (s'_0, s'_1, \ldots, s'_{nd}, s_0, s_1, \ldots, s_{nd})$, and hence $\langle (\tilde{\mathbf{s}}', \tilde{\mathbf{s}}), W_{\boldsymbol{\lambda},k} \rangle = Z_1 + \cdots + Z_{2Ld}$ is the sum of $2Ld$ i.i.d. random variables. Note that $Z_1 \sim \left\langle (s_0, s_1, \ldots, s_7), \mathbf{e}^{(0)} \cdot \boldsymbol{\lambda} \right\rangle$ and $\forall\, i = 2, \ldots, 2Ld$, $Z_i \sim Z_1$. So we can focus on obtaining the PDF of Z_1 since the PDF of the sum $Z_1 + \cdots + Z_{2Ld}$ can be computed from the PDF of Z_1 using the probability generating function defined in Appendix A.1 and Lemma A.1.

C.2.1 PDF of the dot product $Z_1 \sim \langle (s_0, s_1, \ldots, s_7), \mathbf{e}^{(0)} \cdot \boldsymbol{\lambda} \rangle$ for $\boldsymbol{\lambda} \in \mathbf{VR}_{E_8}^{(1)}$

We aim to find the probability density function of $\langle (s_0, s_1, \ldots, s_7), \mathbf{e}^{(0)} \cdot \boldsymbol{\lambda} \rangle$ for each $\boldsymbol{\lambda} \in \mathrm{VR}_{E_8}^{(1)}$. For simplicity, we will take the vector $\boldsymbol{\lambda} = (1, 1, 0, 0, 0, 0, 0, 0)$. The computation for other values $\boldsymbol{\lambda} \in \mathrm{VR}_{E_8}^{(1)}$ can be performed in a similar way.
In polynomial form, for $\mathbf{e}^{(0)} \equiv (e_0, \ldots, e_7)$, the product $\mathbf{e}^{(0)} \cdot \boldsymbol{\lambda}$ modulo $X^8 + 1$ is equal to

$$(e_0 - e_7) + (e_0 + e_1)X + (e_1 + e_2)X^2 + (e_2 + e_3)X^3 + (e_3 + e_4)X^4 + (e_4 + e_5)X^5 + (e_5 + e_6)X^6 + (e_6 + e_7)X^7.$$

In vector form, this is equivalent to

$$(e_0 - e_7, e_0 + e_1, e_1 + e_2, e_2 + e_3, e_3 + e_4, e_4 + e_5, e_5 + e_6, e_6 + e_7).$$

It follows that the dot product $\langle (s_0, s_1, \ldots, s_7), \mathbf{e}^{(0)} \cdot \boldsymbol{\lambda} \rangle$ can be rewritten as

$$(e_0-e_7)s_0+(e_0+e_1)s_1+(e_1+e_2)s_2+(e_2+e_3)s_3+(e_3+e_4)s_4+(e_4+e_5)s_5+(e_5+e_6)s_6+(e_6+e_7)s_7. \tag{C.1}$$

Using the fact that

$$\mathbb{P}\left\{\langle (s_0, s_1, \ldots, s_7), \mathbf{e}^{(0)} \cdot \boldsymbol{\lambda} \rangle = C\right\} = \sum_{(e_0,\ldots,e_7)} \mathbb{P}\left\{\langle (s_0, s_1, \ldots, s_7), \mathbf{e}^{(0)} \cdot \boldsymbol{\lambda} \rangle = C \;\middle|\; (e_0, e_1, \ldots, e_7)\right\} \cdot \mathbb{P}\left\{(e_0, e_1, \ldots, e_7)\right\}$$

we can obtain the distribution of Z_1 by computing

$$\mathbb{P}\left\{\langle (s_0, s_1, \ldots, s_7), \mathbf{e}^{(0)} \cdot \boldsymbol{\lambda} \rangle = C \;\middle|\; (e_0, e_1, \ldots, e_7)\right\} \cdot \mathbb{P}\left\{(e_0, e_1, \ldots, e_7)\right\}. \tag{C.2}$$

for each value of $(e_0, \ldots, e_7) \in \{-k, k\}^8$. Note that conditioning on $(e_0, \ldots, e_7)$, the sum in (C.1) is a sum of independent random variables, so one can calculate it using the probability generating function.

Let $G_s(X)$ be the probability generating function of s_i, which is the same for all $i = 0, \ldots, 7$, and k the parameter of the centered binomial distribution ψ_k. Let $p = \psi_k$ for simplicity. The function G_s can be expressed as

$$G_s(X) = p(-k)X^{-k} + p(-(k-1))X^{-(k-1)} + \cdots + p(k-1)X^{(k-1)} + p(k)X^k.$$

For instance, the probability generating function of $s_0(e_0 - e_7)$ for a fixed integer value of (e_0, e_7), denoted as F_0, can be written as

$$\begin{aligned} F_0(X) &= p(-k)X^{-k(e_0-e_7)} + p(-(k-1))X^{-(k-1)(e_0-e_7)} + \cdots + p(k)X^{k(e_0-e_7)} \\ &\overset{(*)}{=} p(-k)X^{-k|e_0-e_7|} + p(-(k-1))X^{-(k-1)|e_0-e_7|} + \cdots + p(k)X^{k|e_0-e_7|} \\ &= X^{-k|e_0-e_7|} \underbrace{\left[p(-k) + p(-k+1)X^{|e_0-e_7|} + \cdots + p(k)X^{2k|e_0-e_7|}\right]}_{G_0(X)} \end{aligned} \tag{C.3}$$

The equality $(*)$ is true since $p(-k) = p(k)$. Our aim is to let the exponent of X that is outside the parenthesis (i.e. the exponent of $X^{-k|e_0-e_7|}$) to be negative, and the exponents of X of $G_0(X)$ to be positive.

In the same way as (C.3), for $i \geq 1$, we define F_i to be the probability generating function of $s_i(e_{i-1} + e_i)$ and $G_i(X) = p(-k) + p(-k+1)X^{|e_{i-1}+e_i|} + \cdots + p(k)X^{2k|e_{i-1}+e_i|}$. Finally, the probability generating function F of the expression (C.1) is the product of each $F_i(X)$,

$i = 0, \ldots, 7$. In more details,

$$F(X) = \prod_{i=0}^{7} F_i(X) = X^{\underbrace{-k\,(|e_0 - e_7| + \cdots + |e_6 + e_7|)}_{\texttt{cst}}} \prod_{i=0}^{7} G_i(X). \tag{C.4}$$

Each function $G_i(X)$ can be seen as a probability generating function of some distribution. For example, $G_0(X)$ represents the probability generating function of the random variable that takes values in $[[0, |e_0 - e_7|, \ldots, 2k|e_0 - e_7|]]$ with probabilities $\{p(-k), p(-k+1), \ldots, p(k)\}$ respectively. From this fact, we create the array `arr0` that contains the probability of having $i * |e_0 - e_7|)$ for $i = 0, \ldots, 2k$. These probabilities are placed in `arr0` as follows:

$$\texttt{arr0}(1 + \texttt{i} * |e_0 - e_7|) = \mathbb{P}\{\texttt{i} * |e_0 - e_7|\},\ \forall \texttt{i} = 0, \ldots, 2k.$$

This array will have length of $\texttt{shift\_for\_arr} = 1 + 2k(k+k)$. In the same way we construct the arrays $\texttt{arr1}, \ldots, \texttt{arr7}$ that correspond to the functions $G_1, \ldots, G_7$. The convolution product of these 8 arrays will contain the coefficients of the probability generating function of $\prod_{i=0}^{7} G_i(X)$.

We aim to create an array `scal_pdf` to store the probability distribution of the sum (C.1). In Matlab, we shift the indices since negative indices are not allowed. The dot product in expression (C.1) can have values in $[[-(2k) \cdot k \cdot 8\,;\, (2k) \cdot k \cdot 8]]$; the shift is defined to be `main_shift=(VR1_card*k)*k*8+1`, where `VR1_card` is the number of non-zero components of Voronoi-relevant vectors of $\mathrm{VR}_{E_8}^{(1)}$, which is 2. Finally, `scal_pdf[main_shift+i]` represents the probability that (C.1) takes the value `i`. We can now apply the formula (C.2) in order to obtain `scal_pdf`.

Remark C.1. These simulations have been done $|\mathrm{VR}_{E_8}^{(1)}| = 112$ times since for each $\boldsymbol{\lambda} \in \mathrm{VR}_{E_8}^{(1)}$ we have a different distribution of the dot product.

C.2.2 PDF of the dot product $Z_1 \sim \langle (s_0, s_1, \ldots, s_7), \mathbf{e}^{(0)} \cdot \boldsymbol{\lambda} \rangle$ for $\boldsymbol{\lambda} \in \mathbf{VR}_{E_8}^{(2)}$

The simulations for $\boldsymbol{\lambda} \in \mathrm{VR}_{E_8}^{(2)}$ follow the same strategy as in Subsection C.2.1. We repeat the simulations $|\mathrm{VR}^{(2)}| = 128$ times.

C.3 Distribution of $Z_1 + \cdots + Z_{2Ld}$

Recall that the probability generating function $F(X)$ of Z_1 is defined in (C.4). Hence, to obtain the probability generating function of the sum $Z_1 + \cdots + Z_{2Ld}$, we need to multiply $F(X)$ by itself $2Ld = 2^6 d$ times. For the sake of efficiency, we raise $F(X)$ to the power of d, take the result and raise it to the power-of-two, and repeat the latter 5 times. We have used the Wolfram `MATHEMATICA` software for this part of the computation.

Appendix D

Reconciliation technique for KyberKEM using Barnes-Wall lattices

D.1 Modification of KyberKEM with reconciliation using BW^{n_0}

In this section, we consider again the modified scheme in Table 4.4, where the lattices Λ_1 and Λ_2 are replaced by products of Barnes-Wall lattices BW^{n_0} with n_0 being a power-of-two. To obtain a nested sequence $\Lambda_3 \subseteq \Lambda_2 \subseteq \Lambda_1$, we need to verify the inclusions of the chosen lattices Λ_1, Λ_2 and Λ_3. Accordingly, using Proposition 2.2 which states that for $n \geq 2$ and $k \geq \left\lfloor \frac{\log n}{2} \right\rfloor$, $2^k\mathbb{Z}^n \subseteq BW^n \subseteq \mathbb{Z}^n$, we choose

$$\Lambda_1 = \left(\frac{q}{2^p} BW^{n_0}\right)^L \supseteq \Lambda_2 = (\beta BW^{n_0})^L \supseteq \Lambda_3 = (q\mathbb{Z}^{n_0})^L, \tag{D.1}$$

where q is a power-of-two integer, $n = n_0 \times L = 256$ and $\beta = \frac{q}{2^k}$ for $k \geq \left\lfloor \frac{\log n_0}{2} \right\rfloor$. In order to obtain the best possible minimum distance for the lattice Λ_2, we choose the equality $k = \left\lfloor \frac{\log n_0}{2} \right\rfloor$. This imposes to have $p \geq k$ so that $\Lambda_2 \subseteq \Lambda_1$. Referring to Subsection 2.2.4 and recalling that $\mathrm{Vol}(BW^{n0}) = \sqrt{(n_0/2)^{n_0/2}}$, the number of reconciliation bits transmitted is

$$\log\left(\frac{\mathrm{Vol}(\Lambda_2)}{\mathrm{Vol}(\Lambda_1)}\right) = \log\left(\frac{\mathrm{Vol}(\beta(BW^{n_0})^L)}{\mathrm{Vol}(\frac{q}{2^p}(BW^{n_0})^L)}\right) = \log\left(\frac{\beta^n}{q^n} \times 2^{np}\right) = \log\left(\left(\frac{2^p}{2^k}\right)^n\right) = n(p-k).$$

Moreover, the size of the key (in bits) is

$$\log\left(\frac{\mathrm{Vol}(\Lambda_3)}{\mathrm{Vol}(\Lambda_2)}\right) = \log\left(\frac{\mathrm{Vol}\left(q\mathbb{Z}^n\right)}{\mathrm{Vol}\left(\beta(BW^{n_0})^L\right)}\right) = \log\left(\frac{q^n}{\beta^n\left(\frac{n_0}{2}\right)^{\frac{n}{4}}}\right) = \log\left(\frac{2^{kn}}{\left(\frac{n_0}{2}\right)^{\frac{n}{4}}}\right) = n\left(k - \frac{\log\left(\frac{n_0}{2}\right)}{4}\right).$$

For most dimensions, the number of key bits is not very suitable for the proposed application, since key sizes in cryptography are usually powers-of-two (See Table D.1).

As we did in Chapter 4, Equation (4.4), we can establish a polynomial splitting for n_0 (see Subsection 4.4.1) and get the sufficient condition to obtain $\hat{\mathbf{k}} = \mathbf{k}$ in Table 4.4, that is

$$Q_{\frac{q}{2^k}\left(1-\frac{1}{2^{p-k}}\right)BW^{n_0}}\left((\mathbf{v}-\mathbf{v}')^{(\kappa)}\right) = 0 \text{ for } \kappa = 0, \ldots, L-1, \tag{D.2}$$

which is satisfied whenever $(\mathbf{v}-\mathbf{v}')^{(\kappa)} \in \underbrace{\frac{q}{2^k}\left(1-\frac{1}{2^{p-k}}\right)}_{C}\mathcal{V}\left(BW^{n_0}\right) = C \cdot \mathcal{V}\left(BW^{n_0}\right)$ for $\kappa = 0, \ldots, L-1$. Note that the calculation of the Voronoi-relevant vectors for BW^{n_0} for large n_0 requires a huge amount of computation. That's why we adopt the *sphere bound* approach illustrated in Proposition D.1 below in order to cover more general cases. However, the upper bound for the error probability provided by the sphere bound is not as tight as the bound based on Voronoi-relevant vectors.

Proposition D.1. For any n-dimensional lattice Λ, the ball of radius $d_{\min}/2$ is included in the Voronoi region. In other words, for any $\mathbf{x} \in \mathbb{R}^n$, we have

$$\mathbb{P}\left\{\mathbf{x} \notin \mathcal{V}(\Lambda)\right\} \leq \mathbb{P}\left\{\|\mathbf{x}\| \geq \frac{d_{\min}}{2}\right\}.$$

A visual representation of the sphere bound is in Figure D.1.

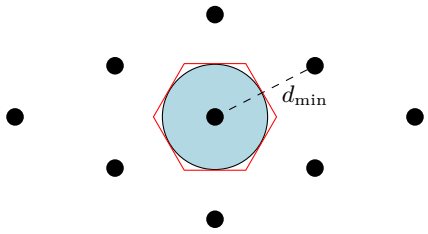

Figure D.1: The sphere of radius $\frac{d_{\min}}{2}$ is inside the Voronoi region.

Recalling that the minimum distance of BW^{n_0} is $\sqrt{n_0/2}$ (see Subsection 2.2.4) and that for a product lattice Λ^L, $d_{\min}\left(\Lambda^L\right) = d_{\min}\left(\Lambda\right)$, we find that $d_{\min}\left(C \cdot BW^{n_0}\right) = \frac{q}{2^k}\left(1-\frac{1}{2^{p-k}}\right)\sqrt{\frac{n_0}{2}}$ is greater than $d_{\min}\left(\frac{q}{2}\left(1-\frac{1}{2^{p'-1}}\right)E_8\right) = \frac{q}{2}\left(1-\frac{1}{2^{p'-1}}\right)\sqrt{2}$ (see Equation (4.4)) for certain values of n_0, p' and p, as shown in Table D.1 and Table D.3. We note that although the minimum distance of BW^{n_0} is increasing in n_0, the inclusion $\Lambda_3 \subseteq \Lambda_2$ in equation (D.1) imposes a scaling factor β which decreases with n_0, so that there is almost no gain in terms of minimum distance when using higher-dimensional lattices.

These are the most promising choices for Λ_2 for which we want to estimate the error probability:

n_0	$k = \lfloor \frac{\log n_0}{2} \rfloor$	$d_{\min}\left(\frac{q}{2}\left(1-\frac{1}{2^{p'-1}}\right)E_8\right) \leq d_{\min}\left(\frac{q}{2^k}\left(1-\frac{1}{2^{p-k}}\right)BW^{n_0}\right)$		Number of reconcilitation bits	Number of key bits
		p'	p		
16	2	≥ 2	$\geq p'+1$	$256(p-2)$	320
32	2	$=2$ ≥ 3	≥ 3 ≥ 4	512	256
64	3	≥ 2	$\geq p'+2$	$256(p-3)$	448
128	3	$=2$ ≥ 3	≥ 4 ≥ 5	512	384
256	4	≥ 2	$\geq p'+3$	$256(p-4)$	576

Table D.1: Values of n_0 and the corresponding minimum values of p' and p for which $d_{\min}\left(\frac{q}{2^k}\left(1-\frac{1}{2^{p-k}}\right)BW^{n_0}\right)$ is greater than or equal to $d_{\min}\left(\frac{q}{2}\left(1-\frac{1}{2^{p'-1}}\right)E_8\right)$. The required reconciliation and key bits are illustrated in the last two columns.

By applying Proposition D.1, condition (D.2) is satisfied if for all $\kappa = 0, \ldots, L-1$,

$$\left\|(\mathbf{v}-\mathbf{v}')^{(\kappa)}\right\| \leq C \cdot \frac{d_{\min}\left(BW^{n_0}\right)}{2} = \frac{q}{2^k}\left(1-\frac{1}{2^{p-k}}\right)\frac{\sqrt{n_0/2}}{2} = \frac{q\sqrt{n_0}}{2^{k+1}\sqrt{2}}\left(1-\frac{1}{2^{p-k}}\right),$$

which is equivalent to

$$\left\|\underbrace{\left(\mathbf{e}_1\mathbf{s}'_1+\cdots+\mathbf{e}_d\mathbf{s}'_d\right)^{(\kappa)}-\left(\mathbf{e}'_1\mathbf{s}_1+\cdots+\mathbf{e}'_d\mathbf{s}_d\right)^{(\kappa)}+\mathbf{e}''^{(\kappa)}}_{\boldsymbol{\omega}^{(\kappa)}}\right\| \leq \frac{q\sqrt{n_0}}{2^{k+1.5}}\left(1-\frac{1}{2^{p-k}}\right);\ d\in\{2,3,4\}. \tag{D.3}$$

Taking the term $\mathbf{e}_\ell\mathbf{s}'_\ell$ from some $\ell \in \{1,\ldots,d\}$, it is a polynomial multiplication modulo X^n+1. It can be identified to

$$(e_{\ell,0}, e_{\ell,1}, \ldots, e_{\ell,n-1}) \cdot (s'_{\ell,0}, s'_{\ell,1}, \ldots, s'_{\ell,n-1}) \mod X^n+1. \tag{D.4}$$

Since for $0 \leq i,j \leq n-1$ the product of $e_{\ell,i} \cdot s'_{\ell,j}$ is bounded by k_1^2 in absolute value (k_1 is the parameter of the centered binomial distribution ψ_{k_1}), then by Proposition 2.4 and Remark 2.4, the resulting vector $\mathbf{e}_\ell\mathbf{s}'_\ell$ in (D.4) has subgaussian components with parameter $\sqrt{n}k_1^2\sqrt{2\pi}$.[1] As a consequence, since by Theorem 2.4 $\mathbf{e}''$ is subgaussian with parameter $\sqrt{k_2\pi}$, each component of the vector $\boldsymbol{\omega}^{(\kappa)}$ in (D.3) is subgaussian with parameter $\sqrt{\pi}\sqrt{2dn(k_1^4+k_1^2k_2^2)+k_2}$, and hence using Proposition 2.5, the whole vector $\boldsymbol{\omega}^{(\kappa)}$ is subgaussian with parameter $\rho = \sqrt{n_0}\left(\sqrt{\pi}\sqrt{2dn(k_1^4+k_1^2k_2^2)+k_2}\right)$. From condition (D.2), an error occurs while decoding if there exists $\kappa \in \{0,\ldots,L-1\}$ such that

$$\left\|\boldsymbol{\omega}^{(\kappa)}\right\| > \frac{q\sqrt{n_0}}{2^{k+1.5}}\left(1-\frac{1}{2^{p-k}}\right).$$

[1]When expanding the product $(e_{\ell,0}, e_{\ell,1}, \ldots, e_{\ell,n-1}) \cdot (s'_{\ell,0}, s'_{\ell,1}, \ldots, s'_{\ell,n-1}) \mod X^n+1$, one can see that it produces an independent sum of $n = 256$ terms in each component, each term of the sum is subgaussian of parameter $k_1^2\sqrt{2\pi}$, and that's why we can use Proposition 2.4.

In addition, applying Theorem 2.3 to $\boldsymbol{\omega}^{(\kappa)}$, we get $\forall \epsilon > 0$ that

$$\mathbb{P}\left\{\left\|\boldsymbol{\omega}^{(\kappa)}\right\| > \frac{\rho}{\sqrt{2\pi}}\sqrt{n_0}\cdot\sqrt{1+2\epsilon+2\epsilon^2}\right\} \leq e^{-n_0\epsilon^2}.$$

That's why, we impose the condition

$$\frac{q\sqrt{n_0}}{2^{k+1.5}}\left(1-\frac{1}{2^{p-k}}\right) \geq \frac{\rho}{\sqrt{2\pi}}\sqrt{n_0}\cdot\sqrt{1+2\epsilon+2\epsilon^2} \tag{D.5}$$

for which the error probability will be bounded by

$$P_e \leq L\cdot\mathbb{P}\left\{\left\|\boldsymbol{\omega}^{(0)}\right\| > \frac{q\sqrt{n_0}}{2^{k+1.5}}\left(1-\frac{1}{2^{p-k}}\right)\right\} \leq L\cdot e^{-n_0\epsilon^2}.$$

Equation (D.5) leads to the following table:

	$n_0 = 16$ $L = 16$ $k = 2$	$n_0 = 32$ $L = 8$ $k = 2$	$n_0 = 64$ $L = 4$ $k = 3$	$n_0 = 128$ $L = 2$ $k = 3$	$n_0 = 256$ $L = 1$ $k = 4$
$d = 2$	$\epsilon \geq 2.488 \mid \rho = 2454$	$\epsilon \geq 1.753 \mid \rho = 3470$	$\epsilon \geq 1.235 \mid \rho = 4908$	$\epsilon \geq 0.870 \mid \rho = 6941$	$\epsilon \geq 0.613 \mid \rho = 9816$
$k_1 = 3$ and $k_2 = 2$	$q = 2^{22} \Longrightarrow p = 3$	$q = 2^{21} \Longrightarrow p = 3$	$q = 2^{22} \Longrightarrow p \in \{4,5\}$	$q = 2^{22} \Longrightarrow p \in \{4,\ldots,9\}$	$q = 2^{24} \Longrightarrow p = 5$
Goal: $P_e \leq 2^{-139}$	$q = 2^{21} \Longrightarrow p \geq 4$	$q = 2^{20} \Longrightarrow p \geq 4$	$q = 2^{21} \Longrightarrow p \geq 6$	$q = 2^{21} \Longrightarrow p \geq 10$	$q = 2^{23} \Longrightarrow p \geq 6$
$d = 3$	$\epsilon \geq 2.697 \mid \rho = 1571$	$\epsilon \geq 1.901 \mid \rho = 2222$	$\epsilon \geq 1.340 \mid \rho = 3143$	$\epsilon \geq 0.945 \mid \rho = 4445$	$\epsilon \geq 0.666 \mid \rho = 6287$
$k_1 = 2$ and $k_2 = 2$	$q = 2^{20} \Longrightarrow p \in \{3,\ldots,8\}$	$q = 2^{21} \Longrightarrow p = 3$	$q = 2^{22} \Longrightarrow p = 4$	$q = 2^{22} \Longrightarrow p = 4$	$q = 2^{23} \Longrightarrow p \in \{5,6\}$
Goal: $P_e \leq 2^{-164}$	$q = 2^{19} \Longrightarrow p \geq 9$	$q = 2^{20} \Longrightarrow p \geq 4$	$q = 2^{21} \Longrightarrow p \geq 5$	$q = 2^{21} \Longrightarrow p \geq 5$	$q = 2^{22} \Longrightarrow p \geq 7$
$d = 4$	$\epsilon \geq 2.776 \mid \rho = 1815$	$\epsilon \geq 1.958 \mid \rho = 2566$	$\epsilon \geq 1.380 \mid \rho = 3630$	$\epsilon \geq 0.973 \mid \rho = 5133$	$\epsilon \geq 0.686 \mid \rho = 7260$
$k_1 = 2$ and $k_2 = 2$	$q = 2^{21} \Longrightarrow p = 3$	$q = 2^{21} \Longrightarrow p = 3$	$q = 2^{22} \Longrightarrow p = 4$	$q = 2^{22} \Longrightarrow p \in \{4,5\}$	$q = 2^{23} \Longrightarrow p \in \{5,6,7\}$
Goal: $P_e \leq 2^{-174}$	$q = 2^{20} \Longrightarrow p \geq 4$	$q = 2^{20} \Longrightarrow p \geq 4$	$q = 2^{21} \Longrightarrow p \geq 5$	$q = 2^{21} \Longrightarrow p \geq 6$	$q = 2^{22} \Longrightarrow p \geq 8$

Table D.2: The values of n_0 and the corresponding values of q and p for which the error probability bounds matches the ones in KyberKEM.

This implies that in order to improve upon KyberKEM, we need q to be at least 2^{19}, with the number of reconciliation bits transmitted being at least 1762 bits. Note that the value n_0 cannot be smaller than 16 because otherwise the size of the key will be less than 256 bits[2].

Although the previous discussion fails to find any advantage in choosing Λ_2 to be a product of higher-dimensional Barnes-Wall lattices, this might be due to the fact that the sphere bound is not tight enough. In the next section, we focus on the case $n_0 = 16$ and try to obtain a better error probability bound.

[2]We note that although the lattice E_8 is similar to the lattice BW^8 in the sense of Definition 2.5, we have taken a different generator matrix for E_8.

n_0	$k = \lfloor \frac{\log n_0}{2} \rfloor$	p'	$d_{\min}\left(\frac{q}{2}\left(1-\frac{1}{2^{p'-1}}\right)E_8\right)$	p	$d_{\min}\left(\frac{q}{2^k}\left(1-\frac{1}{2^{p-k}}\right)BW^{n_0}\right)$
16	2	2	$0.3535q$	3	$0.3535q$
		3	$0.5303q$	4	$0.5303q$
		4	$0.6187q$	5	$0.6187q$
		5	$0.6629q$	6	$0.6629q$
		∞	$0.7071q$	∞	$0.7071q$
32	2	2	$0.3535q$	3	$0.5q$
		3	$0.5303q$	4	$0.75q$
		4	$0.6187q$	5	$0.875q$
		5	$0.6629q$	6	$0.9375q$
		∞	$0.7071q$	∞	1
64	3	2	$0.3535q$	4	$0.3535q$
		3	$0.5303q$	5	$0.5303q$
		4	$0.6187q$	6	$0.6187q$
		5	$0.6629q$	7	$0.6629q$
		∞	$0.7071q$	∞	$0.7071q$
128	3	2	$0.3535q$	4	$0.5q$
		3	$0.5303q$	5	$0.75q$
		4	$0.6187q$	6	$0.875q$
		5	$0.6629q$	7	$0.9375q$
		∞	$0.7071q$	∞	1
256	4	2	$0.3535q$	5	$0.3535q$
		3	$0.5303q$	6	$0.5303q$
		4	$0.6187q$	7	$0.6187q$
		5	$0.6629q$	8	$0.6629q$
		∞	$0.7071q$	∞	$0.7071q$

Table D.3: Explicit values of $d_{\min}\left(\frac{q}{2}\left(1-\frac{1}{2^{p'-1}}\right)E_8\right)$ and $d_{\min}\left(\frac{q}{2^k}\left(1-\frac{1}{2^{p-k}}\right)BW^{16}\right)$ under some instances of p' and p.

D.2 KyberKEM with BW^{16}

In this section, we will work with BW^{16} defined in Subsection 2.2.4 with the generator matrix given in (2.4) in order to obtain a better scaling. With the new choice of the generator matrix, we can have a better inclusion and better minimum distance compared to Section D.1. In fact, the lattice Λ_2 defined in inclusion (D.1) as $\left(\frac{q}{4}BW^{16}\right)^{16}$ can be replaced by $\Lambda_2 = \left(\frac{q}{2}BW^{16}\right)^{16}$, thus doubling the scalar β. Accordingly, we make the following choice:

$$\Lambda_1 = \left(\frac{q}{2^p}BW^{16}\right)^{16} \supseteq \Lambda_2 = \left(\frac{q}{2}BW^{16}\right)^{16} \supseteq \Lambda_3 = \left(q\mathbb{Z}^{16}\right)^{16}.$$

The constant C here is $C = \frac{q}{2}\left(1 - \frac{1}{2^{p-1}}\right)$. The number of reconciliation bits transmitted is $n(p-1)$ and the size of the key (in bits) is $n + 64 = 1.25n = 320$.

Note that BW^{16} and E_8 have the same minimal distance equal to $\sqrt{2}$. However, in this section we will not use the sphere bound and we will consider a union bound over Voronoi-relevant vectors similarly to Subsection 4.4.3. Unfortunately, estimating the bound of Subsection 4.4.3 for this lattice would require very cumbersome computations. In this section, we will use a different approach inspired by [ADPS16b].

Our main objective is to calculate the error probability given by

$$P_e \leq \sum_{\kappa=0}^{L-1} \mathbb{P}\left\{\exists \boldsymbol{\lambda} \in C \cdot \mathrm{VR}_{BW^{16}} \,:\, \langle(\tilde{\mathbf{s}'}, \tilde{\mathbf{s}}), W_{\boldsymbol{\lambda},\kappa}\rangle > \frac{\|\boldsymbol{\lambda}\|^2}{2} - \langle \mathbf{e}''^{(\kappa)}, \boldsymbol{\lambda}\rangle\right\}$$

(see Equation (4.7)). In this section, we take $L = 16$. Using Boole's inequality, the above bound can be reduced to:

$$\begin{aligned}
P_e &\leq \sum_{\kappa=0}^{L-1}\sum_{i=1}^{3} \mathbb{P}\left\{\exists \boldsymbol{\lambda} \in C \cdot \mathrm{VR}^{(i)}_{BW^{16}} \,:\, \langle(\tilde{\mathbf{s}'}, \tilde{\mathbf{s}}), W_{\boldsymbol{\lambda},\kappa}\rangle > \frac{\|\boldsymbol{\lambda}\|^2}{2} - \langle \mathbf{e}''^{(\kappa)}, \boldsymbol{\lambda}\rangle\right\} \\
&\leq L\sum_{i=1}^{3} \mathbb{P}\left\{\exists \boldsymbol{\lambda} \in C \cdot \mathrm{VR}^{(i)}_{BW^{16}} \,:\, \langle(\tilde{\mathbf{s}'}, \tilde{\mathbf{s}}), W_{\boldsymbol{\lambda},0}\rangle > \frac{\|\boldsymbol{\lambda}\|^2}{2} - \langle \mathbf{e}''^{(0)}, \boldsymbol{\lambda}\rangle\right\} \\
&\leq L\sum_{i=1}^{3} \sum_{\boldsymbol{\lambda} \in C\cdot \mathrm{VR}^{(i)}_{BW^{16}}} \mathbb{P}\left\{\langle(\tilde{\mathbf{s}'}, \tilde{\mathbf{s}}), W_{\boldsymbol{\lambda},0}\rangle > \frac{\|\boldsymbol{\lambda}\|^2}{2} - \langle \mathbf{e}''^{(0)}, \boldsymbol{\lambda}\rangle\right\} = L\sum_{i=1}^{3} \sum_{\boldsymbol{\lambda} \in C\cdot \mathrm{VR}^{(i)}_{BW^{16}}} P_{\boldsymbol{\lambda}}.
\end{aligned}$$

Recall from Subsection 2.2.4 that there are three types of Voronoi-relevant vectors in this case. Observing that the value of $P_{\boldsymbol{\lambda}}$ is the same for vectors of the same type, it is enough to consider a fixed vector in $\mathrm{VR}^{(i)}_{BW^{16}}$ for each type $i = 1, 2, 3$, such as

$$\begin{aligned}
\boldsymbol{\lambda}_1 &= (1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0) \\
\boldsymbol{\lambda}_2 &= (\tfrac{1}{2},\tfrac{1}{2},\tfrac{1}{2},\tfrac{1}{2},\tfrac{1}{2},\tfrac{1}{2},\tfrac{1}{2},\tfrac{1}{2},0,0,0,0,0,0,0,0) \\
\boldsymbol{\lambda}_3 &= (\tfrac{1}{2},\tfrac{1}{2},\tfrac{1}{2},\tfrac{1}{2},\tfrac{1}{2},\tfrac{1}{2},\tfrac{1}{2},\tfrac{1}{2},1,0,0,0,0,0,0,0),
\end{aligned}$$

and hence obtain the following bound:

$$P_e \leq L \cdot \sum_{i=1}^{3} \left|\mathrm{VR}_{BW^{16}}^{(i)}\right| \mathbb{P}\left\{\langle(\tilde{\mathbf{s}'}, \tilde{\mathbf{s}}), W_{\boldsymbol{\lambda}_i,0}\rangle > C \cdot \frac{\|\boldsymbol{\lambda}_i\|^2}{2} - \langle \mathbf{e}''^{(0)}, \boldsymbol{\lambda}_i\rangle\right\}. \tag{D.6}$$

In order to work with integer-valued vectors, we define the following:

- $\boldsymbol{\lambda}_1 = \boldsymbol{\lambda}_1{}' = (1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$
- $\boldsymbol{\lambda}_2 = \frac{1}{2}\boldsymbol{\lambda}_2{}' = \frac{1}{2}(1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0)$
- $\boldsymbol{\lambda}_3 = \frac{1}{2}\boldsymbol{\lambda}_3{}' = \frac{1}{2}(1,1,1,1,1,1,1,1,2,0,0,0,0,0,0,0)$

In order to go further in our analysis, we will need the following lemma:

Lemma D.1 (Lemma D.2 in [ADPS16b])**.** Let X^n be a subgaussian vector with parameter σ. Then for any $\tau > 0$ and any vector $\mathbf{v} \in \mathbb{R}^n$ we have:

$$\mathbb{P}\left\{\langle X^n, \mathbf{v}\rangle > \|\mathbf{v}\|\tau\frac{\sigma}{\sqrt{2\pi}}\right\} \leq e^{-\tau^2/2}.$$

We can apply the Lemma with $X^n = (\tilde{\mathbf{s}'}, \tilde{\mathbf{s}})$, $\sigma = \sqrt{k_1\pi}$ and $\mathbf{v} = W_{\mathbf{v}_i,0}$ for $\mathbf{v}_i \in \mathrm{VR}_{BW^{16}}^{(i)}$. Then for any $\tau_i > 0$ and any $\mathbf{v}_i \in \mathrm{VR}_{BW^{16}}^{(i)}$ we have:

$$\mathbb{P}\{\langle(\tilde{\mathbf{s}'}, \tilde{\mathbf{s}}), W_{\mathbf{v}_i,0}\rangle > \|W_{\mathbf{v}_i,0}\|\sqrt{k_1/2}\cdot\tau_i\} \leq e^{-\tau_i^2/2}. \tag{D.7}$$

Using the law of total probability, we can restate the error probability in (D.6) as follows: for $\boldsymbol{\lambda}_i \in \mathrm{VR}_{BW^{16}}^{(i)}$ and constants C_i we have

$$P_e \leq L\cdot\sum_{i=1}^{3}\left|\mathrm{VR}_{BW^{16}}^{(i)}\right|\cdot\left[\mathbb{P}\left\{\|W_{\boldsymbol{\lambda}_i,0}\| \geq C_i\right\} + \mathbb{P}\left\{\langle(\tilde{\mathbf{s}'}, \tilde{\mathbf{s}}), W_{\boldsymbol{\lambda}_i,0}\rangle > C\cdot\frac{\|\boldsymbol{\lambda}_i\|^2}{2} - \langle \mathbf{e}''^{(0)}, \boldsymbol{\lambda}_i\rangle \;\middle|\; \|W_{\boldsymbol{\lambda}_i,0}\| < C_i\right\}\right] \tag{D.8}$$

By Replacing $\boldsymbol{\lambda}$ with $\boldsymbol{\lambda}'$ in (D.8) we obtain:

$$\begin{aligned}
P_e \leq\; & L\cdot\left|\mathrm{VR}_{BW^{16}}^{(1)}\right|\mathbb{P}\left\{\|W_{\boldsymbol{\lambda}_1',0}\| \geq C_1\right\} + L\cdot\left|\mathrm{VR}_{BW^{16}}^{(1)}\right|\cdot\mathbb{P}\left\{\langle(\tilde{\mathbf{s}'}, \tilde{\mathbf{s}}), W_{\boldsymbol{\lambda}_1',0}\rangle > C\cdot\frac{\|\boldsymbol{\lambda}_1'\|^2}{2} - \langle \mathbf{e}''^{(0)}, \boldsymbol{\lambda}_1'\rangle \;\middle|\; \|W_{\boldsymbol{\lambda}_1',0}\| < C_1\right\} \\
& + L\cdot\left|\mathrm{VR}_{BW^{16}}^{(2)}\right|\mathbb{P}\left\{\|W_{\boldsymbol{\lambda}_2',0}\| \geq 2C_2\right\} + L\cdot\left|\mathrm{VR}_{BW^{16}}^{(2)}\right|\cdot\mathbb{P}\left\{\langle(\tilde{\mathbf{s}'}, \tilde{\mathbf{s}}), W_{\boldsymbol{\lambda}_2',0}\rangle > C\cdot\frac{\|\boldsymbol{\lambda}_2'\|^2}{4} - \langle \mathbf{e}''^{(0)}, \boldsymbol{\lambda}_2'\rangle \;\middle|\; \|W_{\boldsymbol{\lambda}_2',0}\| < 2C_2\right\} \\
& + L\cdot\left|\mathrm{VR}_{BW^{16}}^{(3)}\right|\mathbb{P}\left\{\|W_{\boldsymbol{\lambda}_3',0}\| \geq 2C_3\right\} + L\cdot\left|\mathrm{VR}_{BW^{16}}^{(3)}\right|\cdot\mathbb{P}\left\{\langle(\tilde{\mathbf{s}'}, \tilde{\mathbf{s}}), W_{\boldsymbol{\lambda}_3',0}\rangle > C\cdot\frac{\|\boldsymbol{\lambda}_3'\|^2}{4} - \langle \mathbf{e}''^{(0)}, \boldsymbol{\lambda}_3'\rangle \;\middle|\; \|W_{\boldsymbol{\lambda}_3',0}\| < 2C_3\right\}
\end{aligned}$$

Let C_1' be a lower bound for $C\cdot\frac{\|\boldsymbol{\lambda}_1'\|^2}{2} - \langle \mathbf{e}''^{(0)}, \boldsymbol{\lambda}_1'\rangle$, and C_i' $(i = 2,3)$ be a lower bound for

$C \cdot \frac{\|\boldsymbol{\lambda}_i'\|^2}{4} - \langle \mathbf{e}''^{(0)}, \boldsymbol{\lambda}_i' \rangle$. Then we can write

$$\begin{aligned} P_e \leq\ & L \cdot \left|\mathrm{VR}_{BW^{16}}^{(1)}\right| \mathbb{P}\left\{\|W_{\boldsymbol{\lambda}_1',0}\| \geq C_1\right\} + L \cdot \left|\mathrm{VR}_{BW^{16}}^{(1)}\right| \cdot \mathbb{P}\left\{\langle(\tilde{\mathbf{s}}', \tilde{\mathbf{s}}), W_{\boldsymbol{\lambda}_1',0}\rangle > C_1' \;\middle|\; \|W_{\boldsymbol{\lambda}_1',0}\| < C_1\right\} \\ & + L \cdot \left|\mathrm{VR}_{BW^{16}}^{(2)}\right| \mathbb{P}\left\{\|W_{\boldsymbol{\lambda}_2',0}\| \geq 2C_2\right\} + L \cdot \left|\mathrm{VR}_{BW^{16}}^{(2)}\right| \cdot \mathbb{P}\left\{\langle(\tilde{\mathbf{s}}', \tilde{\mathbf{s}}), W_{\boldsymbol{\lambda}_2',0}\rangle > C_2' \;\middle|\; \|W_{\boldsymbol{\lambda}_2',0}\| < 2C_2\right\} \\ & + L \cdot \left|\mathrm{VR}_{BW^{16}}^{(3)}\right| \mathbb{P}\left\{\|W_{\boldsymbol{\lambda}_3',0}\| \geq 2C_3\right\} + L \cdot \left|\mathrm{VR}_{BW^{16}}^{(3)}\right| \cdot \mathbb{P}\left\{\langle(\tilde{\mathbf{s}}', \tilde{\mathbf{s}}), W_{\boldsymbol{\lambda}_3',0}\rangle > C_3' \;\middle|\; \|W_{\boldsymbol{\lambda}_3',0}\| < 2C_3\right\} \end{aligned} \tag{D.9}$$

Next, we aim to find suitable values for the constants C_1', C_2', C_3' in (D.9).

Using the Cauchy-Schwarz inequality, and depending on how many zeros are in $\boldsymbol{\lambda}_i'$, one can bound $\langle \mathbf{e}''^{(0)}, \boldsymbol{\lambda}_i' \rangle$ as follows:

- $\langle \mathbf{e}''^{(0)}, \boldsymbol{\lambda}_1' \rangle \leq \left|\langle \mathbf{e}''^{(0)}, \boldsymbol{\lambda}_1' \rangle\right| \leq \sqrt{2} k_2 \|\boldsymbol{\lambda}_1'\|$. In fact,

  $$\begin{aligned} \langle \mathbf{e}''^{(0)}, \boldsymbol{\lambda}_1' \rangle &= \langle (e_0, \ldots, e_{15}), (1,1,0,\ldots,0) \rangle \\ &= \langle (e_0, e_1, 0, \ldots, 0), (1,1,0,\ldots,0) \rangle \\ &\leq \|(e_0, e_1, 0, \ldots, 0)\| \cdot \|(1,1,0,\ldots,0)\| \\ &\leq \sqrt{2} k_2 \left\|\boldsymbol{\lambda}_1'\right\|. \end{aligned}$$

  This implies that we can choose $C_1' = C \cdot \frac{\|\boldsymbol{\lambda}_1'\|^2}{2} - \sqrt{2} k_2 \|\boldsymbol{\lambda}_1'\|$

- $\langle \mathbf{e}''^{(0)}, \boldsymbol{\lambda}_2' \rangle \leq \left|\langle \mathbf{e}''^{(0)}, \boldsymbol{\lambda}_2' \rangle\right| \leq \sqrt{8} k_2 \|\boldsymbol{\lambda}_2'\|$. This implies that $C_2' = C \cdot \frac{\|\boldsymbol{\lambda}_2'\|^2}{4} - 2\sqrt{2} k_2 \|\boldsymbol{\lambda}_2'\|$.

- $\langle \mathbf{e}''^{(0)}, \boldsymbol{\lambda}_3' \rangle \leq \left|\langle \mathbf{e}''^{(0)}, \boldsymbol{\lambda}_3' \rangle\right| \leq \sqrt{9} k_2 \|\boldsymbol{\lambda}_3'\|$. This implies that $C_3' = C \cdot \frac{\|\boldsymbol{\lambda}_3'\|^2}{4} - 3 k_2 \|\boldsymbol{\lambda}_3'\|$.

Now for some $\tau_i > 0$ set

$$C_1' = C_1 \sqrt{k_1/2} \cdot \tau_1 \,;\; C_2' = 2C_2 \sqrt{k_1/2} \cdot \tau_2 \text{ and } C_3' = 2C_3 \sqrt{k_1/2} \cdot \tau_3.$$

Then, for $L = 16$ we obtain from (D.9) and (D.7):

$$\begin{aligned} P_e &\leq 16 \cdot 480 \cdot \mathbb{P}\left\{\|W_{\boldsymbol{\lambda}_1',0}\| \geq C_1\right\} + 16 \cdot 3839 \cdot \mathbb{P}\left\{\|W_{\boldsymbol{\lambda}_2',0}\| \geq 2C_2\right\} + 16 \cdot 61441 \cdot \mathbb{P}\left\{\|W_{\boldsymbol{\lambda}_3',0}\| \geq 2C_3\right\} \\ &\quad + 16 \cdot 480 \cdot e^{-\tau_1^2/2} + 16 \cdot 3839 \cdot e^{-\tau_2^2/2} + 16 \cdot 61441 \cdot e^{-\tau_3^2/2} \\ &\leq 2^{13} \cdot \mathbb{P}\left\{\|W_{\boldsymbol{\lambda}_1',0}\| \geq C_1\right\} + 2^{16} \cdot \mathbb{P}\left\{\|W_{\boldsymbol{\lambda}_2',0}\| \geq 2C_2\right\} + 2^{20} \cdot \mathbb{P}\left\{\|W_{\boldsymbol{\lambda}_3',0}\| \geq 2C_3\right\} \\ &\quad + 2^{13} \cdot e^{-\tau_1^2/2} + 2^{16} \cdot e^{-\tau_2^2/2} + 2^{20} \cdot e^{-\tau_3^2/2}. \end{aligned}$$

In order to obtain the best probability trade-off, we need to choose C_i and τ_i such that

$$\mathbb{P}\left\{\|W_{\boldsymbol{\lambda}_i',0}\| \geq C_1\right\} \approx e^{-\tau_i^2/2};\ i = 1, 2, 3.$$

The distribution of the norm $\|W_{\boldsymbol{\lambda}_i',0}\|$ is calculated via computer simulations for $d \in \{2, 3, 4\}$ by

	Parameters: $q = 2^{11}$ and $p = 5$			
	(C_1, τ_1)	(C_2, τ_2)	(C_3, τ_3)	Resulting P_e
$d = 2$	—	—	—	—
$d = 3$	$(70.71, 13.52)$	$(165.83, 11.52)$	$(213.30, 13.40)$	2^{-77}
$d = 4$	$(77.78, 12.29)$	$(178.88, 10.68)$	$(228.03, 12.53)$	2^{-65}

Table D.4: Error probability bounds for different KyberKEM levels.

observing that

$$\|W_{\boldsymbol{\lambda}'_i,0}\|^2 = \sum_{j=0}^{2Ld-1} \|\mathbf{e}^{(j)} \cdot \boldsymbol{\lambda}'_i\|^2.$$

We calculated the probability bound for KyberKEM-768 and KyberKEM-1024. In the case of KyberKEM-512, since the centered binomial distribution k_1 is equal to 3, the calculations are out of reach. In fact it requires $(2k_1 + 1)^{16} \approx 2^{45}$ operations. Our results are presented in Table D.4.

In conclusion, we were not able to obtain a good error probability bound for $q = 2^{11}$. This may be due to our sub-optimal error bound technique or to the fact that the choice of the lattice BW^{16} is ill-suited for this application. In fact, the key rate is larger than needed. We remark that in the first version of the NewHope paper [ADPS16b], where similar bounds based on Lemma D.1 were used, the error bound was of the same order (2^{-60}), which prevented the authors from proving CCA-security.

Bibliography

[A+20] Roberto Avanzi et al. CRYSTALS-Kyber algorithm specifications and supporting documentation. *NIST PQC Round*, 2020. `https://pq-crystals.org/kyber/data/kyber-specification-round3.pdf"`.

[AAB+] 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. `https://cryptojedi.org/papers/newhopenist-20171128.pdf`.

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H) IBE in the standard model. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 553–572. Springer, 2010.

[ACKS20] Divesh Aggarwal, Yanlin Chen, Rajendra Kumar, and Yixin Shen. Improved (provable) algorithms for the shortest vector problem via bounded distance decoding. *arXiv preprint arXiv:2002.07955*, 2020.

[ADPS16a] Erdem Alkim, Leo Ducas, Thomas Pöppelmann, and Peter Schwabe. NewHope without reconciliation. *IACR Cryptology ePrint Archive*, page 1157, 2016.

[ADPS16b] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange-a new hope. In *USENIX Security Symposium*, 2016.

[ADRSD15] Divesh Aggarwal, Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. Solving the shortest vector problem in 2^n time using discrete gaussian sampling. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 733–742, 2015.

[ADSD15] Divesh Aggarwal, Daniel Dadush, and Noah Stephens-Davidowitz. Solving the Closest Vector Problem in 2^n Time–The Discrete Gaussian Strikes Again! In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 563–582. IEEE, 2015.

[AG11] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In *International Colloquium on Automata, Languages, and Programming*, pages 403–415. Springer, 2011.

[AGL+10a] Carlos Aguilar, Philippe Gaborit, Patrick Lacharme1, Julien Schrek, and Gilles Zémor. Noisy Diffie-Hellman protocols or code-based key exchange and encryption without masking, 2010. `https://rump2010.cr.yp.to/fae8cd8265978675893352329786cea2.pdf`.

[AGL+10b] Carlos Aguilar, Philippe Gaborit, Patrick Lacharme1, Julien Schrek, and Gilles Zémor. Noisy Diffie-Hellman protocols, Recent results session at Post-Quantum Cryptography-3rd international workshop, PQCrypto, 2010. `https://www.yumpu.com/en/document/read/53051354/noisy-diffie-hellman-protocols`.

[AGV09] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *Theory of Cryptography Conference*, pages 474–495. Springer, 2009.

[AGVW17] Martin R Albrecht, Florian Göpfert, Fernando Virdia, and Thomas Wunderer. Revisiting the expected cost of solving usvp and applications to lwe. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 297–322. Springer, 2017.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 99–108, 1996.

[Ajt98] Miklós Ajtai. The shortest vector problem in L_2 is *NP*-hard for randomized reductions. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 10–19, 1998.

[AKS01] Miklós Ajtai, Ravi Kumar, and Dandapani Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 601–610, 2001.

[ALNSD20] Divesh Aggarwal, Jianwei Li, Phong Q Nguyen, and Noah Stephens-Davidowitz. Slide reduction, revisited—filling the gaps in SVP approximation. In *Annual International Cryptology Conference*, pages 274–295. Springer, 2020.

[APS15] Martin R Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

[AR05] Dorit Aharonov and Oded Regev. Lattice problems in NP ∩ coNP. *Journal of the ACM (JACM)*, 52(5):749–765, 2005.

[Ari09] Erdal Arikan. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE Transactions on information Theory*, 55(7):3051–3073, 2009.

[ASD17] Divesh Aggarwal and Noah Stephens-Davidowitz. Just Take the Average! An Embarrassingly Simple 2^n-Time Algorithm for SVP (and CVP). *arXiv preprint arXiv:1709.01535*, 2017.

[AV00] Dakshi Agrawal and Alexander Vardy. Generalized minimum distance decoding in Euclidean space: Performance analysis. *IEEE Transactions on Information Theory*, 46(1):60–83, 2000.

[Bab86] László Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

[Ban93] Wojciech Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(1):625–635, 1993.

[BB98] Amir H Banihashemi and Ian F Blake. Trellis complexity and minimal trellis diagrams of lattices. *IEEE Transactions on Information Theory*, 44(5):1829–1847, 1998.

[BBGM+17] Hayo Baan, Sauvik Bhattacharya, Oscar Garcia-Morchon, Ronald Rietman, Ludo Tolhuizen, Jose-Luis Torre-Arce, and Zhenfei Zhang. Round2: KEM and PKE based on GLWR. *Cryptology ePrint Archive*, 2017.

[BCLV17] Daniel J Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. NTRU prime: reducing attack surface at low cost. In *International Conference on Selected Areas in Cryptography*, pages 235–260. Springer, 2017.

[BCNS15] Joppe W Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In *2015 IEEE Symposium on Security and Privacy*, pages 553–570. IEEE, 2015.

[BDGL16] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 10–24. SIAM, 2016.

[BDK+18] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-Kyber: a CCA-secure module-lattice-based KEM. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 353–367. IEEE, 2018.

[BGJ+20] Alessandro Budroni, Qian Guo, Thomas Johansson, Erik Mårtensson, and Paul Stankovski Wagner. Making the BKW Algorithm Practical for LWE. In *International Conference on Cryptology in India*, pages 417–439. Springer, 2020.

[BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.

[BJRLW20] Katharina Boudgoust, Corentin Jeudy, Adeline Roux-Langlois, and Weiqiang Wen. Towards classical hardness of module-lwe: The linear rank case. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 289–317. Springer, 2020.

[BK18] Johannes Blomer and Kathlén Kohn. Voronoi cells of lattices with respect to arbitrary norms. *SIAM Journal on Applied Algebra and Geometry*, 2(2):314–338, 2018.

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 575–584, 2013.

[BLRL+18] Shi Bai, Tancrède Lepoint, Adeline Roux-Langlois, Amin Sakzad, Damien Stehlé, and Ron Steinfeld. Improved security proofs in lattice-based cryptography: using the Rényi divergence rather than the statistical distance. *Journal of Cryptology*, 31(2):610–640, 2018.

[BLS16] Shi Bai, Thijs Laarhoven, and Damien Stehlé. Tuple lattice sieving. *LMS Journal of Computation and Mathematics*, 19:146–162, 01 2016.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 719–737. Springer, 2012.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *Annual Cryptology Conference*, pages 868–886. Springer, 2012.

[BRL20] Olivier Bernard and Adeline Roux-Langlois. Twisted-PHS: Using the Product Formula to Solve Approx-SVP in Ideal Lattices. In *ASIACRYPT 2020*, pages 349–380. Springer, 2020.

[BS99] Johannes Blömer and Jean-Pierre Seifert. On the complexity of computing short linearly independent vectors and short bases in a lattice. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 711–720, 1999.

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *Annual cryptology conference*, pages 505–524. Springer, 2011.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *Annual cryptology conference*, pages 505–524. Springer, 2011.

[BV14a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. *SIAM Journal on Computing*, 43(2):831–871, 2014.

[BV14b] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. *SIAM Journal on Computing*, 43(2):831–871, 2014.

[BW59] Eric Stephen Barnes and Gordon Elliott Wall. Some extreme forms defined in terms of abelian groups. *Australian Mathematical Society*, 1(1):47–63, 1959.

[Che13] Yuanmi Chen. Lattice reduction and concrete security of fully homomorphic encryption. *Dept. Informatique, ENS, Paris, France, PhD thesis*, 2013.

[CHKP12] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. *Journal of cryptology*, 25(4):601–639, 2012.

[CL21] André Chailloux and Johanna Loyer. Lattice sieving via quantum random walks. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 63–91. Springer, 2021.

[CN11] Yuanmi Chen and Phong Q Nguyen. BKZ 2.0: Better lattice security estimates. In *ASIACRYPT 2011*, pages 1–20. Springer, 2011.

[CS82a] John Conway and Neil Sloane. Fast quantizing and decoding algorithms for lattice quantizers. *IEEE Trans Inform Theory*, 28(2):227–232, 1982.

[CS82b] John Conway and Neil Sloane. Laminated lattices. *Annals of Mathematics*, pages 593–620, 1982.

[CS82c] John Conway and Neil Sloane. Voronoi regions of lattices, second moments of polytopes, and quantization. *IEEE transactions on information theory*, 28(2):211–226, 1982.

[CS84] John Conway and Neil Sloane. On the voronoi regions of certain lattices. *SIAM Journal on Algebraic Discrete Methods*, 5(3):294–305, 1984.

[CS13] John Conway and Neil *Sphere packings, lattices and groups*, volume 290. Springer Science & Business Media, 2013.

[CT65] James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

[DCRVV15] Ruan De Clercq, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. Efficient software implementation of ring-LWE encryption. In *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 339–344. IEEE, 2015.

[DGJ+19] Jan-Pieter D'Anvers, Qian Guo, Thomas Johansson, Alexander Nilsson, Frederik Vercauteren, and Ingrid Verbauwhede. Decryption failure attacks on IND-CCA secure lattice-based schemes. In *IACR International Workshop on Public Key Cryptography*, pages 565–598. Springer, 2019.

[DGK+10] Yevgeniy Dodis, Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Public-key encryption schemes with auxiliary inputs. In *Theory of Cryptography Conference*, pages 361–381. Springer, 2010.

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[DJD88] Sudhakar Dharmadhikari and Kumar Joag-Dev. *Unimodality, convexity and applications*. Elsevier, 1988.

[DKL+18] Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 238–268, 2018.

[DKRV18] 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 *International Conference on Cryptology in Africa*, pages 282–305. Springer, 2018.

[DKS98] Irit Dinur, Guy Kindler, and Shmuel Safra. Approximating-CVP to within almost-polynomial factors is NP-hard. In *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No. 98CB36280)*, pages 99–109. IEEE, 1998.

[DRSD14] Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. On the closest vector problem with a distance guarantee. In *2014 IEEE 29th Conference on Computational Complexity (CCC)*, pages 98–109. IEEE, 2014.

[DSSV09] Mathieu Dutour Sikirić, Achill Schürmann, and Frank Vallentin. Complexity and algorithms for computing Voronoi cells of lattices. *Mathematics of computation*, 78(267):1713–1731, 2009.

[DVV19] Jan-Pieter D'Anvers, Frederik Vercauteren, and Ingrid Verbauwhede. The impact of error dependencies on Ring/Mod-LWE/LWR based schemes. In *International Conference on Post-Quantum Cryptography*. Springer, 2019.

[DWZ18] Wei Dai, William Whyte, and Zhenfei Zhang. Optimizing polynomial convolution for NTRUEncrypt. *IEEE Transactions on Computers*, 67(11):1572–1583, 2018.

[DXL12] Jintai Ding, Xiang Xie, and Xiaodong Lin. A simple provably secure key exchange scheme based on the learning with errors problem. *IACR Cryptol. ePrint Arch.*, 2012:688, 2012.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Annual International Cryptology Conference*, pages 537–554. Springer, 1999.

[FO13] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of cryptology*, 26(1):80–101, 2013.

[For88] G David Forney. Coset codes. II. Binary lattices and related codes. *IEEE Transactions on Information Theory*, 34(5):1152–1187, 1988.

[FS10] Claus Fieker and Damien Stehlé. Short bases of lattices over number fields. In *International Algorithmic Number Theory Symposium*, pages 157–173. Springer, 2010.

[FV96] G David Forney and Alexander Vardy. Generalized minimum-distance decoding of Euclidean-space codes and lattices. *IEEE Transactions on Information Theory*, 42(6):1992–2026, 1996.

[Gat18] Fletcher Gates. *Reduction-Respecting Parameters for Lattice-Based Cryptosystems.* PhD thesis, 2018.

[GG00] Oded Goldreich and Shafi Goldwasser. On the limits of nonapproximability of lattice problems. *Journal of Computer and System Sciences*, 60(3):540–563, 2000.

[GGH96] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Collision-free hashing from lattice problems. Electronic Colloquium on Computational Complex- ity (ECCC), 1996.

[GHS12] Craig Gentry, Shai Halevi, and Nigel P Smart. Fully homomorphic encryption with polylog overhead. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 465–482. Springer, 2012.

[GJN20] Qian Guo, Thomas Johansson, and Alexander Nilsson. A key-recovery timing attack on post-quantum primitives using the fujisaki-okamoto transformation and its application on frodokem. In *Annual International Cryptology Conference*, pages 359–386. Springer, 2020.

[GKPV10] Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. Tsinghua University Press, 2010.

[GMSS99] Oded Goldreich, Daniele Micciancio, Shmuel Safra, and J-P Seifert. Approximating shortest lattice vectors is not harder than approximating closest lattice vectors. *Information Processing Letters*, 71(2):55–61, 1999.

[GN08] Nicolas Gama and Phong Q Nguyen. Finding short lattice vectors within Mordell's inequality. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 207–216, 2008.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 197–206, 2008.

[Gro96] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[Has88] Johan Hastad. Dual vectors and lower bounds for the nearest lattice point problem. *Combinatorica*, 8(1):75–81, 1988.

[Hel85] Bettina Helfrich. Algorithms to construct Minkowski reduced and Hermite reduced lattice bases. *Theoretical Computer Science*, 41:125–139, 1985.

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In *Theory of Cryptography Conference*. Springer, 2017.

[HK17] Gottfried Herold and Elena Kirshanova. Improved algorithms for the approximate k-list problem in euclidean norm. In *IACR International Workshop on Public Key Cryptography*, pages 16–40. Springer, 2017.

[HKM18] Gottfried Herold, Elena Kirshanova, and Alexander May. On the asymptotic complexity of solving LWE. *Designs, Codes and Cryptography*, 86(1):55–83, 2018.

[HKZ12] Daniel Hsu, Sham Kakade, and Tong Zhang. A tail inequality for quadratic forms of subgaussian random vectors. *Electronic Communications in Probability*, 17, 2012.

[HPS11] Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. Analyzing blockwise lattice algorithms using dynamical systems. In *Annual Cryptology Conference*, pages 447–464. Springer, 2011.

[HR07] Ishay Haviv and Oded Regev. Tensor-based hardness of the shortest vector problem to within almost polynomial factors. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 469–477, 2007.

[HS07] Guillaume Hanrot and Damien Stehlé. Improved analysis of Kannan's shortest lattice vector algorithm. In *Annual international cryptology conference*, pages 170–186. Springer, 2007.

[Joz01] Richard Jozsa. Quantum factoring, discrete logarithms, and the hidden subgroup problem. *Computing in science & engineering*, 3(2):34–43, 2001.

[JZ20] Zhengzhong Jin and Yunlei Zhao. AKCN-E8: Compact and Flexible KEM from Ideal Lattice. *IACR Cryptol. ePrint Arch.*, 2020:56, 2020.

[Kan83] Ravi Kannan. Improved algorithms for integer programming and related lattice problems. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 193–206, 1983.

[Kan87] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of operations research*, 12(3):415–440, 1987.

[KF15] Paul Kirchner and Pierre-Alain Fouque. An improved BKW algorithm for LWE with applications to cryptography and lattices. In *Annual Cryptology Conference*, pages 43–62. Springer, 2015.

[Kho05] Subhash Khot. Hardness of approximating the shortest vector problem in lattices. *Journal of the ACM (JACM)*, 52(5):789–808, 2005.

[Kle00] Philip Klein. Finding the closest lattice vector when it's unusually close. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 937–941, 2000.

[Kra08] Gerhard Kramer. *Topics in multi-user information theory.* now publishers inc, 2008.

[KTX07] Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. Multi-bit cryptosystems based on lattice problems. In *International Workshop on Public Key Cryptography*, pages 315–329. Springer, 2007.

[L+19] Eunsang Lee et al. Modification of FrodoKEM using Gray and error-correcting codes. *IEEE Access*, 7, 2019.

[Laa15a] Thijs Laarhoven. *Search problems in cryptography: From fingerprinting to lattice sieving.* PhD thesis, Eindhoven University of Technology., 2015.

[Laa15b] Thijs Laarhoven. Sieving for shortest vectors in lattices using angular locality-sensitive hashing. In *Annual Cryptology Conference*, pages 3–22. Springer, 2015.

[LdW15] Thijs Laarhoven and Benne de Weger. Faster sieving for shortest lattice vectors using spherical locality-sensitive hashing. In *International Conference on Cryptology and Information Security in Latin America*, pages 101–118. Springer, 2015.

[LLB13] Cong Ling, Laura Luzzi, and Matthieu R Bloch. Secret key generation from gaussian sources using lattice hashing. In *2013 IEEE International Symposium on Information Theory*, pages 2621–2625. IEEE, 2013.

[LLL82] Arjen K Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[LLM06] Yi-Kai Liu, Vadim Lyubashevsky, and Daniele Micciancio. On bounded distance decoding for general lattices. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 450–461. Springer, 2006.

[LLS90] Jeffrey C Lagarias, Hendrik W Lenstra, and Claus-Peter Schnorr. Korkin-Zolotarev bases and successive minima of a lattice and its reciprocal lattice. *Combinatorica*, 10(4):333–348, 1990.

[LLZ+18] Xianhui Lu, Yamin Liu, Zhenfei Zhang, Dingding Jia, Haiyang Xue, Jingnan He, Bao Li, Kunpeng Wang, Zhe Liu, and Hao Yang. LAC: Practical ring-LWE based public-key encryption with byte-level modulus. *IACR Cryptol. ePrint Arch.*, page 1009, 2018.

[LMVDP15] Thijs Laarhoven, Michele Mosca, and Joop Van De Pol. Finding shortest lattice vectors faster using quantum search. *Designs, Codes and Cryptography*, 77(2):375–400, 2015.

[LN16] Patrick Longa and Michael Naehrig. Speeding up the number theoretic transform for faster ideal lattice-based cryptography. In *International Conference on Cryptology and Network Security*, pages 124–139. Springer, 2016.

[LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In *Cryptographers' Track at the RSA Conference*, pages 319–339. Springer, 2011.

[LPMSW19] Changmin Lee, Alice Pellet-Mary, Damien Stehlé, and Alexandre Wallet. An LLL algorithm for module lattices. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 59–90. Springer, 2019.

[LPR] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-LWE cryptography. In *EUROCRYPT 2013*, pages 35–54. Springer.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT 2010*, pages 1–23. Springer, 2010.

[LS71] John Leech and NJA Sloane. Sphere packings and error-correcting codes. *Canadian Journal of Mathematics*, 23(4):718–745, 1971.

[LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015.

[LSS14] Adeline Langlois, Damien Stehlé, and Ron Steinfeld. Gghlite: More efficient multilinear maps from ideal lattices. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 239–256. Springer, 2014.

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 738–755. Springer, 2012.

[Mah15] Mohammad Mahmoody. Topics in cryptography - *CS 4501-650*. Lecture 7, 2015. `https://piazza.com/class_profile/get_resource/i5d2rsalxrq1s4/i9cudz7fv0m552`.

[MG12] Daniele Micciancio and Shafi Goldwasser. *Complexity of lattice problems: a cryptographic perspective*, volume 671. Springer Science & Business Media, 2012.

[Mic01] Daniele Micciancio. The shortest vector in a lattice is hard to approximate to within some constant. *SIAM journal on Computing*, 30(6):2008–2035, 2001.

[Min05] Hermann Minkowski. Diskontinuitätsbereich für arithmetische äquivalenz. 1905.

[MN08] Daniele Micciancio and Antonio Nicolosi. Efficient bounded distance decoders for Barnes-Wall lattices. In *2008 IEEE International Symposium on Information Theory*, pages 2484–2488. IEEE, 2008.

[Moo21] Dustin Moody. The End is Near: The NIST PQC competition. *European Cyber Week*, 2021.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 700–718. Springer, 2012.

[MV10] Daniele Micciancio and Panagiotis Voulgaris. Faster exponential time algorithms for the shortest vector problem. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1468–1480. SIAM, 2010.

[MV13] Daniele Micciancio and Panagiotis Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. *SIAM Journal on Computing*, 42(3):1364–1391, 2013.

[MW14] Daniele Micciancio and Michael Walter. Fast lattice point enumeration with minimal overhead. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 276–294. SIAM, 2014.

[N+20] Michael Naehrig et al. FrodoKEM. tech. rep. In *National Institute of Standards and Technology*, 2020. `https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions`.

[NV08] Phong Q Nguyen and Thomas Vidick. Sieve algorithms for the shortest vector problem are practical. *Journal of Mathematical Cryptology*, 2(2):181–207, 2008.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 333–342, 2009.

[Pei10] Chris Peikert. An efficient and parallel Gaussian sampler for lattices. In *Annual Cryptology Conference*, pages 80–97. Springer, 2010.

[Pei14] Chris Peikert. Lattice cryptography for the internet. In *International Workshop on Post-Quantum Cryptography*, pages 197–219. Springer, 2014.

[Pei15] Chris Peikert. A decade of lattice cryptography. *Cryptology ePrint Archive*, 2015.

[PG13] Thomas Pöppelmann and Tim Güneysu. Towards practical lattice-based public-key encryption on reconfigurable hardware. In *International Conference on Selected Areas in Cryptography*, pages 68–85. Springer, 2013.

[PMHS19] Alice Pellet-Mary, Guillaume Hanrot, and Damien Stehlé. Approx-SVP in ideal lattices with pre-processing. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 685–716. Springer, 2019.

[PP19] Chris Peikert and Zachary Pepin. Algebraically structured LWE, revisited. In *Theory of Cryptography Conference*, pages 1–23. Springer, 2019.

[Pre21] Thomas Prest. Lattice-based NIST Candidates - Abstractions and Ninja Tricks. *European Cyber Week*, 2021.

[PRSD17] Chris Peikert, Oded Regev, and Noah Stephens-Davidowitz. Pseudorandomness of ring-LWE for any ring and modulus. In *Proc. of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 461–473, 2017.

[PS09] Xavier Pujol and Damien Stehlé. Solving the shortest lattice vector problem in time 2ˆ 2.465 n. *Cryptology ePrint Archive*, 2009.

[Puc05] Riccardo Pucella. Foundations of cryptography II: Basic applications. 2005.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *Annual international cryptology conference*, pages 554–571. Springer, 2008.

[PW11] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. *SIAM Journal on Computing*, 40(6):1803–1844, 2011.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):34, 2009.

[RVM+14] Sujoy Sinha Roy, Frederik Vercauteren, Nele Mentens, Donald Donglong Chen, and Ingrid Verbauwhede. Compact ring-LWE cryptoprocessor. In *International workshop on cryptographic hardware and embedded systems*, pages 371–391. Springer, 2014.

[SB95] Gottfried Schnabl and Martin Bossert. Soft-decision decoding of reed-muller codes as generalized multiple concatenated codes. *IEEE transactions on information theory*, 41(1):304–308, 1995.

[SCC19] Xinxia Song, Zhigang Chen, and Liang Chen. A multi-bit fully homomorphic encryption with shorter public key from LWE. *IEEE Access*, 7:50588–50594, 2019.

[Sch87] Claus-Peter Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical computer science*, 53(2-3):201–224, 1987.

[SE94a] Claus-Peter Schnorr and Martin Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical programming*, 66(1):181–199, 1994.

[SE94b] Claus-Peter Schnorr and Martin Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical programming*, 66(1):181–199, 1994.

[Sei18] Gregor Seiler. Faster AVX2 optimized NTT multiplication for Ring-LWE lattice cryptography. *Cryptology ePrint Archive*, 2018.

[SLL21] Charbel Saliba, Laura Luzzi, and Cong Ling. A reconciliation approach to key generation based on Module-LWE. In *IEEE International Symposium on Information Theory (ISIT)*, 2021.

[Slo81] N Sloane. Tables of sphere packings and spherical codes. *IEEE Transactions on Information Theory*, 27(3):327–338, 1981.

[SLS+20] Minki Song, Seunghwan Lee, Dong-Joon Shin, Eunsang Lee, Young-Sik Kim, and Jong-Seon No. Analysis of error dependencies on NewHope. *IEEE Access*, 8:45443–45456, 2020.

[Ste14] Damien Stehlé. Advanced cryptographic primitives - *The Learning With Errors Problem*. Course 3, 2014. `https://perso.ens-lyon.fr/damien.stehle/downloads/Course3.pdf`.

[TU16] Ehsan Ebrahimi Targhi and Dominique Unruh. Post-quantum security of the fujisaki-okamoto and oaep transforms. In *Theory of Cryptography Conference*, pages 192–216. Springer, 2016.

[VB96] Emanuele Viterbo and Ezio Biglieri. Computing the Voronoi cell of a lattice: the diamond-cutting algorithm. *IEEE Transactions on Information Theory*, 42(1):161–171, 1996.

[vEB81] Peter van Emde Boas. Another NP-complete problem and the complexity of computing short vectors in a lattice. *Tecnical Report, Department of Mathmatics, University of Amsterdam*, 1981.

[Via17] Maryna S Viazovska. The sphere packing problem in dimension 8. *Annals of Mathematics*, pages 991–1015, 2017.

[vP16] AH van Poppelen. Cryptographic decoding of the Leech lattice. Master's thesis, Utrecht University, 2016.

[Wan20] Jiabo Wang. Coding techniques in lattice-based cryptography. 2020.

[WZ76] A. Wyner and J. Ziv. The rate-distortion function for source coding with side information at the decoder. *IEEE Transactions on Information Theory*, 22(1):1–10, 1976.

[Zam14] Ram Zamir. *Lattice Coding for Signals and Networks: A Structured Coding Approach to Quantization, Modulation, and Multiuser Information Theory*. Cambridge University Press, 2014.

[ZJGS17] Y Zhao, Z Jin, B Gong, and G Sui. A modular and systematic approach to key establishment and public-key encryption based on LWE and its variants. *NIST PQC Round*, 1:4, 2017.

[ZSE02] Ram Zamir, Shlomo Shamai, and Uri Erez. Nested linear/lattice codes for structured multiterminal binning. *IEEE Transactions on Information Theory*, 48(6):1250–1276, 2002.

Index