Lattice Cryptography: Understanding Kyber (ML-KEM) and Dilithium (ML-DSA)

Vadim Lyubashevsky

IBM Research Europe, Zurich

(Last updated: June 17, 2024)

Contents

1 Prelude

The beginning of modern lattice-based cryptography traces back to two works from the mid 1990's. The first was Ajtai's result [Ajt96] that showed that being able to solve a random instance of a certain problem was as hard as solving some believed-to-be-hard problem for *every* lattice. The second was the NTRU cryptosystem [HPS98], which constructed an efficient cryptosystem based on a new, potentially-hard lattice problem. These two results garnered some interest from the cryptographic community leading to a series of works throughout the 2000's [Mic07, MR07, Reg09, PR06, LM06, GPV08, Lyu09, LPR10] that aimed to combine the two branches of research by creating efficient (at least in an asymptotic sense) encryption, signature and identity-based encryption schemes which were based on solid theoretical foundations.

Lattice-based cryptography gained further momentum in the early 2010's due to the first realization, from any assumption, of a fully-homomorphic encryption scheme [Gen09] and the increase in research aimed towards building a quantum computer that would be able to break all cryptography based on factoring and discrete log [Sho97]. The latter part of the 2010's saw lattice-based schemes go from being just asymptotically-efficient to being truly practical. By the time the NIST post-quantum cryptography standardization process started in late 2017, lattice-based schemes were among the fastest and most compact quantum-resistant primitives, and even surpassed their number-theoretic counterparts in terms of raw performance.

This tutorial focuses on describing all the concepts used in the schemes from the last category and the hope is that by the end, the reader will have all the tools needed to understand the concrete instantiations and design decisions that were made in many of the two "main" lattice-based encryption – CRYSTALS-Kyber (ML-KEM) [BDK+18, NIS23b] – and signature – CRYSTALS-Dilithium (ML-DSA) [DKL+18, NIS23a] – schemes selected by NIST for standardization and included in the CNSA 2.0 algorithmic suite [Age24]. On the way, we will also give the main ideas behind other lattice-based KEMs like Frodo [BCD+16] and NTRU [HPS98].

The target audience for this manuscript are people who are already familiar with basic public-key cryptography and ideas from proving security of cryptographic schemes using reduction arguments. It's therefore assumed that concepts such as indistinguishability, CPA-secure encryption, random oracles, the Fiat-Shamir transform, etc. are not too foreign. Those familiar with these concepts most likely first saw them when studying public key cryptographic primitives based on the hardness of the discrete log or RSA problems. Keeping the discrete logarithm constructions, such as El Gamal encryption and Schnorr signatures, in mind will actually be very useful for understanding the high-level ideas of the lattice constructions.

The main difference that many find between classical and lattice cryptography is that lattice cryptography is "messy". Instead of there being one parameter from which security is derived, there are several which affect the security in various ways. There are also many optimization tricks which involve operations such as dropping some low-order bits, rounding, etc. On the other hand, many lattice constructions require very few "deep" mathematical concepts – one could even argue that Euler's theorem, upon which the RSA cryptosystem relies, is a deeper mathematical fact than anything needed for understanding how Kyber and Dilithium work.[1] It takes a little practice and patience to get used to the issues that these constructions present and so the reader is encouraged to go through this manuscript with some pencil and paper to work out the details and get comfortable with

[1] This is not to say that there is no math in lattice cryptography. Cryptanalysis requires fairly deep concepts from the geometry of numbers and algebraic number theory, and even understanding the security of constructions involving trapdoor sampling, such as the FALCON [PFH+17] digital signature scheme, requires Fourier Analysis – and even some fairly non-trivial number theory to understand how to efficiently perform key generation. But none of these are needed in this manuscript.

the mess.

There is a lot about lattices that this manuscript does not cover. For example, it does not say anything about more "advanced" constructions beyond encryption or signatures – a good overview of some of those may be found in [Pei16]. It also doesn't say much about cryptanalysis or the geometric aspects of lattice cryptography which are crucial to understanding some more advanced constructions. A good reference for the geometrical foundations of lattices is [MG02] and the lecture notes [Mic19]. Some recent papers related to algorithms and cryptanalysis relating to the problems upon which the lattice-based NIST standardization candidates are based are [ACD+18, AM18, ADH+19]. Notably missing is also the other lattice-based signature scheme chosen to be standardized by NIST – FALCON [PFH+17], understanding which would require a deeper dive into the geometry of lattices, which can be began by studying the seminal work of [GPV08] which explains how to create a lattice-based trapdoor sampling algorithm.

While the main reason for the interest in lattice-based encryption and signatures is their presumed resistance to quantum attacks, we do not broach this topic. The random oracle model (ROM) where, in the security proof, a concrete cryptographic hash function is replaced with a perfect random function to which the adversary only has oracle access, has an analogy in the quantum setting (QROM) where the adversary is allowed to also make quantum queries (i.e. querying the oracle on a superposition of inputs) to such a function. While security proofs in the ROM do not immediately carry over to the QROM setting, there have been many recent works bridging the two closer together (e.g. [HHK17, SXY18, KLS18, DFMS19, LZ19]) with the only remaining difference being the tightness of, or the concrete problem in, the security reduction. At the time of this writing, if one just replaces all the cryptographic functions with their counterparts that have the requisite quantum security, then there aren't any known improved attacks on real-world schemes proven secure in the ROM if the adversary is given the extra quantum power of querying the random oracle on a superposition of inputs.

Outline. In section 2, we present the framework for lattice-based encryption over the ring $\mathbb{Z}_q$ based on the hardness of the LWE problem that stems from the original work of Regev [Reg09] up to its most efficient instantiation [BCD+16]. We also discuss many small, but crucial, tricks for compressing the outputs. Some of these tricks are fairly general and will also come in handy in the constructions presented in the later sections. In Section 3, we introduce lattices and describe the connection between LWE and SIS and the difficulty of solving lattice problems. In Section 4, we review polynomial rings and instantiate the analogous version of the encryption scheme from Section 2, which results in the Kyber (ML-KEM) encryption scheme. Finally in Section 5, we present all the needed techniques for optimized lattice-based analogues of the Schnorr signature schemes, to which Dilithium (ML-DSA) belongs [DKL+18]. The signature scheme is build by applying the Fiat-Shamir transform to a lattice-based Σ-protocol. The latter also serves as a gateway to understanding more advanced lattice-based constructions such as zero-knowledge proofs.

2 Encryption

We begin our study of lattice-based cryptography by constructing a CPA-secure (i.e. Chosen Plaintext Attack Secure) public key encryption scheme. Recall that CPA-secure encryption scheme consists of three algorithms: key generation, encryption, and decryption. The key generation algorithm outputs a public key/secret key pair. The encryption algorithm takes as input the public key and a message, and produces a ciphertext. The decryption algorithm, in turn, takes a ciphertext and the secret key and outputs the message. The scheme is said to be CPA-secure if for any two messages of the adversary's choosing, the encryption of the two messages are computationally indistinguishable.

We only discuss CPA-secure encryption in this section, as there are generic transformations (e.g. Fujisaki-Okamoto (c.f. [FO99]) that convert CPA-secure encryption schemes to CCA-secure ones (in which the adversary also has access to a decryption oracle) as well as to key exchange protocols secure against active attackers (Section 4.7).

We point out that the schemes in this chapter, unlike their discrete log and factoring analogues, are most efficiently instantiated in a way that gives rise to decryption errors. That is, even if the key generation and encryption algorithms are run correctly, the produced ciphertext may, with a very small probability, not decrypt to the message that was encrypted. In general, having a small-enough decryption error (say, around 2^{-150}) doesn't appear to hurt the practical security of schemes; one just has to account for these in the proofs of the transformations (e.g. [HHK17, SXY18]).

2.1 Some Notation

All operations in this section will be performed in the ring $(\mathbb{Z}_q, +, \times)$ with the usual addition and multiplication of integers modulo q. For a set S, we write $a \leftarrow S$ to mean that a is chosen uniformly at random from the set S. For any positive integer β, we will define the set

$$[\beta] = \{-\beta, \ldots -1, 0, 1, \ldots, \beta\}. \tag{1}$$

This notation naturally extends to vectors (and matrices) by writing $[\beta]^{n\times m}$. By default, all our vectors will be column vectors. One can also indicate that a vector $\mathbf{v}$ is in $[\beta]^m$ by writing $\|\mathbf{v}\|_\infty \leq \beta$. For an integer x, we write $\lceil x \rfloor$ to denote the closest integer to x, with ties being broken upwards.

2.2 A Motivating Example

Let's pretend for a second that for positive integers $q, n \geq m$, and $\beta \ll q$, the following two distributions are computationally indistinguishable (in the security parameter related to m):

1. $(\mathbf{A}, \mathbf{As})$, where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$ and $\mathbf{s} \leftarrow [\beta]^m$
2. $(\mathbf{A}, \mathbf{u})$, where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$ and $\mathbf{u} \leftarrow \mathbb{Z}_q^n$.

That is, we're assuming that there exists no efficient algorithm which can tell whether the sample he is given comes from the first or the second distribution. This indistinguishability assumption is clearly false because one can use Gaussian elimination to invert $\mathbf{A}$ (or an $m \times m$ sub-matrix of $\mathbf{A}$) and check whether there is indeed an $\mathbf{s} \in [\beta]^m$ satisfying $\mathbf{As} = \mathbf{u}$. Please bear with this example, because a slightly modified version of this assumption forms the foundation of most lattice cryptography. We will now use this assumption to construct a simple CPA-secure public key encryption scheme that very much resembles the discrete logarithm based El Gamal encryption scheme. The secret and public keys of the scheme are

$$\mathsf{sk}: \mathbf{s} \leftarrow [\beta]^m, \ \mathsf{pk}: (\mathbf{A} \leftarrow \mathbb{Z}_q^{m\times m}, \mathbf{t} = \mathbf{As}). \tag{2}$$

To encrypt a message $\mu \in \mathbb{Z}_q$, the encryptor picks a random vector $\mathbf{r} \leftarrow [\beta]^m$ and outputs the ciphertext $(\mathbf{u}, v) \in \mathbb{Z}_q^m \times \mathbb{Z}_q$ where

$$(\mathbf{u}^T = \mathbf{r}^T\mathbf{A}, v = \mathbf{r}^T\mathbf{t} + \mu). \tag{3}$$

To decrypt, one simply computes

$$\mu = v - \mathbf{u}^T\mathbf{s}. \tag{4}$$

Correctness of the scheme follows because

$$v - \mathbf{u}^T\mathbf{s} = \mathbf{r}^T\mathbf{t} + \mu - \mathbf{r}^T\mathbf{A}\mathbf{s} = \mathbf{r}^T\mathbf{A}\mathbf{s} + \mu - \mathbf{r}^T\mathbf{A}\mathbf{s} = \mu. \tag{5}$$

The security of the scheme follows by our assumption and the hybrid argument. By the assumption, the public key $(\mathbf{A}, \mathbf{t})$ in (2) is indistinguishable from uniform. The public key matrix $\mathbf{A}' = [\mathbf{A} \mid \mathbf{t}] \in \mathbb{Z}_q^{m\times(m+1)}$ is therefore indistinguishable from uniform, and using our assumption once more we obtain that the distribution $(\mathbf{A}', \mathbf{r}^T\mathbf{A}')$ is also indistinguishable from uniform. It therefore follows that the distribution of $(\mathbf{A}, \mathbf{t}, \mathbf{u}, v)$ is indistinguishable from uniform, and therefore the scheme is CPA-secure.

Notice that we used our indistinguishability assumption twice – once for arguing that the public key looks random, and the other time to argue that the ciphertext does. Instead of choosing $\mathbf{A} \leftarrow \mathbb{Z}_q^{m\times m}$, we could have chosen it randomly from $\mathbb{Z}_q^{n\times m}$ for $m \neq n$. If we set $m \gg n$, then one can show[2] that the public key $(\mathbf{A}, \mathbf{t})$ is actually statistically-close to being uniform. But then forming the ciphertext would still require us to use the absurd indistinguishability assumption. So using the assumption at least once is inescapable.

2.3 The LWE Problem

We will now make a "small" adjustment to the assumption from the previous section which will make it plausibly valid and still allow us to construct a cryptosystem following the same outline. We now define a simple version of the Learning with Errors Problem (LWE) [Reg09] upon which a lot of lattice cryptography rests.

Definition 1. For positive integers m, n, q, and $\beta < q$, the $\mathsf{LWE}_{n,m,q,\beta}$ problem asks to distinguish between the following two distributions:

1. $(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e})$, where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}, \mathbf{s} \leftarrow [\beta]^m, \mathbf{e} \leftarrow [\beta]^n$
2. $(\mathbf{A}, \mathbf{u})$, where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$ and $\mathbf{u} \leftarrow \mathbb{Z}_q^n$.

The crucial part that makes $\mathsf{LWE}_{n,m,q,\beta}$ hard is the presence of the additional "error" vector $\mathbf{e}$ which makes the simple Gaussian elimination attack inapplicable. The exact hardness of the problem depends on the parameters n, m, q, and β. The problem becomes harder as m and β/q grow. The parameter n is not known to have a large impact on the hardness of the problem except in extreme cases where n is as large as approximately $m^{2\beta+1}$, in which case one can build a distinguisher in time approximately $m^{2\beta}$ by linearization techniques [AG11]. In the constructions that we will present throughout this chapter, n will never need to be so large. Since the parameter n will not be particularly important, we will sometimes omit it when stating the hardness assumption and just write $\mathsf{LWE}_{m,q,\beta}$.

We also mention that there is nothing too special about using the uniform distribution for our secret and error terms – it just makes the presentation simpler and so we choose to use this distribution for illustrative purposes. In the original definition of LWE, the error distribution was chosen as a rounded Gaussian distribution – that is, one generates a continuous random 0-centered Gaussian with some standard deviation and rounds it to the nearest integer. This distribution was necessary for the average-case to worst-case

[2] Using a simple application of the Leftover Hash Lemma [IZ89, IN96]

reduction proofs [Reg09, Pei09] showing that LWE is at least as hard as some worst-case lattice problems. This restriction has since been shown to not be strictly necessary, and one can use, for example, the uniform distribution [DM13, MP13]. Some practical implementations, notably Kyber, use the binomial distribution to generate the errors (c.f. [ADPS16, BDK+18]) because in practice it is sometimes faster to generate a string of bits and add them up instead of generating a uniform element in $[\beta]$.

To account for different distributions that one could use, we can define the LWE problem relative to the the distribution of the secrets ψ as:

Definition 2. For positive integers m, n, q, and a distribution ψ, the $\mathsf{LWE}_{n,m,q,\psi}$ problem asks to distinguish between the following two distributions:

1. $(\mathbf{A}, \mathbf{As} + \mathbf{e})$, where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}, \mathbf{s} \leftarrow \psi^m, \mathbf{e} \leftarrow \psi^n$
2. $(\mathbf{A}, \mathbf{u})$, where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$ and $\mathbf{u} \leftarrow \mathbb{Z}_q^n$.

For concreteness, we will mostly use the LWE from Definition 1, but once one accounts for the specific properties of ψ (specifically the norm of the secrets generated by this distribution), everything we say equally applies to the problem in Definition 2 as well.

Something to note is that in the definition of LWE, we are not setting any conditions on the relative sizes of m, n and q. It's therefore possible that some choices will lead to trivially hard LWE instances. For example, if $n < m$ and β is large-enough, then the distribution $(\mathbf{A}, \mathbf{As} + \mathbf{e})$ could indeed be statistically-close to $(\mathbf{A}, \mathbf{u})$. And then one clearly should not be able to build an encryption scheme based on just this assumption, since we would have an unconditionally-secure public key encryption scheme. The reason why things will indeed not work out will become clear when we consider correctness of the decryption. It is of course also possible to set parameters so that LWE is not a hard problem. For example, if $m = 1$, then it is not difficult to figure out whether $\mathbf{As} + \mathbf{e}$ is close to a multiple of the *vector* $\mathbf{A}$. We will discuss the hardness of the LWE problem in Section 3. Another note is that it is also possible to define LWE where the secret $\mathbf{s}$ is chosen to be uniform in $\mathbb{Z}_q^m$ (being careful to then take n large enough so that the LWE problem does not become trivially hard); and this is indeed how LWE was originally defined in [Reg09]. A simple proof [ACPS09], however, shows that taking $\mathbf{s}$ to be from the same distribution as $\mathbf{e}$ results in an essentially equally-hard problem. For applications, it is usually more efficient to take $\mathbf{s}$ to be smaller, and so we will only consider this version of the LWE problem.

2.3.1 An LWE-Based Encryption Scheme

In the rest of this section, we will present cryptosystems which stem from the original work in [Reg09] and have been improved and generalized via a series of observations in various follow-up works (c.f. [ACPS09, Pei09, LPS10, LP11, BCD+16]). We can view the scheme in this section as a modification of the encryption scheme from Section 2.2, but based on the hardness of $\mathsf{LWE}_{m,q,\beta}$ instead of the clearly false assumption made there. Our first modification will be the message μ – rather than being an arbitrary element in $\mathbb{Z}_q$, it will now come from the set $\{0, 1\}$. The key generation from (2) is modified to:

$$\mathsf{sk} : \mathbf{s} \leftarrow [\beta]^m, \ \mathsf{pk} : (\mathbf{A} \leftarrow \mathbb{Z}_q^{m \times m}, \mathbf{t} = \mathbf{As} + \mathbf{e}_1), \text{ where } \mathbf{e}_1 \leftarrow [\beta]^m. \tag{6}$$

To encrypt a message $\mu \in \{0, 1\}$, the encryptor chooses $\mathbf{r}, \mathbf{e}_2 \leftarrow [\beta]^m$ and $e_3 \leftarrow [\beta]$, and outputs

$$\left(\mathbf{u}^T = \mathbf{r}^T\mathbf{A} + \mathbf{e}_2^T, v = \mathbf{r}^T\mathbf{t} + e_3 + \frac{q}{2}\mu\right). \tag{7}$$

Let's first discuss a little notation. We are working in $\mathbb{Z}_q$, but the above equation has a strange-looking term $q/2$. What we mean by this is an element in $\mathbb{Z}_q$ that's closest to

the rational number $q/2$ (e.g. if $q = 13$, then $q/2 = 6$). So the division operation is not in $\mathbb{Z}_q$ – i.e. we are not multiplying by the inverse of 2. (If we will ever want to do division in $\mathbb{Z}_q$, we will instead write it as multiplication by an inverse). Also, to be precise, we really should write $\lceil q/2 \rfloor$, and such rounded terms will indeed come in often, but we omit this for presentation-purposes.

Before discussing decryption, let's go through the security argument to see why the scheme is based on the hardness of $\mathsf{LWE}_{m,q,\beta}$. The argument is the same as in Section 2.2. The indistinguishability of the public key $(\mathbf{A}, \mathbf{t})$ from the uniform distribution over $\mathbb{Z}_q^{m\times(m+1)}$ stems directly from the $\mathsf{LWE}_{m,q,\beta}$ assumption. Rewriting the public key $(\mathbf{A}, \mathbf{t})$ as a matrix $\mathbf{A}' = [\mathbf{A} \mid \mathbf{t}]$, we see that the $\mathsf{LWE}_{m,q,\beta}$ assumption again implies that the distribution $\left(\mathbf{A}', \mathbf{r}^T\mathbf{A}' + \begin{bmatrix}\mathbf{e}_2\\ e_3\end{bmatrix}^T\right)$ is also indistinguishable from uniform. Thus $(\mathbf{A}, \mathbf{T}, \mathbf{u}, v)$ is indistinguishable from uniform for any $\mu \in \{0, 1\}$ based on $\mathsf{LWE}_{m,q,\beta}$. Note that we used the $\mathsf{LWE}_{m,q,\beta}$ assumption twice and the parameter m came into play as the number of columns of $\mathbf{A}$ in the argument showing that the public key looks random, and then as the number of rows in $\mathbf{A}$ in the argument that the ciphertext looks random. This is intuitively the reason why it makes sense to set the number of rows and columns in $\mathbf{A}$ to be equal when trying to minimize the combined size of the public key and ciphertext in public key encryption. We discuss this topic further in Section 2.4, and in Section 2.5.5 also discuss some applications, and a sightly-modified cryptosystem, in which one may not want to have the number of rows equal to the number of columns.

To decrypt, one computes $v - \mathbf{u}^T\mathbf{s}$. But rather than this cleanly giving us the message μ as in (4), we instead obtain

$$v - \mathbf{u}^T\mathbf{s} = \mathbf{r}^T(\mathbf{A}\mathbf{s} + \mathbf{e}_1) + e_3 + \frac{q}{2}\mu - \left(\mathbf{r}^T\mathbf{A} + \mathbf{e}_2^T\right)\mathbf{s} \tag{8}$$

$$= \mathbf{r}^T\mathbf{e}_1 + e_3 + \frac{q}{2}\mu - \mathbf{e}_2^T\mathbf{s} \tag{9}$$

As in (5), the $\mathbf{r}^T\mathbf{A}\mathbf{s}$ terms in the above equation cancels out; we are however left with some combinations of "error" terms. Luckily, all these error terms have coefficients bounded by $\pm\beta$, and so the vector products $\mathbf{r}^T\mathbf{e}_1$ and $\mathbf{e}_2^T\mathbf{s}$ in (9) each consist of m terms of magnitude at most β^2 each. Therefore one can rewrite (9) as $e + \frac{q}{2}\mu$ where $e \in [2m\beta^2 + \beta]$, and so if the parameters are set such that $2m\beta^2 + \beta < q/4$ the decryptor can determine μ from looking at $v - \mathbf{u}^T\mathbf{s}$ by checking whether the preceding value is closer to 0 or to $q/2$.

2.3.2 Bounding the Total Error, Exactly

The value $2m\beta^2 + \beta$ that we computed above is the upper bound on the error magnitude. Because the coefficients of $\mathbf{r}, \mathbf{s}, \mathbf{e}_1$, and $\mathbf{e}_2$ are chosen randomly in the range $[\beta]$, which is centered around 0, it is however quite unlikely that the error will actually be so large. Due to cancellations, it will in fact be closer to $\mathcal{O}(\sqrt{m}\beta^2)$. In general, we would be fine getting a bound on the error such that with very high probability (say $1 - 2^{-150}$) the error will be below $q/4$. This would imply that the probability of a decryption error is at most 2^{-150}. Such small errors can be tolerated in applications and when using this CPA-encryption scheme as a component of other constructions (e.g. CCA-secure encryption).

There are asymptotic ways to approximately compute such bounds, but when dealing with concrete parameters and having β not too large, it is possible (using simple scripts) to get the exact value for

$$\Pr_{\mathbf{s},\mathbf{r},\mathbf{e}_1,\mathbf{e}_2\leftarrow[\beta]^m, e_3\leftarrow[\beta]}[\mathbf{r}^T\mathbf{e}_1 + e_3 - \mathbf{e}_2^T\mathbf{s} \in [\alpha]] \tag{10}$$

using the fact that probability distributions of sums of random variables can be modeled as products of polynomials. Suppose that A and B are random variables (possibly with

different distributions) over the finite set $[\gamma]$. For all $i \in [\gamma]$, let A_i (resp. B_i) be the probability that A (resp. B) is equal to i. Now define the polynomials

$$A(X) = \sum_{i=-\gamma}^{\gamma} A_i X^i, \;\; B(X) = \sum_{i=-\gamma}^{\gamma} B_i X^i.$$

Let

$$C(X) = A(X) \cdot B(X) = \sum_{i=-2\gamma}^{2\gamma} C_i X^i$$

be the product of $A(X)$ and $B(X)$. One can now make a direct connection between the coefficients C_i and the probability that $A + B = i$. In particular,

$$\Pr[A + B = i] = C_i. \tag{11}$$

This implies that

$$\Pr[A + B \in [\alpha]] = \sum_{i=-\alpha}^{\alpha} C_\alpha. \tag{12}$$

This directly carries over to computing the probability in (10) by noticing that $\mathbf{r}^T\mathbf{e}_1 - \mathbf{e}_2^T\mathbf{s}$ is distributed as a sum of $2m$ *independent* random variables A, where

$$A_i = \Pr[A = i] = \Pr_{x,y \leftarrow [\beta]}[xy = i],$$

and e_3 is just a uniformly distributed random variable over $[\beta]$. So, for example, if $\beta = 2$, then

$$A_{-4} = A_4 = \frac{2}{25},\; A_{-2} = A_2 = \frac{4}{25},\; A_{-1} = A_1 = \frac{2}{25},\; A_0 = \frac{9}{25},$$

and all the other A_i are 0. If we then write

$$\begin{aligned} C(X) = &\left(\frac{2}{25}X^{-4} + \frac{4}{25}X^{-2} + \frac{2}{25}X^{-1} + \frac{9}{25} + \frac{2}{25}X + \frac{4}{25}X^2 + \frac{2}{25}X^4\right)^{2m} \\ &\cdot \left(\frac{1}{5}X^{-2} + \frac{1}{5}X^{-1} + \frac{1}{5} + \frac{1}{5}X + \frac{1}{5}X^2\right), \end{aligned}$$

then the probability in (10) is exactly $\sum_{i=-\alpha}^{\alpha} C_i$, where C_i are again the coefficients associated to X^i in $C(X)$. By setting $\alpha = q/4 - 1$, we will obtain the probability that the decryption algorithm in Section 2.3.1 will correctly decrypt.

2.4 Public Key and Ciphertext Size Trade-offs

We now know how to set the parameters m, q, β relative to one another so that the encryption scheme in Section 2.3.1 correctly decrypts with overwhelming probability. We haven't yet discussed how one should set the parameters in order for it to be secure, and we will defer this to a bit later. But we can still compute the sizes of the public key and the ciphertext in terms of the parameters q, m, β.

The public key (see (6)) consists of a random matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times m}$ and a vector $\mathbf{t} \in \mathbb{Z}_q^m$. Since $\mathbf{A}$ is completely random, there is no need to store it – if one creates $\mathbf{A}$ by choosing a 256-bit seed ρ and then defining $\mathbf{A}$ as the expansion of ρ using some cryptographic PRF (e.g. SHAKE based on the SHA-3 function), then one only needs to store the 256 bits ρ instead of the $m^2 \log q$ bits of $\mathbf{A}$. One will need to expand $\mathbf{A}$ from ρ when encrypting and decrypting, but it's often a worthwhile trade-off to do so in lieu of storing the $\mathbf{A}$ – and in

particular when the scheme is used as a KEM, one would much rather transmit ρ than the entire $\mathbf{A}$. The other part of the public key, $\mathbf{t}$, depends on the secret key and so cannot be compressed in a similar way, and thus the total public key size is $256 + m \log q$ bits. The ciphertext $(\mathbf{u}, v) \in \mathbb{Z}_q^m \times \mathbb{Z}_q$ can be represented using $(m+1)\log q$ bits. Concrete secure parameter settings will require taking $m \approx 700$ and $q \approx 2^{13}$, and so it is somewhat inefficient to have the encryption of just one plaintext bit be so large. We will now give a generalization of the LWE encryption scheme that allows for various trade-offs between the public key and ciphertext size.

The reason that our current LWE-based encryption scheme has such a large ciphertext expansion is that $\mathbf{u}$ consists of $m \log q$ bits. To reduce the ciphertext expansion, we will give a variant of the scheme that amortizes the ciphertext part $\mathbf{u}$ among the encryption of many messages. The trade-off is that the public key will be larger. Suppose that we would like to encrypt $N = k\ell$ bits, which we arrange into a matrix $\mathbf{M} \in \{0,1\}^{k \times \ell}$. The key generation procedure will now be

$$\mathsf{sk}: \mathbf{S} \leftarrow [\beta]^{m \times \ell}, \ \mathsf{pk}: \left(\mathbf{A} \leftarrow \mathbb{Z}_q^{m \times m}, \mathbf{T} = \mathbf{AS} + \mathbf{E}_1\right), \text{ where } \mathbf{E}_1 \leftarrow [\beta]^{m \times \ell}. \tag{13}$$

Observe that, compared to (6), the public key size is now increased to $256 + \ell m \log q$ bits. The encryption algorithm proceeds analogously where the encryptor chooses $\mathbf{R}, \mathbf{E}_2 \leftarrow [\beta]^{k \times m}$ and $\mathbf{E}_3 \leftarrow [\beta]^{k \times \ell}$, and outputs the ciphertext

$$\left(\mathbf{U} = \mathbf{RA} + \mathbf{E}_2, \mathbf{V} = \mathbf{RT} + \mathbf{E}_3 + \frac{q}{2}\mathbf{M}\right). \tag{14}$$

which is comprised of $km \log q + k\ell \log q$ bits. To obtain a trade-off between the public key and ciphertext sizes, one can vary the parameters k and ℓ, while keeping its product (the total number of bits N) fixed. To obtain the minimum combined public key and ciphertext size, one should set $k \approx \ell \approx \sqrt{N}$. In this way, encrypting N bits requires $\approx 256 + 2\sqrt{N} m \log q + N \log q$ bits, which in practice will be dominated by the $2\sqrt{N} m \log q$ term because one generally never needs to use public key encryption to encrypt more than $N = 256$ bits before switching to symmetric encryption.

The security of this cryptosystem is again directly based on $\mathsf{LWE}_{m,q,\beta}$. Notice that the public key $(\mathbf{A}, \mathbf{AS} + \mathbf{E}_1)$ can be rewritten as $(\mathbf{A}, \mathbf{As}_1 + \mathbf{e}_1, \ldots, \mathbf{As}_\ell + \mathbf{e}_\ell)$ where $\mathbf{s}_i$ and $\mathbf{e}_i$ are, respectively, the i^{th} columns of $\mathbf{S}$ and $\mathbf{E}_1$. The indistinguishability of $(\mathbf{A}, \mathbf{T})$ from uniform then directly follows from $\mathsf{LWE}_{m,q,\beta}$ using the usual hybrid argument with a loss of $\log \ell$ bits of security.[3]

Writing $\mathbf{A}' = [\mathbf{A} \mid \mathbf{T}]$, we again use the $\mathsf{LWE}_{m,q,\beta}$ assumption (and the hybrid argument) to note that the distribution $(\mathbf{A}', \mathbf{RA}' + [\mathbf{E}_2 \mid \mathbf{E}_3])$ is indistinguishable from the uniform distribution, and therefore

$$(\mathbf{A}, \mathbf{T}, \mathbf{U}, \mathbf{V}) = \left(\mathbf{A}', \mathbf{RA}' + [\mathbf{E}_2 \mid \mathbf{E}_3 + \frac{q}{2}\mathbf{M}]\right) \tag{15}$$

is indistinguishable from uniform for any fixed message $\mathbf{M}$.

Decryption is done exactly the same way as we've been doing it for all the other schemes in this section. Given the ciphertext $(\mathbf{U}, \mathbf{V})$, the decryptor computes

$$\mathbf{V} - \mathbf{US} = \mathbf{R}(\mathbf{AS} + \mathbf{E}_1) + \mathbf{E}_3 + \frac{q}{2}\mathbf{M} - (\mathbf{RA} + \mathbf{E}_2)\mathbf{S} \tag{16}$$

$$= \mathbf{RE}_1 + \mathbf{E}_3 + \frac{q}{2}\mathbf{M} - \mathbf{E}_2\mathbf{S}. \tag{17}$$

From above, observe that the $(i,j)^{th}$ coefficient of $\mathbf{V} - \mathbf{US}$ is equal to

$$\mathbf{r}^T\mathbf{e}_1 + e_3 + \frac{q}{2}\mu - \mathbf{e}_2^T\mathbf{s},$$

[3] As with many applications of the hybrid argument, it's not clear in this case whether there is a real security loss or it's just an artefact of the proof.

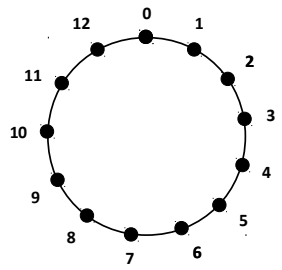

Figure 1: A representation of $\mathbb{Z}_{13}$ as points on a circle. If we define the set $\mathcal{S} = \{\lceil i \cdot 13/4 \rfloor : 0 \leq i < 4\}$, then it consists of the four points $0, 3, 7, 10$. $\mathcal{S}$ can thus be represented by 2 bits and every point in $\mathbb{Z}_{13}$ is within a distance of $\lceil 13/8 \rceil = 2$ of one of the elements of $\mathcal{S}$.

where $\mathbf{r}^T$ and $\mathbf{e}_2^T$ are the i^{th} rows of $\mathbf{R}$ and $\mathbf{E}_2$, $\mathbf{e}_1$ and $\mathbf{s}$ are the j^{th} columns of $\mathbf{E}_1$ and $\mathbf{S}$, and e_3 and μ are the $(i,j)^{th}$ positions of $\mathbf{E}_3$ and $\mathbf{M}$. Since all the vectors are in $\mathbb{Z}_q^m$ and all their coefficients were uniformly chosen from $[\beta]$, we are in the exact same situation as in (9) and so one can set the parameters m, q, β in the exact same way as before to have a small decryption error.

2.5 Some Variations and Optimizations

The scheme presented in the previous section is more of a general framework of what one would do in practice. When instantiating such a scheme with concrete parameters, there are several possible optimizations that one might consider, which we will describe next. We should mention that it's not easy to figure out exactly which optimizations one can use and to exactly set the parameters optimally without actually trying some possibilities and seeing what security / output size one gets.

2.5.1 Reducing the Ciphertext Size by Removing the Low-Order Part

The ciphertext part $\mathbf{V}$ contributes $N \log q$ bits to the total ciphertext size. Suppose that instead of the encryptor publishing all $\log q$ bits of each coefficient of $\mathbf{V}$ as part of the ciphertext, he wanted to transmit only κ bits per coefficient. This is possible, but will add a further error to the decryption equation. Visualizing the additive group $\mathbb{Z}_q$ as points on a circle (c.f. Figure 1), we want to pick a set $\mathcal{S} \subset \mathbb{Z}_q$ of size 2^κ so that the maximum distance between neighboring points in $\mathcal{S}$ (measured in the number of $\mathbb{Z}_q$ points between them) is as small as possible. Note that $q/2^\kappa$ is the smallest we can hope for, so we would like to get as close as possible to this number. One could define such a set as

$$\mathcal{S} = \{\lceil i \cdot q/2^\kappa \rfloor : 0 \leq i < 2^\kappa\}. \tag{18}$$

If $2^\kappa \mid q$, then the distance between all the neighboring points is the same. Otherwise all distances are within 1 of each other, which is as good as one can hope for.

The crucial feature of the set $\mathcal{S}$ is that every $v \in \mathbb{Z}_q$ is within a distance of $\lceil q/2^{\kappa+1} \rceil$ of some element of $\mathcal{S}$. Let us define $\texttt{HIGH}_S(v)$ to be the element in S closest to v and $\texttt{LOW}_S(v)$ to be $v - \texttt{HIGH}_S(v)$. Then instead of transmitting $\mathbf{V} \in \mathbb{Z}_q^{k \times \ell}$ as part of the ciphertext, one could transmit a $\mathbf{V}' \in \mathcal{S}^{k \times \ell} = \texttt{HIGH}_S(\mathbf{V})$. Note that there exists an $\mathbf{E}' \in \left[\lceil q/2^{\kappa+1} \rceil\right]^{k \times \ell} = \texttt{LOW}_S(\mathbf{V})$ such that $\mathbf{V} = \mathbf{V}' + \mathbf{E}'$.

If the ciphertext $(\mathbf{U}, \mathbf{V}')$ is created in this fashion, then the decryption $\mathbf{V}' - \mathbf{US}$ will

produce

$$\mathbf{V}' - \mathbf{US} = \mathbf{RE}_1 + \mathbf{E}_3 - \mathbf{E}' + \frac{q}{2}\mathbf{M} - \mathbf{E}_2\mathbf{S}, \tag{19}$$

where the only difference with the decryption in (17) is the presence of the $\mathbf{E}'$ term. Notice that $\mathbf{E}'$ has limited effect on the decryption error because there other error terms that are involved in an inner product which produce much larger coefficients in (19). In practice, one can usually set κ to be a small constant like 3 or 4 without the decryption error increasing too much. This implies that instead of contributing $N \log q$ bits to the ciphertext, $\mathbf{V}$ contributes just κN bits. Assuming that $\mathbf{V}$ is uniformly distributed, we can obtain the exact distribution of $\mathbf{E}'$ and then compute the decryption error probability using the exact same techniques as in Section 2.3.2.

In some cases it may also make sense to use a similar bit reduction procedure on the ciphertext part $\mathbf{U}$. Here, though, one cannot remove bits without noticeably increasing the decryption error because any error added to $\mathbf{U}$ will get multiplied by $\mathbf{S}$, and so we would get an additional $\mathbf{E}''\mathbf{S}$ term in (19), where $\mathbf{E}''$ is defined analogously to $\mathbf{E}'$. But because $\mathbf{U}$ is the dominant source of ciphertext size, reducing the number of bits in $\mathbf{U}$ even by a little bit, could make a noticeable difference as well. The trick is then to balance the decryption error and the ciphertext size using trial-and-error.

2.5.2 Modulus Switching / Compression / Decompression

We now make the discussion about compression from the previous section more concrete. We first define an operation that takes an element from one set into another. When the target set is smaller, then this can be seen as rounding, or compression. When the target set is larger, this can be seen as lifting or decompression.

Definition 3. For an element $x \in \mathbb{Z}_q$ and some positive integer p, we define a mapping from $\mathbb{Z}_q$ to $\mathbb{Z}_p$ as

$$\lceil x \rfloor_{q \to p} = \left\lceil \frac{x \cdot p}{q} \right\rfloor \in \mathbb{Z}_p.$$

One should observe that the resulting element of $\mathbb{Z}_p$ is independent of the exact representative element x in $\mathbb{Z}_q$ – that is, all elements in the set $x + q\mathbb{Z}$ produce the same result in $\mathbb{Z}_p$ (since $\left\lceil \frac{(x+q\mathbb{Z})p}{q} \right\rfloor = \left\lceil \frac{xp}{q} + p\mathbb{Z} \right\rfloor = \left\lceil \frac{xp}{q} \right\rfloor + p\mathbb{Z}$); and thus the above definition is well-formed. We now prove the core Lemma about how one could use the above function to meaningfully compress and decompress data. In particular, it states that if one first compresses an element in $\mathbb{Z}_q$ to one in $\mathbb{Z}_p$ (where $p < q$) and then decompresses back to $\mathbb{Z}_q$, the result will not be too far away from the original element.

Lemma 1. *For integers $p < q$ and $x \in \mathbb{Z}_q$, it holds that*

$$\left\lceil \lceil x \rfloor_{q \to p} \right\rfloor_{p \to q} = x + \eta \in \mathbb{Z}_q,$$

for some $\eta \in \mathbb{Z}$ satisfying $|\eta| \leq \frac{q}{2p} + \frac{1}{2}$.

Proof. By definition of rounding, there exists a $\delta \in \mathbb{Q}$, $|\delta| \leq \frac{1}{2}$, such that for $x \in \mathbb{Z}_q$,

$$\lceil x \rfloor_{q \to p} = \left\lceil \frac{xp}{q} \right\rfloor = \frac{xp}{q} + \delta \in \mathbb{Z}_p.$$

Therefore,

$$\left\lceil \lceil x \rfloor_{q \to p} \right\rfloor_{p \to q} = \left\lceil \frac{\left(\frac{xp}{q} + \delta\right) \cdot q}{p} \right\rfloor = \left\lceil x + \frac{\delta q}{p} \right\rfloor = b + \frac{\delta q}{p} + \delta' \in \mathbb{Z}_q,$$

for some $|\delta'| \leq \frac{1}{2}$, which is a result of the rounding, and $\frac{\delta q}{p} \leq \frac{q}{2p}$. □

To relate the modulus switching operation from Definition 3 to the example in Section 2.5.1, we relate the definitions of $\mathcal{S}, \kappa, \texttt{HIGH}_\mathcal{S}, \texttt{LOW}_\mathcal{S}$ from the previous section to the notions introduced here.

1. $2^\kappa = p$
2. $\mathcal{S} = \left\{ \lceil x \rfloor_{p \to q}, \text{ for } x \in \mathbb{Z}_p \right\}$
3. $\texttt{HIGH}_\mathcal{S}(x) = \left\lceil \lceil x \rfloor_{q \to p} \right\rfloor_{p \to q}$
4. $\texttt{LOW}_\mathcal{S}(x) = x - \texttt{HIGH}_\mathcal{S}(x)$

And Lemma 1 proves that $\texttt{LOW}_\mathcal{S}(x) \in [\lceil q/2p \rceil]$. In implementations of compression, we would of course not send the element $\texttt{HIGH}_S(x) \in \mathbb{Z}_q$, but rather $\lceil x \rfloor_{q \to p} \in \mathbb{Z}_p$, because the latter can be more compactly represented. One should think of $\texttt{HIGH}_S(x) \in \mathbb{Z}_q$ as the decompression of the compression of x.

We also point out that recovering the message m from the noisy decryption output as in (9) can also be done using the compression function. In particular, for an element $x \in \mathbb{Z}_q$, $\lceil x \rfloor_{q \to 2}$ will map to 0 if x is closer to 0 than to $q/2$, and to 1 otherwise. Thus one can rewrite the decryption procedure (e.g. (8)) as

$$\left\lceil v - \mathbf{u}^T \mathbf{s} \right\rfloor_{q \to 2}. \tag{20}$$

Another point worth mentioning is that, in terms of minimizing the distance between all points $\texttt{HIGH}_\mathcal{S}$ when $\mathcal{S}$ is fixed to be a certain size, the definition of $\mathcal{S}$ as in item (2) is optimal for all p, q. For some particular values of p, q, we could, however, define the set $\mathcal{S}$ differently, while still achieving the same optimality. The reason we may want to do this is to avoid performing the $\lceil x \rfloor_{q \to p}$ operation which requires division by (usually a prime) q. For example, if we have $p = 4$ and $q = 33$, then we could define the set $\mathcal{S} = \{0, 8, 16, 24\}$. Notice that to compute $\texttt{HIGH}_S(x)$ for this set will only require modular reductions and divisions modulo 8, which is usually a very efficient operation in CPUs since just register shifts are required.

In the Kyber encryption scheme (Section 4.6), because the modulus q is small, we are not able to find a prime that satisfies a condition we need for optimal multiplication efficiency (see Section 4.5.2) and to have a set $\mathcal{S}$ such that rounding to it involves only the nice operations above. Thus we use the generic definition of $\mathcal{S}$ as in item (2). In the Dilithium signature scheme (Section 5.7), the modulus q is larger and so we are able to set it so that we have a nice $\mathcal{S}$ and still have fast multiplication.

2.5.3 Learning with Rounding

By the LWE assumption, the distribution $(\mathbf{A}, \mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}) \in \mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^n$ looks indistinguishable from uniform when the coefficients of $\mathbf{s}$ and $\mathbf{e}$ come from $[\beta]$. If we round each coefficient of $\mathbf{t}$ to the nearest point in some subset $\mathcal{S} \subseteq \mathbb{Z}_q$, as in Section 2.5.1, then the distribution of $(\mathbf{A}, \texttt{HIGH}_S(\mathbf{t}))$ is still indistinguishable from $(\mathbf{A}, \texttt{HIGH}_S(\mathbf{u}))$, where $\mathbf{u}$ is uniformly random. Let's now examine $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}$. Since $\mathbf{e}$ has coefficients in a small subset $[\beta]$, it may turn out that it doesn't have *any* effect on the value of $\texttt{HIGH}_S(\mathbf{A}\mathbf{s} + \mathbf{e})$. In particular, when $\mathcal{S}$ is defined as in (18), then the probability (over the randomness of $\mathbf{A}, \mathbf{s}, \mathbf{e}$) that $\texttt{HIGH}_S(\mathbf{A}\mathbf{s} + \mathbf{e}) = \texttt{HIGH}_S(\mathbf{A}\mathbf{s})$ is approximately $\left(1 - \frac{\beta|\mathcal{S}|}{2q}\right)^n$. To see this, notice that if some coefficient of $\mathbf{A}\mathbf{s}$ is not within $\beta/2$ of the midpoint between two points in $\mathcal{S}$, then adding an error in $[\beta]$ will not change the point in $\mathcal{S}$ to which it gets rounded.

So whenever q is large with respect to β and $|\mathcal{S}|$, adding $\mathbf{e}$ doesn't make a difference, and so there is no reason to add it in the first place! Distinguishing the distribution

$(\mathbf{A}, \mathtt{HIGH}_S(\mathbf{As}+\mathbf{e}))$ from uniform is called the Learning with Rounding (LWR) problem, and it is at least as hard as LWE whenever adding $\mathbf{e}$ doesn't affect the rounded output [BPR12, AKPW13, BGM+16]. Sometimes this assumption is used in constructions of encryption schemes even when the parameters do not permit a reduction from LWE (i.e. q is not large enough). In this case, it's a separate assumption and its relationship with LWE is unclear.

The advantage of making the LWR assumption is that it permits smaller parameters due to the fact that it does not add the error $\mathbf{e}$. As an illustration, let's take a look at (19). The error $\mathbf{E}_3$ that was added in (14) is a term needed for the LWE assumption, whereas the error $\mathbf{E}'$ naturally occurred due to the rounding. Since we already have $\mathbf{E}'_3$, the $\mathbf{E}_3$ might be unnecessary. Thus under the LWR assumption, the rounding is performing two functions – it adds a uniform-like error vector and it reduces the ciphertext size. Similarly, instead of adding an error to the ciphertext $\mathbf{U}$, we can simply round it to some (different) set $\mathcal{S}$ which has the effect of adding deterministic error to $\mathbf{U}$ – thus making the $\mathbf{E}_2$ term in (14) possibly redundant and unnecessary.

2.5.4 Encrypting More Bits per Slot

Our encryption scheme packed the N-bit message into a matrix $\mathbf{M} \in \{0,1\}^{k \times \ell}$. We could have instead, for example, packed it into a matrix

$$\mathbf{M} \in \{0, 1, \ldots, 2^b - 1\}^{(k/\sqrt{b}) \times (\ell/\sqrt{b})}.$$

For decryption to work, we would need to make two small changes to the encryption and decryption algorithms, while also adjusting the parameters. We will replace the $\frac{q}{2}\mathbf{M}$ term in the encryption algorithm with $\frac{q}{2^b}\mathbf{M}$. The part of the decryption algorithm computing $\mathbf{V} - \mathbf{US}$ will then result in the $\frac{q}{2}$ term being similarly replaced with $\frac{q}{2^b}$ in (17). This means that in order for decryption to produce the correct result, one would need to have the remaining error terms (i.e. the terms in (17) not involving $\mathbf{M}$) be in $[q/2^{b+1}]$ rather than $[q/4]$ as before.

The trivial modification of the parameters in order for decryption to again work would involve simply increasing q by a factor of approximately 2^{b-1}. Note that this could have a positive effect of the size of the ciphertext and public key. Take, for example, the size of the public key $256 + \ell m \log q$. If we increase q by a factor of 2^{b-1}, but decrease ℓ by a factor of 2, we will obtain $256 + \frac{\ell m}{2}(\log q + b - 1)$. In other words, the size is smaller when $b - 1 < \log q$. But increasing q while keeping eveything else constant has the effect of decreasing security (as mentioned in the beginning of Section 2.3, the problem becomes harder as the ratio β/q grows), so we will need to either increase β or increase m. The way to achieve the optimal parameters is to try a few possible options.

2.5.5 LWE Encryption with a "Non-Square" Public Key

In all the versions of public key encryption that we presented thus far, the security proof used the LWE assumption to argue that the public key is computationally indistinguishable from uniform, and then used the uniformity of the public key and the LWE assumption one more time to argue that the ciphertext is computationally indistinguishable from uniform. It is sometimes, however, useful to have the public key be truly uniform. Or, similarly, have the ciphertext be truly uniform under the (computational) assumption that the public key was. In other words, for some applications we may want to apply the LWE assumption only once. We'll now explain how to construct such a cryptosystem and also give some intuition for when something like this may actually be useful in practice.

To make the public key or the ciphertext uniform, we will need to use the *leftover hash lemma*. This lemma applied to our scenario roughly states that if $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$ (where q is a prime) and $\mathbf{s} \leftarrow [\beta]^m$, where $(2\beta+1)^m \gg q^n$, then the distribution of $(\mathbf{A}, \mathbf{As})$ is

statistically-close to the distribution of $(\mathbf{A}, \mathbf{u})$, where $\mathbf{u} \leftarrow \mathbb{Z}_q^n$. In other words, if $\mathbf{s}$ is chosen uniformly from some set, then the size of this set should be larger than the size of the range of the function $\mathbf{As}$. With the leftover hash lemma in hand, it's fairly easy to see how one would modify either the key generation (13) or the encryption (14) procedure to ensure that either the public key or the ciphertext are random.

If we would like the public key to be uniformly random, we replace (13) with

$$\mathsf{sk}: \mathbf{S} \leftarrow [\beta']^{m \times \ell},\ \mathsf{pk}: \left(\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}, \mathbf{T} = \mathbf{AS}\right),\ \text{where } (2\beta'+1)^m > q^n. \tag{21}$$

Note that since we're not using the LWE assumption twice, there is no longer a reason to have β in the key generation be the same as in encryption – and so we give them different names. The encryption and decryption equation can remain exactly as in (14) and (17). When setting the parameters to make sure that decryption returns the correct answer, one should pay attention to the fact that the term $\mathbf{E}_2\mathbf{S}$ in (17) is larger due to the fact that m and/or β' are larger than before, and also to the fact that the term $\mathbf{RE}_1$ in (17) is 0 because $\mathbf{E}_1$ does not exist in the key generation procedure.

The necessary modifications for the scenario where we would like to have the ciphertext be uniformly-random (after applying the LWE assumption to conclude that the public key in (13) is indistinguishable from random) uses the exact same principle. In particular, the dimension of $\mathbf{A}$ and the distribution of $\mathbf{R}$ should be such that $([\mathbf{A} \mid \mathbf{T}], \mathbf{R}[\mathbf{A} \mid \mathbf{T}]$ where $[\mathbf{A} \mid \mathbf{T}] \leftarrow \mathbb{Z}_q^{n \times m}$ are indistinguishable from $([\mathbf{A} \mid \mathbf{T}], [\mathbf{U} \mid \mathbf{V}])$, where $\mathbf{U}, \mathbf{V}$ are uniform.

Without going into much detail, we will now touch upon why one would want to have either the public key or the ciphertext be truly uniform. The two examples we provide are by no means exhaustive, but give a flavor about where and why sacrificing some efficiency may be required. The main application of a uniform public key is in the lattice construction of identity-based encryption [GPV08]. In this scenario, the public key of a user with identity $x \in \{0,1\}^*$ is $(\mathbf{A}, \mathbf{t}_x)$, where $\mathbf{t}_x = \mathcal{H}(x)$ for a cryptographic hash function $\mathcal{H}$ (modeled as a random oracle) that maps $\{0,1\}^*$ to $\mathbb{Z}_q^n$. In other words, $\mathbf{A}$ is common to all users, while $\mathbf{t}_x$ is unique to every user and is uniformly-random. In an IBE scheme, the public key of user x should be computable by anyone. It is therefore not possible to generate it from some secret information (e.g. like the secret key) as in (6). On the other hand, there needs to exist a way to associate a secret key with the public key. The solution to this problem is for the master authority to possess a "trap-door" to $\mathbf{A}$, which allows him to create a low-norm vector $\mathbf{s}_x$ satisfying $\mathbf{As}_x = \mathbf{t}_x$. This $\mathbf{s}_x$ is then the secret decryption key of user x. Interestingly, note that the key generation is *not* done as in (21) – in particular, the secret key is created *after* the public key. This is only possible because the master authority created $\mathbf{A}$ together with a trapdoor. Despite this difference in the order of key creation, the main result of [GPV08] provides algorithms such that the distribution of $\mathbf{A}, \mathbf{s}, \mathbf{t}$ will be the same whether one chooses $\mathbf{s}$ before computing $\mathbf{t}$ or the other way around.

An application where one would want the ciphertext to be uniformly random comes up when there is a chance that something about the ciphertext randomness $\mathbf{r}$ is leaked. It was observed in [AGV09] that one could leak something about $\mathbf{r}$, and by the leftover hash lemma, the ciphertext would still remain uniformly random.

2.6 Non-Interactive Key Exchange

The encryption schemes described in this section can be trivially converted to a passively-secure *key transport* scheme in which the two parties wish to agree on a shared symmetric key (e.g. an AES key). The protocol would simply involve the first party creating a public key (as in (13)) and sending it to the second party. The second party then chooses the AES key and encrypts it as the message $\mathbf{M}$ and sends the ciphertext (as in (14)). The first party then decrypts the shared key $\mathbf{M}$.

While the above protocol is good enough for most purposes where classical key exchange (e.g. Diffie-Hellman) is used, there is a critical order to the flow of the protocol. The user sending $\mathbf{M}$ cannot send his message before receiving the public key. In the classical Diffie-Hellman protocol, on the other hand, either user can send his g^{x_i} first. One can similarly create a protocol with this property from the LWE problem, but it will be a much less efficient way of exchanging keys where this arbitrary flow property isn't needed.

We will describe this simple protocol for the case of agreeing on one random bit. To agree on more bits, one would apply the same ideas as in Section 2.4. There is a public random matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times m}$ that is trusted by everyone to have been honestly generated (e.g. it is expanded by SHAKE from the seed 0). Both parties choose vectors $\mathbf{s}_1, \mathbf{s}_2, \mathbf{e}_1, \mathbf{e}_2 \leftarrow [\beta]^m$. The first party sends $\mathbf{u}_1^T = \mathbf{s}_1^T\mathbf{A} + \mathbf{e}_1^T$, while the second sends $\mathbf{u}_2 = \mathbf{A}\mathbf{s}_2 + \mathbf{e}_2$. Upon receiving the message from its respective counter-party, the first party computes $\mathbf{s}_1^T\mathbf{u}_2$ and if this value is closer to $q/2$ than to 0 (i.e. it's between $q/4$ and $3q/4$), it will set the shared bit $b_1 = 1$ (otherwise it will set $b_1 = 0$). The second party computes $\mathbf{u}_1^T\mathbf{s}_2$ and sets $b_2 = 0$ or 1 using the same rule. In short, they end up with

$$\mathbf{s}_1^T\mathbf{u}_2 = \mathbf{s}_1^T\mathbf{A}\mathbf{s}_2 + \mathbf{s}_1^T\mathbf{e}_2 \tag{22}$$

$$\mathbf{u}_1^T\mathbf{s}_2 = \mathbf{s}_1^T\mathbf{A}\mathbf{s}_2 + \mathbf{e}_1^T\mathbf{s}_2. \tag{23}$$

and hope that the error terms terms $\mathbf{s}_1^T\mathbf{e}_2$ and $\mathbf{e}_1^T\mathbf{s}_2$ (which have maximum magnitude $m\beta^2$) don't cause $b_1 \neq b_2$.

Notice that the probability that $b_1 \neq b_2$ is at most the probability that $\mathbf{s}_1^T\mathbf{A}\mathbf{s}_2$ falls into the "dangerous" ranges near $3q/4$ and $q/4$. In particular,

$$\Pr[b_1 \neq b_2] < \Pr\left[\mathbf{s}_1^T\mathbf{A}\mathbf{s}_2 \in \left[\frac{3q}{4} + m\beta^2, \frac{3q}{4} - m\beta^2\right] \text{ or } \left[\frac{q}{4} + m\beta^2, \frac{q}{4} - m\beta^2\right]\right]. \tag{24}$$

The probability that $\mathbf{s}_1^T\mathbf{A}\mathbf{s}_2$ is any partiular value in $\mathbb{Z}_q$ is $1/q$ and so the above probability is at most $4m\beta^2/q$. In practice, one would use the techniques from Section 2.3.2 to reduce this probability, but one would still end up with a probability of $\Omega(\beta^2\sqrt{m}/q)$ of a mismatch in b_1 and b_2. This is quite different than the situation we had with the encryption schemes where we could set parameters so that the decryption error is exponentially small (or even 0) with respect to q. In the non-interactive key agreement, getting a small error will require setting q to be very large which has an adverse effect on the communication size.

3 Hardness of LWE and Other Lattice Problems

We will now give a geometric view of the LWE problem. While this connection is not really necessary for understanding how most cryptographic constructions work, it is crucial for understanding their security.

3.1 Lattices

At the core of the geometric interpretation of the LWE problem are objects known as *lattices*. An m-dimensional integer lattice Λ is simply a subgroup of the group $(\mathbb{Z}^m, +)$. Such a group can be described via a generating set called a basis. In particular, a lattice Λ defined by a (full-rank) basis $\mathbf{B} \in \mathbb{Z}^{m \times m}$ is

$$\Lambda = \mathcal{L}(\mathbf{B}) = \{\mathbf{v} \in \mathbb{Z}^m \ : \ \exists \mathbf{z} \in \mathbb{Z}^m \text{ s.t. } \mathbf{B}\mathbf{z} = \mathbf{v}\}. \tag{25}$$

In this chapter, we will just restrict ourselves to special types of lattices called *q-ary integer lattices*, as these are the ones that are used in cryptographic constructions. They also have the nice theoretical property that, asymptotically, solving some problem over random instances of these lattices is as hard as solving some problem for *any* lattice. This is the celebrated worst-case to average-case reduction line of research [Ajt96, Reg09] that formed the foundation, and spearheaded the development, of lattice-based cryptography.

For a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, the q-ary lattice Λ defined by $\mathbf{A}$ is

$$\Lambda = \mathcal{L}_q^{\perp}(\mathbf{A}) = \{\mathbf{v} \in \mathbb{Z}^m \ : \ \mathbf{A}\mathbf{v} \equiv 0 \pmod q\} \tag{26}$$

It's not hard to see that under the usual vector addition operation, the above set is a group. For reader's familiar with linear codes, the above two definitions of lattices are akin to describing a code using a generating matrix (25) or a parity-check matrix (26). Even more specifically, the lattices that we will be dealing with are

$$\Lambda = \mathcal{L}_q^{\perp}([\mathbf{A} \mid \mathbf{I}_n]), \tag{27}$$

where $\mathbf{I}_n$ is the $n \times n$ identity matrix. This is not much of a restriction because if the $\mathbf{A}$ in (26) contains n columns that are linearly independent over $\mathbb{Z}_q$ (without loss of generality, suppose that $\mathbf{A} = [\mathbf{A}_1 \mid \mathbf{A}_2]$ where $\mathbf{A}_2 \in \mathbb{Z}_q^{n \times n}$ is invertible), then we can write $\mathbf{A}_2^{-1}\mathbf{A} = [\mathbf{A}_2^{-1}\mathbf{A}_1 \mid \mathbf{I}]$ and $\mathcal{L}_q^{\perp}(\mathbf{A}) = \mathcal{L}_q^{\perp}(\mathbf{A}_2^{-1}\mathbf{A})$, where the latter is in the form of (27). For lattices in the form of (27), it is also easy to switch between the "generator" matrix representation in (25) and the "parity check" matrix representation in (26). It's an easy exercise to check that

$$\mathcal{L}_q^{\perp}([\mathbf{A} \mid \mathbf{I}_n]) = \mathcal{L}\left(\begin{bmatrix} -\mathbf{I}_m & \mathbf{0} \\ \mathbf{A} & q\mathbf{I}_n \end{bmatrix}\right). \tag{28}$$

3.1.1 The Quotient Group and Determinant

The determinant of a full-rank lattice $\Lambda \subseteq \mathbb{Z}^m$, written as $\det(\Lambda)$, is the inverse of the density of Λ in the space $\mathbb{Z}^m$. That is, if we define the set $S_r = \{\mathbf{z} \in \mathbb{Z}^m \ : \ \|\mathbf{z}\| < r\}$, then

$$\det(\Lambda) = \lim_{r \to \infty} \frac{|S_r|}{|\Lambda \cap S_r|}.$$

If $\Lambda = \mathcal{L}(\mathbf{B})$ for a full-rank matrix $\mathbf{B} \in \mathbb{Z}^{m \times m}$, then $\det(\Lambda) = \det(\mathbf{B})$, where the right-hand side is the usual matrix determinant of $\mathbf{B}$. For example, the determinant of the $(n+m)$-dimensional lattice in (28) is $\det(\Lambda) = \det(\mathbf{B}) = q^n$. Another equivalent definition of the determinant of a full-rank m-dimensional lattice Λ is the size of the quotient group $\mathbb{Z}^m/\Lambda$.

The parity-check representation of a lattice, $\Lambda = \mathcal{L}_q^{\perp}(\mathbf{A})$, is convenient for checking whether two vectors in $\mathbb{Z}^m$ are in the same coset of $\mathbb{Z}^m/\Lambda$. In particular, $\mathbf{z}_1$ and $\mathbf{z}_2$ are in the same coset if and only if $\mathbf{A}\mathbf{z}_1 \equiv \mathbf{A}\mathbf{z}_2 \pmod q$. With this observation it's easy to see that when $\Lambda = \mathcal{L}_q^{\perp}([\mathbf{A} \mid \mathbf{I}_n])$, there are exactly q^n cosets; which is consistent with our previous observation that the determinant of the lattice in (28) is q^n.

3.1.2 Distance to the Lattice

For an m-dimensional lattice Λ and any vector $\mathbf{r} \in \mathbb{Z}^m$ (not necessarily in Λ), the ℓ_p-norm distance from $\mathbf{r}$ to the lattice is defined as

$$\Delta_p(\mathbf{r}, \Lambda) = \min_{\mathbf{v} \in \Lambda} \|\mathbf{v} - \mathbf{r}\|_p. \tag{29}$$

Observe that for any two elements $\mathbf{r}_1$ and $\mathbf{r}_2$ belonging to the same coset of $\mathbb{Z}^m/\Lambda$, $\Delta_p(\mathbf{r}_1, \Lambda) = \Delta_p(\mathbf{r}_2, \Lambda)$, and so distances are well-defined notions for cosets as well. Therefore if $\Lambda = \mathcal{L}_q^{\perp}(\mathbf{A})$ and $\mathbf{t} \equiv \mathbf{A}\mathbf{z} \pmod q$ defines a coset $\mathbf{z} + \Lambda$, then we write

$$\Delta_p^C(\mathbf{t}, \Lambda) = \Delta_p(\mathbf{z}, \Lambda).$$

For convenience, we write Δ^C instead of Δ to denote that $\mathbf{t}$ is the image of the coset under $\mathbf{A}$, rather than some coset representative.

We will now prove some statements about the (non)-existence of short vectors in random lattices. Lemmas 2 and 3 show that random cosets are far from random q-ary lattices and that q-ary lattices don't have very short vectors. Lemma 4 proves a partial converse, giving a lower bound on the length of the shortest vector in any q-ary lattice.

Lemma 2. *For any q and any $\mathbf{t} \in \mathbb{Z}_q^n$ that has a coefficient t_i such that $\gcd(t_i, q) = 1$,*

$$\Pr_{\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}}[\exists \mathbf{z} \in [\beta]^{n+m} \ s.t. \ [\mathbf{A} \mid \mathbf{I}_n]\mathbf{z} \equiv \mathbf{t} \pmod q] \leq (2\beta+1)^{n+m}/q^n$$

Proof. Since $\mathbf{t}$ has some coefficient relatively prime to q, it must also be the case that some coefficient of $\mathbf{z}$ must also be relatively prime to q. Without loss of generality, assume that it's the first one. Then we have that for a fixed $\mathbf{z}$,

$$\begin{aligned}
\Pr_{\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}}[[\mathbf{A} \mid \mathbf{I}_n]\mathbf{z} \equiv \mathbf{t} \pmod q] &= \Pr_{\mathbf{a} \leftarrow \mathbb{Z}_q^n, \mathbf{A}' \leftarrow \mathbb{Z}_q^{n \times (m-1)}}[[\mathbf{a} \mid \mathbf{A}' \mid \mathbf{I}_n] \begin{bmatrix} z_1 \\ \mathbf{z}' \end{bmatrix} \equiv \mathbf{t} \pmod q] \\
&= \Pr_{\mathbf{a} \leftarrow \mathbb{Z}_q^n}[\mathbf{a} z_1 \equiv \mathbf{t} - [\mathbf{A}' \mid \mathbf{I}_n]\mathbf{z}' \pmod q] \\
&= \Pr_{\mathbf{a} \leftarrow \mathbb{Z}_q^n}[\mathbf{a} \equiv z_1^{-1}(\mathbf{t} - [\mathbf{A}' \mid \mathbf{I}_n]\mathbf{z}') \pmod q] = q^{-n},
\end{aligned}$$

where $z^{-1} \pmod q$ exists because we assumed that $\gcd(z_1, q) = 1$. Since there are $(2\beta+1)^{n+m}$ possible vectors in $[\beta]^{n+m}$, the statement in the lemma follows by the union bound. □

Corollary 1.

$$\Pr_{\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}, \mathbf{t} \leftarrow \mathbb{Z}_q^n}[\Delta_\infty^C(\mathbf{t}, \Lambda) \leq \beta] \leq (1 - |\mathbb{Z}_q^*|/q)^n + (2\beta+1)^{n+m}/q^n,$$

where $\Lambda = \mathcal{L}_q^{\perp}([\mathbf{A} \mid \mathbf{I}_n])$. □

Some "popular" settings for q in lattice cryptography are prime q and q that are powers of 2. In both of these cases, the first term in the probability bound is negligible in n ($(1/q)^n$ and 2^{-n}, respectively). And so, whenever $\beta^{1+m/n} \ll q$, random cosets will be more than distance β away from Λ. □

The next lemma states that the probability, over the choice of $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$, that the lattice $\mathcal{L}_q^\perp([\mathbf{A} \mid \mathbf{I}_n])$ has a short non-zero vector is small. We only prove this lemma for a prime q, as it's somewhat more messy for other choices. The proof of the lemma is virtually identical to that of Lemma 2.

Lemma 3. *For any prime q,*

$$\Pr_{\mathbf{A}\leftarrow\mathbb{Z}_q^{n\times m}}[\exists \mathbf{z} \in [\beta]^{n+m} \setminus \{\mathbf{0}\} \text{ s.t. } [\mathbf{A} \mid \mathbf{I}_n]\mathbf{z} \equiv \mathbf{0} \pmod q] \leq (2\beta+1)^{n+m}/q^n.$$

□

The next Lemma is a converse of the one above; it gives a lower bound on the length of an existing non-zero vector.

Lemma 4. *For any q and any $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$,*

$$\exists \mathbf{z} \in \left[q^{n/(n+m)}\right]^{n+m} \setminus \{\mathbf{0}\} \text{ s.t. } [\mathbf{A} \mid \mathbf{I}_n]\mathbf{z} \equiv \mathbf{0} \pmod q$$

Proof. The proof is by the pigeonhole principle. There are more than $(q^{n/(n+m)})^{n+m} = q^n$ vectors in $\mathbb{Z}^{n+m}$ whose coefficients are between 0 and $q^{n/(n+m)}$. Since there are only q^n possibilities for the value of $\mathbf{Az} \bmod q$, there must exist two distinct $\mathbf{z}_1, \mathbf{z}_2$ with coefficients in the aforementioned range such that $\mathbf{Az}_1 \equiv \mathbf{Az}_2 \pmod q$. Thus $\mathbf{z}_1 - \mathbf{z}_2 \in \left[q^{n/(n+m)}\right]^{n+m}$ and $\mathbf{A}(\mathbf{z}_1 - \mathbf{z}_2) \equiv \mathbf{0} \pmod q$. □

Looking at the statements of Lemmas 3 and 4, we see that the boundary between there existing a vector in $\Lambda = \mathcal{L}_q^\perp([\mathbf{A} \mid \mathbf{I}_n])$ with coefficients in $[\beta]$ and there not being one with high probability is pretty sharp. Lemma 4 states that when $\beta = q^{n/(n+m)}$, such a vector always exists. On the other hand, if we set $\beta < \frac{1}{4}q^{n/(n+m)}$, then the probability of there being a vector in Λ with coefficients in $[\beta]$ is less than $2^{-(n+m)}$.

3.2 Finding Short Vectors in Random Lattices (the SIS Problem)

A fundamental computational question one can ask about a lattice is to find a "short" (non-zero) vector in it. When specifically tailored to the lattices we've been dealing with above, the question simply becomes to find a non-zero $\mathbf{z} \in [\beta]^{n+m}$ such that $[\mathbf{A} \mid \mathbf{I}_n]\mathbf{z} \equiv \mathbf{0} \pmod q$. Lemma 4 states that when $\beta = q^{n/(n+m)}$, such a vector surely exists, but the proof does not give us any way of finding it. As of today, all known (quantum) algorithms for finding such vectors (for uniformly random $\mathbf{A}$) take $2^{\Omega(m+n)}$ time (c.f. [AKS01, ADRS15, AS18]).

The problem does get easier as β increases. Clearly if $\beta = q/2$, then the problem is trivially solved by just setting the coefficients of $\mathbf{z}$ that are being multiplied by $\mathbf{I}_n$ to the target coefficients. For smaller value of β, one would run an algorithm that finds a vector in the lattice that is some factor larger than the smallest vector. All modern efficient (i.e. polynomial-time) algorithms for finding such short vectors are descendants of the famous LLL algorithm [LLL82] which guarantees to find a vector of length at most $2^{O(n+m)}$ times larger than that of the shortest vector in the lattice. Lemma 4 then implies that for a random $\mathbf{A}$, the LLL algorithm will find a vector $\mathbf{z} \in \left[2^{\Omega(n+m)} \cdot q^{n/(n+m)}\right]^{n+m}$ in $\mathcal{L}_q^\perp([\mathbf{A} \mid \mathbf{I}_n])$.

While the length of the vector the LLL algorithm is guaranteed to find is exponentially larger (in the dimension of the lattice) than the length of the shortest vector, in practice,

this exponent is not too large. Until experiments were run with lattices of sufficiently high dimension (at least 100), it wasn't even clear whether the real approximation factor of LLL was exponential or just linear. As it turns out, the approximation factor is indeed exponential in the dimension, but the base of the exponent is rather small.

Experiments in [GN08, MR08] showed that one can find vectors in random lattices Λ of the form (27) (of dimension $m+n$) of length approximately

$$\min\left\{q, \det(\Lambda)^{1/(n+m)} \cdot \delta^{n+m}\right\} = \min\left\{q, q^{n/(n+m)} \cdot \delta^{n+m}\right\} \tag{30}$$

where δ depends on how much time the algorithm takes. Deducing a good approximation to the running time of an LLL-type algorithm for a particular value of δ is fairly involved (c.f. [ACD+18, ADH+19]) and is out of scope of this manuscript. As a very rough rule of thumb, $\delta = 1.01$ is considered within reach, whereas $\delta = 1.005$ may never be achieved for lattices of high-enough dimension (e.g. more than 500).

Note that if the dimension of the lattice Λ is very large, one can just remove arbitrarily many columns from $\mathbf{A}$ and run LLL on the resulting lattice. In particular, it is optimal to have the dimension of the lattice be $\sqrt{n \log q / \log \delta}$ (see [MR08, Chapter 3]), which results in the size of the found vector being

$$\min\left\{q, 2^{2\sqrt{n \log q \log \delta}}\right\}. \tag{31}$$

The problem of finding a short vector in a random lattice as in (27) is called the SIS (Short Integer Solution) problem. It is known that solving random instances of this problem is at least as hard as solving some related problem in all lattices [Ajt96, MR07]. We will write $\mathsf{SIS}_{n,m,q,\beta}$ to be the problem of finding a vector with coefficients in $[\beta]$ when given a random lattice as in (27).

Definition 4. For positive integers m, n, q, and $\beta < q$, the $\mathsf{SIS}_{n,m,q,\beta}$ problem asks to find, for a randomly-chosen matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$, vectors $\mathbf{s}_1 \in [\beta]^m$ and $\mathbf{s}_2 \in [\beta]^n$ such that $\mathbf{A}\mathbf{s}_1 + \mathbf{s}_2 = \mathbf{0} \pmod q$.

Notice that by Lemma 3, the $\mathsf{SIS}_{n,m,q,\beta}$ problem is vacuously hard when $\beta \ll \frac{1}{2} q^{n/(n+m)}$ and (31) states that the problem gets easier as β grows. Also note from (31) that once m is larger than $\sqrt{n \log q / \log \delta}$, then it has no impact on the hardness of the problem since it does not figure in the formula for the size of the found vector. This is quite similar to the situation in the $\mathsf{LWE}_{n,m,q,\beta}$ problem, where the parameter n does not matter much. And as in that case, we will just write $\mathsf{SIS}_{n,q,\beta}$.

3.3 The LWE Lattices

Let us now rephrase the $\mathsf{LWE}_{n,m,q,\beta}$ problem from Section 2.3 in the language of lattices. If we choose a random $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$ and random $\mathbf{s} \leftarrow [\beta]^m, \mathbf{e} \leftarrow [\beta]^n$, and output $(\mathbf{A}, \mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e})$, then this is the same as outputting a lattice $\Lambda = \mathcal{L}_q^{\perp}([\mathbf{A} \mid \mathbf{I}_n)$ for a random $\mathbf{A}$ and a coset $\mathbf{t}$ in $\mathbb{Z}_q^m / \Lambda$ such that $\Delta_\infty^C(\mathbf{t}, \Lambda) \le \beta$. On the other hand, outputting $(\mathbf{A}, \mathbf{u})$ for a random $\mathbf{u} \leftarrow \mathbb{Z}_q^n$ is akin to outputting the lattice Λ and a random coset of $\mathbb{Z}^m / \Lambda$. The $\mathsf{LWE}_{n,m,q,\beta}$ problem can therefore be restated as trying to distinguish between cosets that are close to the lattice and random cosets.

Our encryption scheme had $m = n$ and for correctness we needed $\beta^2 = O(q/\sqrt{m})$, and therefore $\beta \ll \sqrt{q}$. From Corollary 1, this implies that random cosets will be further away than β from the lattice. Thus the $\mathsf{LWE}_{n,m,q,\beta}$ problem for parameters that make the encryption scheme work can be seen as distinguishing between cosets that are close to the lattice with those that are far away.

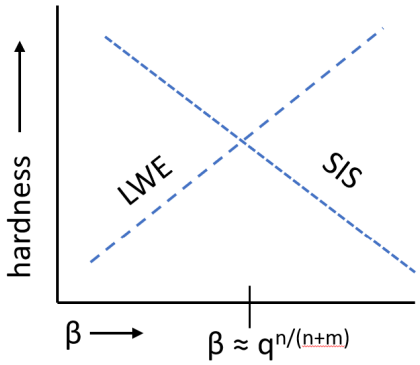

Figure 2: The hardness of $\mathsf{LWE}_{n,m,q,\beta}$ and $\mathsf{SIS}_{n,m,q,\beta}$ for fixed n, m, q, and varying β. The lines are not meant to describe the concrete hardness of these problems, but rather to illustrate the dependence of the hardness of these problems on β. The intersection point is approximately at $\beta = q^{n/(n+m)}$.

We can now show how to use an algorithm that solves SIS to solve LWE. If we're given an $\mathsf{LWE}_{n,m,q,\beta}$ instance $(\mathbf{A}, \mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e})$, then the idea for distinguishing this from random is to find short vectors $\mathbf{r}_1, \mathbf{r}_2$ such that

$$\mathbf{r}_1^T \cdot \mathbf{A} + \mathbf{r}_2^T = \mathbf{0}. \tag{32}$$

Once we find such vectors, we compute $\mathbf{r}_1^T \cdot \mathbf{t}$. If $\mathbf{t}$ is uniformly random, then this will be a random element in $\mathbb{Z}_q$. On the other hand, if $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}$, then

$$\mathbf{r}_1^T \cdot \mathbf{t} = \mathbf{r}_1^T \cdot \mathbf{A} \cdot \mathbf{s} + \mathbf{r}_1^T \cdot \mathbf{e} = -\mathbf{r}_2^T \cdot \mathbf{s} + \mathbf{r}_1^T \cdot \mathbf{e}. \tag{33}$$

Since $\mathbf{s}, \mathbf{e}$ have small norms, and we assumed that we also found short $\mathbf{r}_1, \mathbf{r}_2$, the above implies that $\mathbf{r}_1^T \cdot \mathbf{t}$ will also have a small norm, and this will allow us to distinguish an LWE instance from a random one, and thus solve the LWE problem.

Finding $\mathbf{r}_1, \mathbf{r}_2$ in (32) is equivalent to finding a short vector in the lattice $\mathcal{L}_q^{\perp}([\ \mathbf{A}^T \mid \mathbf{I}_m\ ])$. We know from (31) that we can find a vector in this lattice of norm $2^{2\sqrt{m \log q \log \delta}}$ (note that because we used $\mathbf{A}^T$, the n in (31) becomes an m), which implies that $\|(\mathbf{r}_1, \mathbf{r}_2)\| \leq 2^{2\sqrt{m \log q \log \delta}}$. If $\mathbf{s}, \mathbf{e}$ have coefficients chosen uniformly at random from $[\beta]$, then the variance of each of each coordinate is

$$\frac{1}{2\beta+1} \sum_{i=-\beta}^{\beta} i^2 = \frac{2}{2\beta+1} \sum_{i=1}^{\beta} i^2 = \frac{2}{2\beta+1} \cdot \frac{\beta \cdot (\beta+1) \cdot (2\beta+1)}{6} = \frac{\beta \cdot (\beta+1)}{3}, \tag{34}$$

and so the standard deviation is $\sqrt{\frac{\beta \cdot (\beta+1)}{3}}$.

If we assume that each coefficient is, instead of uniformly, normally-distributed with standard deviation $\sqrt{\frac{\beta \cdot (\beta+1)}{3}}$ (this can be justified asymptotically by the central limit theorem, but it is already a very good approximation for parameters used in lattice cryptography), then we know that the distribution of $-\mathbf{r}_2^T \cdot \mathbf{s} + \mathbf{r}_1^T \cdot \mathbf{e}$ is a normal distribution with standard deviation

$$\|(\mathbf{r}_1, \mathbf{r}_2)\| \cdot \sqrt{\frac{\beta \cdot (\beta+1)}{3}} \approx 2^{2\sqrt{m \log q \log \delta}} \cdot \sqrt{\frac{\beta \cdot (\beta+1)}{3}} \tag{35}$$

Table 1: Approximate values of δ-hardness of the $\mathsf{LWE}_{m,q,\beta}$ problem for some parameters that resemble those used in the Kyber encryption (ML-KEM) scheme

$\mathsf{LWE}_{m,q,\beta}$ Parameters			
m	β	q	δ
512	2	2^{12}	1.0043
768	2	2^{12}	1.0029
1024	2	2^{12}	1.0022

Table 2: Approximate values of δ-hardness of the $\mathsf{LWE}_{m,q,\beta}$ and $\mathsf{SIS}_{n,q,\beta}$ problems for some parameters that resemble those used in the Dilithium (ML-DSA) signature scheme.

$\mathsf{LWE}_{m,q,\beta}$ Parameters			
m	β	q	δ
1024	2	2^{23}	1.004
1280	4	2^{23}	1.003
1792	2	2^{23}	1.0023

$\mathsf{SIS}_{n,q,\beta}$ Parameters			
n	β	q	δ
1024	2^{18}	2^{23}	1.0039
1536	2^{20}	2^{23}	1.0032
2048	2^{20}	2^{23}	1.0024

It is known (see [MR07]) that if we have a normally-distributed random variable with standard deviation greater than $\sqrt{3} \cdot q$ and we reduce it modulo q, then the result is statistically-close (within $\approx 2^{-80}$) to the uniform distribution. Therefore if (35) is greater than $\sqrt{3} \cdot q$, then the algorithm will not work. In other words, $\mathsf{LWE}_{n,m,q,\beta}$ will be secure (at least against this attack, which seems to be as good as any other known approach) whenever

$$\sqrt{\beta \cdot (\beta + 1)} > 3 \cdot q \cdot 2^{-2\sqrt{m \log q \log \delta}}. \tag{36}$$

3.4 Practical Parameters

As we saw in Section 2.3.1, one can build an encryption scheme based on the LWE problem. In Table 1, we list some sample parameters similar to those that are used in the Kyber (ML-KEM) scheme that will be described in Section 4.6. When building our signature scheme in Section 5, the security of the signature scheme depends both on the hardness SIS problem and LWE. In Table 2, we give sample parameters that are used in this signature scheme.

4 Encryption Over Polynomial Rings

The main inefficiency with the LWE-based encryption scheme in Section 2.3 was that it required a fairly large ciphertext for encrypting one bit – in particular, the ciphertext expansion was linear in the security parameter. This inefficiency was somewhat mitigated by the scheme in Section 2.4 by making the ciphertext expansion only square root in the security parameter at the expense of blowing up the public key by the same factor. In this section, we will show how to get rid of this square root blow-up by considering the LWE problem not over $\mathbb{Z}_q$, but over higher degree polynomial rings.

4.1 Polynomial Rings

The polynomial ring $(\mathbb{Z}[X], +, \times)$, with an indeterminate X, consists of elements of the form $a(X) = \sum_{i=0}^{\infty} a_i X^i$, for $a_i \in \mathbb{Z}$, with the usual polynomial addition and multiplication operations. For convenience, we will often omit the indeterminate X, and simply write a instead of $a(X)$. The degree of a, denoted $\deg(a)$, is the largest i for which $a_i \neq 0$. A polynomial a is *monic* if $a_{deg(a)} = 1$ and it is irreducible if it cannot be written as $a = bc$ where $b, c \in \mathbb{Z}[X]$ and $\deg(b), \deg(c) < \deg(a)$.

The encryption schemes that we saw in Section 2 involved operations over the ring $(\mathbb{Z}, +, \times)$. A generalization of this ring, with which we will be working with for the remainder of the chapter, is the ring $(\mathcal{R}_f, +, \times)$, where $f \in \mathbb{Z}[X]$ is a monic polynomial of degree d.[4] The elements of $\mathcal{R}_f$ are the polynomials $a = \sum_{i=0}^{d-1} a_i X^i$, where $a_i \in \mathbb{Z}$. The sum of two elements in $\mathcal{R}_f$ simply involves summing the corresponding coefficients in $\mathbb{Z}$. That is,

$$a + b = \sum_{i=0}^{d-1} (a_i + b_i) X^i.$$

So the addition of polynomials in $\mathcal{R}_f$ can be seen as addition of vectors over $\mathbb{Z}^d$. Multiplication of a polynomial by an element in $\mathbb{Z}$ therefore also has the same interpretation as multiplying a vector by a constant.

Multiplication of two polynomials in $\mathcal{R}_f$ involves performing a normal polynomial multiplication followed by a reduction modulo f. Reduction modulo f means (just like for integers), the remainder after a division by f is performed. In particular, any polynomial $a \in \mathcal{R}_f$ can be uniquely written as $a = bf + r$ for $b, r \in \mathcal{R}_f$ where $\deg(r) < d$. And so $a \bmod f = r$.

To see that any a can indeed be decomposed in this fashion, we use induction. If $\deg(a) < d$, then $a = r$ (and $b = 0$) and we're done. Now suppose that all a of degree $k-1$ can be written as $a = bf + r$ and let a' have degree k. Then $a' - a'_k f X^{k-d}$ has degree $k-1$ and by the inductive hypothesis can be written as $bf + r$ for some b and r. We can therefore write $a' = \left(a' X^{k-d} + b\right) f + r$.

To prove that the above decomposition of a is unique, assume for contradiction that there are distinct $(b, r) \neq (b', r')$ such that $bf + r = b'f + r'$. This implies that $(b - b')f + (r - r') = 0$. Since $\deg(r - r') < d$, it must be that $r = r'$. Then we know that if $(b - b') \neq 0$, then $\deg((b - b')f) = deg(b - b') + deg(f) \neq 0$. And so $b = b'$ and we have a contradiction.

Note that the above proof of existence of b and r gives rise to a very simple algorithm for computing $a \bmod f$ – multiply f by an appropriate monomial αX^i and subtract it away from a to create a polynomial of a smaller degree and continue until getting a polynomial

[4] In the lattice literature, this ring is often written as $\mathbb{Z}[X]/(f(X))$.

of degree less than d. For example if we would like to reduce $2X^3+8X^2+5X+1$ modulo $f=X^2-2X+1$, we write

$$\begin{aligned}2X^3+8X^2+5X+1 &\equiv 2(2X^2-X)+8X^2+5X+1 \equiv 12X^2+3X+1 \pmod f\\ &\equiv 12(2X-1)+3X+1 \equiv 27X-11 \pmod f.\end{aligned}$$

As a simple observation, note that the usual ring $(\mathbb{Z},+,\times)$ is a special instantiation of the ring $(\mathcal{R}_f,+,\times)$ in which the polynomial f is defined to be $f=X$ (in fact $f=X-\alpha$, for any $\alpha\in\mathbb{Z}$ is also fine).

4.1.1 Polynomials and Linear Algebra

A useful observation is that polynomial multiplication modulo f can be written as a multiplication of a matrix in $\mathbb{Z}^{d\times d}$ with a vector in $\mathbb{Z}^d$. Observe that the product $ab \bmod f$ can be written as:

$$ab \bmod f = a\cdot\left(\sum_{i=0}^{d-1} b_i X^i\right) \bmod f = \sum_{i=0}^{d-1}(aX^i \bmod f)b_i. \tag{37}$$

Since each $aX^i \bmod f$ is a polynomial of degree less than d, it can be thought of as a vector in $\mathbb{Z}^d$. The multiplication ab can therefore be seen as a linear combination (with weights b_i) of these d vectors, and thus can be represented as a matrix-vector multiplication. For example, the product

$$(2X^2-1)(X^2-X+2) \bmod X^3-X+1 = 5X^2-3X$$

can be written as

$$\begin{bmatrix}-1 & -2 & 0\\ 0 & 1 & -2\\ 2 & 0 & 1\end{bmatrix}\cdot\begin{bmatrix}2\\-1\\1\end{bmatrix}=\begin{bmatrix}0\\-3\\5\end{bmatrix}. \tag{38}$$

When treating polynomials $a=\sum_{i=0}^{d-1} a_iX^i \in \mathcal{R}_f$ as vectors and matrices, it will be convenient to use the notation $\mathcal{V}_a\in\mathbb{Z}^d$ and $\mathcal{M}_a\in\mathbb{Z}^{d\times d}$, where

$$\mathcal{V}_a=\begin{bmatrix}a_0\\a_1\\\dots\\a_{d-1}\end{bmatrix}\in\mathbb{Z}^d \text{ , and } \mathcal{M}_a=\begin{bmatrix}\mathcal{V}_a & \mathcal{V}_{aX \bmod f} & \dots & \mathcal{V}_{aX^{d-1} \bmod f}\end{bmatrix}\in\mathbb{Z}^{d\times d} \tag{39}$$

We did not include f as part of the notation for $\mathcal{V}_a$ and $\mathcal{M}_a$, but the f should always be evident from context.

Using this notation, we can rewrite (38) as

$$\mathcal{M}_{2X^2-1}\cdot\mathcal{V}_{X^2-X+2}=\mathcal{V}_{5X^2-3X}.$$

We can also extend the above notation to matrices of polynomials. For a vector $\mathbf{a}=\begin{bmatrix}a_1\\\dots\\a_n\end{bmatrix}\in\mathcal{R}_f^n$ and matrix $\mathbf{A}=\begin{bmatrix}a_{1,1} & \dots & a_{1,m}\\ \dots & \dots & \dots\\ a_{n,1} & \dots & a_{n,m}\end{bmatrix}\in\mathcal{R}_f^{n\times m}$, we define $\mathcal{V}_{\mathbf{a}}$ and $\mathcal{M}_{\mathbf{A}}$ as

$$\mathcal{V}_{\mathbf{a}}=\begin{bmatrix}\mathcal{V}_{a_1}\\\dots\\\mathcal{V}_{a_n}\end{bmatrix}\in\mathbb{Z}^{dn} \text{ , and } \mathcal{M}_{\mathbf{A}}=\begin{bmatrix}\mathcal{M}_{a_{1,1}} & \dots & \mathcal{M}_{a_{1,m}}\\ \dots & \dots & \dots\\ \mathcal{M}_{a_{n,1}} & \dots & \mathcal{M}_{a_{n,m}}\end{bmatrix}\in\mathbb{Z}^{dn\times dm}. \tag{40}$$

From the above definitions, one can check that for any $\mathbf{A}\in\mathcal{R}_f^{n\times m}$ and $\mathbf{b}\in\mathcal{R}_f^m$, we have

$$\mathcal{M}_{\mathbf{A}}\cdot\mathcal{V}_{\mathbf{b}}=\mathcal{V}_{\mathbf{Ab}}\in\mathbb{Z}^{dn} \tag{41}$$

4.1.2 Coefficient Growth

When a and b are integers, the magnitude of their product is trivial to compute – it's just the absolute value of ab. For $a, b \in \mathcal{R}_f$, bounding the magnitude of the product is a bit more involved and is heavily dependent on the polynomial f. Because multiplication $ab \in \mathcal{R}_f$ can be written as $\mathcal{M}_a\mathcal{V}_b$, a simple bound on the maximum coefficient of the product is $d\|\mathcal{M}_a\|_\infty \cdot \|\mathcal{V}_b\|_\infty$, where $\|\cdot\|_\infty$ is the absolute value of the largest coefficient. One could obtain better bounds (in the ℓ_2 norm) by computing the maximum singular value of $\mathcal{M}_a$. But either way, the effect of f on the size of the coefficients in $\mathcal{M}_a$ is crucial to bounding the product.

The best we could hope for, in terms of keeping $\mathcal{M}_a$ small, is that $\|\mathcal{M}_a\|_\infty = \|\mathcal{V}_a\|_\infty$. The only two polynomials f for which this holds are $X^d \pm 1$. For polynomials of the form $X^d \pm X^{d/2} + 1$ and $\sum_{i=0}^{d} X^i$, we have $\|\mathcal{M}_a\|_\infty \leq 2\|\mathcal{V}_a\|_\infty$. Here are some example matrices $\mathcal{M}_a$ for a polynomial $a = a_0 + a_1X + a_2X^2 + a_3X^3$ for several f's of degree 4.

$$f = X^4 - 1 \longrightarrow \mathcal{M}_a = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \tag{42}$$

$$f = X^4 + 1 \longrightarrow \mathcal{M}_a = \begin{bmatrix} a_0 & -a_3 & -a_2 & -a_1 \\ a_1 & a_0 & -a_3 & -a_2 \\ a_2 & a_1 & a_0 & -a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \tag{43}$$

$$f = X^4 - X^2 + 1 \longrightarrow \mathcal{M}_a = \begin{bmatrix} a_0 & -a_3 & -a_2 & -a_1 - a_3 \\ a_1 & a_0 & -a_3 & -a_2 \\ a_2 & a_1 + a_3 & a_0 + a_2 & -a_3 + a_1 \\ a_3 & a_2 & a_1 + a_3 & a_0 + a_2 \end{bmatrix} \tag{44}$$

$$f = X^4 + X^3 + X^2 + X + 1 \longrightarrow \mathcal{M}_a = \begin{bmatrix} a_0 & -a_3 & -a_2 + a_3 & -a_1 + a_2 \\ a_1 & a_0 - a_3 & -a_2 & -a_1 + a_3 \\ a_2 & a_1 - a_3 & a_0 - a_2 & -a_1 \\ a_3 & a_2 - a_3 & a_1 - a_2 & a_0 - a_1 \end{bmatrix} \tag{45}$$

There are also polynomials f for which $\|\mathcal{M}_a\|_\infty \gg \|\mathcal{V}_a\|_\infty$. Unsurprisingly, if f itself has large coefficients, then $\|\mathcal{M}_a\|$ will as well. But there are also f with small coefficients that produce $\mathcal{M}_a$'s with exponentially larger coefficients than a – one such example is $f = X^d + 2X^{d-1} + 1$. Polynomials f that result in matrices $\mathcal{M}_a$ having much larger coefficients than a are not useful for cryptographic purposes. In general, we prefer to use polynomials that result in the ratio between $\|\mathcal{M}_a\|_\infty$ and $\|\mathcal{V}_a\|_\infty$ to be 1 or 2.

4.2 The Generalized-LWE and SIS Problems

Let us extend the notation from Section 2 and for a polynomial $a = \sum_{i=0}^{d} a_i X^i \in \mathcal{R}_f$, we write $a \leftarrow [\beta]$ to mean that all integer coefficients a_i are chosen uniformly from $[\beta]$. Similarly, for a vector $\mathbf{a} \in \mathcal{R}_f^m$, we write $\mathbf{a} \leftarrow [\beta]^m$ to be the distribution in which every coefficient of every polynomial in $\mathbf{a}$ is chosen uniformly from $[\beta]$.

We now present a version of the LWE problem that is defined over general rings $\mathcal{R}_f$, rather than just over $\mathbb{Z}$ as in Definition 1. Analogously, we will be working over the ring $\mathcal{R}_{q,f}$, which is like the ring $\mathcal{R}_f$ except that the polynomial coefficients are in $\mathbb{Z}_q$ rather than in $\mathbb{Z}$. The ring $\mathcal{R}_{q,f}$ is often written in the lattice literature as $\mathbb{Z}_q[X]/(f(X))$.

Definition 5. For positive integers $m, n, q, \beta < q$, and ring $\mathcal{R}_{q,f}$, the $\mathcal{R}_{q,f}$-$\mathsf{LWE}_{n,m,\beta}$ problem asks to distinguish between the following two distributions:

1. $(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e})$, where $\mathbf{A} \leftarrow \mathcal{R}_{q,f}^{n \times m}, \mathbf{s} \leftarrow [\beta]^m, \mathbf{e} \leftarrow [\beta]^n$

2. $(\mathbf{A}, \mathbf{u})$, where $\mathbf{A} \leftarrow \mathcal{R}_{q,f}^{n \times m}$ and $\mathbf{u} \leftarrow \mathcal{R}_{q,f}^{n}$.

As before, the parameter n does not have any known effect on the hardness of the problem, unless it is large, and so we will usually just write $\mathcal{R}_{q,f}$-$\mathsf{LWE}_{m,\beta}$. The preceding definition of the generalized LWE problem and the following cryptosystem follows the line of work (which related its security to worst-case instances of certain lattice problems) of [LPR10, BV11, LPR13b, LS15].

We can analogously generalize the SIS problem from Definition 4 as follows.

Definition 6. For positive integers m, n, q, and $\beta < q$, and ring $\mathcal{R}_{q,f}$, the $\mathcal{R}_{q,f}$-$\mathsf{SIS}_{n,m,\beta}$ problem asks to find, for a randomly-chosen matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$, vectors $\mathbf{s}_1 \in [\beta]^m$ and $\mathbf{s}_2 \in [\beta]^n$ such that $\mathbf{A}\mathbf{s}_1 + \mathbf{s}_2 = \mathbf{0} \pmod q$.

4.3 Generalized-LWE Encryption

The description of the encryption scheme is virtually identical to that in Section 2.3.1 with the ring $\mathbb{Z}$ being replaced with $\mathcal{R}_f$. The main advantage of the scheme will be that the message μ, being in $\mathcal{R}_f$, allows us to pack d bits into it.

$$\mathsf{sk} : \mathbf{s} \leftarrow [\beta]^m, \ \mathsf{pk} : (\mathbf{A} \leftarrow \mathcal{R}_{q,f}^{m \times m}, \mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}_1), \text{ where } \mathbf{e}_1 \leftarrow [\beta]^m. \tag{46}$$

To encrypt a message $\mu \in \mathcal{R}_f$ whose coefficients are in $\{0, 1\}$, the encryptor samples $\mathbf{r}, \mathbf{e}_2 \leftarrow [\beta]^m$ and $e_3 \leftarrow [\beta]$, and outputs

$$\left(\mathbf{u}^T = \mathbf{r}^T\mathbf{A} + \mathbf{e}_2^T, v = \mathbf{r}^T\mathbf{t} + e_3 + \frac{q}{2}\mu\right). \tag{47}$$

The security argument based on $\mathcal{R}_{q,f}$-$\mathsf{LWE}_{m,\beta}$ is identical to the proof based on $\mathsf{LWE}_{m,q,\beta}$ in Section 2.3.1.

To decrypt, one computes

$$v - \mathbf{u}^T\mathbf{s} = \mathbf{r}^T(\mathbf{A}\mathbf{s} + \mathbf{e}_1) + e_3 + \frac{q}{2}\mu - \left(\mathbf{r}^T\mathbf{A} + \mathbf{e}_2^T\right)\mathbf{s} \tag{48}$$

$$= \mathbf{r}^T\mathbf{e}_1 + e_3 + \frac{q}{2}\mu - \mathbf{e}_2^T\mathbf{s} \tag{49}$$

To compute the decryption error, we can rewrite the above equation (excluding $\frac{q}{2}\mu$) as

$$\mathcal{M}_{\mathbf{r}^T}\mathcal{V}_{\mathbf{e}_1} + \mathcal{V}_{e_3} - \mathcal{M}_{\mathbf{e}_2^T}\mathcal{V}_{\mathbf{s}} \tag{50}$$

and then apply the techniques in Section 2.3.2 to compute the probability that none of the d coefficients has magnitude greater than $q/4$. When $f = X^d \pm 1$, then for each of the d coefficients, computing this probability is the same as when we worked over the integers because the coefficients in every row of $\mathcal{M}_{\mathbf{r}^T}$ and $\mathcal{M}_{\mathbf{e}_2^T}$ are independent (see (42) and (43)). One then applies the union bound to bound the probability that all d decryption errors are small. For rings over other polynomials, one can still apply the techniques in Section 2.3.2 if one can rewrite the matrix-vector multiplication as a sum of independent random variables.

4.3.1 Optimizations and Efficiency

The main advantage of the Generalized-LWE scheme is that one does not need to increase the size of the public key, as in Section 2.4, in order to be able to encrypt a larger message. When f has degree d, the ring naturally supports the encryption of d bits. So as long as we set $d \geq 256$, which is the length of an AES (or any symmetric cipher) key that one

would encrypt using public key encryption, we are able to use an optimally-small public key. Similarly, there is no need for packing more than one message bit per coefficient as in Section 2.5.4. The optimization in Section 2.5.1 is stil very useful and is applied in exactly the same manner as before. The Learning with Rounding problem and cryptosystem (Section 2.5.3), as well as the non-interactive key exchange (Section 2.6), are also defined analogously for the ring $\mathcal{R}_f$.

4.3.2 Security and Connection to Integer Lattices

The connection between polynomial operations in $\mathcal{R}_f$ and linear algebra over $\mathbb{Z}$ in (41) allows us to make useful connections between the lattices we saw in Section 3.1 and (short) solutions to polynomial equations involved in the $\mathcal{R}_{q,f}$-$\mathsf{LWE}_{n,m,\beta}$ definition in the previous section. The hardness of the LWE-based encryption scheme relied on the hardness of finding, for a random $\mathbf{A} \in \mathbb{Z}_q^{m\times m}$ and a $\mathbf{t} \in \mathbb{Z}_q^m$, integer vectors $\mathbf{s}, \mathbf{e} \in \mathbb{Z}_q^m$ with small coefficients satisfying $\mathbf{As} + \mathbf{e} = \mathbf{t}$. We saw that this was equivalent to finding a particular vector with small coefficients in the lattice $\mathcal{L}_q^\perp([\mathbf{A} \mid \mathbf{t} \mid \mathbf{I}_m])$. And we based the concrete security of the LWE scheme on the hardness of this latter problem.

The security of the more efficient encryption scheme in this section is analogously based on the hardness of finding, for a random $\mathbf{A} \in \mathcal{R}_{q,f}^{m\times m}$ and a $\mathbf{t} \in \mathcal{R}_{q,f}^m$, polynomial vectors $\mathbf{s}, \mathbf{e} \in \mathcal{R}_f^m$ with small coefficients satisfying $\mathbf{As} + \mathbf{e} = \mathbf{t}$. By (41), this is equivalent to finding integer vectors $\mathcal{V}_\mathbf{s}, \mathcal{V}_\mathbf{e} \in \mathbb{Z}^{dm}$ with small coefficients satisfying $\mathcal{M}_\mathbf{A}\mathcal{V}_\mathbf{s} + \mathcal{V}_\mathbf{e} = \mathcal{V}_\mathbf{t}$. Since this equation is over $\mathbb{Z}$, we can transform it into a problem about finding short vectors in integer lattices – that is, finding a particular vector with short coefficients in the lattice $\mathcal{L}_q^\perp([\mathcal{M}_\mathbf{A} \mid \mathcal{V}_\mathbf{t} \mid \mathbf{I}_{dm}])$.

Because the number of columns in $\mathcal{M}_\mathbf{A}$ is dm, the dimension of the lattice

$$\mathcal{L}_q^\perp([\mathcal{M}_\mathbf{A} \mid \mathcal{V}_\mathbf{t} \mid \mathbf{I}_{dm}])$$

is $d(m+1)+1$. If the algebraic structure of $\mathcal{R}_f$ does not exhibit any weaknesses (see Section 4.5.1 below), then the hardness of finding a short vector in this lattice is the same as in the $\mathsf{LWE}_{n',m',q,\beta}$ lattice in (**??**) with $n' = dn$ and $m' = dm$. For the $\mathcal{R}_{q,f}$-$\mathsf{LWE}_{n,m,\beta}$ problem, therefore, the important value is dm – the product of the degree of f in $\mathcal{R}_f$ and the number of columns of $\mathbf{A} \in \mathcal{R}_f^{n\times m}$. Similarly, for the $\mathcal{R}_{q,f}$-$\mathsf{SIS}_{n,m,\beta}$ problem, the important value is dn – the product of the degree of f and the number of rows in $\mathbf{A}$.

4.4 NTRU

The NTRU cryptosystem [HPS98] was the first truly efficient lattice-based encryption scheme, and was also the first to propose using polynomial rings – specifically $\mathcal{R}_{q,X^d-1}$ – in lattice-based cryptography. The scheme was originally proposed as a trapdoor one-way function, which can be seen as a OW-CPA cryptosystem.[5] There is also a simple modification that allows us to construct a CPA-secure encryption scheme [SS11]. In most use cases, however, the trapdoor one-way function is sufficient since there is a black-box transformation from such primitives to CCA-secure encryption schemes (c.f. [Den02]).

4.4.1 The NTRU Trapdoor 1-way Function

In the previous sections, we worked with the decision version of the Generalized-LWE problem, but it is also very natural to define a *search* version of it. This problem could be stated as finding the e when being given $(a, as + e)$ for $a \leftarrow \mathcal{R}_{q,f}$ and $s, e \leftarrow [\beta]$. Notice that if a is invertible in $\mathcal{R}_{q,f}$, then finding e immediately also implies finding s.

[5] An encryption scheme is OW-CPA secure if an attacker, in possession of the public key, cannot recover the message from a ciphertext of a randomly-chosen message.

The NTRU problem is very similar to the above, except that the polynomial a is not chosen at random from $\mathcal{R}_{q,f}$, but is rather the product of the integer $p = (2\beta + 1)$ and polynomials g_1 and g_2^{-1} where $g_i \leftarrow [\beta]$ (conditioned on g_2 being invertible) and p being relatively prime to q (e.g. $\beta = 1$ is a popular choice). The hardness behind NTRU relies on the assumption that the search $\mathcal{R}_{q,f}$-$\mathsf{LWE}_{n,m,\beta}$ problem (with $n = m = 1$) is still hard whenever a is not uniformly random, but is instead the product $pg_1g_2^{-1}$.

The NTRU problem is formally defined as follows:

Definition 7. Let $p = 2\beta + 1$. Given $(a, as + e)$, where $a = pg_1g_2^{-1}$ for $g_1, g_2, s, e \leftarrow [\beta]$ and g_2 being invertible in $\mathcal{R}_{q,f}$ and $\mathcal{R}_{p,f}$, find e.

Based on the presumed hardness of the above problem, we can construct a trapdoor one-way function family as follows: to generate a random element from the family, we choose secret invertible polynomials $g_1, g_2 \leftarrow [\beta]$ with g_2 being invertible in $\mathcal{R}_{q,f}$ and $\mathcal{R}_{p,f}$, and output the public key to be

$$a = pg_1g_2^{-1}. \tag{51}$$

The secret key is g_2.

The one-way function mapping $s, e \leftarrow [\beta]$ to $\mathcal{R}_{q,f}$ computes

$$b = as + e \in \mathcal{R}_{q,f}. \tag{52}$$

Observe that in order to recover e, s, it is enough to recover these modulo p since there is a 1-1 correspondence between elements in $[\beta]$ and residues modulo $p = 2\beta + 1$. To recover s, e modulo p using the secret key, one first computes

$$g_2b \bmod p = pg_1s + g_2e \bmod p = g_2e \bmod p. \tag{53}$$

The first equality (in the ring $\mathcal{R}_{q,f}$) simply follows from the definition of a and b. The second equality only holds true if the preceding equation holds true not only in $\mathcal{R}_{q,f}$, but also over $\mathcal{R}_f$. Here is where we again use the fact that the coefficients of g_i, s, and e are small with respect to q, and can bound them to be less than q (either always or with very high probability). If this is the case, then we have

$$(g_2b \bmod p)g_2^{-1} \bmod p = e, \tag{54}$$

where we multiplied by the inverse of g_2 in $\mathcal{R}_{p,f}$. Once we have $e \pmod p$, which is the same as $e \in [\beta]$, we can also compute

$$(b - e)a^{-1} = s. \tag{55}$$

4.4.2 Security

If we don't presume any special structure of the polynomial a, then recovering s, e in (52) is identical to the attack on the public key of the Generalized-LWE instance in which we try to find a short vector in the lattice

$$\mathcal{L}_q^{\perp}([\mathcal{M}_a \mid \mathcal{V}_b \mid \mathbf{I}_d]). \tag{56}$$

We can also try to recover the secret key g_1, g_2 (or some other short polynomials related to it) from $a = pg_1g_2^{-1}$ by looking for a short vector in the lattice

$$\mathcal{L}_q^{\perp}([\mathcal{M}_{p^{-1}a} \mid \mathbf{I}_d]). \tag{57}$$

The lattices in (56) and (57) look very similar, with the only relevant difference being the presence of one extra vector $\mathcal{V}_b$ in (56). It was therefore somewhat surprising that in certain scenarios, where q is significantly larger than β (but not so large as to be in the

space where generic lattice reduction algorithms clearly work), it was significantly easier to find short vectors in the lattice in (57) than in (56) [ABD16, CJL16, KF17]. While these attacks don't translate to attacks against NTRU paramters, they do prevent the NTRU assumption from being used in advanced primitives (e.g. FHE) that require large moduli and small noise. For this reason, schemes based on Generalized-LWE (whose security relies on essentially (56)) are used in such scenarios.

4.5 Exploiting the Algebraic Structure ...

4.5.1 ... for Attacks

Suppose that we are working over the ring $\mathcal{R}_{q,f}$ with $f = X^d - 1$. Because $X - 1$ is a factor of $X^d - 1$, there is a ring homomorphism from $\mathcal{R}_{q,f}$ to $\mathcal{R}_{q,X-1}$ which maps elements $a = \sum_{i=0}^{d-1} a_i X^i \in \mathcal{R}_{q,f}$ to $a' = \sum_{i=0}^{d-1} a_i \in \mathcal{R}_{q,X-1}$. Because the ring $\mathcal{R}_{q,X-1}$ is exactly the ring $\mathbb{Z}_q$ with the usual addition and multiplication modulo q, we actually have a ring homomorphism from $\mathcal{R}_{q,f}$ to $\mathbb{Z}_q$. What's particularly special about this homomorphism is that if the coefficients of a are small, then the image of a under the homomorphism is small as well (i.e. can only be a factor of d larger). This implies the following simple attack on the $\mathcal{R}_{q,f}$-$\mathsf{LWE}_{m,\beta}$ problem where we're given $\mathbf{A} \in \mathcal{R}_{q,f}^{n \times m}, \mathbf{t} \in \mathcal{R}_{q,f}^n$ and are asked to decide whether there exist $\mathbf{s}, \mathbf{e}$ with coefficients in $[\beta]$ satisfying $\mathbf{As} + \mathbf{e} = \mathbf{t}$ (the analogous attack on the $\mathcal{R}_{q,f}$-$\mathsf{SIS}_{n,m,\beta}$ problem was given in [PR06, LM06]):

Let $\mathbf{A}' \in \mathbb{Z}_q^{n \times m}, \mathbf{t}' \in \mathbb{Z}_q^n$ be images of $\mathbf{A}, \mathbf{t}$ under the homomorphism. If there really existed $\mathbf{s} \in [\beta]^m \subset \mathcal{R}_f^m$ and $\mathbf{e} \in [\beta]^n \subset R^n$ satisfying $\mathbf{As} + \mathbf{e} = \mathbf{t}$, then there exist $\mathbf{s}' \in [d\beta]^m \subset \mathbb{Z}^m, \mathbf{e}' \in [d\beta]^n \subset \mathbb{Z}^n$ satisfying $\mathbf{A}'\mathbf{e}' + \mathbf{s}' = \mathbf{t}'$. Because m and n are fairly small, one can efficiently find such a vector by looking for the shortest vector in the lattice set up as in (**??**). If we find such $\mathbf{s}', \mathbf{e}'$, then we claim that $\mathbf{s}, \mathbf{e}$ with coefficients in $[\beta]$ exist. If we don't find such $\mathbf{s}', \mathbf{e}'$, then we say that $\mathbf{A}, \mathbf{t}$ were random. If $\mathbf{A}, \mathbf{t}$ were uniformly random, then so are $\mathbf{A}', \mathbf{t}'$, and one can use Lemma 2 to upper-bound the probability that $\mathbf{s}', \mathbf{e}'$ with small coefficients can exist. If this upper bound is $1 - \epsilon$, then the above distinguisher has advantage at least ϵ.

The most crucial element enabling the above attack on $\mathcal{R}_{q,f}$-$\mathsf{LWE}_{m,\beta}$ was that there existed a homomorphism into a ring of a smaller degree which *did not increase the coefficient size* by much. If f had a factor of, for example, $X - 2$, then the above attack would not work because the homomorphism would map an element $a = \sum_{i=0}^{d-1} a_i X^i \in \mathcal{R}_{q,f}$ to $a' = \sum_{i=0}^{d-1} a_i 2^i \in \mathbb{Z}_q$, and so a' would be in a range that is exponentially dependent on d. Interestingly, the worst-case to average-case reductions showing that solving $\mathcal{R}_{q,f}$-$\mathsf{LWE}_{m,\beta}$ and $\mathcal{R}_{q,f}$-$\mathsf{SIS}_{n,m,\beta}$ implies finding short vectors in ideal / module lattices [PR06, LM06, LPR13a, LS15, PRS17] only require that f be irreducible (or at least have a large irreducible part in the case of $\mathcal{R}_{q,f}$-$\mathsf{SIS}_{n,m,\beta}$) over the ring $\mathbb{Z}[X]$ and not $\mathbb{Z}_q[X]$. As we will see in the next section, it is actually quite advantageous, in terms of implementation efficiency, to work over a ring $\mathcal{R}_{q,f}$ where the polynomial f has many low-degree factors in $\mathbb{Z}_q[X]$.

4.5.2 ... for Efficiency (The Number Theoretic Transform)

The most computationally-involved algebraic operation in the $\mathcal{R}_{q,f}$-$\mathsf{LWE}_{n,m,\beta}$ encryption scheme is multiplication of polynomials in $\mathcal{R}_{q,f}$. The basic "school-book" polynomial multiplication of two polynomials of degree d requires $O(d^2)$ operations. There are better methods like Karatsuba and Toom-Cook which take approximately $O(d^{1.5})$ time. The most efficient way to perform polynomial multiplication in $\mathcal{R}_{q,f}$, which requires as few as

$O(d \log d)$ operations over $\mathbb{Z}_q$, is via the NTT (Number Theoretic Transform), which is a special case of the FFT performed over the field $\mathbb{Z}_q^*$ rather than over the complex numbers.

We will now explain the NTT algorithm over the polynomial ring $\mathbb{Z}_q[X]/(X^d + \alpha)$ for $\alpha \in \mathbb{Z}$ and d is a power of 2. Suppose that $-\alpha$ has a square root r in $\mathbb{Z}_q$, and we can therefore write $X^d + \alpha \equiv (X^{d/2} - r)(X^{d/2} + r) \pmod q$. Then computing $ab \in \mathbb{Z}_q[X]/(X^d + \alpha)$ can be done via the Chinese Remainder Theorem. That is, we can first compute

$$(a \bmod X^{d/2} - r, a \bmod X^{d/2} + r), \tag{58}$$

$$(b \bmod X^{d/2} - r, b \bmod X^{d/2} + r), \tag{59}$$

then component-wise multiply to obtain

$$(ab \bmod X^{d/2} - r, ab \bmod X^{d/2} + r), \tag{60}$$

and then the above can be used to reconstruct $ab \bmod X^d + \alpha$.

In Lemma 5, we show that the decomposition (58), (59) requires d additions and $d/2$ multiplications over $\mathbb{Z}_q$, and the reconstruction of $ab \bmod X^d + 1$ from (60) requires the same amount of operations. Thus computing the product ab would require $2d$ additions, d multiplications, and two multiplications over the ring $\mathbb{Z}_q[X]/(X^{d/2} \pm r)$. Since multiplications over the latter ring are still $O(n^2)$ time, it's not clear that we made progress – but if the polynomial $X^{d/2} - r$ can be further factored as $(X^{d/4} - s)(X^{d/4} + s)$ (and similarly $X^{d/2} + r = (X^{d/2} - t)(X^{d/4} + t)$), then we can compute (60) recursively!

In particular, we can now compute the runtime of the entire algorithm using a recurrence relation, where $T(d)$ is the time to multiply two polynomials in $\mathbb{Z}_q[X]/(X^d + \alpha)$, and A and M is the time required to compute one integer addition and multiplication in $\mathbb{Z}_q$, respectively. From the above discussion, this recurrence relation is

$$T(d) = 2 \cdot T(d/2) + 2d \cdot A + d \cdot M. \tag{61}$$

If d is a power of 2 and -1 and $-\alpha$ have d^{th} roots in $\mathbb{Z}_q$, then we can continue the recursion until $X^d + \alpha$ splits into linear factors and thus the above recurrence is well-defined for $d, d/2, d/4, \ldots, 2$.[6] The solution to this recurrence is thus

$$T(d) = d \cdot T(1) + 2d \log d \cdot A + d \log d \cdot M, \tag{62}$$

where $T(1) = M$. So computing a product in the ring $\mathbb{Z}_q[X]/(X^d + \alpha)$ could require as few as $2d \log d$ integer and $d(\log d + 1)$ multiplications over $\mathbb{Z}_q$.

When $\alpha = 1$, the above gives us an efficient multiplication algorithm in the ring $\mathcal{R}_{q,X^d+1} = \mathbb{Z}_q[X]/(X^d + 1)$. In order to be able to split the polynomial $X^d + 1$ all the way down to linear terms, one needs that -1 has a d^{th} root, or in other words, that the multiplicative group $\mathbb{Z}_q^*$ has a $2d^{th}$ root of unity.[7] Such a root of unity exists whenever the modulus q is a prime satisfying $q \equiv 1 \pmod{2d}$ (see Lemma 7). It should be noted that even if q does not satisfy the latter, the NTT algorithm can still be performed; it's just that we will not be able to recurse all the way down to linear polynomials. For example, if $q \equiv 1 \pmod d$, then the polynomial $X^d + 1$ can be split into a product of quadratic

[6] To see why, let $r, s \in \mathbb{Z}_q^*$ be such that $r^d = -\alpha$ and $s^d = -1$. We will prove the claim by induction. At every level of the recursion, the invariant is that all factors will be of the form $X^k \pm s^l r^k$ where $2 \le k \le d$ is a power of 2 and $k|l$. This invariant is satisfied by the base case: $X^d + \alpha = X^d + s^d r^d$. We now move on to the inductive step: because $k|l$, if $k/2$ is an integer, then so is $l/2$, and so the terms of the form $X^k - s^l r^k$ factor into $(X^{k/2} + s^{l/2} r^{k/2})(X^{k/2} - s^{l/2} r^{k/2})$, while those of the form $X^k + s^l r^k$ factor as $(X^{k/2} + \sqrt{-1} s^{l/2} r^{k/2})(X^{k/2} - \sqrt{-1} s^{l/2} r^{k/2})$. Replacing $\sqrt{-1} = s^{d/2}$, the preceding two factors are of the form $(X^{k/2} \pm s^{l/2+d/2} r^{k/2})$. Since $k|l$ and $k|d$, we have $\frac{k}{2} | \frac{l}{2} + \frac{d}{2}$, and the invariant at the next level is satisfied. The recursion stops when $k = 1$.

[7] Recall that an element r is a k^{th} root of unity if $r^k = 1$ and $r^j \neq 1$ for all $0 < j < k$. If r is a k^{th} root of unity, then it must be that $r^{k/2} = -1$.

terms. This means that at the bottom level the component-wise multiplication will be in rings $\mathbb{Z}_q[X]/(X^2 - r_i)$ for some $r_i \in \mathbb{Z}_q^*$. This base multiplication, while requiring more than just 1 multiplication over $\mathbb{Z}_q$, can still be performed with a small constant number of multiplications and additions (i.e. 5 multiplications and 2 additions over $\mathbb{Z}_q$). And because we do one decomposition level less, the total number of operations is virtually identical.[8]

One also doesn't necessarily need the polynomial f to be X^d+1 to benefit from the NTT. There are other (cyclotomic) polynomials that have a factorization tree very similar to X^d+1. For example, for certain $d = 2^k \cdot 3$ and primes q, the polynomial $f = X^d - X^{d/2} + 1$ factors into $(X^{d/2} - r_1)(X^{d/2} - r_2)$ modulo q and then one can apply the NTT recursion algorithm to multiply over the rings $\mathbb{Z}_q[X]/(X^{d/2} - r_i)$ – the main difference is that since d is not a power of 2, we will not be able to factor $X^{d/2} \pm r_i$ into degree 1 polynomials, but rather into degree 3 ones. But overall, multiplication can be performed over such rings (c.f. [LS19]) almost as efficiently as over $\mathcal{R}_{q,X^d+1}$. One can also utilize NTT multiplication over rings $\mathcal{R}_{q,f}$ for arbitrary f by multiplying over $\mathbb{Z}_q[X]$ and then reducing modulo f. Multiplication over $\mathbb{Z}_q[X]$ can be performed by doing multiplication over $\mathcal{R}_{q,X^d+1}$ where the degree d is chosen to be high enough so that reduction modulo $X^d + 1$ never occurs (i.e. d is set to an integer larger than $2 \cdot (\deg(f) - 1)$) – this algorithm very often is still the most efficient way to multiply two elements over polynomial rings (c.f. [CHK+21]).

We now prove the lemma that we alluded to above which shows that the number of operations needed in computing (58) and (59), and also in reconstructing the element in the ring from its CRT representation in (60) requires d additions and $d/2$ multiplications.

Lemma 5. *Suppose that a polynomial $g(X) = X^n - r$ can be written as*

$$X^n - r \equiv (X^{n/2} - \sqrt{r})(X^{n/2} + \sqrt{r}) \pmod{q},$$

and define the function ϕ,

$$\begin{aligned}\phi &: \mathbb{Z}_q[X]/(X^n - r) \to \mathbb{Z}_q[X]/(X^{n/2} - \sqrt{r}) \times \mathbb{Z}_q[X]/(X^{n/2} + \sqrt{r}) \\ \phi(a) &= \left(a \bmod X^{n/2} - \sqrt{r}, a \bmod X^{n/2} + \sqrt{r}\right).\end{aligned}$$

If $\sqrt{r} \pmod q$ and $(\sqrt{r})^{-1} \pmod q$ are pre-computed, then ϕ and $2 \cdot \phi^{-1}$ can each be computed using n additions/subtractions and $n/2$ multiplications over $\mathbb{Z}_q$.

Proof. If we write

$$\begin{aligned}a &= \sum_{i=0}^{n-1} a_i X^i, \\ a \mod X^{n/2} - \sqrt{r} &= \sum_{i=0}^{n/2-1} b_i X^i, \\ a \mod X^{n/2} + \sqrt{r} &= \sum_{i=0}^{n/2-1} c_i X^i,\end{aligned}$$

then we can see that for all $0 \le i < n/2$,

$$\begin{aligned}b_i &= a_i + \sqrt{r} \cdot a_{i+n/2}, \\ c_i &= a_i - \sqrt{r} \cdot a_{i+n/2}.\end{aligned}$$

[8] One can easily solve the recurrence in (61) for the case when we are not able to go all the way down to linear factors.

Thus computing ϕ requires $n/2$ multiplications by $\sqrt{r}$ and n additions (or subtractions) to compute $a_i \pm \sqrt{r} \cdot a_{i+n/2}$. For the reverse direction, we can reconstruct $2 \cdot a$ by observing from above that for all $0 \leq i < n/2$,

$$\begin{aligned} 2 \cdot a_i &= b_i + c_i, \\ 2 \cdot a_{i+n/2} &= (\sqrt{r})^{-1} \cdot (b_i - c_i). \end{aligned}$$

The above two operations similarly require $2n$ additions (or subtractions) and $n/2$ multiplications. □

Something a little peculiar to note in the statement of the above lemma is that we're not computing the inverse ϕ^{-1} , but rather $2 \cdot \phi^{-1}$. The reason that we do this is that it saves multiplications. In the recursive algorithm, the procedure in the lemma is run for several (say $\log d$) iterations, and the doubling of the inverse will keep accumulating so that the final result is $2^{log d} = d$ times larger than the correct answer. Then we simply multiply only this last level by d^{-1} . So instead of performing extra multiplications by 2^{-1} at each of the $\log d$ levels, we perform it only at the final one.

4.5.3 Useful Algebraic Properties of the Ring $\mathcal{R}_{q,X^d+1}$

In the previous section, we saw a very efficient algorithm for multiplication over the ring $\mathcal{R}_{q,X^d+1}$, and we also saw that for security, it would be good to have the polynomial X^d+1 be irreducible over $\mathbb{Z}[X]$. In this section, we state and prove some useful properties of the ring $\mathcal{R}_{q,X^d+1}$. The first lemma states that the polynomial $X^d + 1$ is irreducible over the integers exactly when d is a power of 2.

Lemma 6. *The polynomial $X^d + 1$ is irreducible over $\mathbb{Z}[X]$ if and only if d is a power of 2.*

Proof. Let $\Phi_k(X)$ be the k^{th} cyclotomic polynomial, and recall that for any n

$$X^n - 1 = \prod_{k|n} \Phi_k(X).$$

Thus $(X^d + 1)(X^d - 1) = X^{2d} - 1$, and therefore

$$X^d + 1 = \frac{X^{2d} - 1}{X^d - 1} = \frac{\prod_{k|2d} \Phi_k(X)}{\prod_{k|d} \Phi_k(X)} = \prod_{k\,:\,k|2d\,,\,k\nmid d} \Phi_k(X).$$

If $d = 2^\ell$ for some non-negative integer ℓ, then it follows from the above that $X^d + 1 = \Phi_{2d}(X)$ and is thus irreducible (since all cyclotomic polynomials are irreducible). On the other hand, if $d = 2^\ell \cdot d'$ where $d' > 1$ is odd, then both $2d$ and $2d/d' = 2^{\ell+1}$ divide $2d$, but do not divide d. So, $\Phi_{2d}(X)$ and $\Phi_{2d/d'}(X)$ are distinct factors of $X^d + 1$. □

The next lemma is useful for picking the modulus q such that the polynomial $X^d + 1$ factorizes into low-degree polynomials.

Lemma 7. *Let $d \geq k \geq 1$ where $k \mid d$ and $q \equiv 1 \pmod{2k}$ be a prime. Then there exist k distinct $r_i \in \mathbb{Z}_q^*$ satisfying $r_i^k \equiv -1 \pmod{q}$ such that*

$$X^d + 1 \equiv \prod_{i=1}^{k} (X^{d/k} - r_i) \pmod{q}. \tag{63}$$

Public parameters: $k, \eta_1, \eta_2, d_u, d_v \in \mathbb{Z}^+$

CPA-KeyGen	CPA-Encrypt(pk, m)	CPA-Decrypt$(sk, ciphertext)$
$\mathbf{A} \leftarrow \mathcal{R}^{k\times k}_{3329,X^{256}+1}$	$(\mathbf{r}, \mathbf{e}_1, e_2) \leftarrow \psi^k_{\eta_1} \times \psi^k_{\eta_2} \times \psi_{\eta_2}$	$\mathbf{u}' := \lceil \mathbf{u} \rfloor_{2^{d_u} \to q}$
$(\mathbf{s}, \mathbf{e}) \leftarrow \psi^k_{\eta_1} \times \psi^k_{\eta_1}$	$\mathbf{u}^T := \left\lceil \mathbf{r}^T\mathbf{A} + \mathbf{e}_1^T \right\rfloor_{q\to 2^{d_u}}$	$v' := \lceil v \rfloor_{2^{d_v} \to q}$
$\mathbf{t} := \mathbf{A}\mathbf{s} + \mathbf{e}$	$v := \left\lceil \mathbf{r}^T\mathbf{t} + \frac{q-1}{2}m \right\rfloor_{q\to 2^{d_v}}$	$m' := \left\lceil v' - \mathbf{u}'^T\mathbf{s} \right\rfloor_{q\to 2}$
$pk = (\mathbf{A}, \mathbf{t}), sk = \mathbf{s}$	$ciphertext = (\mathbf{u}, v)$	

Figure 3: The CRYSTALS-Kyber CPA-secure Encryption Scheme

Proof. Because the prime q is such that $q \equiv 1 \pmod{2k}$, we have $2k \mid q-1$, and so there exists an element $r \in \mathbb{Z}_q^*$ of order $2k$, and thus $r^k \equiv -1 \pmod q$. Furthermore, for all odd $i \in \{0, \ldots, k-1\}$, all r^{2i+1} are distinct modulo q and satisfy $(r^{2i+1})^k \equiv -1 \pmod q$. The k elements r^{2i+1} are thus the roots of X^k+1 and we therefore have $X^k + 1 \equiv \prod_{i=0}^{k-1}(X - r^{2i+1}) \pmod q$. The lemma follows by substituting X with $X^{d/k}$ in the preceding equality. □

The final lemma is not used anywhere in this manuscript, but we state it here for completeness as it is useful in some advanced applications. It states that the polynomial $X^d + 1$ factorizes for all prime q, so $\mathcal{R}_{q,X^d+1}$ is never a field.

Lemma 8. *Let q be an odd prime and d be a multiple of 4. Then the polynomial $X^d + 1$ factors into at least 2 polynomials over $\mathbb{Z}_q[X]$.*

Proof. If $q \equiv 1 \pmod 4$, then the factors of $X^d + 1$ modulo q are as in Lemma 7 when setting $k = 2$.

If $q \equiv 3 \pmod 4$, then for all $x \in \mathbb{Z}_q^*$ exactly one of x or $-x$ is a quadratic residue modulo q.[9] Set $b = 1$ if 2 is a quadratic residue modulo q, and $b = -1$ if -2 is. Finally let r be such that $r^2 \equiv 2 \cdot b \pmod q$. Then

$$\begin{aligned} X^d + 1 &\equiv X^d + (2b - r^2) \cdot X^{d/2} + 1 \pmod q \\ &\equiv (X^{d/2} + b)^2 - r^2 \cdot X^{d/2} \pmod q \\ &\equiv (X^{d/2} + b + rX^{d/4}) \cdot (X^{d/2} + b - rX^{d/4}) \pmod q, \end{aligned}$$

as claimed. □

4.6 The Encryption Scheme CRYSTALS-Kyber (ML-KEM)

In this section, we will put everything together by giving a full description of the NIST standardized CRYSTALS-Kyber (named ML-KEM by NIST) scheme, which is based on the hardness of the generalized LWE problem. The scheme works over the ring $\mathcal{R}_{3329,X^{256}+1}$ and the distribution of the secrets is drawn from the binomial distribution, rather from the uniform one that we have been working with so far. The main reason for using the binomial distribution (which we will denote as ψ_η for a positive integer η) is simply because it is easier to sample.

[9] To see this, assume that they're both quadratic non-residues or quadratic residues, and therefore their product must be a quadratic residue, and so there is an r such that $-x^2 \equiv r^2 \pmod q$. There is an integer k such that $q = 3 + 4k$, and we raise both sides of the preceding congruence to the power $2k+1$ obtaining $-x^{4k+2} \equiv r^{4k+2} \pmod q$. Since $4k + 2 = q - 1$, Fermat's little theorem implies that x^{4k+2} and r^{4k+2} are congruent to 1 and we have the contradiction $-1 \equiv 1 \pmod q$.

Table 3: Parameters for the three instantiations of Kyber. The security of the three schemes are approximately equivalent to that of AES-128, AES-192, and AES-256, respectively.

	k	η_1	η_2	d_u	d_v	decryption error	pk size	ciphertext size
Kyber-512	3	3	2	10	4	2^{-139}	800 B	768 B
Kyber-768	4	2	2	10	4	2^{-164}	1184 B	1088 B
Kyber-1024	5	2	2	11	5	2^{-174}	1568 B	1568 B

Definition 8. For an integer η, an element from the *binomial distribution* ψ_η is generated as follows: generate random $a_1, \ldots a_\eta, b_1, \ldots, b_\eta \leftarrow \{0,1\}$ and output $\sum a_i - \sum b_i$. This definition is naturally extended to polynomials, where for $a \in \mathcal{R}_f$ we would write $a \leftarrow \psi_\eta$ to denote that every integer coefficient of a is sampled independently according to ψ_η. And similarly, for a vector (of polynomials) of dimension k, we write $\mathbf{a} \leftarrow \psi_\eta^k$ to denote that every element gets sampled according to ψ_k.

At the heart of the CRYSTALS-Kyber CCA-secure KEM is a CPA-secure encryption scheme. The transformation from the latter to a CCA-secure KEM is generic and we briefly describe it in Section 4.7. For the rest of the Section, though, we focus exclusively on the CPA-secure encryption.

The CPA-secure encryption scheme Kyber is presented in Figure 3. The security parameters for Kyber are k, η_1, η_2, where k is the main parameter that is being varied between the security levels. The parameters d_u and d_v specify the (log of) the size of set $\mathcal{S}$ (c.f. Figure 1, (18)) to which the different parts of the ciphertext get rounded to. These values determine the ciphertext size and the decryption error.

The key generation procedure is exactly as in (46) except the secret vectors $\mathbf{s}, \mathbf{e} \in \mathcal{R}^k_{X^{256}+1}$ are generated according to the binomial distribution instead of uniform. The encryption procedure consists of generating vectors $\mathbf{r}, \mathbf{e}_1 \in \mathcal{R}^k_{X^{256}+1}$ and a ring element $e_2 \in \mathcal{R}_{X^{256}+1}$ from the binomial distribution ψ_η (with possibly different values of η – we will explain the intuition for this a bit later) and computing the uncompressed ciphertext as in (47). We then apply the compression function to the ciphertext (see Sections 2.5.1 and 2.5.2) to reduce the ciphertext size. The decryption function is as in (49) with the compression function being used to recover the 0/1 coefficients as in (20).

Correctness and Decryption Error. To compute the decryption error, we look at the term

$$\begin{aligned}
\left\lceil v' - \mathbf{u}'^T\mathbf{s} \right\rfloor_{q\to 2} &= \left\lceil \lceil v \rfloor_{2^{d_v}\to q} - \lceil \mathbf{u}^T \rfloor_{2^{d_u}\to q}\mathbf{s} \right\rfloor_{q\to 2} \\
&= \left\lceil \left\lceil \left\lceil \mathbf{r}^T\mathbf{t} + \frac{q-1}{2}m \right\rfloor_{q\to 2^{d_v}} \right\rfloor_{2^{d_v}\to q} - \left\lceil \left\lceil \mathbf{r}^T\mathbf{A} + \mathbf{e}_1^T \right\rfloor_{q\to 2^{d_u}} \right\rfloor_{2^{d_u}\to q} \mathbf{s} \right\rfloor_{q\to 2} \\
&= \left\lceil \mathbf{r}^T\mathbf{t} + \frac{q-1}{2}m + e' - \left(\mathbf{r}^T\mathbf{A} + \mathbf{e}_1^T + \mathbf{e}''^T\right)\mathbf{s} \right\rfloor_{q\to 2}
\end{aligned}$$

for $e' \in \mathcal{R}_{X^{256}+1}, \mathbf{e}'' \in \mathcal{R}^k_{X^{256}+1}$ whose coefficients correspond to the η in Lemma 1. Replacing $\mathbf{t}$ with $\mathbf{As} + \mathbf{e}$, we obtain

$$\left\lceil v' - \mathbf{u}'^T\mathbf{s} \right\rfloor_{q\to 2} = \left\lceil \mathbf{r}^T\mathbf{e} + \frac{q-1}{2}m + e' - \left(\mathbf{e}_1 + \mathbf{e}''\right)^T\mathbf{s} \right\rfloor_{q\to 2} \tag{64}$$

The result of the above will be m if all the coefficients of

$$\mathbf{r}^T\mathbf{e} + e' - \left(\mathbf{e}_1 + \mathbf{e}''\right)^T\mathbf{s} \tag{65}$$

are less than $q/4$ in magnitude. Computing this probability is done as in Section 2.3.2. We already discussed in Section 4.3 (see (50)) how one would adapt those techniques to the ring setting. The only difference between (50) and (65) is that the latter also includes the terms e' and $\mathbf{e}''$ which resulted from the compression and decompression operations. By assuming that the terms being compressed are uniformly-random, one can easily compute the exact probability distribution of each coefficient of e' and $\mathbf{e}''$ – i.e. the probability distribution of $\left\lceil \lceil x \rfloor_{q \to 2^{d_v}} \right\rfloor_{2^{d_v} \to q}$ for random $x \leftarrow \mathbb{Z}_q$ (resp. with d_u instead of d_v). Once we compute the exact probability of error for one coefficient, we can apply the union bound (i.e. multiply by 256) to obtain an upper bound on the error probability. The values of these are given in Table 3.

Security. The security of Kyber is based on the hardness of the $\mathcal{R}_{3329,X^{256}+1}$-$\mathsf{LWE}_{k,\psi_2}$ problem. The security proof works exactly as in the proof of the scheme in Sections 2.3.1 and 4.3. One thing to notice is that in Kyber-512, the value of η_1 was set to 3, while $\eta_2 = 2$. This means that distinguishing the public key $(\mathbf{A}, \mathbf{t})$ from a uniformly-random one is based on the hardness of the $\mathcal{R}_{3329,X^{256}+1}$-$\mathsf{LWE}_{k,\psi_3}$, while the hardness of distinguishing the ciphertext from uniform is based on a "hybrid" distribution where the coefficients of $\mathbf{r}$ are chosen from ψ_3, while those of $\mathbf{e}_1$ and e_2 are from ψ_2. This is at least as hard as the $\mathcal{R}_{3329,X^{256}+1}$-$\mathsf{LWE}_{k,\psi_2}$ problem, but note that we don't output $\mathbf{r}^T\mathbf{A} + \mathbf{e}_1^T$, but rather $\left\lceil \mathbf{r}^T\mathbf{A} + \mathbf{e}_1^T \right\rfloor_{q \to 2^{d_u}}$. This means that some extra error is being added like in the Learning with Rounding Problem (see Section 2.5.3). The total combined error of coefficients from ψ_2 and the error created by compressing from 3329 to a set of size $2^{d_u} = 1024$ is actually somewhat larger than ψ_3. So while, technically, the hardness of distinguishing the ciphertext from a uniform string is based on the hardness of $\mathcal{R}_{3329,X^{256}+1}$-$\mathsf{LWE}_{k,\psi_2}$, in practice, we get a few extra bits of heuristic security and the problem should be as hard as $\mathcal{R}_{3329,X^{256}+1}$-$\mathsf{LWE}_{k,\psi_3}$. The price we pay for having the coefficients of $\mathbf{s}, \mathbf{e}, \mathbf{r}$ being sampled from ψ_3 instead of ψ_2, and thus achieving this extra heuristic security, is that the decryption error grows.

Computational Efficiency. There are several tricks that one can do to noticeably optimize the efficiency of the scheme in Figure 3. Firstly, one need not store the public key part $\mathbf{A}$ because it is generated uniformly at random. One can thus simply store a 256-bit seed ρ and create $\mathbf{A}$ as $\mathcal{H}(\rho)$ where $\mathcal{H}$ is some cryptographic hash function (e.g. SHAKE) that can expand a seed into an arbitrary-length random-looking string. The public key would therefore consist of just $(\rho, \mathbf{t})$.

As discussed in Section 4.5.2, the NTT algorithm allows for very efficient multiplication in rings of the form $\mathcal{R}_{q,X^d+1}$ whenever q is of the form (Lemma 7) that allows for factorization of $X^d + 1$ into low-degree polynomials. The modulus in Kyber (i.e. 3329) is congruent to 1 modulo 256, and therefore the polynomial $X^{256} + 1$ splits into a product of degree 2 polynomials of the form $(X^2 - r_i)$. As notation, for a polynomial $a \in \mathcal{R}_{X^{256}+1}$, let us denote by $\hat{a}$ its *NTT representation*:

$$\hat{a} = (a \bmod X^2 - r_1, \ldots, a \bmod X^2 - r_{128}). \tag{66}$$

Converting a into $\hat{a}$ (and $\hat{a}$ into a) takes $O(d \log d)$ operations. While this is quite fast, the existence of the NTT representation allows us even further optimizations.

Notice that the polynomials comprising the matrix $\mathbf{A}$ are sampled at random. And in order to efficiently do the multiplication $\mathbf{As}$, we need to first convert the polynomials $\mathbf{A}$ to their NTT representation as in (66). The simple observation is that because there is a 1-1 correspondence between polynomials and their NTT representations, we can just sample $\mathbf{A}$ randomly in its NTT representation already! Furthermore, the public key $\mathbf{t}$ can be stored in its NTT representation and so there is no need to do an inverse NTT after

Public parameters: The public parameters of the CPA-Encryption scheme from Figure 3

KEM-KeyGen	KEM-Encaps(pk)	KEM-Decaps(sk, c, h, z)
$(pk, sk) \leftarrow$ CPA-KeyGen $pk := (\mathbf{A}, \mathbf{t}), sk := \mathbf{s}$	$m \leftarrow \{0,1\}^{256} \in \mathcal{R}_{X^{256}+1}$ $(K, \rho) := \mathcal{H}(m, pk) \in \{0,1\}^{512}$ $c :=$ CPA-Encrypt(pk, m, ρ) Shared Key $:= K$, ctxt $:= c$	$m' :=$ CPA-Decrypt(sk, c) $(K', \rho') := \mathcal{H}(m', pk)$ $c' :=$ CPA-Encrypt(pk, m', ρ') if $c \neq c'$, then $K' := \perp$ Shared Key $:= K'$

Figure 4: CCA-secure Key Encapsulation Scheme constructed using the Fujisaki-Okamoto transform. The functions $\mathcal{H}$ and $\mathcal{G}$ are modeled as random oracles. The CPA-KeyGen, CPA-Encrypt, and CPA-Decrypt algorithms are as in Figure 3 (though the construction is rather generic), with the value $\rho \in \{0,1\}^{256}$ added to the CPA-Encrypt input represents the random coins used in the procedure which is used in the generation of $(\mathbf{r}, \mathbf{e}_1, e_2)$.

the multiplication. And since $\mathbf{t}$ in its NTT representation will be needed anyway for the computation of $\mathbf{r}^T\mathbf{t}$ in the encryption algorithm, this is a double-win.

On the other hand, note that we cannot sample $\mathbf{s}, \mathbf{e}, \mathbf{r}, \mathbf{e}_1$, and e directly in their NTT representation because their distribution is not uniformly random. We also cannot perform the compression operations $\lceil \cdot \rfloor_{q \to p}$ when the element is in its NTT representation. Therefore some NTT computations will be necessary. Nevertheless, because the matrix $\mathbf{A}$ consists of k polynomials, by sampling it in NTT form already, we avoid doing k^2 NTT computations – which is a very substantial saving.

4.7 From CPA Encryption to a CCA-KEM

A Key Encapsulation Mechanism (KEM) allows two parties to exchange a random message (shared key). It consists of three algorithms – KEM-KeyGen, KEM-Encaps, and KEM-Decaps. The key generation algorithm outputs a secret key and a public key. The encapsulation algorithm takes the public key as input and outputs a shared key and a ciphertext. The decapsulation algorithm, in turn, takes the ciphertext and the secret key as inputs and produces the same shared key as output. A CPA-secure KEM is one in which an adversary cannot distinguish the shared key from uniform when given the public key and ciphertext. Such a KEM can be constructed from any CPA-secure public key encryption scheme by simply encrypting a random message and setting it as the shared key.[10] To be considered CCA-secure, the indistinguishability of the shared key from random should be preserved even when the adversary has access to a decapsulation oracle which it may use on anything other than the given ciphertext.

The conversion from a CPA-secure public key encryption scheme to a CCA-Secure KEM follows the Fujisaki-Okamoto (FO) transform. The intuition behind the Fujisaki-Okamoto transform is to make the decapsulation oracle "useless" to the adversary so that the only time it produces a non-$\perp$ output is when it is given a ciphertext for which the adversary already knows the message. The way the latter is accomplished is by making the randomness used in constructing the ciphertext dependent on the message [11] and having the decapsulation algorithm first decrypt the ciphertext to obtain the message, and then re-encrypt it, and output $\perp$ if the ciphertexts don't match. This generic conversion is

[10] To achieve slightly more "advanced" security notions, which we do not discuss here, one would not just output a message as the shared key, but rather a hash of it with the public key, as in Figure 4."

[11] This makes the encryption scheme deterministic, but this does not pose problems because we're always encrypting random messages.

presented in Figure 4.

Some Small Modification for Lattices. There are a few modifications to Figure 4 that are present in the standardized ML-KEM (i.e. Kyber) owing to the particulars of lattice-based encryption. Unlike in discrete logarithm schemes, the size of the public key in lattice-based scheme is rather large ($\approx$ 1KB) and the algebraic operations that use the NTT are very fast compared to the much slower exponentiation or elliptic curve multiplication operations in classical cryptography. Thus hashing the public key in the encapsulation and decapsulation fnctions is actually a very computationally-noticeable operation which could be somewhere between 30 and 50 percent of the running time when the scheme is implemented using AVX-2 instructions. Because in many practical scenarios, the decapsulation algorithm is run more often than key generation (e.g. the public key of a party could be fixed), we can pre-hash the public key as $h = \mathcal{G}(pk)$ in the KEM-KeyGen algorithm and store it, and then use $\mathcal{G}(pk)$ in the KEM-Encaps algorithm as input to $\mathcal{H}$ (nothing is saved here) and use h in the KEM-Decaps algorithm instead of pk. In the latter algorithm, the savings entail hashing only 32 bytes instead of $\approx$ 1KB.

Another change in the Kyber KEM is that $\perp$ is never output, but rather in the case that the ciphertexts do not match, a random key is output which is a hash of the input ciphertext and some random secret value created during the key generation. The reasoning for this is somewhat technical, and it is not really clear whether this adds any security in practice.

5 Digital Signatures from Σ-Protocols

In this section, we will strive towards a construction of a lattice-based digital signature scheme whose high-level structure is similar to the classic discrete-log based Schnorr signature scheme [Sch89]. In a Schnorr digital signature scheme, the public key consists of two elements g, h in a finite field and the signature is a non-interactive Zero-Knowledge Proof of Knowledge (ZKPoK) of an exponent x such that $g^x = h$. The non-interactive proof is obtained in two steps. First, we construct a 3-move interactive Σ-protocol that's an honest-verifier ZKPoK of an x satisfying $g^x = y$. And secondly, we transform the interactive proof into a non-interactive one using the Fiat-Shamir transform.

5.1 The Goal for the ZKPoK

We would like to now follow a similar road-map for constructing a signature scheme from the $\mathcal{R}_{q,f}$-$\mathsf{LWE}_{n,m,\beta}$ and $\mathcal{R}_{q,f}$-$\mathsf{SIS}_{n,m,\beta}$ problems. We can start with a $\mathcal{R}_{q,f}$-$\mathsf{LWE}_{n,m,\beta}$ instance $(\mathbf{A}, \mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2)$, where $\mathbf{A} \leftarrow \mathcal{R}_{q,f}^{n\times m}$, $\mathbf{s}_1 \leftarrow [\beta]^m, \mathbf{s}_2 \leftarrow [\beta]^n$, make $\mathbf{A}, \mathbf{t}$ the public key and hope to be able to create a zero-knowledge proof that proves knowledge of $\mathbf{s}_1, \mathbf{s}_2$ in the appropriate range.[12] Creating such a proof turns out to be less straight-forward and significantly less efficient than in the discrete logarithm setting. The reason is that in addition to proving the algebraic relation that the $\mathbf{s}_i$ satisfy $\mathbf{A}\mathbf{s}_1 + \mathbf{s}_2 = \mathbf{t}$, we also need to prove that the coefficients of $\mathbf{s}_i$ fall into a particular range (ideally $[\beta]$, but $[\bar{\beta}]$ for some $\bar{\beta}$ a little larger than β is also OK).

The most efficient signatures that stem from Σ-protocols go around the need for proving knowledge of small $\mathbf{s}_1, \mathbf{s}_2$ satisfying

$$\mathbf{A}\mathbf{s}_1 + \mathbf{s}_2 = \mathbf{t}, \tag{67}$$

and instead prove knowledge of a "relaxed" solution to the equation. In particular, we will be giving protocols that allow a prover in possession of $\mathbf{s}_1 \in [\beta]^m, \mathbf{s}_2 \in [\beta]^n$ satisfying the above equation to prove knowledge of $\bar{\mathbf{s}}_1, \bar{\mathbf{s}}_2$ with coefficients in a somewhat larger interval than $[\beta]$, and another element $\bar{c}$ with small coefficients, satisfying

$$\mathbf{A}\bar{\mathbf{s}}_1 + \bar{\mathbf{s}}_2 = \bar{c}\mathbf{t}. \tag{68}$$

A simple proof below shows that proving knowledge of (68) still proves something meaningful. In particular, it will allow us to conclude that being able to produce such $\bar{\mathbf{s}}_1, \bar{\mathbf{s}}_2$, and $\bar{c}$ means that one can either solve Ring-LWE or Ring-SIS related to the matrix $\mathbf{A}$.

Lemma 9. *Suppose that there is an algorithm, that when given $\mathbf{A} \leftarrow \mathcal{R}_{q,f}^{n\times m}, \mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$ for $\mathbf{s}_1 \leftarrow [\beta]^m, \mathbf{s}_2 \leftarrow [\beta]^n$, is able to come up with $\bar{\mathbf{s}}_1 \in [\bar{\beta}]^m, \bar{\mathbf{s}}_2 \in [\bar{\beta}]^n$ and $\bar{c} \in [2]$ such that $\mathbf{A}\bar{\mathbf{s}}_1 + \bar{\mathbf{s}}_2 = \bar{c}\mathbf{t}$. Then there is another algorithm, running in the same time, and having the same probability of success, that solves either the $\mathcal{R}_{q,f}$-$\mathsf{LWE}_{n,m,\beta}$ or the $\mathcal{R}_{q,f}$-$\mathsf{SIS}_{n,m+1,\bar{\beta}}$ problems.*

Proof. Given a uniformly-random matrix $\bar{\mathbf{A}} = [\mathbf{A} \mid \mathbf{t}] \in \mathcal{R}_{q,f}^{n\times(m+1)}$, which is an instance of the $\mathcal{R}_{q,f}$-$\mathsf{SIS}_{n,m+1,\bar{\beta}}$ problem, the reduction sets $(\mathbf{A}, \mathbf{t})$ as the public key to the protocol in Figure 5. By the $\mathcal{R}_{q,f}$-$\mathsf{LWE}_{n,m,\beta}$ assumption, this is computationally-indistinguishable from the real key distribution. If there is an algorithm which can produce $\bar{\mathbf{s}}_1, \bar{\mathbf{s}}_2, \bar{c}$ with coefficients at most $\bar{\beta}$ satisfying $\mathbf{A}\bar{\mathbf{s}}_1 + \bar{\mathbf{s}}_2 = \bar{c}\mathbf{t}$, then we have a solution to the $\mathcal{R}_{q,f}$-$\mathsf{SIS}_{n,m+1,\bar{\beta}}$ instance $\bar{\mathbf{A}}$. □

[12] Recall the notation from Section 4.2: for a polynomial s, $s \leftarrow [\beta]$ means that all the coefficients of s are chosen uniformly at random from the set $[\beta] = \{-\beta, \ldots, 0, \ldots, \beta\}$ and $\mathbf{s} \leftarrow [\beta]^n$ means that all the n polynomials in the polynomial vector $\mathbf{s}$ have their coefficients chosen uniformly from the set $[\beta]$.

Note that in the above reduction, the Ring-SIS problem has dimension $m+1$ (rather than m) and completely ignores the fact that $\bar{c} \in [2]$ rather than in $[\bar{\beta}]$. Thus the Ring-SIS problem from which it reduces is slightly easier than what is actually needed for a forgery. A different argument allows us to obtain a reduction in which the dimension of the Ring-SIS instance is not increased, and we present it below for completeness.

Lemma 10. *Suppose that there is an algorithm, that when given* $\mathbf{A} \leftarrow \mathcal{R}_{q,f}^{n \times m}, \mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$ *for* $\mathbf{s}_1 \leftarrow [\beta]^m, \mathbf{s}_2 \leftarrow [\beta]^n$*, is able to come up with* $\bar{\mathbf{s}}_1 \in [\bar{\beta}]^m, \bar{\mathbf{s}}_2 \in [\bar{\beta}]^n$ *and* $\bar{c}$ *with* $\|\bar{c}\|_1 = \bar{\eta}$ *such that* $\mathbf{A}\bar{\mathbf{s}}_1 + \bar{\mathbf{s}}_2 = \bar{c}\mathbf{t}$*; and this algorithm succeeds with probability* ϵ*. Set* β' *such that* $(2\beta'+1)^{n+m} > q^n \cdot 2^{128/d}$*. Then there exists another algorithm which can solve (at least one of)* $\mathcal{R}_{q,f}$-$\mathsf{LWE}_{n,m,\beta}$*,* $\mathcal{R}_{q,f}$-$\mathsf{LWE}_{n,m,\beta'}$ *or* $\mathcal{R}_{q,f}$-$\mathsf{SIS}_{n,m,\bar{\beta}+\beta'\cdot\bar{\eta}}$*, which runs in the same time and succeeds with probability* $\epsilon - 2^{-128}$*.*[13]

Proof. Let $\mathcal{A}$ be such an algorithm and suppose that we are given a matrix $(\mathbf{A}, \mathbf{t})$. We pick random $\mathbf{s}_1' \leftarrow [\beta']^m, \mathbf{s}_2' \leftarrow [\beta']^n$ and construct $\mathbf{t}' = \mathbf{A}\mathbf{s}_1' + \mathbf{s}_2'$. By the $\mathcal{R}_{q,f}$-$\mathsf{LWE}_{n,m,\beta}$ and $\mathcal{R}_{q,f}$-$\mathsf{LWE}_{n,m,\beta'}$ assumptions, the distributions $(\mathbf{A}, \mathbf{t})$ and $(\mathbf{A}, \mathbf{t}')$ are computationally indistinguishable and so $\mathcal{A}$ should produce $\bar{\mathbf{s}}_1 \in [\bar{\beta}]^m, \bar{\mathbf{s}}_2 \in [\bar{\beta}_2]$, and $\bar{c} \in \mathcal{R}_{q,f}$ with $\|\bar{c}\|_1 \leq \bar{\eta}$ satisfying $\mathbf{A}\bar{\mathbf{s}}_1 + \bar{\mathbf{s}}_2 = \bar{c}\mathbf{t}'$. This implies that

$$\mathbf{A}(\bar{\mathbf{s}}_1 - \bar{c}\mathbf{s}_1') + (\bar{\mathbf{s}}_2 - \bar{c}\mathbf{s}_2') = \mathbf{0}. \tag{69}$$

Since all the coefficients of $\bar{\mathbf{s}}_i - \bar{c}\mathbf{s}_i'$ are in $[\bar{\beta} + \beta' \cdot \bar{\eta}]$, we have a solution to $\mathcal{R}_{q,f}$-$\mathsf{SIS}_{n,m,\bar{\beta}+\beta'\cdot\bar{\eta}}$ – as long as at least one of $\bar{\mathbf{s}}_i - \bar{c}\mathbf{s}_i'$ is non-zero. We will now show that this bad case can happen with probability at most 2^{-128}. There are $(2\beta'+1)^{(n+m)\cdot d}$ possible values for $\mathbf{s}_1', \mathbf{s}_2'$, while there are only q^{nd} possible values for $\mathbf{t}'$. It is thus information-theoretically impossible to guess the pre-image $(\mathbf{s}_1', \mathbf{s}_2')$ from $\mathbf{t}'$ except with probability $q^{nd}/(2\beta'+1)^{(n+m)d}$.[14] By our choice of β', this probability is at most 2^{-128}. □

5.1.1 The Challenge Space.

The size of the coefficients in $\bar{c}$ (as well as $\bar{\mathbf{s}}_1$ and $\bar{\mathbf{s}}_2$) will be dependent on the challenge space of the ZKPoK. Since we would like these to be small, we want to define our challenge space so that it consists of polynomials with small norms.

If we're working over the ring $\mathcal{R}_f$ where the degree of f is d, then define η to be the smallest integer such that $2^\eta \cdot \binom{d}{\eta} > 2^{256}$ (we assume that d is large enough for such an η to exist). Then we define the challenge set $\mathcal{C} \subset \mathcal{R}_f$ as

$$\mathcal{C} = \{c \in [1], \|c\|_1 = \eta\}, \tag{70}$$

and the set of all (non-zero) differences as

$$\bar{\mathcal{C}} = \{\bar{c} = c_1 - c_2, \text{ for } c_1 \neq c_2 \in \mathcal{C}\}. \tag{71}$$

So $\mathcal{C}$ consists of all polynomials in $\mathcal{R}_f$ with exactly η non-zero coefficients taken from the set $\{-1, 1\}$. By the definition of η, the size of $\mathcal{C}$ is exactly $2^\eta \cdot \binom{d}{\eta}$. Note that we could have defined $\mathcal{C}$ to also include polynomials with fewer than η non-zero coefficients, but this would increase the complexity of sampling a random element in $\mathcal{C}$ while not increasing its size by much.[15]

[13] Because β' is approximately $q^{n/(n+m)}$, the $\mathcal{R}_{q,f}$-$\mathsf{LWE}_{n,m,\beta'}$ problem is at the apex of its hardness (see Figure 2), and so the reliance on this extra assumption presents no additional constraint in practice. The main constraints will be the security of $\mathcal{R}_{q,f}$-$\mathsf{LWE}_{n,m,\beta}$ and $\mathcal{R}_{q,f}$-$\mathsf{SIS}_{n,m,\bar{\beta}+\beta'\cdot\bar{\eta}}$.

[14] To see this, observe that the optimal strategy for guessing the pre-image $(\mathbf{s}_1', \mathbf{s}_2')$ when seeing some value $\mathbf{t}'$, is to deterministically guess the most likely pre-image. So the optimal guesser will output at most q^{nd} possible pre-images. Since there are $(2\beta'+1)^{(n+m)d}$ total pre-images, each equally-likely to be picked, only a $q^{nd}/(2\beta'+1)^{(n+m)d}$ fraction of them is ever output by the optimal guesser.

[15] Sampling a random vector of length d with η ± 1's can be done by initializing a d-dimensional vector with η 1's, then performing a shuffle (e.g. Fisher-Yates) to obtain a random permutation of this vector, and then randomly negating (or not) each of the 1's.

5.2 The Basic Σ-Protocol

We now describe the basic scheme, given in Figure 5, which was presented in [Lyu09]. An unusual feature of this protocol is that it does not have perfect completeness. In order to keep the coefficients of the output small, a rejection sampling step in the last round of the Σ-protocol is performed in order to make sure that the distribution is independent of the secret. In all the protocols in this manuscript, the rejection sampling step will be quite simple – just checking whether all the coefficients are in a certain range. One can perform slightly more complicated rejection sampling steps, which require sampling and rejecting according to a discrete Gaussian distribution, which results in slightly smaller outputs [Lyu12, DDLL13]. The main downside of these latter algorithms is that the rejection sampling step is more complex and a slightly incorrect implementation may end up leaking the secret key – it may also be more complicated to defend against side-channel attacks. A simpler algorithm for such a widely-used primitive, like a digital signature, may thus be preferable in practice.

The main consequence of the rejection sampling step is that the running time of the signing algorithm will be a random variable (but independent of $\mathbf{s}_1, \mathbf{s}_2$), rather than fixed. Other than this, a reader familiar with Schnorr-type proofs will find a lot of similarities in this protocol. The first stage of the protocol consists of creating the masking variables $\mathbf{y}_1$ and $\mathbf{y}_2$, the second step is the challenge, and the last step consists of adding the mask to the product of the challenge and the secret. A rejection sampling step is then performed, and if the prover sends $\perp$, he aborts and will need to restart the protocol. The transformation of this protocol to a signature scheme will use the usual Fiat-Shamir transform, in which the challenge is created as a hash of the message and the first message of the prover.

A common trick for keeping the size of the communication compact in an interactive scheme is to send a hash of the first message instead of the message itself. Because the unhashed first message can be recovered from those sent in the next rounds, the verifier is able to compute the hash of this message during verification. In the lattice setting where we use rejection sampling, this trick has an additional advantage in that it will allow us to simulate transcripts in which a rejection occurs. This hashing is, however, unnecessary for proving the security of the signature scheme because the signer does not see the aborted signing attempts. For this reason, we will present the interactive scheme without the hashing and only prove zero-knowledge in the case that no aborts (i.e. $\perp$ is not sent) occur.

5.2.1 Honest Verifier Zero-Knowledge

We will now prove that the protocol in Figure 5 is HVZK. That is, we will show how to create valid transcripts without knowledge of the secret $\mathbf{s}_1, \mathbf{s}_2$. The key to the proof is the below lemma which shows that for all $\mathbf{s}_1, \mathbf{s}_2$ with bounded coefficients, the probability of $\perp$ is the same and that the distribution of $\mathbf{z}_1, \mathbf{z}_2$ is independent of $\mathbf{s}_1, \mathbf{s}_2$.

Lemma 11. *If $\gamma \in \mathbb{Z}^+$ is such that for all polynomials $s \in [\beta], c \in \mathcal{C}$, we have $cs \in [\gamma]$, then for all $\mathbf{s}_i, c$ as in the protocol in Figure 5*

$$\Pr_{\mathbf{y}_1,\mathbf{y}_2}[(\mathbf{z}_1, \mathbf{z}_2) \neq \perp] = \left(\frac{2\bar{\beta}+1}{2(\bar{\beta}+\gamma)+1}\right)^{d(m+n)} \tag{72}$$

and

$$\forall \mathbf{z}_1' \in [\beta]^m, \mathbf{z}_2' \in [\beta]^n, \Pr_{\mathbf{y}_1,\mathbf{y}_2}[(\mathbf{z}_1, \mathbf{z}_2) = (\mathbf{z}_1', \mathbf{z}_2') \mid (\mathbf{z}_1, \mathbf{z}_2) \neq \perp] = \left(\frac{1}{2\bar{\beta}+1}\right)^{d(m+n)} \tag{73}$$

Proof. Consider $\begin{bmatrix}\mathbf{z}_1\\ \mathbf{z}_2\end{bmatrix} = \begin{bmatrix}c\mathbf{s}_1\\ c\mathbf{s}_2\end{bmatrix} + \begin{bmatrix}\mathbf{y}_1\\ \mathbf{y}_2\end{bmatrix}$ as a sum of integer vectors, which written in notation

Private information: $\mathbf{s}_1 \in [\beta]^m, \mathbf{s}_2 \in [\beta]^n$
Public information: $\mathbf{A} \in \mathcal{R}_{q,f}^{n\times m}, \mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2 \in \mathcal{R}_{q,f}^n$

Prover		Verifier
$\mathbf{y}_1 \leftarrow [\gamma + \bar{\beta}]^m$ $\mathbf{y}_2 \leftarrow [\gamma + \bar{\beta}]^n$, $\mathbf{w} := \mathbf{A}\mathbf{y}_1 + \mathbf{y}_2$		
	$\xrightarrow{\mathbf{w}}$	
		$c \leftarrow \mathcal{C}$
	$\xleftarrow{c}$	
$\mathbf{z}_1 := c\mathbf{s}_1 + \mathbf{y}_1$ $\mathbf{z}_2 := c\mathbf{s}_2 + \mathbf{y}_2$ if $\mathbf{z}_1 \notin [\bar{\beta}]^m$ or $\mathbf{z}_2 \notin [\bar{\beta}]^n$ then $(\mathbf{z}_1, \mathbf{z}_2) := \perp$		
	$\xrightarrow{(\mathbf{z}_1, \mathbf{z}_2)}$	
		Accept iff $\mathbf{z}_1 \in [\bar{\beta}]^m$ and $\mathbf{z}_2 \in [\bar{\beta}]^n$ and $\mathbf{A}\mathbf{z}_1 + \mathbf{z}_2 - c\mathbf{t} = \mathbf{w}$

Figure 5: The basic Zero-Knowledge Proof System in which the prover knows $\mathbf{s}_1 \in [\beta]^m, \mathbf{s}_2 \in [\beta]^n$ satisfying (67) and gives a ZKPoK of knowledge of $\bar{\mathbf{s}}_1 \in [2\bar{\beta}]^m$, $\bar{\mathbf{s}}_2 \in [2\bar{\beta}]^n$, and a $\bar{c} \in \bar{\mathcal{C}}$ satisfying (68). The value γ is defined in Lemma 11, and the value of $\bar{\beta}$ affects the completeness of the protocol (i.e. the probability that $\perp$ is not sent) as specified in Lemma 11.

from 4.1.1, is

$$\mathbf{z} = \begin{bmatrix} \mathcal{V}_{\mathbf{z}_1} \\ \mathcal{V}_{\mathbf{z}_2} \end{bmatrix} = \begin{bmatrix} \mathcal{V}_{c\mathbf{s}_1} \\ \mathcal{V}_{c\mathbf{s}_2} \end{bmatrix} + \begin{bmatrix} \mathcal{V}_{\mathbf{y}_1} \\ \mathcal{V}_{\mathbf{y}_2} \end{bmatrix} \in \mathbb{Z}^{d(n+m)}.$$

The probability that the i^{th} coefficient (for any i) of $\mathbf{z}$ is a particular coefficient $\nu_z \in [\bar{\beta}]$ is exactly $\frac{1}{2\bar{\beta}+1}$. The reason is the following: suppose that the i^{th} coefficient in $\begin{bmatrix} \mathcal{V}_{c\mathbf{s}_1} \\ \mathcal{V}_{c\mathbf{s}_2} \end{bmatrix}$ is ν_s, then the i^{th} coefficient ν_y of the vector $\begin{bmatrix} \mathcal{V}_{\mathbf{y}_1} \\ \mathcal{V}_{\mathbf{y}_2} \end{bmatrix}$ will need to be exactly $\nu_z - \nu_s$. Notice that $\nu_z \in [\bar{\beta}]$ and $\nu_s \in [\gamma]$ implies that $\nu_z - \nu_s \in [\bar{\beta} + \gamma]$, which is exactly the range that the coefficient ν_y gets selected from. Therefore the probability that ν_y will be this value is exactly $\frac{1}{2(\bar{\beta}+\gamma)+1}$. Thus

$$\forall \mathbf{z}_1' \in [\bar{\beta}]^m, \mathbf{z}_2' \in [\bar{\beta}]^n, \Pr_{\mathbf{y}_1,\mathbf{y}_2}[(\mathbf{z}_1, \mathbf{z}_2) = (\mathbf{z}_1', \mathbf{z}_2')] = \left(\frac{1}{2(\bar{\beta} + \gamma) + 1}\right)^{d(m+n)}. \quad (74)$$

And since there are $(2\bar{\beta} + 1)^{d(m+n)}$ possible valid $(\mathbf{z}_1', \mathbf{z}_2') \in [\bar{\beta}]^m \times [\bar{\beta}]^n$ that could be sent, we obtain the claim in the first part of the lemma (i.e. (72)).

To obtain the second part of the lemma, we observe that (73) = (74)/(72). □

The probability of the prover not sending $\perp$ is

$$\left(\frac{2\bar{\beta} + 1}{2(\bar{\beta} + \gamma) + 1}\right)^{d(m+n)} > \left(\frac{\bar{\beta}}{\bar{\beta} + \gamma}\right)^{d(m+n)} = \left(1 + \frac{\gamma}{\bar{\beta}}\right)^{-d(m+n)} \approx e^{-\gamma d(m+n)/\bar{\beta}}, \quad (75)$$

and so setting $\bar{\beta} = \gamma d(m + n)$ would result in the protocol requiring an expected number of e repetitions before a non-$\perp$ value is sent. One could of course set $\bar{\beta}$ to be smaller at the cost of a higher number of expected repetitions.

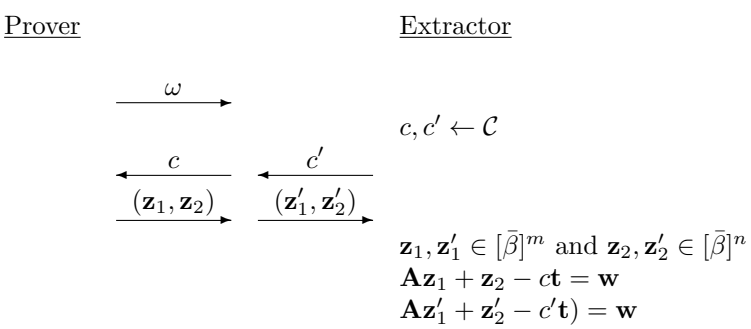

Figure 6: Extraction procedure for (68).

We now use Lemma 11 to show how to simulate a non-aborting transcript with the correct probability, which will show that the protocol in Figure 5 is honest-verifier zero-knowledge in the case that $\perp$ is not sent.

The simulator chooses random $\mathbf{z}_1 \leftarrow [\bar{\beta}]^m$, $\mathbf{z}_2 \leftarrow [\bar{\beta}]^n$, $c \leftarrow \mathcal{C}$, sets $\mathbf{w} := \mathbf{A}\mathbf{z}_1 + \mathbf{z}_2 - c\mathbf{t}$ and outputs $(\mathbf{w}, c, \mathbf{z}_1, \mathbf{z}_2)$. This distribution perfectly simulates the non-aborting transcript because c is uniform, and by Lemma 11, the values of $\mathbf{z}_1, \mathbf{z}_2$ are uniformly random (for any c); and $\mathbf{w}$ is uniquely determined by the other variables. This completes the proof that the protocol in Figure 5 is HVZK when $\perp$ is not sent.

Before continuing, we would like to remark that the probability with which the honest prover will send $\perp$ (and thus will have to repeat the protocol) is *independent* of the secret $\mathbf{s}_1, \mathbf{s}_2$ (Lemma 11). This is important in real-world applications where any dependence of the running time on the secret would lead to side-channel attacks where the adversary attempts to deduce some information about the secret by observing the running time of the prover. The protocol in Figure 5 is therefore immune to this particular attack.

5.2.2 Proof of Knowledge

To show that the protocol is a PoK, we use the usual rewinding argument (see Figure 6) in which the prover sends ω and then successfully replies to two challenges c, c' with $(\mathbf{z}_1, \mathbf{z}_2)$ and $(\mathbf{z}_1', \mathbf{z}_2')$. If we can extract two such transcripts $(\mathbf{w}, c, \mathbf{z}_1, \mathbf{z}_2)$ and $(\mathbf{w}, c', \mathbf{z}_1', \mathbf{z}_2')$ satisfying the verification equation, then we have $\mathbf{A}\mathbf{z}_1 + \mathbf{z}_2 - c\mathbf{t} = \mathbf{A}\mathbf{z}_1' + \mathbf{z}_2' - c'\mathbf{t}$. The preceding simplifies to

$$\mathbf{A}(\mathbf{z}_1 - \mathbf{z}_1') + (\mathbf{z}_2 - \mathbf{z}_2') = (c - c')\mathbf{t} \tag{76}$$

which is exactly the statement in (68) with $\bar{\mathbf{s}}_1 \in [2\bar{\beta}]^m, \bar{\mathbf{s}}_2 \in [2\bar{\beta}]^n, \bar{c} \in [2]$.

5.2.3 Putting it All Together.

The Honest Verifier Zero-Knowledge property implies that an adversary gains no information from seeing non-aborted transcripts. The Proof of Knowledge property implies that an adversary who is able to impersonate a prover can produce a solution as in (68), which implies by Lemma 9, that he can solve the Ring-LWE or the Ring-SIS problem. The two properties together imply that (assuming Ring-SIS and Ring-LWE are hard) an adversary cannot impersonate the prover in the protocol in Figure 5 even if he observes previous non-aborting valid interactions. This allows us, using the Fiat-Shamir transform, to construct a digital signature scheme secure, in the random oracle model, based on the hardness of Ring-SIS and Ring-LWE..

5.2.4 Setting the Parameters.

The proof of knowledge argument at the beginning of this section (see (76)) implies that one can extract $\bar{\mathbf{s}}_1, \bar{\mathbf{s}}_2$ with coefficients in $[2\bar{\beta}]$ and $\bar{c}$ such that $\|\bar{c}\|_1 \leq 2\eta$ which satisfy (68). Lemma 9 then states that if $\mathcal{R}_{q,f}$-$\mathsf{LWE}_{n,m,\beta}$ is hard, then the above extraction implies solving $\mathcal{R}_{q,f}$-$\mathsf{SIS}_{n,m+1,\bar{\beta}}$ (or $\mathcal{R}_{q,f}$-$\mathsf{SIS}_{n,m,2(\bar{\beta}+\beta'\eta)}$ by Lemma 10). The optimal parameter setting is therefore when these two problems are equally hard – i.e. the problems are on the same vertical position in Figure 2. Choosing the parameter $\bar{\beta}$ is dictated by Lemma 11. As pointed out in (75), $\bar{\beta}$ should be set to around $\gamma d(m+n)$, where γ is such that $cs \in [\gamma]$. So if c is chosen such that $\|c\|_1 \leq \eta$ and $s \in [\beta]$, then γ can be set to $\eta \cdot \beta$. Thus $\bar{\beta}$ is approximately a factor $\eta d(m+n)$ larger than β.

5.3 Analogies to Discrete Logarithm Schemes: Schnorr, Okamoto, and Katz-Wang.

In this section, we will dig deeper into the analogy between the lattice-based protocol in Figure 5 and discrete log based protocols.[16] As previously mentioned, the lattice protocol and its Fiat-Shamir transformed signature scheme are very analogous to the Schnorr identification and signature schemes [Sch89]. We will see in the following that by simply changing the parameter β (and $\bar{\beta}$ which is derived from it), one can obtain schemes which have slightly different security characteristics which are analogous to other discrete logarithm based schemes in the literature. At the end, it will turn out that the Schnorr instantiation, where one is free to set β without any constraints, is the most efficient variant; but we still we believe that it is instructive to see the other variants to get more intuition for how lattice-based protocols are constructed and instantiated. Indeed, understanding how parameters affect the nature of lattice schemes is an important component of designing lattice cryptography protocols.

Schnorr. The public key in the Schnorr protocol consists of a random g and $h = g^x$, where x is the secret key. In the first move, the prover picks a random masking variable y, computes $w = g^y$, and sends w to the verifier. The verifier sends a random challenge c, to which the prover replies with $z = y + xc$, and the verifier checks that $g^z = t^c \cdot w$.

To show that impersonation (in the honest verifier setting) implies breaking discrete log, upon receiving the discrete log challenge (g, h), we set this to be the public key. Without knowing the x such that $g^x = h$, one can simulate honestly-generated transcripts (w, c, z) by first picking random z, c, and then setting $w = g^z/t^c$. After this, it can be shown that if the adversary is able to impersonate, then the usual rewinding argument obtains two transcripts (w, c, z) and (w, c', z') such that $g^z = t^c \cdot w$ and $g^{z'} = t^{c'} \cdot w$. From this, we obtain $\bar{z} = z - z'$ and $\bar{c} = c - c'$ satisfying $g^{\bar{z}} = t^{\bar{c}}$. And from the latter, one can obtain a valid discrete log solution $\bar{s}/\bar{c}$.

The lattice-based analogue in Figure 5 sets the public key as $(\mathbf{A}, \mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2)$ with the secret key being $\mathbf{s}_1, \mathbf{s}_2$. In the lattice case, we extract the $\bar{\mathbf{s}}_i, \bar{c}$ satisfying (68), and then use Lemma 9 to show that this implies a solution to Ring-SIS for the instance $[\mathbf{A} \mid \mathbf{t}]$. A minor difference is that in the case of Schnorr signatures, the public key (g, g^x) was random, whereas in the lattice case, we needed the Ring-LWE assumption to argue that the public key looks random in Lemma 9.

One should observe that algebraically, the lattice-based and the discrete-log based schemes are quite similar. In both cases, there is some homomorphic one-way function family $\mathcal{F}$, and the public key consists of $f, f(x)$, where f is a randomly-chosen member of

[16] This section is not needed for the sequel and can be skipped if one only wishces to get to the final construction of the signature scheme.

this family, and x is a randomly-chosen secret key. The first move consists of choosing a random mask y and sending $w = f(y)$. And on challenge c, the prover responds with $z = y + xc$. Different properties of discrete log protocols are achieved by varying the relationship between the size of the domain and range of f – and the lattice analogues are obtained using the same blueprint, as we will now see.

Okamoto. The Okamoto protocol [Oka92] is similar in spirit to the Schnorr one with its main differentiating characteristic being that one can prove the Okamoto *identification scheme* secure against *active* adversaries – that is it can be proven secure even if the verifier adversarially chooses the challenges c.[17]. Since the Fiat-Shamir transformation only requires HVZK, this stronger property of the Okamoto signature is not really needed in practice for constructing digital signature schemes. The public key in the Okamoto Scheme consists of random (g_1, g_2) and $h = g_1^{x_1} \cdot g_2^{x_2}$. In the first move, the prover chooses random y_1, y_2 and outputs $w = g_1^{y_1} \cdot g_2^{y_2}$. Upon receiving the challenge c, the prover outputs $z_i = y_i + cs_i$ for $i = 1, 2$. The verifier checks that $g_1^{z_1} \cdot g_2^{z_2} = h^c \cdot w$. To prove the security of the Schnorr scheme, we needed to simulate the transcript by first picking a random z, c and from these deriving w. In the Okamoto scheme, one does not need to simulate – all one needs to do is honestly run the scheme. The reduction from the discrete logarithm problem proceeds as follows: given a discrete log instance (g_1, g_2), where $g_2 = g_1^x$ for some unknown x, the extractor chooses a valid secret key x_1, x_2 and sets the public key to $h = g_1^{x_1} \cdot g_2^{x_2}$. He can thus honestly answer all of the adversary's queries by honestly running the protocol. If the adversary afterwards succeeds in impersonating the prover, then the usual rewinding argument leads to two transcripts $(w, c, z_1, z_2) \neq (w, c', z_1', z_2')$ such that

$$w \cdot h^c = g_1^{z_1} \cdot g_2^{z_2}$$
$$w \cdot h^{c'} = g_1^{z_1'} \cdot g_2^{z_2'},$$

which can be re-written as

$$h^{\bar{c}} = g_1^{\bar{z}_1} \cdot g_2^{\bar{z}_2}, \tag{77}$$

where $\bar{c} = c - c'$ and $\bar{z}_i = z_i - z_i'$. By further rewriting h as $g_1^{x_1} \cdot g_2^{x_2}$, one obtains

$$1 = g_1^{\bar{z}_1 - x_1\bar{c}} \cdot g_2^{\bar{z}_2 - x_2\bar{c}}. \tag{78}$$

Now observe that as long as $\bar{z}_i - x_i\bar{c}$ are not both 0, we can obtain a solution to the discrete logarithm problem – that is find an x such that $g_1^x = g_2$.

Showing that with very high probability $\bar{z}_i - x_i\bar{c}$ will not equal to 0, we note that given $g_1, g_2, h = g_1^{x_1} \cdot g_2^{x_2}$, there are many possible x_1', x_2' such that $g_1^{x_1'} \cdot g_2^{x_2'} = h$ – indeed, if $g_1^x = g_2$, then any x_1', x_2' satisfying $x_1' + x \cdot x_2' = x_1 + x \cdot x_2$ are valid. And any one of these pairs is equally likely to have been chosen as the original secret key. We then need to prove that the distribution of the transcripts in the Okamoto protocol (i.e. the w, c, z_1, z_2) is identical regardless of which of the valid secret keys was chosen (even when the adversary has control of the c).[18] Once this is established, one sees that if the algorithm impersonating the prover can send $\bar{z}_1, \bar{z}_2$ such that $\bar{z}_i - x_i\bar{c} = 0$, then he knows the values $x_i = \bar{z}_i/\bar{c}$. But these are information-theoretically hidden, and so there is not an overwhelming probability that an (even all-powerful) impersonator can output such $\bar{z}_i$. Thus an impersonator can be used to solve discrete log. Notice that the reason the same proof does not work in the

[17] While there is no reduction from the standard discrete log or DDH problems to the actively-secure Schnorr signature scheme, one could prove the scheme secure based on certain "knowledge assumptions" [BP02]

[18] This follows from the fact that the z_i are uniform because the y_i are uniform, and then w is a deterministic function of the z_i and c.

Schnorr protocol is that the public key is $(g, h = g^x)$, and there is only one possible secret key x.

In the lattice setting, all one needs to do to move between Schnorr-like and Okamoto-like schemes is to set the secret key $\mathbf{s}_1, \mathbf{s}_2$ such that the public key $(\mathbf{A}, \mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2)$ does not, with high probability, uniquely determine the secret key. This simply requires us to choose a larger β. In particular, if we choose β such that $(2\beta+1)^{n+m} > q^n \cdot 2^{128/d}$, then there is only a 2^{-128} probability that any (all-powerful) algorithm can recover the exact $(\mathbf{s}_1, \mathbf{s}_2)$.[19] And the fact that the transcripts do not leak any information about the $\mathbf{s}_i$ is proved in Lemma 11.

But just like the Okamoto signature scheme being less efficient than the Schnorr one, the added requirement on β will make this lattice-based instantiation less optimal. We will discuss this in more detail after presenting the next Schnorr variant.

Katz-Wang. Another Schnorr-like protocol with slightly different security properties is the Katz-Wang protocol [KW03, Section 3]. The differentiating feature of this scheme is that it is based entirely on the DDH problem and its security proof does not require rewinding. The advantage of not doing rewinding is that the security reduction is tighter.[20] Thus, in theory, the [KW03] scheme has a tighter connection to a discrete-log type problem. In the lattice setting, if one wanted to prove the security of the scheme in the Quantum Random Oracle Model (QROM), where the adversary is assumed to be quantum and cannot be straightforwardly rewound (because quantum states cannot be copied), then having rewinding makes the security reduction even less tight. Both in the classical and lattice settings, however, the tightness of the reduction from a discrete log or lattice problem does not seem to affect the security of the signature schemes and so the Okamoto and Katz-Wang schemes, as well as their lattice versions, remain mostly of only theoretical interest.

The public key in the Katz-Wang scheme consists of random (g_1, g_2) and $(h_1 = g_1^x, h_2 = g_2^x)$ for a random secret x. To sign, one first creates a random mask y, then computes $(w_1 = g_1^y, w_2 = g_2^y)$, and sends the w_i to the verifier. Upon receiving a challenge c, the prover computes $z = y + cx$. The verifier then checks whether $g_i^z = h_i^c \cdot w_i$.

The security reduction is from the DDH problem. Given a DDH instance (g_1, g_2, h_1, h_2), we are tasked with determining whether these elements are all random or whether there is an x such that $h_i = g_i^x$.[21] Given this (g_1, g_2, h_1, h_2), we simply publish it as the public key. One can then create transcripts of the interaction in the usual way by first selecting z, then c, and then deriving the values for w_i. When it's the adversary's turn to impersonate, we observe that if (g_1, g_2, h_1, h_2) is random, then, information-theoretically, he only has a negligible chance of succeeding in outputting a valid z. In particular, if we write $h_1 = g_1^{x_1}, h_2 = g_2^{x_2}$ for $x_1 \neq x_2$, and $w_1 = g_1^{r_1}$, $w_2 = g_2^{r_2}$, for some r_i, then

$$\begin{aligned}\Pr_c[\exists z \text{ s.t. } g_1^z = h_1^c \cdot w_1 \wedge g_2^z = h_2^c \cdot w_2] &= \Pr_c[\exists z \text{ s.t. } z = x_1 c + r_1 \wedge z = x_2 c + r_2] \\ &= \Pr_c[c = (r_2 - r_1)/(x_1 - x_2)] \\ &= 1/|\mathcal{C}|,\end{aligned}$$

where $\mathcal{C}$ is the domain of the challenge space from which c is chosen. Thus seeing whether the adversary can impersonate immediately gives us a solution to the DDH problem.

[19] To see this, observe that the optimal strategy for guessing the pre-image $(\mathbf{s}_1, \mathbf{s}_2)$ when seeing some value $\mathbf{t}$, is to deterministically guess the most likely pre-image. Because the domain size of $\mathbf{t}$ is q^{nd}, the optimal guesser will output at most q^{nd} possible pre-images. And since there are $(2\beta+1)^{(n+m)d}$ total pre-images, each equally-likely to be picked, only a $q^{nd}/(2\beta+1)^{(n+m)d}$ fraction of them is ever output by the optimal guesser.

[20] In the reduction that uses rewinding, there is a loss of a multiplicative factor of the number of random oracle queries.

[21] This formulation of the DDH problem is equivalent to the more common formulation where one is asked to distinguish (g, g^a, g^b, g^{ab}) from uniform (simply define $g_2 = g^a$).

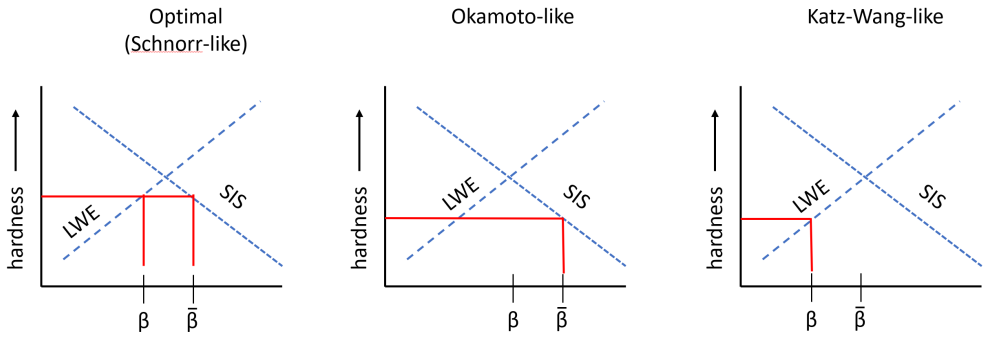

Figure 7: Sketching the intuition for optimal parameter selection of the Schnorr, Okamoto, and Katz-Wang analogues of the lattice protocol from Figure 5. The constraints imposed by the Okamoto and Katz-Wang variants result in less hard instances of the Ring-SIS / Ring-LWE problems, which will require increasing the parameters (like n and m) in order to increase the security of the scheme.

In the case of the lattice scheme from the protocol in Figure 5, the analogy to the Katz-Wang scheme is obtained by setting the parameters so that $\bar{\beta}$ is small-enough so that, information-theoretically, there will not exist a valid response $\mathbf{z}_1, \mathbf{z}_2$ when the public key is uniformly-random $(\mathbf{A}, \mathbf{t})$. Note that in this case, seeing whether the adversary can succeed, will allow us distinguish uniformly-random $(\mathbf{A}, \mathbf{t})$ from $(\mathbf{A}, \mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2)$, which is exactly the Ring-LWE problem. In order to send $\mathbf{z}_1, \mathbf{z}_2$ that will make the verifier accept, we need them to have coefficients in $[\bar{\beta}]$ and satisfy $\mathbf{A}\mathbf{z}_1 + \mathbf{z}_2 = \mathbf{t}c + \mathbf{w}$. What we would like to show is that for all $\mathbf{w}$, if the public key is chosen at random, then there is only one possible challenge c for which there will exist such valid $\mathbf{z}_i$.

For contradiction, assume that there are two $c, c' \in \mathcal{C}$ for which there exist $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}'_1, \mathbf{z}'_2$ such that

$$\mathbf{A}\mathbf{z}_1 + \mathbf{z}_2 = \mathbf{t}c + \mathbf{w},$$
$$\mathbf{A}\mathbf{z}_1 +' \mathbf{z}'_2 = \mathbf{t}c' + \mathbf{w}.$$

Combining the two equations, we get

$$\mathbf{A}\bar{\mathbf{z}}_1 + \bar{\mathbf{z}}_2 = \mathbf{t}\bar{c}, \tag{79}$$

where $\bar{\mathbf{z}}_i = \mathbf{z}_i - \mathbf{z}'_i$ and $\bar{c} = c - c'$. We can now use an argument similar to that in Lemma 2 to conclude that with high probability, for a random $(\mathbf{A}, \mathbf{t})$ such $\bar{\mathbf{z}}_i$ with coefficients in $[2\bar{\beta}]$ do not exist.[22]

In order to set the parameter $\bar{\beta}$ such that a solution to (79) doesn't exist, with high probability, we need $\bar{\beta}$ to be somewhat less than $q^{n/(n+m)}$ (as in Lemma 2), and so β would have to be even smaller, where the relationship between the two variables is still dictated by Lemma 11. This, as in the case of the Katz-Wang scheme, results in a less efficient instantiation than the Schnorr analogue.

Comparing the Efficiency/Security. The Okamoto and Katz-Wang analogues of the lattice-based protocol introduce some constraints on the parameters β and $\bar{\beta}$ that result in a less efficient instantiation than if one were freely allowed to choose these values (subject to the relationship between them established in Lemma 11). In Figure 2, we sketched how the

[22] To use the analogy of that lemma over $\mathcal{R}_{q,f}$ instead of $\mathbb{Z}_q$, we need to make sure that the polynomials $z \in [2\bar{\beta}]$ are invertible. One can ensure this by appropriately setting the parameters of the ring $\mathcal{R}_{q,f}$ (c.f. [LS18, Corollary 1.2]).

Private information: $\mathbf{s}_1 \in [\beta]^m, \mathbf{s}_2 \in [\beta]^n$
Public information: $\mathbf{A} \in \mathcal{R}_{q,f}^{n \times m}, \mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2 \in \mathcal{R}_{q,f}^n$

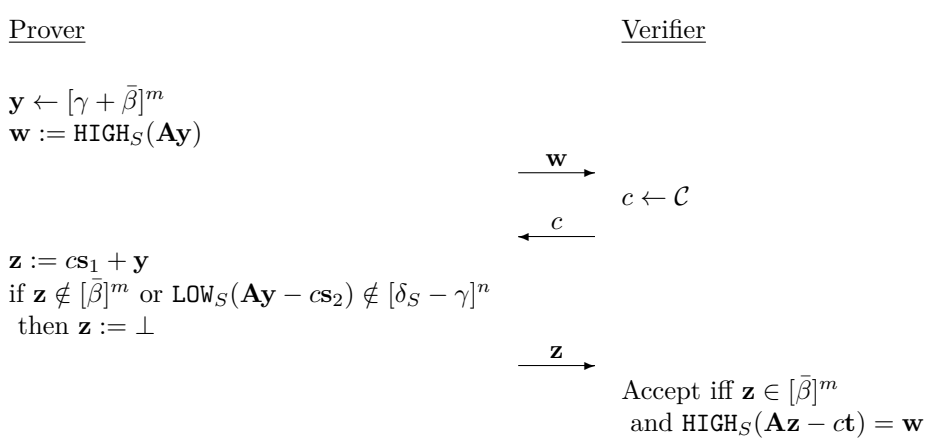

Figure 8: Basic Zero-Knowledge Proof System with a smaller output. The set $\mathcal{S} \in \mathbb{Z}_q$ has size 2^κ and the function $\mathtt{HIGH}_S, \mathtt{LOW}_S$, and the constant δ_S are defined as in the text of Section 5.4. The prover, who knows $\mathbf{s}_1 \in [\beta]^m, \mathbf{s}_2 \in [\beta]^n$ satisfying (67), produces a ZKPoK of $\bar{\mathbf{s}}_1 \in [2\bar{\beta}]^m$, $\bar{\mathbf{s}}_2 \in [q/2^\kappa]^n$, and a $\bar{c} \in \bar{\mathcal{C}}$ satisfying (68). The value γ is defined in Lemma 11, and the value of $\bar{\beta}$ affects the completeness of the protocol (i.e. the probability that $\perp$ is not sent) as specified in (84).

security of the LWE and SIS problems evolve based on the parameter β. The intersection point where $\mathsf{LWE}_{n,m,q,\beta}$ and $\mathsf{SIS}_{n,m,q,\beta}$ problems (and their polynomial versions) meet in hardness is at approximately $q^{n/(n+m)}$. The optimal setting of the parameters for the signature scheme from Figure 5 will be where β and $\bar{\beta}$ are on different sides of this intersection. In the Okamoto-like scheme, however, we need to set $\beta > q^{n/(n+m)}$ in order for the public key to not uniquely determine the secret key, which puts β and $\bar{\beta}$ on the same side. In the Katz-Wang-like scheme, we need to have $\bar{\beta} < q^{n/(n+m)}$ in order to use the information-theoretic argument, which again puts β and $\bar{\beta}$ on the same side. We pictorially demonstrate this in Figure 7, which should give the intuition for why the unconstrained Schnorr-like variant of the scheme leads to the most efficient parameters.

5.4 Reducing the Proof Size

In this section we will show how to reduce the size of the proof of the protocol in Figure 5 by essentially removing the need to send $\mathbf{z}_2$. The intuition is that to prove knowledge of (68), it's enough to output a proof corresponding to the $\bar{\mathbf{s}}_1$ such that $\mathbf{A}\bar{\mathbf{s}}_1 \approx \bar{c}\mathbf{t}$. Thus one does not, in principle, need to send the value corresponding to $\mathbf{z}_2$. One needs to be careful, though, to change the protocol so that it still remains zero-knowledge. The idea for not sending $\mathbf{z}_2$ appeared in [GLP12, BG14], and the protocol we present in Figure 8 is due to [BG14].

The idea, and execution, of not sending $\mathbf{z}_2$ is somewhat similar in spirit to the bit-dropping idea for shortening the ciphertext in Section 2.5.1. As in that section, suppose that we pick a set $\mathcal{S} \subset \mathbb{Z}_q$ of size 2^κ so that the distance between any two elements in this set is $\approx q/2^\kappa$ (see (18)). Let us also recall the notation from that section with which we can uniquely represent any $w \in \mathbb{Z}_q$ as $w = \mathtt{HIGH}_S(w) + \mathtt{LOW}_S(w)$, where $\mathtt{HIGH}_s(w) \in S$ and $\mathtt{LOW}_S(w) = w - \mathtt{HIGH}_S(w) \in [q/2^{\kappa+1}]$. This notation can be naturally extended to vectors

over $\mathcal{R}_{q,f}$ by applying this decomposition to each integer coefficient of each polynomial.

Also, define δ_S to be the largest integer such that for all 2^κ elements $s_i \in \mathcal{S}$, the sets $s_i + [\delta_S]$ are all disjoint. If we pick the points in $\mathcal{S}$ such that they are equidistant from each other (with distances varying by at most 1 since q may not be divisible by 2^κ) on the circle representing $\mathbb{Z}_q$, then δ_S will be approximately $q/2^{\kappa+1}$, which is also (again, within 1) the maximum value, over all $w \in \mathbb{Z}_q$, of $\texttt{LOW}_S(w)$. For the rest of the section, we will assume that $\mathcal{S}$ is picked in such a manner. An important simple observation is that for all positive $\gamma < \delta_S$,

$$\texttt{LOW}_S(w) \in [\delta_S - \gamma] \text{ and } s \in [\gamma] \Longrightarrow \texttt{HIGH}_S(w) = \texttt{HIGH}_S(w+s) \tag{80}$$

With the above notation, consider the protocol in Figure 8. We will show that it is a proof of knowledge of $\bar{\mathbf{s}}_1 \in [2\bar{\beta}]^m, \bar{\mathbf{s}}_2 \in [q/2^\kappa]^n, \bar{c} \in \bar{\mathcal{C}}$ satisfying (68).

5.4.1 Correctness

For correctness (in the case that $\mathbf{z} \neq \perp$), we need to show that $\texttt{HIGH}_S(\mathbf{Ay}) = \texttt{HIGH}_S(\mathbf{Az} - c\mathbf{t})$. If we write

$$\mathbf{Az} - c\mathbf{t} = \mathbf{A}(c\mathbf{s}_1 + \mathbf{y}) - c(\mathbf{As}_1 + \mathbf{s}_2) = \mathbf{Ay} - c\mathbf{s}_2, \tag{81}$$

we know by one of the Prover's conditions for not sending $\perp$ that $\texttt{LOW}_S(\mathbf{Ay} - c\mathbf{s}_2) \in [\delta_S - \gamma]^n$. The latter implies, by the observation in (80), that $\texttt{HIGH}_S(\mathbf{Ay}) = \texttt{HIGH}_S(\mathbf{Ay} - c\mathbf{s}_2)$, and therefore $\mathbf{w} = \texttt{HIGH}_S(\mathbf{Az} - c\mathbf{t})$.

5.4.2 Zero-Knowledge

As before, we will only show how to simulate transcripts in which $\perp$ is not sent. From Lemma 11, we know that conditioned on $\mathbf{z} \in [\bar{\beta}]^n$, it is uniformly random. Thus our simulation chooses a $\mathbf{z}$ uniformly random in $[\bar{\beta}]^n$ and $c \in \mathcal{C}$. He then checks whether $\texttt{LOW}_S(\mathbf{Az} - c\mathbf{t}) \in [\delta_S - \gamma]^n$. If it is not, then it resamples it resamples $\mathbf{z}$ and c and tries again. Once he is successful, he sets $\mathbf{w} := \texttt{HIGH}_S(\mathbf{Az} - c\mathbf{t})$ and outputs the view $(\mathbf{w}, c, \mathbf{z})$. Because $\mathbf{z}$ has the correct distribution after the first check passes and the second check is exactly the same in the real proof and the simulation, the simulation perfectly simulates the non-aborting transcripts.

In the above simulation, it is *crucial* that the simulator is able to perfectly simulate the real prover's check that

$$\texttt{LOW}_S(\mathbf{Ay} - c\mathbf{s}_2) \in [\delta_S - \gamma]^n. \tag{82}$$

This is indeed why this check, rather than a different and possibly less restrictive one, is performed in the real proof. Indeed, one does not need (82) to hold in order for the verification equation to be satisfied. The scheme would still be complete if the prover directly checks that the verifier will accept – i.e. simply check that $\mathbf{w} = \texttt{HIGH}_S(\mathbf{Az} - c\mathbf{t})$. But the simulator cannot perform this check because he does not know $\mathbf{w}$ ($\mathbf{w}$ is set in the simulation after setting $\mathbf{z}$ and checking (82)) and so the scheme would lose its ZK property. The check in (82) is thus necessary in order to simultaneously ensure correctness and simulatability (and thus security).

5.4.3 Computing the probability of $\perp$

Similarly to the calculation in Lemma 11, we know that $\Pr_{\mathbf{y}}\left[\mathbf{z} \in [\bar{\beta}]^m\right] = \left(\frac{2\bar{\beta}+1}{2(\bar{\beta}+\gamma)+1}\right)^{dm} \approx e^{-\gamma dm/\bar{\beta}}$ (see (75)). To compute the probability that $\texttt{LOW}_S(\mathbf{Ay} - c\mathbf{s}_2) \in [\delta_S - \gamma]^n$, we make the heuristic assumption that the distribution of $\mathbf{Ay} - c\mathbf{s}_2$ is uniform in $\mathcal{R}_{q,f}^n$, and thus

$\texttt{LOW}_S(\mathbf{Ay} - c\mathbf{s}_2)$ is uniformly distributed in $[\delta_S]^n$. Therefore the probability that a random element in $[\delta_S]^n$ is inside $[\delta_S - \gamma]^n$ is

$$\left(\frac{2(\delta_S - \gamma) + 1}{2\delta_S + 1}\right)^{dn} > \left(1 - \frac{\gamma}{\delta_S}\right)^{dn} \approx e^{-\gamma dn/\delta_S}. \tag{83}$$

Combining the two probabilities, we obtain that

$$\Pr_{\mathbf{y}}[\mathbf{z} \neq \perp] \approx e^{-\gamma d(m/\bar{\beta} + n/\delta_S)}. \tag{84}$$

Note that a larger $\bar{\beta}$ and a larger δ_S increase the correctness probability of the protocol. But as we will see below, the larger these values are, the larger the coefficients in the extracted $\bar{\mathbf{s}}_1, \bar{\mathbf{s}}_2$ satisfying (68) will be.

We point out that even though we make the heuristic assumption to calculate the probability that $\texttt{LOW}_S(\mathbf{Ay} - c\mathbf{s}_2) \in [\delta_S - \gamma]^n$, we do not need to use any heuristics to argue that the probability of outputting $\perp$ is independent of the secret key (which is needed to prevent side-channel attacks). This is because the distribution of $\mathbf{z}$ is independent of the secret key (Lemma 11), and $\texttt{LOW}_S(\mathbf{Ay} - c\mathbf{s}_2) = \texttt{LOW}_S(\mathbf{Az} - c\mathbf{t})$, and thus this is also independent of the secret key.

5.4.4 Proof of Knowledge

Via rewinding, the extractor can obtain two transcripts $(\mathbf{w}, c, \mathbf{z})$ and $(\mathbf{w}, c', \mathbf{z}')$ that satisfy the verification equations, and thus $\texttt{HIGH}_S(\mathbf{Az} - c\mathbf{t}) = \texttt{HIGH}_S(\mathbf{Az}' - c'\mathbf{t})$. By definition, we have

$$\begin{aligned}\mathbf{Az} - c\mathbf{t} &= \texttt{HIGH}_S(\mathbf{Az} - c\mathbf{t}) + \texttt{LOW}_S(\mathbf{Az} - c\mathbf{t})\\ \mathbf{Az}' - c'\mathbf{t} &= \texttt{HIGH}_S(\mathbf{Az}' - c'\mathbf{t}) + \texttt{LOW}_S(\mathbf{Az}' - c'\mathbf{t})\end{aligned}$$

where $\texttt{LOW}_S(\mathbf{Az} - c\mathbf{t}), \texttt{LOW}_S(\mathbf{Az} - c\mathbf{t}) \in [q/2^{\kappa+1}]^n \approx [\delta_S]^n$. Subtracting the two above equations, we obtain

$$\mathbf{A}(\mathbf{z} - \mathbf{z}') - (c - c')\mathbf{t} = \texttt{LOW}_S(\mathbf{Az} - c\mathbf{t}) - \texttt{LOW}_S(\mathbf{Az}' - c'\mathbf{t}) \in [q/2^{\kappa}]^n \approx [2\delta_S]^n, \tag{85}$$

which is equivalent to (68) when we set $\bar{\mathbf{s}}_2$ to $\texttt{LOW}_S(\mathbf{Az} - c\mathbf{t}) - \texttt{LOW}_S(\mathbf{Az}' - c'\mathbf{t})$.

5.5 Reducing the Public Key Size

We now continue improving the efficiency of the signature scheme by showing how to decrease the size of the public key. Similarly to the intuition for removing the need to transmit $\mathbf{z}_2$ from the protocol in Figure 5 in the previous section due to the fact that $\mathbf{Az}_1 \approx c\mathbf{t}$, we also notice that if we write $\mathbf{t} = \texttt{HIGH}_{\mathcal{T}}(\mathbf{t}) + \texttt{LOW}_{\mathcal{T}}(\mathbf{t})$ (where $\mathcal{T} \subset \mathbb{Z}_q$), then $\mathbf{Az}_1 \approx c \cdot (\texttt{HIGH}_T(\mathbf{t}) + \texttt{LOW}_T(\mathbf{t})) \approx c \cdot \texttt{HIGH}_T(\mathbf{t})$. In other words, the verifier does not need to know the low-order bits of $\mathbf{t}$ in order to *approximately* verify the verification equation.

By not making $\texttt{LOW}_{\mathcal{T}}(\mathbf{t})$ a part of the public key, we again run into a similar problem as in the previous section in that without some adjustments to the protocol, the verifier will want to compute $\texttt{HIGH}_S(\mathbf{Az} - c\mathbf{t})$ because now he doesn't even know $\mathbf{t}$, but only $\texttt{HIGH}_{\mathcal{T}}(\mathbf{t})$. We will now describe the techniques needed to make the proof work and present the protocol, from [DKL+18], in Figure 9.

Similarly to the way in which we defined the set $S \subset \mathbb{Z}_q$ in the previous section, we define a set $\mathcal{T} \subset \mathbb{Z}_q$ that consists of 2^ℓ elements each a distance of approximately $q/2^\ell$ apart. Instead of outputting the entire vector $\mathbf{t} \in \mathcal{R}_{q,f}^n$ as part of the public key, we will decompose it as $\mathbf{t} = \mathbf{t}_1 + \mathbf{t}_0$, where $\mathbf{t}_1 = \texttt{HIGH}_{\mathcal{T}}(\mathbf{t})$ and $\mathbf{t}_0 = \texttt{LOW}_{\mathcal{T}}(\mathbf{t})$. The public key

Private information: $\mathbf{s}_1 \in [\beta]^m, \mathbf{s}_2 \in [\beta]^n$
Public information: $\mathbf{A} \in \mathcal{R}_{q,f}^{n \times m}, \mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$ (not used by the verifier), $\mathbf{t}_1 = \mathtt{HIGH}_{\mathcal{T}}(\mathbf{t})$

Prover		Verifier
$\mathbf{y} \leftarrow [\gamma + \bar{\beta}]^m$ $\mathbf{w} := \mathtt{HIGH}_S(\mathbf{A}\mathbf{y})$		
	$\xrightarrow{\mathbf{w}}$	
		$c \leftarrow \mathcal{C}$
	$\xleftarrow{c}$	
$\mathbf{z} := c\mathbf{s}_1 + \mathbf{y}$ if $\mathbf{z} \notin [\bar{\beta}]^m$ or $\mathtt{LOW}_S(\mathbf{A}\mathbf{y} - c\mathbf{s}_2) \notin [\delta_S - \gamma]^n$ then $(\mathbf{z}, \mathbf{h}) := \perp$ if $c\mathbf{t}_0 \notin [\delta_S]^n$ then $(\mathbf{z}, \mathbf{h}) := \perp$ if $(\mathbf{z}, \mathbf{h}) \neq \perp$ then $\mathbf{h} := \mathtt{HINT}(\mathbf{A}\mathbf{z} - c\mathbf{t}_1, c\mathbf{t}_0)$		
	$\xrightarrow{(\mathbf{z}, \mathbf{h})}$	
		Accept iff $\mathbf{z} \in [\bar{\beta}]^m$ and $\mathtt{USEHINT}(\mathbf{A}\mathbf{z} - c\mathbf{t}_1, \mathbf{h}) = \mathbf{w}$

Figure 9: The Zero-Knowledge Proof System with a smaller proof and a smaller public key. The set $\mathcal{S} \in \mathbb{Z}_q$ has size 2^κ and the function $\mathtt{HIGH}_S, \mathtt{LOW}_S$, and the constant $\delta_S \approx q/2^{\kappa+1}$ are defined as in the text of Section 5.4. The set $\mathcal{T}$ has size 2^ℓ and we require that with high probability (over the choice of c and $\mathbf{t}$), $c \cdot \mathtt{LOW}_T(\mathbf{t}) \in [\delta_S]^n$. The functions $\mathtt{HINT}$ and $\mathtt{USEHINT}$ are defined as in the text of Section 5.5. The prover, who knows $\mathbf{s}_1 \in [\beta]^m, \mathbf{s}_2 \in [\beta]^n$ satisfying (67), produces a ZKPoK of $\bar{\mathbf{s}}_1 \in [2\bar{\beta}]^m$, $\bar{\mathbf{s}}_2 \in [q/2^{\kappa-1}]^n$, and a $\bar{c} \in \bar{\mathcal{C}}$ satisfying (88). The value γ is defined in Lemma 11, and the value of $\bar{\beta}$ affects the completeness of the protocol (i.e. the probability that $\perp$ is not sent) as specified in (84).

will only consist of $\mathbf{t}_1$, which requires $nd\ell$ bits to represent instead of $nd \log q$ required to represent the entire $\mathbf{t}$.

The place where the verifier used $\mathbf{t}$ in the protocol in Figure 8 was in the computation of $\mathbf{A}\mathbf{z} - c\mathbf{t}$. If the verifier only has $\mathbf{t}_1$, then he can compute $\mathbf{A}\mathbf{z} - c\mathbf{t}_1 = \mathbf{A}\mathbf{z} - c\mathbf{t} + c\mathbf{t}_0$. In order for verification to work out, we would need

$$\mathtt{HIGH}_S(\mathbf{A}\mathbf{z} - c\mathbf{t}) = \mathtt{HIGH}_S(\mathbf{A}\mathbf{z} - c\mathbf{t} + c\mathbf{t}_0). \tag{86}$$

At this point, the natural question is why not resolve this issue by applying the same technique as in the previous section? That is, we can make sure that $c\mathbf{t}_0 < \gamma'$ for some γ' and then use the observation in (80) to force the above equation to always hold. The problem with this approach is that it's wasteful and unlikely to result in a useful reduction in size. The reason we upper-bounded the coefficients of $c\mathbf{s}_2$ (over all possible secrets $\mathbf{s}_2$) by γ was that we needed to keep $\mathbf{s}_2$ a secret. On the other hand, we do not need to keep $\mathbf{t}_0$ a secret – we simply want it to be unnecessary for verification. So it's perfectly fine if the verifier is able to compute something that depends on $\mathbf{t}_0$ – i.e. $\mathbf{A}\mathbf{z} - c\mathbf{t} + c\mathbf{t}_0$. Our only goal here is to make sure that a verifier who knows $\mathtt{HIGH}_S(\mathbf{A}\mathbf{z} - c\mathbf{t} + c\mathbf{t}_0)$ is able to derive $\mathtt{HIGH}_S(\mathbf{A}\mathbf{z} - c\mathbf{t})$.

The main observation is that if $c\mathbf{t}_0 \in [\delta_S]^n$, then a verifier who can compute $\mathtt{HIGH}_S(\mathbf{A}\mathbf{z} - c\mathbf{t} + c\mathbf{t}_0)$ would only require one bit of extra information (per coefficient) to determine $\mathtt{HIGH}_S(\mathbf{A}\mathbf{z} - c\mathbf{t})$. In particular, if some integer point v is between points s_i and s_{i+1} in

the set $\mathcal{S}$ on the circle representing $\mathbb{Z}_q$, and we add to it an integer point $v' \in [\delta_S]$, then the closest point to $v + v'$ in S can only be s_i or s_{i+1} (i.e. $\texttt{HIGH}_S(v + v') = s_i$ or s_{i+1}). The prover, who knows, v and v' can provide this one bit of information to the verifier. In other words, he can tell the verifier whether $\texttt{HIGH}_S(\mathbf{Az} - c\mathbf{t}) = \texttt{HIGH}_S(\mathbf{Az} - c\mathbf{t} + c\mathbf{t}_0)$ or not. Either way, the verifier can now determine $\texttt{HIGH}_S(\mathbf{Az} - c\mathbf{t})$. Using this technique, one can significantly reduce the size of the public key at the cost of adding an extra dn "hint" bits to the signature.

As notation, we will write $\texttt{HINT}(\mathbf{Az} - c\mathbf{t}_1, c\mathbf{t}_0) = \mathbf{h} \in \{0,1\}^{dn}$ to be the vector of hints necessary to recover $\texttt{HIGH}_S(\mathbf{Az} - c\mathbf{t})$ from $\mathbf{Az} - c\mathbf{t}_1$. We'll write $\texttt{USEHINT}(\mathbf{Az} - c\mathbf{t}_1, \mathbf{h})$ to be this recovery procedure. In particular, $\texttt{USEHINT}(\mathbf{Az} - c\mathbf{t}_1, \mathbf{h})$ computes $\mathbf{v} = \mathbf{Az} - c\mathbf{t}_1$ and then for each integer coefficient of $\mathbf{v}$ (which lies between s_i and s_{i+1} in $\mathcal{S}$), it uses the corresponding bit of $\mathbf{h}$ to either output the closer s_i (if the hint bit is 0) or the one that is further away (if the hint bit is 1). In practice, it'll usually turn out that the majority of the time the hint bit will be 0, and so the hint vector can be more efficiently represented by just enumerating the positions where the hint bit should be 1.

With this notation, the algorithm with compressed signatures and public keys is presented in Figure 9. It will be important for the security proof to note that for any vector $\mathbf{v} \in \mathcal{R}_{q,f}^n$ and a hint vector $\mathbf{h} \in \{0,1\}^{dn}$, we have

$$\mathbf{v} - \texttt{USEHINT}(\mathbf{v}, \mathbf{h}) \in [q/2^\kappa]^n \approx [2\delta_{\mathcal{S}}]^n. \tag{87}$$

The above is directly implied by the USEHINT procedure and the fact that the distance between two neighboring points in $\mathcal{S}$ is $q/2^\kappa$.

The statement that is proved by the protocol in Figure 9 is the knowledge of $\bar{\mathbf{s}}_1 \in [2\bar{\beta}]^m$, $\bar{\mathbf{s}}_2 \in [q/2^{\kappa-1}]^n$, and $\bar{c} \in \bar{\mathcal{C}}$ satisfying

$$\mathbf{A}\bar{\mathbf{s}}_1 + \bar{\mathbf{s}}_2 = \bar{c}\mathbf{t}_1. \tag{88}$$

If we would like to relate this to the original $\mathbf{t}$ and satisfy (68), we can substitute $\mathbf{t}_1 = \mathbf{t} - \mathbf{t}_0$ and rewrite the above equation as $\mathbf{A}\bar{\mathbf{s}}_1 + (\bar{\mathbf{s}}_2 + \bar{c}\mathbf{t}_0) = \bar{c}\mathbf{t}$ and so the length of the vector $\bar{\mathbf{s}}_2$ gets increased by the maximum possible value (over all $\bar{c} \in \bar{\mathcal{C}}$) of $\bar{c}\mathbf{t}_0$.

5.5.1 Correctness and Zero-Knowledge

An important point that is worth repeating is that while the verifier does not need the value of $\mathbf{t}_0 = \texttt{LOW}_{\mathcal{T}}(\mathbf{t})$ for verification, one *should not* consider the value $\mathbf{t}_0$ to be secret because outputting $\mathbf{h}$ leaks some information about $\mathbf{t}_0$. The way to think about $\mathbf{t}$ for the proofs is that the verifier knows the entire $\mathbf{t}$, but only uses $\mathbf{t}_1$ in verification. The zero-knowledge property of the scheme immediately follows from the zero-knowledge property of the scheme in Figure 8, which we already established in Section 5.4.2, because the only difference in the prover's output is the construction of the hint $\mathbf{h}$, which can be done knowing $\mathbf{z}$ and $\mathbf{t}$.

The correctness of the scheme follows whenever $c\mathbf{t}_0 \in [\delta_S]^n$. Note that it doesn't affect the security of the scheme if $\mathbf{t}_0$ is chosen such that $c\mathbf{t}_0$ is sometimes not in $[\delta_S]^n$ – though this will require additional restarts on the part of the prover.

5.5.2 Proof of Knowledge

Via rewinding, one obtains the transcripts $(\mathbf{w}, c, (\mathbf{z}, \mathbf{h}))$ and $(\mathbf{w}, c', (\mathbf{z}', \mathbf{h}'))$, and we thus have

$$\texttt{USEHINT}(\mathbf{Az} - c\mathbf{t}_1, \mathbf{h}) = \texttt{USEHINT}(\mathbf{Az}' - c'\mathbf{t}_1, \mathbf{h}'). \tag{89}$$

From (87), we know that

$$\mathbf{Az} - c\mathbf{t}_1 - \texttt{USEHINT}(\mathbf{Az} - c\mathbf{t}_1, \mathbf{h}) \in [q/2^\kappa]^n$$
$$\mathbf{Az}' - c'\mathbf{t}_1 - \texttt{USEHINT}(\mathbf{Az}' - c'\mathbf{t}_1, \mathbf{h}) \in [q/2^\kappa]^n,$$

Private information: $\mathbf{s}_1 \leftarrow [\beta]^m, \mathbf{s}_2 \leftarrow [\beta]^n$
Public information: $\mathbf{A} \in \mathcal{R}_{q,f}^{n \times m}, \mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$ (not used by the verifier), $\mathbf{t}_1 = \texttt{HIGH}_{\mathcal{T}}(\mathbf{t})$

Signer		Verifier
$\mathbf{y} \leftarrow [\gamma + \bar{\beta}]^m$ $c := \mathcal{H}(\texttt{HIGH}_S(\mathbf{A}\mathbf{y}), \mu) \in \mathcal{C}$ $\mathbf{z} := c\mathbf{s}_1 + \mathbf{y}$ if $\mathbf{z} \notin [\bar{\beta}]^m$ or $\texttt{LOW}_S(\mathbf{A}\mathbf{y} - c\mathbf{s}_2) \notin [\delta_S - \gamma]^n$ then RESTART if $c\mathbf{t}_0 \notin [\delta_{\mathcal{S}}]^n$, then RESTART $\mathbf{h} := \texttt{HINT}(\mathbf{A}\mathbf{z} - c\mathbf{t}_1, c\mathbf{t}_0)$	$\xrightarrow{(\mathbf{z}, \mathbf{h})}$	Accept iff $\mathbf{z} \in [\bar{\beta}]^m$ and $\mathcal{H}(\texttt{USEHINT}(\mathbf{A}\mathbf{z} - c\mathbf{t}_1, \mathbf{h}), \mu) = c$

Figure 10: Digital Signature Scheme obtained as a result of applying the Fiat-Shamir transform to the protocol in Figure 9 that signs a message (digest) μ.

which together with (89) implies that

$$\mathbf{A}(\mathbf{z} - \mathbf{z}') - (c - c')\mathbf{t}_1 \in [q/2^{\kappa-1}]^n \approx [4\delta_{\mathcal{S}}]^n, \tag{90}$$

which in turn directly implies knowledge of $\bar{\mathbf{s}}_1, \bar{\mathbf{s}}_2$, and $\bar{c}$ as in (88).

5.6 Digital Signatures

The signing procedure in Figure 10 is the Fiat-Shamir transform of the protocol in Figure 9, where the secret keys $\mathbf{s}_1, \mathbf{s}_2$ are chosen uniformly at random from their respective domains. An important point, which we previously mentioned, is that because the protocol is no longer interactive, there is never a need for the prover to send $\perp$ – he can simply keep restarting the protocol until the rejection sampling step is successful. This is the reason that we only needed to prove the zero-knowledge property only in the case that $\perp$ is not output.

The correctness and the zero-knowledge properties of the protocol follow directly from these respective properties of the interactive protocol in Figure 9. The security follows from the security of the identification scheme. Via the generic properties of the Fiat-Shamir transform, we know that one can extract from a successful signer the same thing as from a successful prover in Section 5.5.2 – $\bar{\mathbf{s}}_1, \bar{\mathbf{s}}_2$, and $\bar{c}$ satisfying (88). Applying Lemma 9 then implies that extracting these values is as hard as Ring-LWE or Ring-SIS.

5.7 The Signature Scheme CRYSTALS-Dilithium (ML-DSA)

We now give a sample instantiation for a digital signature scheme very much resembling [DKL+18], which is (conservatively) estimated to have 192 bits of security.

We will be working over the ring $\mathcal{R}_{q,f}$ for $f = X^{256} + 1$ and $q = 2^{23} - 2^{13} + 1$. This choice of prime q allows for efficient NTT, since $q \equiv 1 \pmod{512}$, as described in Section 4.5.2. The challenge set $\mathcal{C}$ consists of polynomials with $0, \pm 1$ coefficients with exactly 207 $0's$ (and thus 49 non-zeros) and the secrets $\mathbf{s}_1, \mathbf{s}_2$ have coefficients randomly chosen from $[\beta]$ for $\beta = 4$.

Table 4: Sample parameters for a digital signature scheme very similar to the NIST Level 3 parameter set (i.e. one that should be as hard as AES-192) CRYSTALS-Dilithium [DKL+18].

q	$2^{23} - 2^{13} + 1$
$f(X)$	$X^{256} + 1$
β	4
(n, m)	$(6, 5)$
$\mathcal{C}$	$c \in [1]$ with 49 $\pm 1's$ and 207 $0's$
γ	$49 * 4 = 196$
$\bar{\beta}$	$2^{19} - \gamma - 1$
$\mathcal{S}$	$\{i \cdot (q-1)/16 \mid 0 \le i \le 15\}$
$\delta_{\mathcal{S}}$	$(q-1)/32 - 1$
$\mathcal{T}$	$\{i \cdot 2^{13} \mid 0 \le i \le (q-1)/2^{13}\}$

The set $\mathcal{S}$ and $\mathcal{T}$ are then defined as in Table 4. The definition of set $\mathcal{S}$ implies that every point in $\mathbb{Z}_q$ is at most $(q-1)/32 + 1$ away from any element in $\mathcal{S}$. Similarly, the definition of $\mathcal{T}$ implies that all the coefficients of $\mathbf{t}_0$ are in $[2^{12}]$. One can now check that $c\mathbf{t}_0$ has coefficients in $[\delta_{\mathcal{S}}]$ with high probability and so the scheme will be correct (with high probability). To make the scheme always correct, the prover should additionally check that $c\mathbf{t}_0 \in [\delta_{\mathcal{S}}]^6$, and we set the parameters so that it happens with a very small probability ($< 1\%$)[23]

The probability that a restart does not occur is computed from (84) as approximately

$$e^{-196 \cdot 256 \cdot (5/2^{19} + 6 \cdot 32/(q-1))} \approx 0.2,$$

which means that one needs, on average, about 5 signing attempts before a signature is produced. This makes the signing procedure noticeably slower than verification. Still, optimized implementations of the signing procedure runs in well under a millisecond on a standard personal computer.

The public key consists of a 256-bit seed ρ that is used in the expansion of the public matrix $\mathbf{A}$ and the vector $\mathbf{t}_1 = \texttt{HIGH}_{\mathcal{T}}(\mathbf{A}\mathbf{s}_1 + \mathbf{s}_2)$. Since the set $\mathbf{S}$ can be described with 10 bits, and there are $6 \cdot 256$ integer elements in $\mathbf{t}_1$, the description of $\mathbf{t}_1$ requires $6 \cdot 256 \cdot 10$ bits. Thus the public key consists of 1952 bytes.

The signature consists of the vector $\mathbf{z}_1$, the challenge c, and the hint vector $\mathbf{h}$. Since the coefficients of $\mathbf{z}_1$ are in $[\beta]$, its representation requires 20 bits per coefficients for a total of $256 \cdot 5 \cdot 20$ bits. The challenge c consists of approximately 256 bits, while the hint vector $\mathbf{h}$ is a binary vector of dimension $256 \cdot 6$, and so requires at most that many bits to represent. Thus the total signature size is around 3424 bytes.

Some Optimizations. We now discuss some small optimizations to the protocol in Figure 10 that are used in the ML-DSA (Dilithium) NIST standard. If the message μ is very long, then the computation $\mathcal{H}(\texttt{HIGH}_S(\mathbf{A}\mathbf{y}), \mu)$ done after every restart will be needlessly inefficient. For this reason, it makes sense to first hash the real message μ' using SHA-512 to obtain a 512-bit digest μ, and only use this digest in the signing process.

In order to make the sampling of $\mathbf{y}$ as efficient as possible (since it also gets sampled once with every restart), one can make the range in which each coefficient is sampled a power of 2, so exactly 20 bits. So instead of sampling from $[\gamma + \bar{\beta}] = [2^{19} - 1]$, where each

[23] In fact, it never happens for the parameter set given in Table 4 because $\|\mathbf{t}_0\|_\infty \le 2^{12}$ and $\|c\|_1 = 49$, and so $\|c\mathbf{t}_0\|_\infty \le 49 \cdot 2^{12} < \delta_{\mathcal{S}}$.

coefficient comes from a domain of size $2^{20}-1$, we would instead sample each coefficient from the set $\{-(2^{19}-1),\ldots,2^{19}-1,2^{19}\}$.

Another optimization mentioned above is to send a compact representation of the hint vector $\mathbf{h}$. Because $\mathbf{h}$ consists of mostly zeros (with high probability for the parameters in Table 4, it will have at most 55 ones, and thus $256\cdot 6-55$ zeros. Instead of sending a $256\cdot 6$-bit string, one can instead specify the positions of the 0's within the polynomial (which takes 8 bits per non-zero coefficient, so $8\cdot 55$ bits) and also specify the boundaries between the polynomials, which requires $5\cdot 6=30$ bits, for a total of 470 bits instead of the naive $256\cdot 8=1536$, which reduces the signature size to about 3290 bytes. This will require the signer to perform a restart in case the number of non-zero entries in $\mathbf{h}$ is greater than 55, but experimentally, this occurs with a very small probability and does not really affect the run-time.

Security. As previously mentioned (see Section 3.3), the security of the signature scheme relies on both the $\mathcal{R}_{q,f}$-$\mathsf{SIS}_{n,m,\beta}$ and $\mathcal{R}_{q,f}$-$\mathsf{LWE}_{n,m,\beta}$ problems. The $\mathcal{R}_{q,f}$-$\mathsf{LWE}_{n,m,\beta}$ assumption is used to prove that the public key $(\mathbf{A}.\mathbf{t})$ is indistinguishable from uniform, and the $\mathcal{R}_{q,f}$-$\mathsf{SIS}_{n,m,\beta}$ assumption is used to show that an adversary cannot forge a signature and thus find a solution for the $\mathcal{R}_{q,f}$-$\mathsf{SIS}_{n,m,\beta}$ problem in (88) where $\bar{\mathbf{s}}_1$ has coefficients in $[2\cdot\bar{\beta}]$ and $\bar{\mathbf{s}}_2$ has coefficients in $[4\delta_{\mathcal{S}}]$, where both sets are approximately $[2^{20}]$ for the parameters in Table 4. This therefore corresponds to the correspond to the middle row of the parameter set in Table 2 and one can see that the value of δ, which controls the hardness of the problems, is roughly similar for both $\mathcal{R}_{q,f}$-$\mathsf{LWE}_{n,m,\beta}$ and $\mathcal{R}_{q,f}$-$\mathsf{SIS}_{n,m,\beta}$.

References

[ABD16] Martin R. Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes. In *CRYPTO (1)*, volume 9814 of *Lecture Notes in Computer Science*, pages 153–178. Springer, 2016.

[ACD+18] Martin R. Albrecht, Benjamin R. Curtis, Amit Deo, Alex Davidson, Rachel Player, Eamonn W. Postlethwaite, Fernando Virdia, and Thomas Wunderer. Estimate all the {LWE, NTRU} schemes! In *SCN*, volume 11035 of *Lecture Notes in Computer Science*, pages 351–367. Springer, 2018.

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO*, pages 595–618, 2009.

[ADH+19] Martin R. Albrecht, Léo Ducas, Gottfried Herold, Elena Kirshanova, Eamonn W. Postlethwaite, and Marc Stevens. The general sieve kernel and new records in lattice reduction. In *EUROCRYPT (2)*, volume 11477 of *Lecture Notes in Computer Science*, pages 717–746. Springer, 2019.

[ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In *USENIX Security Symposium*, pages 327–343. USENIX Association, 2016.

[ADRS15] Divesh Aggarwal, Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. Solving the shortest vector problem in 2^n time using discrete gaussian sampling: Extended abstract. In *STOC*, pages 733–742. ACM, 2015.

[AG11] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In *ICALP (1)*, pages 403–415, 2011.

[Age24] National Security Agency. The commercial national security algorithm suite 2.0 and quantum computing faq. Technical report, 2024. https://media.defense.gov/2022/Sep/07/2003071836/-1/-1/0/CSI_CNSA_2.0_FAQ_.PDF.

[AGV09] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 474–495. Springer, 2009.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *STOC*, pages 99–108, 1996.

[AKPW13] Joël Alwen, Stephan Krenn, Krzysztof Pietrzak, and Daniel Wichs. Learning with rounding, revisited - new reduction, properties and applications. In *CRYPTO (1)*, volume 8042 of *Lecture Notes in Computer Science*, pages 57–74. Springer, 2013.

[AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *STOC*, pages 601–610. ACM, 2001.

[AM18] Divesh Aggarwal and Priyanka Mukhopadhyay. Improved algorithms for the shortest vector problem and the closest vector problem in the infinity norm. In *ISAAC*, volume 123 of *LIPIcs*, pages 35:1–35:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

[AS18] Divesh Aggarwal and Noah Stephens-Davidowitz. Just take the average! an embarrassingly simple 2ˆn-time algorithm for SVP (and CVP). In *SOSA@SODA*, volume 61 of *OASICS*, pages 12:1–12:19. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

[BCD+16] Joppe W. Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! practical, quantum-secure key exchange from LWE. In *ACM Conference on Computer and Communications Security*, pages 1006–1018. ACM, 2016.

[BDK+18] Joppe W. 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 *2018 IEEE European Symposium on Security and Privacy, EuroS&P*, pages 353–367, 2018.

[BG14] Shi Bai and Steven D. Galbraith. An improved compression technique for signatures based on learning with errors. In *CT-RSA*, pages 28–47, 2014.

[BGM+16] Andrej Bogdanov, Siyao Guo, Daniel Masny, Silas Richelson, and Alon Rosen. On the hardness of learning with rounding over small modulus. In *TCC (A1)*, volume 9562 of *Lecture Notes in Computer Science*, pages 209–224. Springer, 2016.

[BP02] Mihir Bellare and Adriana Palacio. GQ and schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 162–177. Springer, 2002.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 719–737. Springer, 2012.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *CRYPTO*, pages 505–524, 2011.

[CHK+21] Chi-Ming Marvin Chung, Vincent Hwang, Matthias J. Kannwischer, Gregor Seiler, Cheng-Jhih Shih, and Bo-Yin Yang. NTT multiplication for ntt-unfriendly rings new speed records for saber and NTRU on cortex-m4 and AVX2. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(2):159–188, 2021.

[CJL16] Jung Hee Cheon, Jinhyuck Jeong, and Changmin Lee. An algorithm for ntru problems and cryptanalysis of the ggh multilinear map without a low-level encoding of zero. *LMS Journal of Computation and Mathematics*, 19(A):255–266, 2016.

[DDLL13] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In *CRYPTO (1)*, pages 40–56, 2013.

[Den02] Alexander W. Dent. A designer's guide to kems. *IACR Cryptology ePrint Archive*, 2002. http://eprint.iacr.org/2002/174.

[DFMS19] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Security of the fiat-shamir transformation in the quantum random-oracle model. In *CRYPTO (2)*, volume 11693 of *Lecture Notes in Computer Science*, pages 356–383. Springer, 2019.

[DKL+18] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):238–268, 2018.

[DM13] Nico Döttling and Jörn Müller-Quade. Lossy codes and a new variant of the learning-with-errors problem. In *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 18–34. Springer, 2013.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO*, pages 537–554, 1999.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

[GLP12] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In *CHES*, pages 530–547, 2012.

[GN08] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In *EUROCRYPT*, pages 31–51, 2008.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the fujisaki-okamoto transformation. In *TCC*, pages 341–371, 2017.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In *ANTS*, pages 267–288, 1998.

[IN96] Russell Impagliazzo and Moni Naor. Efficient cryptographic schemes provably as secure as subset sum. *J. Cryptology*, 9(4):199–216, 1996.

[IZ89] Russell Impagliazzo and David Zuckerman. How to recycle random bits. In *FOCS*, pages 248–253. IEEE Computer Society, 1989.

[KF17] Paul Kirchner and Pierre-Alain Fouque. Revisiting lattice attacks on overstretched NTRU parameters. In *EUROCRYPT (1)*, volume 10210 of *Lecture Notes in Computer Science*, pages 3–26, 2017.

[KLS18] Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of fiat-shamir signatures in the quantum random-oracle model. In *EUROCRYPT (3)*, volume 10822 of *Lecture Notes in Computer Science*, pages 552–586. Springer, 2018.

[KW03] Jonathan Katz and Nan Wang. Efficiency improvements for signature schemes with tight security reductions. In *CCS*, pages 155–164. ACM, 2003.

[LLL82] Arjen Lenstra, Hendrik Lenstra Jr., and Laszlo Lovasz. Factoring polynomials with rational coefficients. *Mathematische Annalen*, (261):513–534, 1982.

[LM06] Vadim Lyubashevsky and Daniele Micciancio. Generalized compact knapsacks are collision resistant. In *ICALP (2)*, pages 144–155, 2006.

[LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for lwe-based encryption. In *CT-RSA*, pages 319–339, 2011.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, pages 1–23, 2010.

[LPR13a] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 60(6):43, 2013. Preliminary version appeared in EUROCRYPT 2010.

[LPR13b] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-lwe cryptography. In *EUROCRYPT*, pages 35–54, 2013.

[LPS10] Vadim Lyubashevsky, Adriana Palacio, and Gil Segev. Public-key cryptographic primitives provably as secure as subset sum. In *TCC*, pages 382–400, 2010.

[LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptography*, 75(3):565–599, 2015.

[LS18] Vadim Lyubashevsky and Gregor Seiler. Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs. In *EUROCRYPT (1)*, volume 10820 of *Lecture Notes in Computer Science*, pages 204–224. Springer, 2018.

[LS19] Vadim Lyubashevsky and Gregor Seiler. NTTRU: truly fast NTRU using NTT. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(3):180–201, 2019.

[Lyu09] Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In *ASIACRYPT*, pages 598–616, 2009.

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT*, pages 738–755, 2012.

[LZ19] Qipeng Liu and Mark Zhandry. Revisiting post-quantum fiat-shamir. In *CRYPTO (2)*, volume 11693 of *Lecture Notes in Computer Science*, pages 326–355. Springer, 2019.

[MG02] Daniele Micciancio and Shafi Goldwasser. *Complexity of lattice problems - a cryptograhic perspective*, volume 671 of *The Kluwer international series in engineering and computer science*. Springer, 2002.

[Mic07] Daniele Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. *Computational Complexity*, 16(4):365–411, 2007.

[Mic19] Daniele Micciancio. Lattices: Algorithms and applications, 2019. `http://cseweb.ucsd.edu/classes/fa17/cse206A-a/`.

[MP13] Daniele Micciancio and Chris Peikert. Hardness of sis and lwe with small parameters. In *CRYPTO (1)*, pages 21–39, 2013.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007.

[MR08] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors, *Chapter in Post-quantum Cryptography*, pages 147–191. Springer, 2008.

[NIS23a] NIST. Module-lattice-based digital signature standard. Technical report, 2023.

[NIS23b] NIST. Module-lattice-based key-encapsulation mechanism standard. Technical report, 2023.

[Oka92] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *CRYPTO*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53. Springer, 1992.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *STOC*, pages 333–342, 2009.

[Pei16] Chris Peikert. A decade of lattice cryptography. *Foundations and Trends in Theoretical Computer Science*, 10(4):283–424, 2016.

[PFH+17] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, , and Zhenfei Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2017. https://csrc.nist.gov/projects/post-quantum-cryptography/ round-1-submissions.

[PR06] Chris Peikert and Alon Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *TCC*, pages 145–166, 2006.

[PRS17] Chris Peikert, Oded Regev, and Noah Stephens-Davidowitz. Pseudorandomness of ring-lwe for any ring and modulus. In *STOC*, pages 461–473. ACM, 2017.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009.

[Sch89] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO*, pages 239–252, 1989.

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.

[SS11] Damien Stehlé and Ron Steinfeld. Making NTRU as secure as worst-case problems over ideal lattices. In *EUROCRYPT*, pages 27–47, 2011.

[SXY18] Tsunekazu Saito, Keita Xagawa, and Takashi Yamakawa. Tightly-secure key-encapsulation mechanism in the quantum random oracle model. In *EUROCRYPT (3)*, volume 10822 of *Lecture Notes in Computer Science*, pages 520–551. Springer, 2018.