Introduction to Lattice-based Cryptography

Dmytro Zakharov

Abstract

In recent years, the interest in lattice-based cryptography has been steadily increasing. One primary reason is, of course, that problems related to lattices and their derivatives are widely believed to be *hard even for quantum computers*. Furthermore, there are settings where the algebraic structure of lattices is natural: e.g., for fully-homomorphic encryption.

This brief paper is a technical introduction to lattice-based cryptography. We give formal definitions of lattices and introduce key related hardness assumptions such as SVP, CVP, and several variants of LWE. We show how to build public-key encryption (PKE) and digital signature schemes based on LWE assumptions and partially prove the security of key constructions. Finally, we provide a high-level specification of Kyber PKE and Dilithium signature schemes.

We aim to make these notes accessible while preserving rigorousness where appropriate. Basic knowledge of cryptography and algebra is required.

Contents

1 Lattices

Notation and Basic Notions

All vectors are written in bold (e.g., $\mathbf{x}$ or $\mathbf{y}$) and considered *column* vectors by default. All real vector spaces are Euclidean: that is, for every $\mathbf{x}, \mathbf{y} \in V \subseteq \mathbb{R}^d$, we define the inner product $\langle \cdot, \cdot \rangle$ and ℓ_2 norm $\| \cdot \|$ in a standard fashion:

$$\langle \mathbf{x}, \mathbf{y} \rangle \triangleq \sum_{i \in [d]} x_i y_i, \quad \|\mathbf{x}\| \triangleq \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle},$$

where we use $[d]$ to denote the set $\{1, \dots, d\}$. Additionally, we use (d) to denote the ranging set using 0-indexed notation: $\{0, \dots, d-1\}$. Other norms will be introduced on the go when needed. Finally, we use notation $x \leftarrow\!\$\ \mathcal{X}$ to denote sampling from the distribution $\mathcal{X}$. If $\mathcal{X}$ is a set, the underlying distribution is uniform over $\mathcal{X}$.

1.1 Definition

We start our discussion with the definition of the lattice. Following [dBvW25], we provide two definitions: one is rather algebraic, while the other is more intuitive and practical. In fact, both definitions will be helpful since some facts are sometimes easier to prove using the first definition.

Definition 1.1. Let V be a real Euclidean vector space of finite dimension d. The following two definitions of **lattice** $\Lambda \subseteq V$ of rank $r \leq d$ are equivalent:

(i) Λ is a discrete additive subgroup of V (with $\Lambda \cong \mathbb{Z}^r$ as $\mathbb{Z}$-modules).

(ii) $\Lambda = \mathcal{L}(B) \triangleq \{B\mathbf{z} : \mathbf{z} \in \mathbb{Z}^d\}$ for some matrix $B \in \mathbb{R}^{r \times d}$ where its columns, further denoted as $\{\mathbf{b}_1, \dots, \mathbf{b}_r\} \subseteq \mathbb{R}^d$, are linearly independent over V.

While the second definition is clear, the first one requires some deciphering. Since we work in the Euclidean space, it makes sense to define what it means for the set $S \subseteq V$ to be *discrete*. Intuitively, it should mean that any two points $\mathbf{s}_1, \mathbf{s}_2 \in S$ are separated by some "considerable" distance. To translate it into the mathematical language, we require that:

$$\inf_{\mathbf{s}_1 \neq \mathbf{s}_2 \in S} \|\mathbf{s}_1 - \mathbf{s}_2\| > 0.$$

Besides Λ being discrete, additive subgroup structure of Λ guarantees that Λ is closed under any $\mathbb{Z}$-linear combinations: that is, for any $\mathbf{x}, \mathbf{y} \in \Lambda$ and $\lambda, \mu \in \mathbb{Z}$, we have $\lambda\mathbf{x} + \mu\mathbf{y} \in \Lambda$. Showing this property might sometimes be easier than finding the matrix B in definition *(ii)* explicitly.

Geometric Intuition. While definition *(i)* is useful, the definition *(ii)* gives an intuitive way of thinking of lattice $\Lambda = \mathcal{L}(B)$. If vector space V is typically thought as a set of all $\mathbb{R}$-linear combinations* $\sum_{i \in [d]} \mathbf{e}_i \mathbb{R}$ over some basis $\{\mathbf{e}_i\}_{i \in [d]}$, the lattice Λ can be thought as all $\mathbb{Z}$-linear combinations $\sum_{i \in [r]} \mathbf{b}_i \mathbb{Z}$ over real basis $\{\mathbf{b}_i\}_{i \in [r]} \subseteq \mathbb{R}^d$. See Figure 1.1 for illustration.

Remark 1.2. Note that the basis of lattice Λ is not required to lie in $\mathbb{Z}^d$, compared to coefficients of the linear combination. However, in practice they usually do: note that we still can represent basis in memory only in the rational form (that is, $\mathbf{b}_i \in \mathbb{Q}^d$). With the proper rescaling, representation in $\mathbb{Z}^d$ would suffice.

Remark 1.3. In this section we consider *full-rank* lattices only: that is, $\Lambda \cong \mathbb{Z}^d$ for $d = \dim(V)$.

*We further use notation $S + S' := \{s + s' : s \in S, s' \in S'\}$ and for $a \in S$, set $aS := \{as : s \in S\}$.

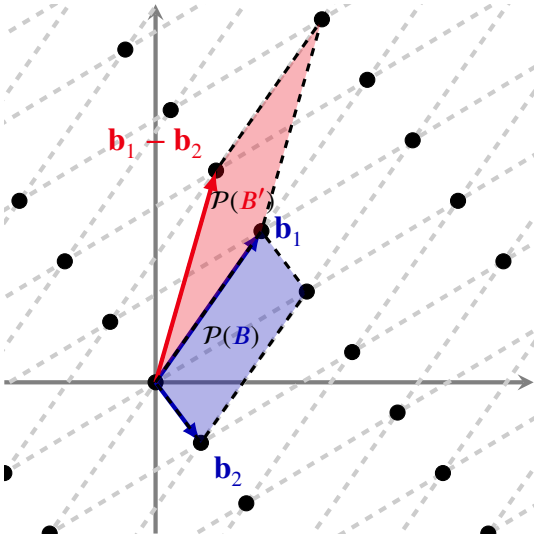

Figure 1.1: Lattice $\Lambda = \mathcal{L}(B)$ of rank 2 with basis $B = \{\mathbf{b}_1, \mathbf{b}_2\}$. B is an example of a "good" basis where basis vectors are almost orthogonal. The basis $B' = \{\mathbf{b}_1, \mathbf{b}_1 - \mathbf{b}_2\}$ of the same lattice $\Lambda = \mathcal{L}(B')$ is "worse" (the vectors seem more parallel and fundamental domain $\mathcal{P}(B')$ is skewed).

1.2 Fundamental Measures

1.2.1 Fundamental Domain

A basis B of $\Lambda = \mathcal{L}(B)$ is associated with the fundamental domain $\mathcal{P}(B) \triangleq \sum_{i\in[d]} \mathbf{b}_i[0,1)$[†].

What is special about this parallelepiped is that it represents all elements in the quotient group V/Λ: any element $\mathbf{x} \in V$ can be uniquely written as $\mathbf{x} = \mathbf{v} + \mathbf{p}$ where $\mathbf{v} \in \Lambda$ and $\mathbf{p} \in \mathcal{P}(B)$. Moreover, the shape of this $\mathcal{P}(B)$ determines the "quality" of the basis B: if $\mathcal{P}(B)$ is skewed and is not concentrated in the origin, it is typically considered "bad" as some computational problems become much harder to solve. In turn, a non-skewed basis B' primarily concentrated around the origin is considered "good" as $\{\mathbf{b}'_i\}_{i\in[d]}$ much more closely resembles the orthogonal basis than one of B (and of course, the orthogonal basis in a sense is "perfect" because it is very "nice" to work with). Again, see Figure 1.1 for illustration.

Another natural quantity to consider is the volume of such parallelepiped, called *covolume*.

Definition 1.4 (Covolume)**.** The **covolume** $\mathrm{Vol}(\Lambda)$ of lattice Λ is defined as:

$$\mathrm{Vol}(\Lambda) \triangleq \mathrm{Vol}(V/\Lambda) = \sqrt{|\det(B^\top B)|}.$$

Note that the definition of $\mathcal{P}(B)$ depends on the choice of B since the basis of Λ is generally not unique. Thus, it might seem that covolume should also depend on the choice of B. However, we show the following.

Proposition 1.5. The **covolume** $\mathrm{Vol}(\Lambda)$ is an invariant of Λ: that is, it does not depend on the choice of B.

Proof. Suppose $\Lambda = \mathcal{L}(B_0) = \mathcal{L}(B_1)$. Since B_0 and B_1 both generate Λ, one has $B_1 = U B_0$ for

[†]Sometimes, the interval $[-1/2, 1/2)$ is used instead of $[0,1)$ to center $\mathcal{P}(B)$ with respect to origin: in such case, $\mathcal{P}(B)$ is called a *centered parallelepiped of* B

some matrix $U \in \mathrm{GL}_d(\mathbb{Z})$. Since U is an integer matrix with integer inverse, it is unimodular: that is, $|\det U| = 1$. It is then easy to see $|\det B_1^\top B_1| = |\det B_0^\top U^\top U B_0| = |\det B_0^\top B_0| = \mathrm{Vol}(\Lambda)^2$. □

1.2.2 Successive Minima

Turns out that one of the hardest computational problems is computing the shortest vectors of Λ. However, we need some preliminary work to formalize this statement.

Denote $B_\rho(\mathbf{x}_0) := \{\mathbf{x} \in V : \|\mathbf{x} - \mathbf{x}_0\| < \rho\}$ — an open ball in V of radius ρ centered at $\mathbf{x}_0$.

Definition 1.6 (Successive minima)**.** The i^{th} **minimum** $\lambda_i(\Lambda)$ is the radius of the smallest sphere centered at the origin containing i linearly independent lattice vectors:

$$\lambda_i(\Lambda) \triangleq \inf\{\rho \in \mathbb{R}_{\geq 0} : \dim(\mathrm{span}(\Lambda \cap B_\rho(\mathbf{0}))) \geq i\}.$$

In particular, $\lambda_1(\Lambda) = \min_{\mathbf{x}\neq\mathbf{y}\in\Lambda} \|\mathbf{x} - \mathbf{y}\| = \min_{\mathbf{x}\in\Lambda\setminus\{\mathbf{0}\}} \|\mathbf{x}\|$ is called the *first minimum* while the vector of length $\lambda_1(\Lambda)$ — the shortest vector. The natural question to ask though is whether it exists at all: after all, in $\mathbb{R}^d$, for instance, it is clearly undefined. We give affirmative answer.

Gram-Schmidt Orthogonalization*

This is probably a good place to remind the reader of the Gram-Schmidt Orthogonalization. Recall that the main goal of Gram-Schmidt Orthogonalization is, based on sequence of vectors $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$, produce orthogonal vectors $\{\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*\}$ such that $\mathrm{span}\{\mathbf{b}_i\}_{i\in[n]} = \mathrm{span}\{\mathbf{b}_i^*\}_{i\in[n]}$.

For that, set $\mathbf{b}_1^* := \mathbf{b}_1$. Now we shall produce the vector $\mathbf{b}_2^*$ that is orthogonal to $\mathbf{b}_1$. For that, we set $\mathbf{b}_2^*$ to the component of $\mathbf{b}_2$ that is orthogonal to $\mathbf{b}_1$. Recall that the angle θ between $\mathbf{b}_1$ and $\mathbf{b}_2$ can be found as $\cos\theta = \langle \mathbf{b}_1, \mathbf{b}_2\rangle / (\|\mathbf{b}_1\| \cdot \|\mathbf{b}_2\|)$. The component of $\mathbf{b}_2$ parallel to $\mathbf{b}_1$ can be thus found as $\widehat{\mathbf{b}}_1 \cdot \|\mathbf{b}_2\| \cos\theta = \langle \mathbf{b}_1, \mathbf{b}_2\rangle \mathbf{b}_1 / \|\mathbf{b}_1\|^2$ and so $\mathbf{b}_2^* := \mathbf{b}_1 - \langle \mathbf{b}_1, \mathbf{b}_2\rangle \mathbf{b}_1 / \|\mathbf{b}_1\|^2$.

In a similar fashion, one obtains the general relations for the rest of vectors:

$$\mathbf{b}_i^* := \mathbf{b}_i - \sum_{j=0}^{i-1} \mu_{i,j} \mathbf{b}_j^*, \quad \mu_{i,j} := \frac{\langle \mathbf{b}_i, \mathbf{b}_j^*\rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^*\rangle}.$$

We shall further denote the result of Gram-Schmidt orthogonalization of B by B^*. The question that might naturally arise: given such orthogonalization, can't we then always turn a "bad" basis of lattice Λ into a "good" basis? Unfortunately, typically $\mathcal{L}(B) \neq \mathcal{L}(B^*)$: the orthogonalization of a lattice basis is (typically) not a lattice basis. However, the idea of Gram-Schmidt orthogonalization lies in the core of many attacks on lattices. Besides, it has several nice properties which we list below:

Proposition 1.7. Suppose $\Lambda = \mathcal{L}(B)$ is a lattice and B^* is the Gram-Schmidt orthogonalization of B with columns $\{\mathbf{b}_j^*\}_{j\in[d]}$. Then:

(a) $\mathrm{Vol}(\Lambda) = \prod_{j=1}^d \|\mathbf{b}_j^*\|$.

(b) $\lambda_1(\Lambda) \geq \min_{j\in[d]} \|\mathbf{b}_j^*\| > 0$.

Proof.

Part (a). Note that $\det(B) = \det(B^*)$: the process of Gram-Schmidt orthogonalization only comprises adding one row to another with some factor which does not change the determinant. Then, $\mathrm{Vol}(\Lambda) = \sqrt{\det(B^\top B)} = \sqrt{\det((B^*)^\top B^*)} = \sqrt{\det \mathrm{diag}\{\|\mathbf{b}_1^*\|^2, \ldots, \|\mathbf{b}_d^*\|^2\}} = \prod_{j\in[d]} \|\mathbf{b}_j^*\|$.

Part (b) (adapted from [MG02]). Consider a generic non-zero lattice vector $B\mathbf{x}$ ($\mathbf{x} \in \mathbb{Z}^d \setminus \{\mathbf{0}\}$) and let i be the biggest index such that $x_i \neq 0$. We first prove that $|\langle B\mathbf{x}, \mathbf{b}_i^* \rangle| \geq \|\mathbf{b}_i^*\|^2$. From the definition of i, we know $B\mathbf{x} = \sum_{j=1}^{i} \mathbf{b}_j x_j$. Thus:

$$\begin{aligned}
\langle B\mathbf{x}, \mathbf{b}_i^* \rangle = \sum_{j=1}^{i} \langle \mathbf{b}_j, \mathbf{b}_i^* \rangle x_j &= \langle \mathbf{b}_i, \mathbf{b}_i^* \rangle x_i \\
&= \langle \mathbf{b}_i^* + \sum_{j<i} \mu_{i,j} \mathbf{b}_j^*, \mathbf{b}_i^* \rangle x_i \\
&= \langle \mathbf{b}_i^*, \mathbf{b}_i^* \rangle x_i + \sum_{j<i} \mu_{i,j} \langle \mathbf{b}_j^*, \mathbf{b}_i^* \rangle x_i \\
&= \|\mathbf{b}_i^*\|^2 x_i.
\end{aligned}$$

From this follows $|\langle B\mathbf{x}, \mathbf{b}_i^* \rangle| \geq \|\mathbf{b}_i^*\|^2 |x_i| \geq \|\mathbf{b}_i^*\|^2$ and therefore $\|B\mathbf{x}\| \cdot \|\mathbf{b}_i^*\| \geq \|\mathbf{b}_i^*\|^2$. Clearly $\|\mathbf{b}_i^*\| \neq 0$ since $\mathbf{b}_i$'s are linearly independent, so $\|B\mathbf{x}\| \geq \|\mathbf{b}_i^*\| \geq \min_{j \in [d]} \|\mathbf{b}_j^*\|$. By definition of $\lambda_1(\Lambda)$, the result follows. □

Proposition 1.8. For lattice Λ, there exists $\mathbf{v} \in \Lambda$ such that $\|\mathbf{v}\| = \lambda_1(\Lambda)$.

Proof. By the definition of $\lambda_1(\Lambda)$, there exists a sequence $\{\mathbf{v}_n\}_{n \in \mathbb{N}} \subseteq \Lambda$ such that $\lim_{n \to \infty} \|\mathbf{v}_n\| = \lambda_1(\Lambda)$. From Proposition 1.7, $\lambda_1(\Lambda) > 0$, so for all sufficiently large n it must be $\|\mathbf{v}_n\| \leq 2\lambda_1(\Lambda)$, i.e., lattice vector $\mathbf{v}_n$ belongs to the closed ball $\overline{B}_{2\lambda_1(\Lambda)}(\mathbf{0})$. But this set is compact, so we might extract a convergent subsequence $\{\mathbf{v}_{i_j}\}_{j \in \mathbb{N}}$ with limit $\mathbf{w} = \lim_{j \to \infty} \mathbf{v}_{i_j}$ and $\|\mathbf{w}\| = \lambda_1(\Lambda)$. We claim that $\mathbf{w} \in \Lambda$. By definition of $\mathbf{w}$, $\lim_{j \to \infty} \|\mathbf{v}_{i_j} - \mathbf{w}\| = 0$. Therefore for all sufficiently large j, $\|\mathbf{v}_{i_j} - \mathbf{w}\| < \lambda_1(\Lambda)/2$. By triangle inequality for sufficiently large j and all $k > j$, $\|\mathbf{v}_{i_j} - \mathbf{v}_{i_k}\| \leq \|\mathbf{v}_{i_j} - \mathbf{w}\| + \|\mathbf{w} - \mathbf{v}_{i_k}\| < \lambda_1(\Lambda)$. But $\mathbf{v}_{i_j} - \mathbf{v}_{i_k} \in \Lambda$, and no non-zero lattice vector can have length strictly less than $\lambda_1(\Lambda)$. Thus, $\mathbf{v}_{i_k} = \mathbf{v}_{i_j}$ for all $k > j$. Thus $\mathbf{w} = \mathbf{v}_{i_j} \in \Lambda$. □

1.3 Minkowski's Theorem

Clearly, one expects the dependence between the covolume of a lattice $\mathrm{Vol}(\Lambda)$ and the first minimum $\lambda_1(\Lambda)$. In particular, for smaller covolume we expect the smaller first minimum $\lambda_1(\Lambda)$ (or alternatively, one can say the lattice with large covolume cannot have a small first minimum). Moreover, by dimensionality analysis one should expect that $\lambda_1(\Lambda) \propto \mathrm{Vol}(\Lambda)^{1/d}$. The Minkowski theorem gives the inequality between these two quantities.

Theorem 1.9 (Minkowski's Theorems)**.** Let $\Lambda = \mathcal{L}(B)$ be the lattice of rank d with covolume $\mathrm{Vol}(\Lambda) = \det B$. Then,

First Minkowski's Theorem $\lambda_1(\Lambda) < \sqrt{d}\mathrm{Vol}(\Lambda)^{1/d}$.

Second Minkowski's Theorem $\prod_{i \in [d]} \lambda_i^{1/d} < \sqrt{d}\mathrm{Vol}(\Lambda)^{1/d}$.

Proof. See [MG02, Section 1.3].

While the Minkowski's Theorem turns out to be asymptotically tight for "bad" lattices, in general $\lambda_1(\Lambda)$ might be much smaller than $\sqrt{d}\mathrm{Vol}(\Lambda)^{1/d}$. Consider a very simple example: let $d = 2$ and $\mathbf{b}_1 := \varepsilon \mathbf{e}_1$, $\mathbf{b}_2 := (1/\varepsilon)\mathbf{e}_2$ (where $\mathbf{e}_1, \mathbf{e}_2$ is the canonical basis for $\mathbb{R}^2$). Minkowski's Theorem gives $\lambda_1(\Lambda) \leq \sqrt{2}$, however one easily sees that $\lambda_1(\Lambda) = \varepsilon$ can be made arbitrarily small.

Moreover, Minkowski's Theorem proof is not constructive: we know that a vector of length

below $\sqrt{d}\mathrm{Vol}(\Lambda)^{1/d}$ exists, but there is no concrete algorithm to find it, let alone $\lambda_1(\Lambda)$. This became the core of one of the most well-known lattice computational problem: finding shortest vector in Λ. We introduce this and related problems in the next subsection.

1.4 Lattice Computational Problems

In this section, we finally define some problems that are currently believed to be unsolvable in polynomial time (under reasonable choice of parameters, of course).

1.4.1 Shortest Vector Problem (SVP)

As previously announced, we start with the *shortest vector problem* (SVP), which requires to find the vector of length $\lambda_1(\Lambda)$.

Definition 1.10 (Lattice Generation Algorithm)**.** A **lattice generation algorithm** $\mathsf{LGen}(1^\mu)$ is a probabilistic polynomial time (PPT) algorithm that takes the security parameter 1^μ and outputs the lattice basis[a] $\boldsymbol{B} \in \mathbb{Z}^{\mu\times\mu}$ where the lattice $\Lambda = \mathcal{L}(\boldsymbol{B})$ is of rank μ.

[a] As previously noted in Remark 1.2, we might assume that $\boldsymbol{B}$ is given in the integer representation.

Definition 1.11 (Shortest Vector Problem)**.** For a given adversary $\mathcal{A}$ and lattice generation algorithm $\mathsf{LGen}(1^\mu)$, define the game $\underline{\mathsf{Game\,SVP}}^{\mathcal{A}}_{\mathsf{LGen}}$ as follows:

1. Sample the lattice basis $\boldsymbol{B} \leftarrow \mathsf{LGen}(1^\mu)$ and send to $\mathcal{A}$;
2. $\mathcal{A}$ outputs some $\mathbf{x} \in \mathbb{Z}^\mu \setminus \{\mathbf{0}\}$ based on $\boldsymbol{B} \in \mathbb{Z}^{\mu\times\mu}$;
3. **return** 1 iff $\|\boldsymbol{B}\mathbf{x}\| = \lambda_1(\Lambda)$ and 0 otherwise.

We define the $\mathcal{A}$'s **advantage in solving the shortest vector problem** as:

$$\mathsf{Adv}^{\mathrm{SVP}}_{\mathcal{A},\mathsf{LGen}}(\mu) := \Pr\left[\underline{\mathsf{Game\,SVP}}^{\mathcal{A}}_{\mathsf{LGen}}(\mu) = 1\right]$$

Definition 1.12 (SVP Assumption)**.** We say that **Shortest Vector Problem (SVP) assumption** holds if for any PPT adversary $\mathcal{A}$,

$$\mathsf{Adv}^{\mathrm{SVP}}_{\mathcal{A},\mathsf{LGen}}(\mu) = \mathrm{negl}(\mu).$$

1.4.2 Approximate Shortest Vector Problem (SVP$_\gamma$)

Turns out that the setting of Definition 1.11 is too strict: finding the shortest non-zero vector in lattice Λ is too hard. Moreover, one expects that building cryptographic protocols on this assumption solely would be highly hard. So instead, we will require not finding the shortest vector, but "short-enough", where the natural measure of "short" is "the multiple of $\lambda_1(\Lambda)$".

Definition 1.13 (Approximate Shortest Vector Problem)**.** For a given adversary $\mathcal{A}$ and lattice generation algorithm $\mathsf{LGen}(1^\mu)$, define the game $\underline{\mathsf{Game\ SVP}}^{\mathcal{A}}_{\gamma,\mathsf{LGen}}$ as follows:

1. Sample the lattice basis $B \leftarrow \mathsf{LGen}(1^\mu)$ and send to $\mathcal{A}$;
2. $\mathcal{A}$ outputs some $\mathbf{x} \in \mathbb{Z}^\mu \setminus \{\mathbf{0}\}$ based on (B, γ);
3. **return** 1 iff $\|B\mathbf{x}\| \leq \gamma \lambda_1(\Lambda)$ and 0 otherwise.

We define the $\mathcal{A}$'s **advantage in solving the approximate shortest vector problem** as:

$$\mathsf{Adv}^{\mathrm{SVP}_\gamma}_{\mathcal{A},\mathsf{LGen}}(\mu) := \Pr\left[\underline{\mathsf{Game\ SVP}}^{\mathcal{A}}_{\gamma,\mathsf{LGen}}(\mu) = 1\right]$$

Definition 1.14 (Approximate SVP Assumption)**.** We say that **Approximate Shortest Vector Problem (SVP) assumption** with given γ holds if for any PPT adversary $\mathcal{A}$,

$$\mathsf{Adv}^{\mathrm{SVP}_\gamma}_{\mathcal{A},\mathsf{LGen}}(\mu) = \mathrm{negl}(\mu).$$

Typically, γ is analyzed as a function of the rank of the lattice μ. According to [dBvW25, Section 2.2.5], for example, vectors of length below $\gamma\lambda_1(\Lambda)$ for $\gamma = \sqrt{\mu}$ are considered "very short" while $\gamma = \mathsf{poly}(\mu)$ is the most interesting case. As one expects, the Approximate SVP assumption does not hold for $\gamma = 2^\mu$ by means of Lenstra-Lenstra-Lovasz (LLL) algorithm.

1.4.3 Approximate Closest Vector Problem (CVP_γ)

A somewhat similar problem to SVP is, given some target vector $\mathbf{t} \in \mathbb{Z}^\mu$, find the closest vector to $\mathbf{t}$ on the lattice. This problem also turns out to be computationally hard. Similarly to SVP, the raw CVP problem is too strict, so we give formal definition for approximate CVP straight away.

Definition 1.15 (Approximate Closest Vector Problem)**.** For a given adversary $\mathcal{A}$ and lattice generation algorithm $\mathsf{LGen}(1^\mu)$, define the game $\underline{\mathsf{Game\ CVP}}^{\mathcal{A}}_{\gamma,\mathsf{LGen}}$ as follows:

1. Sample the lattice basis $B \leftarrow \mathsf{LGen}(1^\mu)$;
2. Sample target vector $\mathbf{t} \hookleftarrow \$\ \mathbb{Z}^\mu$ and send $(B, \mathbf{t})$ to $\mathcal{A}$;
3. $\mathcal{A}$ outputs some $\mathbf{x} \in \mathbb{Z}^\mu$ based on $(B, \mathbf{t}, \gamma)$;
4. **return** 1 iff $\|B\mathbf{x} - \mathbf{t}\| \leq \gamma \|B\mathbf{y} - \mathbf{t}\|$ for all $\mathbf{y} \neq \mathbf{x}$ and 0 otherwise.

We define the $\mathcal{A}$'s **advantage in solving the approximate closest vector problem** as:

$$\mathsf{Adv}^{\mathrm{CVP}_\gamma}_{\mathcal{A},\mathsf{LGen}}(\mu) := \Pr\left[\underline{\mathsf{Game\ CVP}}^{\mathcal{A}}_{\gamma,\mathsf{LGen}}(\mu) = 1\right]$$

Similarly to Definition 1.14, we say that approximate CVP assumption with the approximate factor γ holds if $\mathsf{Adv}^{\mathrm{CVP}_\gamma}_{\mathcal{A},\mathsf{LGen}}(\mu) = \mathrm{negl}(\mu)$ for all PTT adversaries $\mathcal{A}$.

2 Learning With Errors (LWE)

2.1 Key Generation

Fix some modulo $q > 1$ (further assume it is prime). Our primary goal is to build public key cryptography schemes in the lattice-fashionable way. Turns out that one of such approaches is to use **Learning With Errors** assumptions.

Fix secret key $\mathbf{s} \in \mathbb{Z}_q^n$ which we assume, for the moment, is sampled uniformly randomly from

the corresponding set (we will assume so until Section 3.1). The public key, corresponding to $\mathbf{s}$, will be a vector of perturbed random inner products with $\mathbf{s}$. Specifically, the public information consists of multiple pairs $(\mathbf{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ such that $b = \langle \mathbf{a}, \mathbf{s} \rangle + e$ for an *error* $e \in \mathbb{Z}_q$, typically taken from the distribution that produces very small values of e: for instance, one might think of $e \in [-\eta, \eta] \pmod q$ for $\eta \approx 3$. Denote such distribution by χ.

This way, we can give an intermediate version of our key generation as follows:

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$**:**

1. $\mathbf{s} \leftarrow\$ \mathbb{Z}_q^\lambda$; `// Generate secret`
2. **for** $i \in [\lambda]$:
3. $\quad \mathbf{a}_i \leftarrow\$ \mathbb{Z}_q^\lambda$, $e \leftarrow\$ \chi$;
4. $\quad b_i \leftarrow \langle \mathbf{a}_i, \mathbf{s} \rangle + e$; `// Generate random pertubation`
5. **return** $(\mathsf{pk} = \{(\mathbf{a}_i, b_i)\}_{i \in [\lambda]}, \mathsf{sk} = \mathbf{s})$.

It is then natural to batch all pertubations into a single matrix A and vector $\mathbf{b}$.[‡] This way, our updated key generation scheme is in Figure 2.1 (where by χ^λ we denote the distribution over $\mathbb{Z}_q^\lambda$ where each component is taken from χ).

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$:

1. $(\mathbf{s}, A) \leftarrow\$ \mathbb{Z}_q^\lambda \times \mathbb{Z}_q^{\lambda \times \lambda}$;
2. $\mathbf{e} \leftarrow\$ \chi^\lambda$;
3. $b \leftarrow A\mathbf{s} + \mathbf{e}$;
4. **return** $(\mathsf{pk} = (A, b), \mathsf{sk} = \mathbf{s})$.

Figure 2.1: LWE-based Key Generation Scheme

Note that we of course expect that solving $A\mathbf{s} + \mathbf{e} = \mathbf{b}$ is hard with respect to $\mathbf{s}$. It should also be clear why we introduce the error term $\mathbf{e}$: assuming $\mathbf{e} = \mathbf{0}$, the equation simplifies to $A\mathbf{s} = \mathbf{b}$. This is a simple system of linear equations which can be further dealt with by means of, say, Gaussian elimination, which has $O(\lambda^3)$ complexity. The assumption of hardness of solving this equation for $\mathbf{e} \neq \mathbf{0}$ is what we introduce and treat in the next sections.

2.2 Search-LWE (SLWE) and Decisional-LWE (DLWE) Assumptions

In this section we shall finally formalize main assumptions about learning with errors problem. First, introduce the LWE instance generation algorithm as specified in Figure 2.2 below.

Now, informally, LWE assumption states that, given output of $\mathsf{LWEGen}_{n,m,q,\chi}(1^\lambda, \mathbf{s})$, it is impossible to determine $\mathbf{s}$ (*Search-LWE assumption*). Similarly to many assumptions in other cryptographic systems (such as Diffie-Hellman assumption), LWE also has the decisional analog: distinguish output of $\mathsf{LWEGen}_{n,m,q,\chi}(1^\lambda, \mathbf{s})$ from the uniform sample over $\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$ (*Decisional-LWE assumption*). We state both assumptions more formally below.

[‡]We also note that A might not necessarily be (and is typically not) the square matrix, but just for the sake of exposition we for now assume $A \in \mathbb{Z}_q^{\lambda \times \lambda}$.

$\mathsf{LWEGen}_{n,m,q,\chi}(1^\lambda, \mathbf{s}) \to \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m$:

1. $A \leftarrow\$ \mathbb{Z}_q^{m\times n}$;
2. $\mathbf{e} \leftarrow\$ \chi$ (over $\mathbb{Z}_q^m$);
3. $\mathbf{b} \leftarrow A\mathbf{s} + \mathbf{e}$;
4. **return** $(A, \mathbf{b})$.

Figure 2.2: LWE Instance Generation

Definition 2.1 (Search-LWE Assumption)**.** For the given adversary $\mathcal{A}$ and LWE parameters (n, m, q, χ), define $\underline{\mathsf{Game\ SLWE}}^{\mathcal{A}}_{\mathsf{LWEGen}}$ as specified in Figure 2.3. Define the advantage of $\mathcal{A}$ as

$$\mathsf{Adv}^{\mathrm{SLWE}}_{\mathcal{A},\mathsf{LWEGen}}(\lambda) = \Pr\left[\underline{\mathsf{Game\ SLWE}}^{\mathcal{A}}_{\mathsf{LWEGen}}(\lambda) = 1\right].$$

We say that **Search-LWE assumption** (SLWE) holds if $\mathsf{Adv}^{\mathrm{SLWE}}_{\mathcal{A},\mathsf{LWEGen}}(\lambda) = \mathrm{negl}(\lambda)$.

Definition 2.2 (Decisional-LWE Assumption)**.** For the given adversary $\mathcal{A}$ and LWE parameters (n, m, q, χ), define $\underline{\mathsf{Game\ DLWE}}^{\mathcal{A}}_{\mathsf{LWEGen}}$ as specified in Figure 2.4. Define the advantage of $\mathcal{A}$ as

$$\mathsf{Adv}^{\mathrm{DLWE}}_{\mathcal{A},\mathsf{LWEGen}}(\lambda) = \left|\Pr\left[\underline{\mathsf{Game\ DLWE}}^{\mathcal{A}}_{\mathsf{LWEGen}}(\lambda) = 1\right] - \frac{1}{2}\right|.$$

We say that **Decisional-LWE assumption** (DLWE) holds if $\mathsf{Adv}^{\mathrm{DLWE}}_{\mathcal{A},\mathsf{LWEGen}}(\lambda) = \mathrm{negl}(\lambda)$ for all PPT adversaries $\mathcal{A}$.

$\underline{\mathsf{Game\ SLWE}}^{\mathcal{A}}_{\mathsf{LWEGen}}$:

1. $\mathbf{s} \leftarrow\$ \mathbb{Z}_q^n$;
2. $(A, \mathbf{b}) \leftarrow\$ \mathsf{LWEGen}_{n,m,q,\chi}(\mathbf{s})$;
3. $\widehat{\mathbf{s}} \leftarrow \mathcal{A}(A, \mathbf{b})$;
4. **return** $\mathbf{s} =_? \widehat{\mathbf{s}}$.

Figure 2.3: Search-LWE Game

$\underline{\mathsf{Game\ DLWE}}^{\mathcal{A}}_{\mathsf{LWEGen}}$:

1. $\mathbf{s} \leftarrow\$ \mathbb{Z}_q^n$;
2. $(A_0, \mathbf{b}_0) \leftarrow\$ \mathsf{LWEGen}_{n,m,q,\chi}(\mathbf{s})$;
3. $(A_1, \mathbf{b}_1) \leftarrow\$ \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m$;
4. $c \leftarrow\$ \{0, 1\}$;
5. $\widehat{c} \leftarrow \mathcal{A}(A_c, \mathbf{b}_c)$;
6. **return** $c =_? \widehat{c}$.

Figure 2.4: Decision-LWE Game

The natural question to ask is whether SLWE and DLWE assumptions are equivalent or not. In fact, [Reg09] has proven that these notions are in fact equivalent in case $q = \mathsf{poly}(n)$. We show this in the following proposition.

Proposition 2.3 (Following [Reg09])**.** If $q = \mathsf{poly}(n)$, SLWE and DLWE assumptions are polynomially equivalent. To put it the other way, if the SLWE assumption holds for LWE instance with parameters $(n, m, q = \mathsf{poly}(n), \chi)$, then the DLWE assumption also holds.

Proof. SLWE $\Rightarrow$ DLWE reduction is obvious. We therefore show DLWE $\Rightarrow$ SLWE. Suppose exists adversary $\mathcal{A}$ that wins $\underline{\mathsf{Game\ DLWE}}^{\mathcal{A}}_{\mathsf{LWEGen}}$ in a non-negligible probability. We construct $\mathcal{B}^{\mathcal{A}}$ that wins $\underline{\mathsf{Game\ SLWE}}^{\mathcal{B}}_{\mathsf{LWEGen}}$ is a non-negligible probability as well.

The informal idea is as follows: suppose $\mathcal{B}$ wants to solve $A\mathbf{s} + \mathbf{e} = \mathbf{b}$. Turns out that $\mathcal{B}$ can guess the coordinates of $\mathbf{s}$ one at a time, starting from s_1. Suppose $\mathcal{B}$ picks some $\widetilde{s} \in \mathbb{Z}_q$ and wants to check whether $\widetilde{s} = s_1$. Select some $\boldsymbol{\Delta} \leftarrow\!\$\ \mathbb{Z}_q^m$ and let $\widetilde{A} = (\widetilde{\mathbf{a}}_1, \mathbf{a}_2, \dots, \mathbf{a}_n)$ where $\widetilde{\mathbf{a}}_1 = \mathbf{a}_1 + \boldsymbol{\Delta}$ (and where the initial matrix is given by $A = (\mathbf{a}_1, \dots, \mathbf{a}_n)$) and set $\widetilde{\mathbf{b}} := \mathbf{b} + \widetilde{s}\boldsymbol{\Delta}$. Now, if $\widetilde{s} = s_1$, then $\widetilde{\mathbf{b}} = \widetilde{A}\mathbf{s} + \mathbf{e}$, otherwise $\widetilde{\mathbf{b}} = \widetilde{A}\mathbf{s} + \mathbf{e} + (\widetilde{s} - s_1)\boldsymbol{\Delta}$. Since $\boldsymbol{\Delta}$ is uniform, it follows that in the latter case $\widetilde{\mathbf{b}}$ is uniformly random and independent of $\widetilde{A}$. Thus, using $\mathcal{A}$, adversary $\mathcal{B}$ has an effective test for chosen $\widetilde{s}$, and thus $\mathcal{B}$ can range over all possible q values, which is polynomial in n.

This way, by using $\mathcal{B}^{\mathcal{A}}_{\mathrm{SLWE}}$ defined in Figure 2.5, we obtain:

$$\mathsf{Adv}^{\mathrm{DLWE}}_{\mathcal{A},\mathsf{LWEGen}}(\lambda) \le qn \cdot \mathsf{Adv}^{\mathrm{SLWE}}_{\mathcal{B}^{\mathcal{A}},\mathsf{LWEGen}}(\lambda). \quad \square$$

Remark 2.4. Note that it might seem like the factor of qn is too large to state the equivalence of DLWE and SLWE (indeed, in the elliptic curve cryptography, one typically uses q exponential in the security parameter). However, in practical applications, q is typically not considerably large: for instance, in Kyber [BDK+18], $q = 7681$ and $n = 256$!
Also note that the brute-force complexity without the DLWE oracle is q^n, which is of course overwhelmingly large! (compared to qn).

$\mathcal{B}^{\mathcal{A}}_{\mathrm{SLWE}}(1^\lambda, A = (\mathbf{a}_1, \dots, \mathbf{a}_n), \mathbf{b}) \to \mathbb{Z}_q^m$:

1. $\widehat{\mathbf{s}} \leftarrow \mathbf{0} \in \mathbb{Z}_q^m$; // Prepare the output
2. **for** $i \in [n]$:
3. **for** $\widetilde{s} \in \mathbb{Z}_q$:
4. $\boldsymbol{\Delta} \leftarrow\!\$\ \mathbb{Z}_q^m$; // Generate random offset
5. $\widetilde{A} \leftarrow (\mathbf{a}_1, \dots, \mathbf{a}_{i-1}, \widetilde{\mathbf{a}}_i, \mathbf{a}_{i+1}, \dots, \mathbf{a}_n)$, $\widetilde{\mathbf{a}}_i := \mathbf{a}_i + \boldsymbol{\Delta}$;
6. $\widetilde{\mathbf{b}} \leftarrow \mathbf{b} + \widetilde{s}\boldsymbol{\Delta}$; // Set offset vector $\widetilde{\mathbf{b}}$ and matrix $\widetilde{A}$
7. $c \leftarrow \mathcal{A}(\widetilde{A}, \widetilde{\mathbf{b}})$; // Run DLWE oracle
8. **if** $c = 0$, set $\widehat{s}_i \leftarrow \widetilde{s}$ and **break**;
9. **return** $\widehat{\mathbf{s}}$.

Figure 2.5: Algorithm for finding the solution to LWE instance given oracle to the DLWE.

2.3 LWE-based Public-Key Encryption

Finally, let us define the public-key encryption (PKE) scheme based on SLWE and DLWE assumptions! We remind the reader of the definition of a PKE scheme.

Definition 2.5 (Public-Key Encryption Scheme)**.** A **public-key encryption scheme** Π_{PKE} is a triplet of the following PPT algorithms:

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$ returns the pair of public key and secret key, respectively.
- $\mathsf{Enc}(\mathsf{pk}, m) \to c$ takes the public key and message and produces the ciphertext c.
- $\mathsf{Dec}(\mathsf{sk}, c) \to m$ takes the secret key and ciphertext and restores the message m.

We require **correctness**:

$$\forall m \ : \ \Pr\left[\widehat{m} = m \;\middle|\; \begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \widehat{m} \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m))\end{array}\right] \geq 1 - \mathrm{negl}(\lambda),$$

and **IND-CCA** and **IND-CPA** security. To not overwhelm the reader with the details, we will only focus on the IND-CPA security with the corresponding $\underline{\text{Game INDCPA}}^{\mathcal{A}}_{\Pi}$ defined in Figure 2.6. We naturally define advantage $\mathsf{Adv}^{\text{IND-CPA}}_{\mathcal{A},\Pi_{\text{PKE}}}(\lambda) := 2\left|\Pr[\underline{\text{Game INDCPA}}^{\mathcal{A}}_{\Pi_{\text{PKE}}} = 1] - \frac{1}{2}\right|$ and say that Π_{PKE} is **IND-CPA secure** if $\mathsf{Adv}^{\text{IND-CPA}}_{\mathcal{A},\Pi_{\text{PKE}}}(\lambda) = \mathrm{negl}(\lambda)$ for all PPT $\mathcal{A}$.

$\underline{\text{Game INDCPA}}^{\mathcal{A}}_{\Pi}(1^\lambda)$:

1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$;
2. $(m_0, m_1) \leftarrow \mathcal{A}(\mathsf{pk})$; `// Typically one requires` $|m_0| = |m_1|$
3. $b \leftarrow\!\$\ \{0,1\}$, $c_b \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$;
4. $\widehat{b} \leftarrow \mathcal{A}(\mathsf{pk}, m_0, m_1, c_b)$;
5. **return** $\widehat{b} =_? b$.

Figure 2.6: IND-CPA Security Game. $\mathcal{A}$ might be allowed to perform steps 2–3 $\mathrm{poly}(\lambda)$ times.

Now, we define the following PKE scheme based on (n, m, q, χ) assumption where the message space is $\{0,1\}$. The main idea is that encryption scheme will generate the new LWE instance $(A', \mathbf{b}')$ by taking a "small" linear combination of publicly known $(A, \mathbf{b})$. By additionally adding $m\lceil q/2 \rfloor$ to $\mathbf{b}'$, we obtain the encryption (further denote by $\lceil \cdot \rfloor : \mathbb{Q} \to \mathbb{Z}$ the function that rounds a given rational number to the nearest integer). Upon decryption, the holder of the secret key $\mathbf{s}$ will be able to check whether $A'\mathbf{s} - \mathbf{b}'$ produces the small error. If the term $m\lceil q/2 \rfloor$ was added to form $\mathbf{b}'$, the resultant difference would be large, suggesting that the original bit was 1. Otherwise, decryption would output 0. Let us put this formally in the definition below.

Definition 2.6 (LWE-based PKE)**.** Based on LWE instance with parameters (n, m, q, χ), define the PKE scheme $\Pi^{\text{LWE}}_{\text{PKE}}$ over message space $\{0,1\}$ as follows: KeyGen is specified in Figure 2.1, Enc in Figure 2.7, and Dec in Figure 2.8.

Since we have to work with "small" and "large" elements of $\mathbb{Z}_q$, by $\|x\|_\infty$ for $x \in \mathbb{Z}_q$ we denote x if $x \leq \frac{q}{2}$, and $q - x$ otherwise (for example, naturally, $\|q - 3\|_\infty = 3$).

Now we show that $\Pi^{\text{LWE}}_{\text{PKE}}$ is a correct and secure PKE scheme.

Proposition 2.7. $\Pi^{\text{LWE}}_{\text{PKE}}$ is correct and IND-CPA secure under DLWE assumption.

$\mathsf{Enc}(\mathsf{pk}, m \in \{0,1\}) \to (c \in \mathbb{Z}_q^n \times \mathbb{Z}_q)$:

1. $\Delta\mathbf{r}, \Delta\mathbf{z} \leftarrow\$\ \chi^m$, $\Delta z' \leftarrow\$\ \chi$;
2. $\widetilde{\mathbf{a}} \leftarrow A^\top \Delta\mathbf{r} + \Delta\mathbf{z}$;
3. $\widetilde{b} \leftarrow \mathbf{b}^\top \Delta\mathbf{r} + \Delta z' + m\lceil q/2 \rfloor$;
4. **return** $c = (\widetilde{\mathbf{a}}, \widetilde{b})$.

Figure 2.7: Encryption in $\Pi_{\mathrm{PKE}}^{\mathrm{LWE}}$

$\mathsf{Dec}(\mathsf{sk}, c) \to m$:

1. Parse $(\widetilde{\mathbf{a}}, \widetilde{b}) \leftarrow c$;
2. $\delta \leftarrow \widetilde{b} - \mathbf{s}^\top \widetilde{\mathbf{a}}$;
3. **return** 1 iff $\|\delta\|_\infty > q/4$.

Figure 2.8: Decryption in $\Pi_{\mathrm{PKE}}^{\mathrm{LWE}}$

Proof. (*Correctness*) Let us first show correctness. Assume $\mathsf{sk} = \mathbf{s}$ and $\mathsf{pk} = (A, \mathbf{b})$ is the valid key-pair (that is, $A\mathbf{s} + \mathbf{e} = \mathbf{b}$ for "small" $\mathbf{e} \leftarrow\$\ \chi^m$) and we formed $(\widetilde{\mathbf{a}}, \widetilde{b})$ according to Figure 2.7. Let us try to decode this cipher by means of Figure 2.8:

$$\begin{aligned}\delta &= \mathbf{b}^\top \Delta\mathbf{r} + \Delta z' + m\lceil q/2 \rfloor - \mathbf{s}^\top A^\top \Delta\mathbf{r} - \mathbf{s}^\top \Delta\mathbf{z} \\ &= (\mathbf{b} - A\mathbf{s})^\top \Delta\mathbf{r} + (\Delta z' - \mathbf{s}^\top \Delta\mathbf{z}) + m\lceil q/2 \rfloor \\ &= (\mathbf{e}^\top \Delta\mathbf{r} - \mathbf{s}^\top \Delta\mathbf{z} + \Delta z') + m\lceil q/2 \rfloor\end{aligned}$$

Now note that the quantity marked in blue is small. Indeed, let us show why. Assume that χ produces elements of $\mathbb{Z}_q$ with the modulus not exceeding η. Then[§],

$$\|\mathbf{e}^\top \Delta\mathbf{r} - \mathbf{s}^\top \Delta\mathbf{z} + \Delta z'\|_\infty \le \|\mathbf{e}^\top \Delta\mathbf{r}\|_\infty + \|\mathbf{s}^\top \Delta\mathbf{z}\|_\infty + \|\Delta z'\|_\infty \le n\eta^2 + n\eta + \eta \le 2n\eta.$$

Clearly, $2n\eta$ is a small quantity (assuming $\eta \ll q/8n$). Now, if $m = 0$, we get that $\|\delta\|_\infty = \|(\text{blue part})\|_\infty < q/4$, so decryption works. If $m = 1$, then $\delta = \lceil q/2 \rfloor + (\text{blue part})$, which is a large value, so the decryption outputs 1, as expected.

(*IND-CPA Security*). Suppose $\mathcal{A}$ solves the $\underline{\mathsf{Game\ INDCPA}}_\Pi^{\mathcal{A}}$ with a non-negligible probability. We construct the $\mathcal{B}^{\mathcal{A}}$ that breaks the $\underline{\mathsf{Game\ DLWE}}_{\mathsf{LWEGen}}^{\mathcal{B}}$ using $\mathcal{A}$ as an oracle. The high-level idea of the proof is the following: suppose $\mathcal{B}$ is given a tuple $(A, \mathbf{b})$. If $(A, \mathbf{b})$ is a valid LWE instance, then it is a valid public key in $\Pi_{\mathrm{PKE}}^{\mathrm{LWE}}$, so $\mathcal{B}$ might run $\mathcal{A}$ on $\mathsf{pk}^* \leftarrow (A, \mathbf{b})$. Then, if $\mathcal{A}$ correctly guesses the bit, then likely pk^* was a valid key. In turn, if pk^* is a "garbage", with probability approximately $1/2$, the $\mathcal{A}$ would lose. Thus, $\mathcal{B}$ decision will be based on whether $\mathcal{A}$ guesses the correct bit or not. We encapsulate this in Figure 2.9. One can then show:

$$\mathsf{Adv}_{\mathcal{A},\Pi}^{\text{IND-CPA}}(\lambda) \le 2\mathsf{Adv}_{\mathcal{B}^{\mathcal{A}},\mathsf{LWEGen}}^{\mathrm{DLWE}}(\lambda). \quad \square$$

2.4 Relation to Lattices

The reasonable question that the reader might rightfully have at this point: where are the lattices? So far, we have spent the whole Section 1 describing lattices, while the current section does not explicitly use them in any way.

In fact, turns out that the LWE problem can be viewed as the lattice problem!

[§]In the sequel, we will show that $\mathbf{s}$ can be taken from the same distribution as the error term $\mathbf{e}$ (that is, from χ), making upper bound even smaller.

$\mathcal{B}^{\mathcal{A}}_{\text{DLWE}}(1^\lambda, A, \mathbf{b}) \to \{0,1\}$:

1. pass $\mathsf{pk}^* \leftarrow (A, \mathbf{b})$ to $\mathcal{A}$;
2. get (m_0, m_1) from $\mathcal{A}$;
3. choose $d \leftarrow\$ \{0,1\}$;
4. send $c_d \leftarrow \mathsf{Enc}(\mathsf{pk}^*, m_d)$ to $\mathcal{A}$;
5. $\mathcal{A}$ responds with $\widehat{d}$;
6. **return** $d =_? \widehat{d}$.

Figure 2.9: Algorithm for finding the solution to DLWE instance given oracle to the IND-CPA.

Definition 2.8. For the LWE instance $(A, \mathbf{b})$ over parameters (n, m, q, χ), we define the associated lattice $\Lambda_q(A)$ as follows:

$$\Lambda_q(A) \triangleq \{\mathbf{y} \in \mathbb{Z}^m : \mathbf{y} = A\mathbf{x} \pmod q \text{ for some } \mathbf{x} \in \mathbb{Z}^n\} = A\mathbb{Z}^n + q\mathbb{Z}^m.$$

Now note that the vector $\mathbf{b}$ does not lie on the lattice $\Lambda_q(A)$ (unless $\mathbf{e} = \mathbf{0}$ which makes LWE instance unsecure). But note that due to the modular equation, $\mathbf{b} = A\mathbf{s} + q\mathbf{z} + \mathbf{e}$ for some $\mathbf{z} \in \mathbb{Z}^m$ and since $A\mathbf{s} + q\mathbf{z} \in \Lambda_q(A)$, the distance from $\mathbf{b}$ to $\Lambda_q(A)$ is bounded by $\|\mathbf{e}\|$. That said, although $\mathbf{b}$ is not in the lattice, it is very close to it! Therefore, almost surely, the closest vector to $\mathbf{b}$ is $A\mathbf{s} + q\mathbf{z}$. Now imagine that we know how to solve CVP defined in Section 1.4.3. Then, based on $\mathbf{b}$, we can derive $A\mathbf{s} + q\mathbf{z}$, and then easily obtain $\mathbf{s}$ by solving a system of linear equations. Note that the approximation factor depends on the underlying distribution χ^m, from which $\mathbf{e}$ is sampled.

This way, lattices are crucial for cryptanalysis of LWE-based schemes.

2.5 Variations of LWE

Let us analyze the efficiency of the PKE scheme provided in Definition 2.6.

Proposition 2.9. In the PKE scheme $\Pi^{\text{LWE}}_{\text{PKE}}$ over LWE instance (n, m, q, χ), the public key size is $O(m \log q)$, secret key size is $O(n \log q)$, and the codeword size is $O(m \log q)$: therefore, sending a B-bit message takes $O(Bm \log q)$ space.

Remark 2.10. One might ask why the public key size is not $O(nm \log q)$, given that we need to store the matrix A. Fortunately, A is sampled randomly, so instead of storing the whole matrix, one might store the seed from which one can derive A instead.

Unfortunately, such key and ciphertext sizes are significantly larger than those of elliptic curve-based schemes (in fact, if λ is taken as a security parameter, then ciphertext size scales as $O(\lambda^2)$). This led to the natural question: can we reduce these sizes?

2.5.1 Ring-LWE

Turns out that one possible answer to this question is to use objects other than $\mathbb{Z}_q^n$ as elements that possess certain useful structure: for instance, the popular choice is to use $\mathcal{R}_q := \mathcal{R}/q\mathcal{R}$ for $\mathcal{R} = \mathbb{Z}[T]/(T^n + 1)$ and n being a power-of-two. In such case, one might think of $\mathcal{R}$ either by:

(a) $(n-1)$-degree polynomials with operations modulo $T^n + 1$;

(b) the ring of integers $\mathbb{Z}(\zeta_{2n})$ since $\mathcal{R} \cong \mathbb{Z}[T]/(\Phi_{2n}(T)) \cong \mathbb{Z}(\zeta_{2n})$ where $\zeta_N := e^{2\pi i/N}$.

Now, instead of having $(A, \mathbf{b}) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$ as a public key, we will store the public key as a pair $(a, b) \in \mathcal{R}_q^2$ where $b = as + e$, but over the polynomial space. However, we should clarify what "small" means in the context of $\mathcal{R}_q$ (since e here comes from some small distribution). For this reason, given $f(T) = \sum_{i \in (n)} f_i T^i \in \mathcal{R}_q$, define:

$$\|f\|_\infty \triangleq \max_{i \in (n)} \|f_i\|_\infty.$$

So in other words, polynomial $f(T) \in \mathcal{R}_q$ is large as long as its largest coefficient is large.

Further, we identify Ring-LWE instance by a tuple of $(\mathcal{R}, n, q, \chi)$, where (i) additive group of $\mathcal{R}$ is isomorphic to $\mathbb{Z}^n$, (ii) instead of $\mathbb{Z}_q^n$ in "plain" LWE, we use field $\mathcal{R}_q := \mathcal{R}/q\mathcal{R}$, and (iii) χ is a (small) random distribution over $\mathcal{R}_q$.

Proposition 2.11. Using Ring-LWE structure, we can reduce ciphertext size for a n-bit message of PKE Scheme from Definition 2.6 from $O(nm \log q)$ to $O(n \log q)$. Moreover, computing expression $as + e$ now takes only $O(n \log n)$ complexity instead of $O(nm)$.

The reason for $O(n \log n)$ complexity for computing $as + e$ comes from the fact that we can utilize Number Theoretic Transform (NTT/FFT) for fast polynomial multiplication. The reason for reduced ciphertext size is less evident, but the idea is the following: instead of adding a $m\lceil q/2 \rfloor$ term to the second ciphertext piece in Figure 2.7, we are going to represent n bits of message $(m_i)_{i \in (n)}$ as a polynomial $m(T) := \sum_{i \in (n)} m_i T^i \in \mathcal{R}_q$. We specify details in the definition below.

Definition 2.12. Based on Ring-LWE instance with parameters $(\mathcal{R}, n, q, \chi)$, define the PKE scheme $\Pi_{\text{PKE}}^{\text{RLWE}}$ over message space $\{0,1\}^n$ as follows: KeyGen is specified in Figure 2.10, Enc in Figure 2.11, and Dec in Figure 2.12.

KeyGen$(1^\lambda) \to$ (pk, sk):

1. $(s, a) \leftarrow\!\$\ \mathcal{R}_q^2$;
2. $e \leftarrow\!\$\ \chi$ (over $\mathcal{R}_q$);
3. $b \leftarrow as + e \in \mathcal{R}_q$;
4. **return** $((a, b), s)$.

Figure 2.10: Ring-LWE-based Key Generation

Enc(pk, $(m_i)_{i \in (n)} \in \{0,1\}^n$) $\to c$:

1. $m \leftarrow \sum_{i \in (n)} m_i T^i \in \mathcal{R}_q$;
2. $\Delta r, \Delta z, \Delta z' \leftarrow\!\$\ \chi$;
3. $\widetilde{a} \leftarrow a\Delta r + \Delta z$;
4. $\widetilde{b} \leftarrow b\Delta r + \Delta z' + \lceil \frac{q}{2} \rfloor m$;
5. **return** $c = (\widetilde{a}, \widetilde{b})$.

Figure 2.11: Ring-LWE-based Encryption

Dec(sk, c) $\to m$:

1. Parse $(\widetilde{a}, \widetilde{b}) \leftarrow c$;
2. $\delta \leftarrow \widetilde{b} - s\widetilde{a}$;
3. **return** 1 iff $\|\delta\|_\infty > \frac{q}{4}$.

Figure 2.12: Ring-LWE-based Decryption

Another valid question is whether introducing such structure should worsen the security: so in other words, given appropriate Ring-LWE instance $(\mathcal{R}, n, q, \chi)$, is $\Pi_{\text{PKE}}^{\text{RLWE}}$ secure? Turns out that there are no algorithms for breaking analogous Ring-LWE-based assumption significantly faster than "plain" version of the LWE. To put this assumption formally, we slighly modify Definition 2.1.

Definition 2.13. For the given adversary $\mathcal{A}$ and Ring-LWE parameters $(\mathcal{R}, n, q, \chi)$, define $\underline{\text{Game SRLWE}}^{\mathcal{A}}$ as specified in Figure 2.13. Define the advantage of $\mathcal{A}$ as

$$\mathsf{Adv}^{\text{SRLWE}}_{\mathcal{A}}(\lambda) = \Pr\left[\underline{\text{Game SRLWE}}^{\mathcal{A}}(\lambda) = 1\right].$$

We say that **Search-Ring-LWE assumption** (SRLWE) holds if $\mathsf{Adv}^{\text{SRLWE}}_{\mathcal{A}}(\lambda) = \text{negl}(\lambda)$ for all PPT adversaries $\mathcal{A}$. Decisional version of Ring-LWE is defined similarly to Definition 2.2.

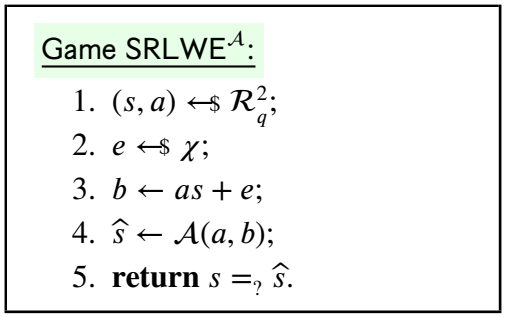

Figure 2.13: Search-Ring-LWE Game

We leave the following proposition without a proof as it closely resembles one specified in Proposition 2.7. For concrete details and survey on possible attacks, see for example [LLZ+18].

Proposition 2.14. $\Pi^{\text{RLWE}}_{\text{PKE}}$ is correct and IND-CPA secure under Decisional Ring-LWE assumption.

2.5.2 Module-LWE

The idea of Module-LWE is to in a way "blend" together Ring-LWE and plain-LWE. Namely, instead of using $\mathcal{R}_q$ in Ring-LWE, we will use vectors from $\mathcal{R}_q^\ell$. Note that equipping $\mathcal{R}_q$ with a vector structure naturally induces metric notions we have used before: e.g., for $\boldsymbol{f} = (f_1, \ldots, f_\ell) \in \mathcal{R}_q^\ell$, we have $\|\boldsymbol{f}\|_\infty := \max_{i\in[\ell]} \|f_i\|_\infty$ and $\langle \boldsymbol{f}, \boldsymbol{g}\rangle = \sum_{i\in[\ell]} f_i g_i$ is a valid inner product that produces output in $\mathcal{R}_q$.

The analogy to plain-LWE equation $\mathbf{b} = A\mathbf{s} + \mathbf{e}$ for $A \in \mathbb{Z}_q^{n\times m}$, $\mathbf{s} \in \mathbb{Z}_q^m$, and $\mathbf{b}, \mathbf{e} \in \mathbb{Z}_q^n$, is now $\mathbf{b}_\mathcal{R} = A_\mathcal{R}\mathbf{s}_\mathcal{R} + \mathbf{e}_\mathcal{R}$ for $A_\mathcal{R} \in \mathcal{R}_q^{k\times\ell}$, $\mathbf{s} \in \mathcal{R}_q^\ell$, and $\mathbf{s}_\mathcal{R}, \mathbf{e}_\mathcal{R} \in \mathcal{R}_q^k$, where operations are done inside ring $\mathcal{R}_q$. We call $(A, \mathbf{b})$ a Module-LWE instance with parameters $(\mathcal{R}, k, \ell, q, \chi)$ if it is an output of $\mathsf{MLWEGen}_{\mathcal{R},k,\ell,q,\chi}$ defined in Figure 2.14. Now we define Module-LWE assumption in the definition below.

Definition 2.15. For the given adversary $\mathcal{A}$ and Module-LWE parameters $(\mathcal{R}, k, \ell, q, \chi)$, define $\underline{\text{Game SMLWE}}^{\mathcal{A}}$ as specified in Figure 2.15. Define the advantage of $\mathcal{A}$ as

$$\mathsf{Adv}^{\text{SMLWE}}_{\mathcal{A}}(\lambda) = \Pr\left[\underline{\text{Game SMLWE}}^{\mathcal{A}}(\lambda) = 1\right].$$

We say that **Search-Module-LWE assumption** (SMLWE) holds if $\mathsf{Adv}^{\text{SMLWE}}_{\mathcal{A}}(\lambda) = \text{negl}(\lambda)$ for all PPT adversaries $\mathcal{A}$. Decisional Module-LWE is defined similarly to Definition 2.2.

At first glance, it might seem that Module-LWE looks much less efficient than Ring-LWE: for

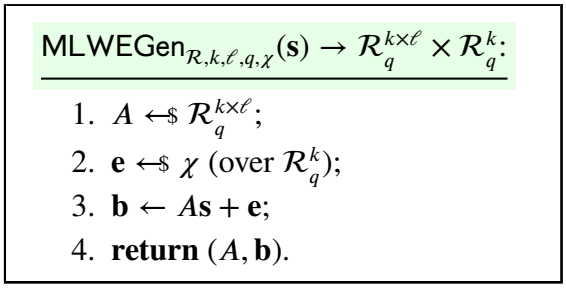

Figure 2.14: Modular-LWE $(\mathcal{R}, k, \ell, q, \chi)$ instance generation

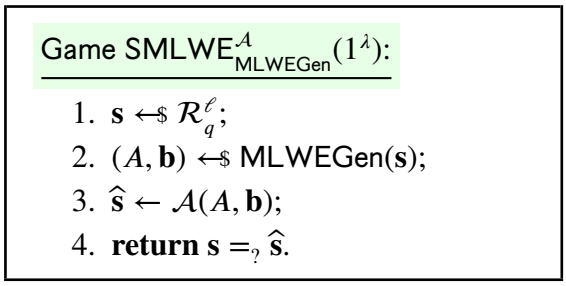

Figure 2.15: Search Modular-LWE Game

instance, why should representing A as a matrix $\mathcal{R}_q^{k\times\ell}$ be any more efficient than using a single $\mathcal{R}_q$? Notice that to adjust the security of Ring-LWE, one must regulate n — the power of cyclotomic polynomial. However, it must be a power of two, which makes such parameter adjusting highly problematic. For Module-LWE, one typically fixes (q, n) (which fully determine $\mathcal{R}_q$-arithmetic efficiency), and then changes ℓ to set target security level.

This way, key sizes (which are primarily determined by a storage of $\mathbf{b}$ by Remark 2.10) turn out to be more or less equal for both Ring-LWE and Module-LWE:

- *Dilithium* [DKL+18], being a Module-LWE, uses parameters $q = 2^{23} - 2^{13} + 1$, $n = 256$, and $(k, \ell) \in \{(3,2), (4,3), (5,4), (6,5)\}$; If we take, say $(k, \ell) = (4, 3)$, we need $kn = 1024$ elements of $\mathbb{Z}_q$ to store $\mathbf{b} \in \mathcal{R}_q^k$.
- *NewHope* [ADPS16], being a Ring-LWE, uses $q = 2^{13} + 2^{12} + 1$ and $n = 1024$. Notice that although we have a single polynomial $b \in \mathcal{R}_q$, we still need 1024 elements of $\mathbb{Z}_q$ for storage.

Turns out that implementing analogous PKE scheme to ones in Definition 2.6 and Definition 2.12 based on Module-LWE assumptions leads to the Kyber PKE description! We specify the details in Section 3.1.

3 Lattice-based Cryptographic Protocols

In Section 2, we have defined the public-key encryption scheme based on LWE assumptions (specifically, based on plain-LWE and Ring-LWE). In this section, we shall see how to implement several familiar cryptographic protocols using these assumptions.

3.1 Kyber Public-Key Encryption

As previously announced, Kyber PKE Scheme [BDK+18] is, in the essence, a Module-LWE-based adaptation of Definition 2.6 and Definition 2.12.

Definition 3.1. Based on Module-LWE instance with parameters $(\mathcal{R}, k, \ell, q, \chi)$, define the PKE scheme $\Pi^{\text{Kyber}}_{\text{PKE}}$ over message space $\{0,1\}^n$ as follows: KeyGen is specified in Figure 3.1, Enc in Figure 3.2, and Dec in Figure 3.3.
In particular, original paper [BDK+18] proposes the following parameters: $n = 256$, $k = \ell = 3$, $q = 7681$, and χ is the centered binomial distribution B_η for $\eta = 4$.

Short-Secret LWE (ss-LWE)*. Before stating the security of the Kyber PKE, notice the subtle difference from our previous PKE schemes: here, the secret value $\mathbf{s}$ is sampled from the same distribution as the error $\mathbf{e}$. This surely optimizes the storage requirements, but it might seem that

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$:

1. $\mathbf{s}, \mathbf{e} \leftarrow\!\$\ \chi^k$ (over $\mathcal{R}_q^k$);
2. $A \leftarrow\!\$\ \mathcal{R}_q^{k \times k}$;
3. $\mathbf{b} \leftarrow A\mathbf{s} + \mathbf{e}$;
4. **return** $((A, \mathbf{b}), \mathbf{s})$.

Figure 3.1: Kyber Key Generation (note that $\ell = k$)

$\mathsf{Enc}(\mathsf{pk}, (m_i)_{i \in (n)} \in \{0,1\}^n) \to c$:

1. $m \leftarrow \sum_{i \in (n)} m_i T^i \in \mathcal{R}_q$;
2. $\Delta\mathbf{r}, \Delta\mathbf{z} \leftarrow\!\$\ \chi^k$;
3. $\Delta z' \leftarrow\!\$\ \chi$;
4. $\widetilde{\mathbf{a}} \leftarrow A^\top \Delta\mathbf{r} + \Delta\mathbf{z}$;
5. $\widetilde{b} \leftarrow \mathbf{b}^\top \Delta\mathbf{r} + \Delta z' + \lceil \frac{q}{2} \rfloor m$;
6. **return** $c = (\widetilde{\mathbf{a}}, \widetilde{b})$.

Figure 3.2: Kyber Encryption

$\mathsf{Dec}(\mathsf{sk}, c) \to m$:

1. Parse $(\widetilde{\mathbf{a}}, \widetilde{b}) \leftarrow c$;
2. $\delta \leftarrow \widetilde{b} - \mathbf{s}^\top \widetilde{\mathbf{a}}$;
3. **return** 1 iff $\|\delta\|_\infty > \frac{q}{4}$.

Figure 3.3: Kyber Decryption

this significantly reduces the security: for starters, the number of possible values of $\mathbf{s}$ reduces from q^{nk} down to η^{nk} (where each coefficient c_i from output of χ is bounded by η: $\|c_i\|_\infty \le \eta$). However, η^{nk} is still an exponentially large value: even for $\eta = 2$, the number of possible values is 2^{768} given Kyber parameters. This motivates us to modify the LWE assumptions to allow $\mathbf{s}$ to be sampled from smaller distributions. Further, we will show results for plain LWE, and the following results can be trivially generalized to Ring and Module LWEs.

Definition 3.2 (ss-LWE)**.** Let $\underline{\mathsf{Game\ ss\text{-}LWE}}^{\mathcal{A}}_{\mathsf{LWEGen}}$ be as specified in Figure 2.3, but $\mathbf{s}$ is sampled from the distribution χ. Similarly define the advantage of adversary $\mathcal{A}$ as follows:

$$\mathsf{Adv}^{\text{ss-LWE}}_{\mathcal{A},\mathsf{LWEGen}}(\lambda) = \Pr\left[\underline{\mathsf{Game\ ss\text{-}LWE}}^{\mathcal{A}}_{\mathsf{LWEGen}}(\lambda) = 1\right].$$

We say that **short-secret LWE assumption** holds if $\mathsf{Adv}^{\text{ss-LWE}}_{\mathcal{A},\mathsf{LWEGen}}(\lambda) = \mathrm{negl}(\lambda)$ for all PPT $\mathcal{A}$.

We claim that ss-LWE and search LWE assumptions are in fact polynomially equivalent (although with different LWE instances parameters). We introduce the following proposition.

Proposition 3.3. The following two statements are true:

(a) if ss-LWE assumption holds with parameters (n, m, q, χ), then LWE assumption with the same parameters (n, m, q, χ) holds as well;

(b) if LWE assumption holds with parameters (n, m, q, χ), then ss-LWE assumption with parameters $(n, m - n, q, \chi)$ holds as well.

Proof. *Part (a).* Assume $\mathsf{Adv}^{\text{SLWE}}_{\mathcal{A},\mathsf{LWEGen}}(\lambda)$ is non-negligible for some $\mathcal{A}$. We build $\mathcal{B}^{\mathcal{A}}$ that uses $\mathcal{A}$ as an oracle and breaks the ss-LWE assumption in Figure 3.4. Note that $\mathbf{b} + A\boldsymbol{\delta}$ is distributed uniformly over $\mathbb{Z}_q^m$, thus $(A, \widetilde{\mathbf{b}})$ is a valid instance of SLWE, which is needed for a valid $\mathcal{A}$ execution at step 3. Thus,

$$\mathsf{Adv}^{\text{SLWE}}_{\mathcal{A},\mathsf{LWEGen}}(\lambda) = \mathsf{Adv}^{\text{ss-LWE}}_{\mathcal{B}^{\mathcal{A}},\mathsf{LWEGen}}(\lambda).$$

Part (b). Now assume $\mathsf{Adv}^{\text{ss-LWE}}_{\mathcal{A},\mathsf{LWEGen}(n,m-n,q,\chi)}(\lambda)$ is non-negligible for some $\mathcal{A}$. We build $\mathcal{B}^{\mathcal{A}}$ that uses $\mathcal{A}$ as an oracle and breaks the SLWE assumption for parameters (n, m, q, χ) in Figure 3.5. The main trick here is to rearrange the original equation $\mathbf{b} = A\mathbf{s} + \mathbf{e}$ to get the ss-LWE instance of smaller size. We do precisely that: following steps 1–5, we obtain $\widetilde{\mathbf{b}} = \widetilde{A}\mathbf{e}_1 + \mathbf{e}_2$, which is clearly a ss-LWE instance with parameters $(n, m - n, q, \chi)$, making oracle call at step 6 valid. Notice that

solving this ss-LWE instance provides $\mathcal{B}$ with an error piece $\mathbf{e}_1$, making it possible to restore $\mathbf{s}$ by solving a linear equation $\mathbf{b}_1 = A_1\mathbf{s} + \mathbf{e}_1$. Thus,

$$\mathsf{Adv}^{\text{ss-LWE}}_{\mathcal{A},\mathsf{LWEGen}(n,m-n,q,\chi)}(\lambda) = \mathsf{Adv}^{\text{SLWE}}_{\mathcal{B}^{\mathcal{A}},\mathsf{LWEGen}(n,m,q,\chi)}(\lambda). \quad \square$$

$\mathcal{B}^{\mathcal{A}_{\text{SLWE}}}_{\text{ss-LWE}}(A, \mathbf{b}) \to \mathbf{s}$:

1. $\boldsymbol{\delta} \leftarrow\$ \mathbb{Z}_q^m$;
2. $\widetilde{\mathbf{b}} \leftarrow \mathbf{b} + A\boldsymbol{\delta}$;
3. $\widetilde{\mathbf{s}} \leftarrow \mathcal{A}_{\text{SLWE}}(A, \widetilde{\mathbf{b}})$;
4. $\widehat{\mathbf{s}} \leftarrow \widetilde{\mathbf{s}} - \boldsymbol{\delta}$
5. **return** $\widehat{\mathbf{s}}$.

Figure 3.4: Solving ss-LWE given Search-LWE solver

$\mathcal{B}^{\mathcal{A}_{\text{ss-LWE}}}_{\text{SLWE}}(A, \mathbf{b}) \to \mathbf{s}$:

1. let $A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$ where $A_1 \in \mathbb{Z}_q^{n\times n}$, $A_2 \in \mathbb{Z}_q^{(m-n)\times n}$;
2. let $\mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}$ where $\mathbf{b}_1 \in \mathbb{Z}_q^{n}$, $\mathbf{b}_2 \in \mathbb{Z}_q^{m-n}$;
3. let $\mathbf{e} = \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \end{bmatrix}$ where $\mathbf{e}_1 \in \mathbb{Z}_q^{n}$, $\mathbf{e}_2 \in \mathbb{Z}_q^{m-n}$;
4. $\widetilde{A} \leftarrow -A_2A_1^{-1} \in \mathbb{Z}_q^{(m-n)\times n}$;
5. $\widetilde{\mathbf{b}} \leftarrow \widetilde{A}\mathbf{b}_1 + \mathbf{b}_2 \in \mathbb{Z}_q^{m-n}$;
6. $\widehat{\mathbf{e}}_1 \leftarrow \mathcal{A}_{\text{ss-LWE}}(\widetilde{A}, \widetilde{\mathbf{b}})$;
7. $\widehat{\mathbf{s}} \leftarrow A_1^{-1}(\mathbf{b}_1 - \mathbf{e}_1)$;
8. **return** $\widehat{\mathbf{s}}$.

Figure 3.5: Solving Search-LWE given ss-LWE solver

Now with the ss-LWE assumption in mind, we can state the security of the Kyber PKE.

Proposition 3.4. $\Pi^{\text{Kyber}}_{\text{PKE}}$ is correct and IND-CPA secure under Short-Secret Decisional Module-LWE assumption.

3.2 Dilithium Digital Signature

In this section, we specify the main idea behind the well-known lattice-based signature scheme *Dilithium* [DKL+18]. Compared to previous discussion, we omit formal proofs of security and concentrate on the construction.

First, recall the definition of the signature scheme.

Definition 3.5 (Signature Scheme)**.** A **signature scheme** Σ is a triplet of the following effective probabilistic algorithms:

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$ outputs the pair of public and secret key, respectively.
- $\mathsf{Sign}(\mathsf{sk}, m) \to \sigma$ signs a message m using secret key sk and thereby produces signature σ.
- $\mathsf{Verify}(\mathsf{pk}, m, \sigma) \to \{0,1\}$ checks whether the signature σ was produced by signing message m using sk corresponding to pk and outputs the decision.

We require **correctness**:

$$\forall m \ : \ \Pr\left[\mathsf{Verify}(\mathsf{pk}, m, \sigma) = 1 \;\middle|\; \begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)\end{array}\right] \geq 1 - \mathrm{negl}(\lambda),$$

and unforgeability against chosen-message attacks, roughly meaning that obtaining $\mathrm{poly}(\lambda)$ pairs (σ_i, m_i) signed by some key-pair $(\mathsf{pk}, \mathsf{sk})$ does now allow $\mathcal{A}$ to produce a valid pair $(\widehat{\sigma}, \widehat{m})$. See [BS23, Section 13] for more details.

3.2.1 Schnorr Signature

Further denote by H the hash function from $\{0,1\}^*$ to the appropriate domain. One of the most widely used digital signatures schemes (especially in the Bitcoin applications) is the *Schnorr Signature Scheme*. We define it below.

Definition 3.6. Fix a cyclic abelian group $\mathbb{G} = \langle g \rangle$ of order q. Define the **Schnorr Signature** scheme Σ_{Schnorr} as a triplet of functions specified in Figure 3.6, Figure 3.7, and Figure 3.8.

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$:

1. $\alpha \leftarrow\!\!\$\ \mathbb{Z}_q$;
2. $h \leftarrow g^\alpha \in \mathbb{G}$;
3. **return** (h, α).

Figure 3.6: Schnorr keypair generation

$\mathsf{Sign}(\alpha, m \in \{0,1\}^*) \to \sigma$:

1. $r \leftarrow\!\!\$\ \mathbb{Z}_q$, $a \leftarrow g^r$;
2. $e \leftarrow H(m, a)$;
3. $z \leftarrow r + \alpha e$;
4. **return** $\sigma := (e, z)$.

Figure 3.7: Schnorr signing

$\mathsf{Verify}(h, m, \sigma) \to \{0,1\}$:

1. parse $(e, z) \leftarrow \sigma$;
2. $a' \leftarrow g^z h^{-e}$;
3. **return** 1 iff $e = H(m, a')$.

Figure 3.8: Schnorr signature verification

Why all of a sudden we decided to bring up the Schnorr signature when considering LWE-based signature? Turns out that in the essence, the structure of Dilithium greatly resembles Schnorr signatures! Note that Schnorr signature is a compiled interactive Σ-protocol. In the terminology of interactive protocols, a in a *commitment*, e is a *challenge*, and z is a *response*. Our hope is that we can construct a similar protocol that will have different yet resembling expressions for (a, e, z), but in the "LWE language".

3.2.2 LWE-like Σ-protocol

Fix $(\mathcal{R}, k, \ell, q)$ Module-LWE instance. We are going to require a couple of additional sets to make our scheme work:

- $\mathcal{Q}_\eta$ is a set of polynomials $\sum_{j \in (n)} \alpha_j T^j$ in $\mathcal{R}_q$ with $\|\alpha_j\|_\infty \leq \eta$.
- $\widetilde{\mathcal{Q}}_\gamma$ is a set of polynomials $\sum_{j \in (n)} \alpha_j T^j$ in $\mathcal{R}_q$ with $\|\alpha_j\|_\infty < \gamma$ for relatively large γ.

- $\mathcal{B}_\tau$ is a set polynomials in $\mathcal{Q}_1$, exactly τ of whose coefficients are ± 1.

Additionally set $\nu := \tau\eta$. For instance, for NIST FIPS 204 standard [oSND+24] one chooses parameters $q = 2^{23} - 2^{13} + 1$, $n = 256$, $(k, \ell) = (8, 7)$, $\eta = 2$, $\gamma = 2^{19}$, $\tau = 60$, $\nu = 120$.

First Attempt. Fix hash function $H : \{0,1\}^* \to \mathcal{B}_\tau$. We present the first attempt of building LWE-based Schnorr-like signature in Figure 3.9, Figure 3.10, and Figure 3.11.

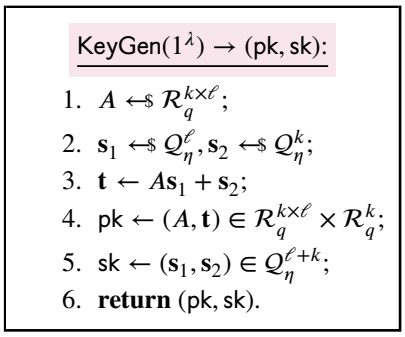

Figure 3.9: First-attempt LWE: key generation

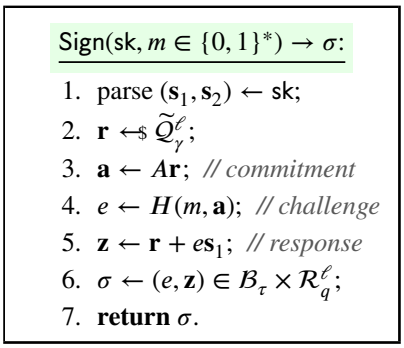

Figure 3.10: First-attempt LWE: signing

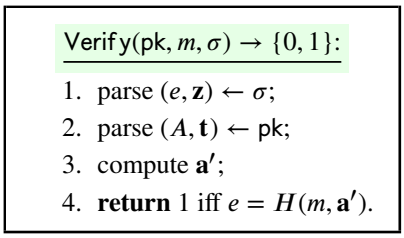

Figure 3.11: First-attempt LWE: signature verification

Notice that the protocol greatly resembles the Schnorr signature scheme previously presented in Definition 3.6. However, the biggest issue so far is step 3 in Figure 3.11: how can the verifier recompute the commitment $\mathbf{a}'$ specifically?

Note that $\mathbf{z} = \mathbf{r} + e\mathbf{s}_1$, thus $A\mathbf{z} = A\mathbf{r} + eA\mathbf{s}_1 = \mathbf{a} + e(\mathbf{t} - \mathbf{s}_2)$ and so we obtain $A\mathbf{z} - e\mathbf{t} = \mathbf{a} - e\mathbf{s}_2$. Unfortunately, this way we only obtain $\mathbf{a} - e\mathbf{s}_2$ where $\mathbf{s}_2$ is unknown to the verifier. However, here where our distributions $\mathcal{Q}_\eta$ and $\mathcal{B}_\tau$ might come into play: notice that $e\mathbf{s}_2$ is a vector of polynomials in $\mathcal{R}_q^k$ with small coefficients (since $\|s_{2,i}\|_\infty \le \eta$ for all $i \in [k]$ and $\|e\|_\infty \le 1$ by construction).

Thus, we might modify our idea in the following way: the signer will use the function called HighBits, which will divide and round each coefficient of $\mathbf{w}$ which hopefully eventually removes the noise $e\mathbf{s}_2$ (we give concrete specifics later on). The signer then composes the challenge $e = H(m, \mathbf{a}_H)$ where $\mathbf{a}_H = \mathsf{HighBits}(\mathbf{a})$.

If LowBits of $\mathbf{a} - e\mathbf{s}_2$ is sufficiently small then, as $e\mathbf{s}_2$ is small, we have $\mathsf{HighBits}(\mathbf{a} - e\mathbf{s}_2) = \mathsf{HighBits}(\mathbf{a})$, and so the verifier can compute $\mathbf{a}'$. This way, we modify the signing procedure by making signer recompute $(e, \mathbf{z})$ while $\mathsf{LowBits}(\mathbf{a} - e\mathbf{s}_2)$ is not sufficiently small.

Second Attempt. We encapsulate observations above in Figure 3.12 and Figure 3.13.

However, this attempt also contains numerous problems we have to mitigate:

- **Issue #1.** Public key $(A, \mathbf{t}) \in \mathcal{R}_q^{k\times\ell} \times \mathcal{R}_q^k$ is very large: we are going to derive A from the public seed ρ and use only "high order bits" of coefficients in $\mathbf{t}$.
- **Issue #2.** Secret key $(\mathbf{s}_1, \mathbf{s}_2) \in \mathcal{Q}_\eta^{\ell+k}$ is similarly too large: we are going to generate these values from the *secret seed* ρ'.
- **Issue #3.** Costly hashing: during signing, on each step we have to perform $H(m, \mathbf{a}_H)$, which is expensive if m is large. Instead, we are going to set $\mu \leftarrow H(m)$ and on each step compute $H(\mu, \mathbf{a}_H)$.
- **Issue #4.** Scheme is non-deterministic (which might be undesirable for some applications) and requires too many random samplings $\mathbf{r} \leftarrow\$ \widetilde{\mathcal{Q}}_\gamma^\ell$. Instead, we will generate $\mathbf{r}$ from the seed ρ'' and counter κ, where ρ'' in turn will be obtained by hashing a secret seed K and μ.

$\mathsf{Sign}(\mathsf{sk}, m \in \{0,1\}^*) \to \sigma$:

1. parse $(\mathbf{s}_1, \mathbf{s}_2) \leftarrow \mathsf{sk}$;
2. **while** $\mathsf{LowBits}(\mathbf{a} - e\mathbf{s}_2)$ is large:
3. $\quad \mathbf{r} \leftarrow\!\!\$\ \widetilde{Q}_\gamma^\ell$;
4. $\quad \mathbf{a} \leftarrow A\mathbf{r}$;
5. $\quad \mathbf{a}_H \leftarrow \mathsf{HighBits}(\mathbf{a})$;
6. $\quad e \leftarrow H(m, \mathbf{a}_H)$;
7. $\quad \mathbf{z} \leftarrow \mathbf{r} + e\mathbf{s}_1$;
8. $\sigma \leftarrow (e, \mathbf{z})$;
9. **return** σ.

Figure 3.12: Second-attempt LWE: signing

$\mathsf{Verify}(\mathsf{pk}, m, \sigma) \to \{0,1\}$:

1. parse $(e, \mathbf{z}) \leftarrow \sigma$;
2. parse $(A, \mathbf{t}) \leftarrow \mathsf{pk}$;
3. $\mathbf{a}'_H \leftarrow \mathsf{HighBits}(A\mathbf{z} - e\mathbf{t})$;
4. **return** 1 iff $e = H(m, \mathbf{a}'_H)$.

Figure 3.13: Second-attempt LWE: signature verification

- **Issue #5.** This protocol is not fully zero-knowledge: by careful inspection of coefficients of $\mathbf{z}$, one might sometimes get some information about these coefficients (e.g., if some coefficient is $\gamma + \nu$, then the corresponding coefficient of $e\mathbf{s}_1$ is ν). The solution is that signer, while producing the signature, will ensure that $\|\mathbf{z}\|_\infty < \gamma - \nu$.

3.2.3 Dilithium Protocol Simplified Specification

HighBits and LowBits. Now, let us define these functions more carefully.

Definition 3.7. Fix even $\alpha \in \mathbb{N}$ such that $q - 1 = m\alpha$. Functions $\mathsf{HighBits}_\alpha(x)$ and $\mathsf{LowBits}_\alpha(x)$ will be informally defined as quotient and remainder of division x by α, respectively. More formally, let $x = x_H\alpha + x_L$ where $|x_L| \le \alpha/2$ and $0 \le x_H \le m$. Then, we set $\mathsf{HighBits}_\alpha(x) := x_H$ and $\mathsf{LowBits}_\alpha(x) := x_L$.

Additional parameters. We slighly modify parameters specified earlier in Section 3.2.2: instead of γ, we use γ_1, and we additionally introduce parameters γ_2 (which in NIST FIPS 204, for instance, is $(q-1)/32 \approx 2^{18}$) and $\alpha := 2\gamma_2$. Additionally, denote by $H(\bullet; d) : \{0,1\}^* \to \{0,1\}^d$ the hash function returning d-bit binary string as an output, $H_{\mathcal{B}}(\bullet) : \{0,1\}^{2\lambda} \to \mathcal{B}_\tau$ the hash function returning polynomial in $\mathcal{B}_\tau$, and by $\leftarrow_\rho$ sampling with seed ρ.

Finally, we are ready to present almost complete specification of the Dilithium protocol.

Definition 3.8 (Dilithium Signature Scheme)**.** We define the **Dilithium** signature scheme $\Sigma_{\text{Dilithium}}$ as a triplet of functions specified in Figure 3.14, Figure 3.15, and Figure 3.16.

Let us again check the *correctness* of the obtained scheme. Notice that

$$A\mathbf{z} - e\mathbf{t} = A(\mathbf{r} + e\mathbf{s}_1) - e(A\mathbf{s}_1 + \mathbf{s}_2) = A\mathbf{r} - e\mathbf{s}_2 = \mathbf{a} - e\mathbf{s}_2.$$

Since $\|\mathsf{LowBits}_{2\gamma_2}(\mathbf{a} - e\mathbf{s}_2)\|_\infty \le \gamma_2 - \nu$ and $\|e\mathbf{s}_2\|_\infty \le \nu$, we have

$$\mathbf{a}'_H = \mathsf{HighBits}_{2\gamma_2}(A\mathbf{z} - e\mathbf{t}) = \mathsf{HighBits}_{2\gamma_2}(\mathbf{a} - e\mathbf{s}_2) = \mathsf{HighBits}_{2\gamma_2}(\mathbf{a}) = \mathbf{a}_H.$$

Thus, valid signatures are accepted.

$\underline{\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk}):}$

1. $(\rho, \rho', K) \leftarrow\!\$\ \{0,1\}^n \times \{0,1\}^{2n} \times \{0,1\}^n$; // Generate seeds
2. $A \leftarrow\!\$_\rho\ \mathcal{R}_q^{k\times\ell}$; // Sample matrix from seed ρ
3. $(\mathbf{s}_1, \mathbf{s}_2) \leftarrow\!\$_{\rho'}\ \mathcal{Q}_\eta^\ell \times \mathcal{Q}_\eta^k$; // Sample secrets from seed ρ'
4. $\mathbf{t} \leftarrow A\mathbf{s}_1 + \mathbf{s}_2$;
5. $\mathsf{tr} \leftarrow H(\rho, \mathbf{t}; 2\lambda)$;
6. $\mathsf{pk} \leftarrow (\rho, \mathbf{t})$; // ρ can be used for restoring A
7. $\mathsf{sk} \leftarrow (\rho', K, \mathsf{tr})$; // ρ' can be used for restoring $\mathbf{s}_1$ and $\mathbf{s}_2$
8. **return** $(\mathsf{pk}, \mathsf{sk})$.

Figure 3.14: Dilithium key generation

$\underline{\mathsf{Sign}(\mathsf{sk}, m) \to \sigma:}$

1. $A \leftarrow\!\$_\rho\ \mathcal{R}_q^{k\times\ell}$; // resample same A from seed ρ
2. $\mu \leftarrow H(\mathsf{tr}, m; 2n)$;
3. $\mathsf{rnd} \leftarrow \mathbf{0}^n$ **if** deterministic mode, $\mathsf{rnd} \leftarrow\!\$\ \{0,1\}^n$ otherwise;
4. $\rho'' \leftarrow H(K, \mathsf{rnd}, \mu; 2n)$;
5. $\kappa \leftarrow 0$; $\mathbf{z} \leftarrow \perp$; // initialize counter
6. **while** $\mathbf{z} = \perp$:
7. $\quad \mathbf{r} \leftarrow\!\$_{(\rho'',\kappa)}\ \widetilde{\mathcal{Q}}_{\gamma_1}^\ell$;
8. $\quad \mathbf{a} \leftarrow A\mathbf{r}$; $\mathbf{a}_H \leftarrow \mathsf{HighBits}_{2\gamma_2}(\mathbf{a})$; // compute high-bits of commitment
9. $\quad \widetilde{e} \leftarrow H(\mu, \mathbf{a}_H; 2\lambda)$;
10. $\quad e \leftarrow H_{\mathcal{B}}(\widetilde{e}) \in \mathcal{B}_\tau$; // compute challenge
11. $\quad \mathbf{z} \leftarrow \mathbf{r} + e\mathbf{s}_1$; // compute response
12. $\quad \boldsymbol{\delta} \leftarrow \mathsf{LowBits}_{2\gamma_2}(\mathbf{a} - e\mathbf{s}_2)$; // compute ''noise'' level
13. $\quad$ **if** $\|\mathbf{z}\|_\infty \geq \gamma_1 - \nu$ or $\|\boldsymbol{\delta}\|_\infty \geq \gamma_2 - \nu$, then set $\mathbf{z} \leftarrow \perp$;
14. $\quad \kappa \leftarrow \kappa + \ell$; // update counter
15. **return** $\sigma = (\widetilde{e}, \mathbf{z})$.

Figure 3.15: Dilithium message signing

$\mathsf{Verify}(\mathsf{pk}, m, \sigma) \to \{0,1\}$:

1. parse $(\rho, \mathbf{t}) \leftarrow \mathsf{pk}$;
2. parse $(\widetilde{e}, \mathbf{z}) \leftarrow \sigma$;
3. **if** $\|\mathbf{z}\|_\infty \geq \gamma_1 - \nu$ then **return** 0;
4. $A \leftarrow\!\!\$_\rho\ \mathcal{R}_q^{k\times\ell}$;
5. $\mathsf{tr} \leftarrow H(\rho, \mathbf{t}; 2n)$; $\mu \leftarrow H(\mathsf{tr}, m; 2n)$;
6. $e \leftarrow H_{\mathcal{B}}(\widetilde{e}) \in \mathcal{B}_\tau$;
7. $\mathbf{a}'_H \leftarrow \mathsf{HighBits}_{2\gamma_2}(A\mathbf{z} - e\mathbf{t})$;
8. **return** 1 **if** $\widetilde{e} = H(\mu, \mathbf{a}'_H; 2\lambda)$ and 0 otherwise.

Figure 3.16: Dilithium signature verification

Finally, we state the security of $\Sigma_{\text{Dilithium}}$ without a proof. Curious reader is of course encouraged to read the original paper [DKL+18] for details.

Proposition 3.9. $\Sigma_{\text{Dilithium}}$ is **zero-knowledge** and **unforgeable against chosen-message** attack assuming (1) Decisional Module-LWE, (2) Module-SIS (Short-Integer Solution)[a], and (3) (Quantum) Random Oracle Model (QROM).

[a] We have not defined it rigorously, but informally we assume finding a *short* solution to $[A|E_{k\times k}]\mathbf{s} = 0$ for $A \in \mathcal{R}_q^{k\times\ell}$ and identity matrix $E_{k\times k}$ is hard.

Acknowledgements

Some of the arguments and ideas used in these notes were particularly inspired by awesome courses "Cryptography 101 with Alfred Menezes"[¶] (Section 3.2 is primarily based on "Kyber and Dilithium" lectures) and lecture materials by François Charles on lattices during ICMU Summer School on Number Theory.

References

[ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange: a new hope. In *Proceedings of the 25th USENIX Conference on Security Symposium*, SEC'16, page 327–343, USA, 2016. USENIX Association.

[BDK+18] Joppe Bos, Leo Ducas, Eike Kiltz, T Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehle. Crystals - kyber: A cca-secure module-lattice-based kem. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 353–367, 2018. doi:10.1109/EuroSP.2018.00032.

[BS23] Dan Boneh and Victor Shoup. *A Graduate Course in Applied Cryptography*. 2023. URL: https://toc.cryptobook.us/.

[dBvW25] Koen de Boer and Wessel van Woerden. Lattice-based cryptography: A survey on the security of the lattice-based NIST finalists. Cryptology ePrint Archive, Paper 2025/304, 2025. URL: https://eprint.iacr.org/2025/304.

[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 Transactions on*

[¶] See https://cryptography101.ca/.

Cryptographic Hardware and Embedded Systems, 2018(1):238–268, Feb. 2018. URL: https://tches.iacr.org/index.php/TCHES/article/view/839, doi:10.13154/tches.v2018.i1.238-268.

[LLZ+18] Xianhui Lu, Yamin Liu, Zhenfei Zhang, Dingding Jia, Haiyang Xue, Jingnan He, Bao Li, and Kunpeng Wang. LAC: Practical ring-LWE based public-key encryption with byte-level modulus. Cryptology ePrint Archive, Paper 2018/1009, 2018. URL: https://eprint.iacr.org/2018/1009.

[MG02] Daniele Micciancio and S. Goldwasser. *Complexity of Lattice Problems*. Kluwer Academic Publishers, USA, 2002.

[oSND+24] National Institute of Standards, Technology (NIST), Thinh Dang, Jacob Lichtinger, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, and Angela Robinson. Module-lattice-based digital signature standard, 2024-08-13 04:08:00 2024. URL: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=958463, doi:10.6028/NIST.FIPS.204.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.