Université de Rennes

Lattice-Based Cryptography

M2 Graduate Course, Research Specialization - Rennes University

Corentin Jeudy (Orange)
corentin.jeudy@orange.com

Contents

Part I

Security Notions

This first part recalls the common security notions necessary in public key cryptography.

1

Definition of Security

Cryptography is a science which aims at designing algorithms that achieve specific security properties. For example, the goal of encryption is to provide confidentiality of data (at rest or in transit), while that of digital signature is to ensure integrity and authenticity of data. It is thus necessary to define algorithms and protocols but also formally model the security notions we expect from them. In public-key cryptography, the security of primitives is based on computational assumptions which must be well-defined and widely studied so that we can trust the security of our constructions. For example, the encryption (and signature) RSA are linked to the factorization problem, while the Diffie-Hellman key exchange is based on the discrete logarithm problem. Finally, each construction requires a proof of security, meaning a proof that the primitive meets the security requirements defined by the model. Usually, we prove that if an adversary can attack the construction (the type of attack is defined by the security model), then there exists an efficient algorithm that can solve a hard computational problem.

The protocols should be public and analyzed by the community to be considered secure. The security must rely on the underlying computational assumption and the good choice of the key, and not on the secrecy of the algorithm itself.

Contents

1.1 Perfect Security

1.1.1 One Time Pad

Before entering the details of security models used in cryptography, we start by the historical result of Shannon of a perfectly secure cipher. A cipher is the usual term to denote a symmetric encryption scheme.

Definition 1.1 (Symmetric Encryption Scheme)

Let $\mathcal{K}$, $\mathcal{M}$ and $\mathcal{C}$ be three sets called key space, plaintext (or message) space and ciphertext space respectively. A symmetric encryption scheme is defined by three algorithms KeyGen,

Enc and Dec as follows.

- KeyGen: Selects a key $k \in \mathcal{K}$.
- Enc: Takes a key $k \in \mathcal{K}$, a plaintext $m \in \mathcal{M}$ and outputs a ciphertext $c \in \mathcal{C}$.
- Dec: Takes a key $k \in \mathcal{K}$, a ciphertext $c \in \mathcal{C}$ and outputs a plaintext $m \in \mathcal{M}$.

The encryption scheme must be *correct*, i.e., verifying

$$\forall k \in \mathcal{K}, \forall m \in \mathcal{M}, \mathsf{Dec}(k, \mathsf{Enc}(k, m)) = m$$

Claude Shannon defined the notion of perfect secrecy. It captures the idea that an adversary cannot learn anything about a message m given a ciphertext of m alone, even if it knows the probability distribution of the plaintexts.

Definition 1.2 (Perfect Secrecy)

Let $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a symmetric cipher with key space $\mathcal{K}$, message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$. Let P be a probability distribution on $\mathcal{M}$ and Q be the probability distribution defined by $\mathsf{Enc}(\mathsf{KeyGen}(), P)$. Let M, C be random variables distributed according to P and Q respectively. The cipher is said perfectly secure if for all $m \in \mathrm{Supp}(P)$ and all $c \in \mathrm{Supp}(Q)$, it holds that

$$\mathbb{P}[M = m | C = c] = \mathbb{P}[M = m]$$

We can then construct the *One Time Pad* and prove that it is both correct and perfectly secure. Let use define the three algorithms. Let ℓ be a positive integer defining $\mathcal{K} = \mathcal{M} = \mathcal{C} = \{0,1\}^\ell$.

Algorithm 1.1: KeyGen (One Time Pad)

Input: 1^ℓ

1. $k \hookleftarrow U(\{0,1\}^\ell)$

Output: k

Algorithm 1.2: Enc (One Time Pad)

Input: Key $k \in \{0,1\}^\ell$, Message $m \in \{0,1\}^\ell$.

1. $c \leftarrow m \oplus k$

Output: c

Algorithm 1.3: Dec (One Time Pad)

Input: Key $k \in \{0,1\}^\ell$, Ciphertext $c \in \{0,1\}^\ell$.

1. $m \leftarrow c \oplus k$

Output: m

Theorem 1.1 (One Time Pad Security)

The One Time Pad cipher defined in Algorithms 1.1, 1.2 and 1.3 is correct and perfectly secure.

Proof (Theorem 1.1). Let k, m be in $\{0,1\}^\ell$. By involution, it holds that $(m \oplus k) \oplus k = m$ and therefore $\mathsf{Dec}(k, \mathsf{Enc}(k, m)) = m$. Now, let K, M be independent random variables following $U(\{0,1\}^\ell)$ and some distribution P respectively, and $C = \mathsf{Enc}(K, M)$. Let m, c be in $\{0,1\}^\ell$.

It holds that

$$\begin{aligned}\mathbb{P}[M \oplus K = c | M = m] &= \mathbb{P}[m \oplus K = c] \\ &= \mathbb{P}[K = m \oplus c] \\ &= 2^{-\ell}.\end{aligned}$$

Now, we can compute $\mathbb{P}[M = m | C = c]$ as follows.

$$\begin{aligned}\mathbb{P}[M = m | C = c] &= \frac{\mathbb{P}[M = m \wedge M \oplus K = c]}{\mathbb{P}[M \oplus K = c]} \\ &= \frac{\mathbb{P}[M = m \wedge K = c \oplus m]}{\sum_{m \in Supp(P)} \mathbb{P}[M \oplus K = c | M = m]} \\ &= \frac{\mathbb{P}[M = m]\mathbb{P}[K = c \oplus m]}{2^{-\ell}} \\ &= \mathbb{P}[M = m].\end{aligned}$$

As a result, the One Time Pad is correct and perfectly secure as claimed.

Since we have a simple encryption scheme that is perfectly secure, it is natural to ask why cryptography is still an active field of research. The first thing is that in the One Time Pad, the key must be the same length as the message. Thence, when encrypting a large amount of data, one must use very large keys which is highly impractical. Additionally, each key can only be used once. Otherwise, when a pair (m, c) is known, one can find the key k. Additionally, if two messages m_1, m_2 are encrypted with the same key k, one can learn $m_1 \oplus m_2 = c_1 \oplus c_2$ which leaks information about m_1 and m_2, thus violating perfect security. This is due to the fact that the One Time Pad is not perfectly secure if one considers the joint probability of two ciphertext of independent messages but with the same key.

Additionally, the One Time Pad offers a solution only for the confidentiality of data, but many other problems remain. The first is how to have two distant users agree on the same key k, especially if this key can only be used once. The other natural question is how to guarantee that the message originates from the correct sender. And how to guarantee that the message has not been tampered with? For example, if an adversary intercepts $c = m \oplus k$ and forwards $c \oplus 1$, then the decryption will be $m \oplus 1$. Hence, without learning anything on m nor k, an adversary is able to flip bits in the decrypted message. As a result, the integrity of the transmitted message is not guaranteed.

These problems are dealt with other cryptographic algorithms and protocols, which are essential to satisfy all the security properties we expect.

1.1.2 Shannon's Theorem

To go further, Shannon derived a sufficient and necessary condition to have perfectly secure ciphers. For that, we first need the following lemma.

Lemma 1.1

Let $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a cipher with key space $\mathcal{K}$, message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$. We call $\mathcal{C}'$ the set of reachable ciphertext, i.e., $\mathcal{C}' = \mathsf{Enc}(\mathcal{K}, \mathcal{M})$. If the cipher is correct and perfectly secure, then it holds that $|\mathcal{K}| \geq |\mathcal{C}'| \geq |\mathcal{M}|$.

Proof (Lemma 1.1). Since the cipher is correct, the encryption function is necessarily injective. Otherwise, either it yields a contradiction on the correctness or the decryption function is ill-defined. Hence $|\mathcal{C}'| \geq |\mathcal{M}|$.

Then, let m be in $\mathcal{M}$. By perfect security, it holds that for all $c \in \mathcal{C}'$ we have

$$\mathbb{P}[C = c | M = m] = \frac{\mathbb{P}[C = c \wedge M = m]}{\mathbb{P}[M = m]} = \frac{\mathbb{P}[C = c]\mathbb{P}[M = m | C = c]}{\mathbb{P}[M = m]} = \mathbb{P}[C = c] > 0$$

It means that for all reachable ciphertext c, there exists a key $k \in \mathcal{K}$ such that $\mathsf{Enc}(k, m) = c$. Hence $|\mathcal{K}| \geq |\mathcal{C}'|$.

Theorem 1.2 (Shannon's Theorem)

Let $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a cipher with key space $\mathcal{K}$, message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$ such that $|\mathcal{K}| = |\mathcal{M}| = |\mathcal{C}|$. The cipher is pefectly secure if and only if (1) the output distribution of KeyGen is $U(\mathcal{K})$, and (2) for each $(m, c) \in \mathcal{M} \times \mathcal{C}$, there exists a unique $k \in \mathcal{K}$ such that $c = \mathsf{Enc}(k, m)$.

1.2 Defining Security

In the previous section, we defined one specific notion of security for symmetric ciphers called perfect security. But as Shannon's theorem shows (Theorem 1.2), it is quite an impractical security notion. We now dive into other forms of security. For that, we first present a few mathematical tools and notations that are going to be helpful.

1.2.1 Perfect Security versus Computational Security

We showed that the One Time Pad mathematically verifies our notion of perfect security. This implies that even an all-powerful adversary with unlimited computing power will not be able to attacks the cipher. The security is due to the fact that the adversary does not have enough information to attack the primitive. We have also seen why it is not usable in practice. We thus need to find new security notions. In modern cryptography, the schemes can be attacked but in an unreasonable amount of time (say the age of the universe). This level of security is enough for practical security. We call it *computational security*. Note that **computational security is weaker than perfect security**.

What this means in practice os that we consider *efficient* adversaries which are limited to running algorithms that terminate in a reasonable amount of time, and we accept that an adversary can succeed in attacking with a very low probability.

Example 1.1 (Time Scales)

We usually measure time with CPU cycles. A computer running at 3.4 GHz performs $3.4 \cdot 10^9$ cycles per second. Imagining 2^{60} cycles, it would take this computer around $2^{60}/(3.4 \cdot 10^9) \approx 3.4 \cdot 10^8$ seconds which is around 11 years. By parallelizing, it may be dropped down to less than a year. For reference, 2^{58} is the estimate of the number of seconds since the big bang. Regarding success probabilities, an event that occurs with probability 2^{-60} every second should thus occur on average once every 100 billion years.

Asymptotic Approach.

The computation time and success probability of an adversary are usually defined as functions of a security parameter λ. The asymptotic approach thus categorizes the scales as asymptotic behaviors. For that, let us recall the usual Landau notations.

Definition 1.3 (Landau Notations)

Let f, g be two functions from $\mathbb{N}$ to $\mathbb{R}^\times$.

- $f = O(g) \iff \exists K \in \mathbb{R}, \exists \lambda_0, \forall \lambda \geq \lambda_0,\ f(\lambda) \leq K g(\lambda)$.
- $f = o(g) \iff \lim_{\lambda \to +\infty} f(\lambda)/g(\lambda) = 0$.
- $f = \Omega(g) \iff g = O(f)$.
- $f = \omega(g) \iff g = o(f)$.
- $f = \Theta(g) \iff f = O(g) \wedge g = O(f)$.

In the asymptotic approach, we say that algorithms run in time polynomial in λ if there exists a constant c such that the algorithm's running time is $O(\lambda^c)$. For success probabilities, we say they are negligible in λ if for all constants c, the success probability is $o(\lambda^{-c})$. Throughout this course negl is usually a negligible function, i.e., $\mathsf{negl}(\lambda) = \lambda^{-\omega(1)}$. Adding two negligible functions

stays negligible, and multiplying a negligible function with a polynomial one stays negligible as well. Examples of negligible functions are $2^{-\lambda}$, $2^{-\sqrt{\lambda}}$ or $\lambda^{-\log \lambda}$.

Concrete Approach.

The asymptotic approach is interesting to understand coarse grain behavior and have a quick idea of the expected behavior. However, it usually hides several elements. Asymptotic notations are relative to constants, and additionally, the polynomial categorization does not say anything about the degree of the polynomial. For example λ^{1000} grows much faster than λ^2, but they are both polynomial functions of λ. In concrete instantiation of cryptographic schemes, the asymptotic approach may be insufficient to obtain a fine-grained behavior.

To have more detailed analyses, we usually define a security standard with the security parameter λ that takes a concrete value. For example, one can choose $\lambda = 128$, in which case we talk about 128 *bits of security*. This means that the best attacks should run in time at least $2^{\lambda} = 2^{128}$ operations. One could also define other cost models which are relevant for adversaries. The ones that are usually considered are memory (amount of storage needed to run an algorithm successfully) and time. Typical security parameters considered in cryptography are therefore $\lambda = 128$, $\lambda = 256$, $\lambda = 512$. They are very conservative in the sense that the former is already more than enough to guarantee that an attacker would need a highly unreasonable amount of ressource (time or memory) to break the security of the scheme. Certain more recent algorithms are categorized as *lightweight* and use $\lambda = 80$. In our case, we will take $\lambda \geq 128$.

Computational Security.

Depending on the desired metrics (time, memory, electric consumption, etc.), we can define what we mean by *computational security*.

Definition 1.4 (Computational Security)

A cryptographic primitive is called (t, m, ε)-secure the probability of an adversary of successfully attacking the primitive in time at most t, with at most m units of memory, is at most ε.

1.2.2 Indistinguishability

We usually consider probability distributions when talking about security. For example, if the ciphertexts are distributed uniformly in the ciphertext space, it means they look random to an adversary and thus do not leak information about the underlying message. But if they are not exactly uniform but close to it, it may give us security guarantees as well. For that, we need to define notions of closeness more formally.

Statistical Indistinguishability.

There are several ways of measuring how close two probability distributions are. A very usual tool in cryptography is called the *statistical distance*, which we define here.

Definition 1.5 (Statistical Distance)

Let S be a countable set, and X and Y be two discrete random variables taking values in S. The statistical distance between X and Y (or the distributions they represent) is defined by

$$\Delta(X, Y) = \frac{1}{2} \sum_{s \in S} \left| \mathbb{P}[X = s] - \mathbb{P}[Y = s] \right|.$$

If X and Y are continuous random variables over a set S and with respective probability density functions f_X and f_Y, the statistical distance between X and Y is defined as

$$\Delta(X, Y) = \frac{1}{2} \int_{s \in S} \left| f_X(s) - f_Y(s) \right| ds.$$

The statistical distance verifies the usual properties of a distance as reminded in the following lemma.

Lemma 1.2

Let X, Y, Z be three random variables taking values in a set S. It holds that (1) $\Delta(X,Y) \geq 0$, (2) $\Delta(X,Y) = 0 \Leftrightarrow X$ and Y are identically distributed, (3) $\Delta(X,Y) = \Delta(Y,X)$ and (4) $\Delta(X,Z) \leq \Delta(X,Y) + \Delta(Y,Z)$.

Proof (Lemma 1.2). Let X, Y, Z be three random variables taking values in a set S. It trivially follows that $\Delta(X,Y) = \frac{1}{2}\sum_{s\in S}\left|\mathbb{P}[X=s] - \mathbb{P}[Y=s]\right| \geq 0$, thus yielding (1).

For (2), first assume that $\Delta(X,Y) = 0$. Since it is a sum of positive reals, it holds that for all $s \in S$, $\mathbb{P}[X=s] = \mathbb{P}[Y=s]$. Reciprocally, assume that X and Y are identically distributed. Then each term of the sum is 0 and therefore $\Delta(X,Y) = 0$.

For (3), it directly holds that

$$\Delta(X,Y) = \frac{1}{2}\sum_{s\in S}\left|\mathbb{P}[X=s] - \mathbb{P}[Y=s]\right| = \frac{1}{2}\sum_{s\in S}\left|\mathbb{P}[Y=s] - \mathbb{P}[X=s]\right| = \Delta(Y,X).$$

Finally, we have

$$\begin{aligned}
\Delta(X,Z) &= \frac{1}{2}\sum_{s\in S}\left|\mathbb{P}[X=s] - \mathbb{P}[Z=s]\right| \\
&= \frac{1}{2}\sum_{s\in S}\left|\mathbb{P}[X=s] - \mathbb{P}[Y=s] + \mathbb{P}[Y=s] - \mathbb{P}[Z=s]\right| \\
&\leq \frac{1}{2}\sum_{s\in S}\left|\mathbb{P}[X=s] - \mathbb{P}[Y=s]\right| + \left|\mathbb{P}[Y=s] - \mathbb{P}[Z=s]\right| \\
&= \Delta(X,Y) + \Delta(Y,Z),
\end{aligned}$$

as claimed in (4).

We also have the following properties which are extremely important when comparing distributions. The following will possibly be proven in the exercise sessions.

Lemma 1.3

Let X, Y be two discrete random variables over a countable set S, and Z a discrete random variable over a countable set S' (possibly different from S). If Z is independent of X and Y, then $\Delta((X,Z),(Y,Z)) = \Delta(X,Y)$.
For any positive integer k and tuples of random variables $(X_i)_{1\leq i\leq k}, (Y_i)_{1\leq i\leq k}$ over a tuple $(S_i)_{1\leq i\leq k}$ of countable sets, if all the X_i and Y_i are independent, then

$$\Delta((X_i)_{1\leq i\leq k}, (Y_i)_{1\leq i\leq k}) \leq \sum_{i=1}^{k}\Delta(X_i,Y_i).$$

Finally, for two random variables X, Y over a countable set S, and for any (possibly randomized) function $f : S \to S'$ with S' a countable set, it holds that $\Delta(f(X), f(Y)) \leq \Delta(X,Y)$.

The statistical distance also benefits from a probability preservation property which is essential when using the statistical distance in reductions or security proof.

Lemma 1.4 (Probability Preservation)

Let P and Q be two discrete probability distributions over a countable set S, and $E \subseteq S$ be an arbitrary event. Then it holds that

$$P(E) \leq \Delta(P,Q) + Q(E).$$

We can now define the notion of statistical indistinguishability.

Definition 1.6 (Statistical Indistinguishability)

Let I be a (possibly infinite) set of indices. Let $(X_i)_{i\in I}$ and $(Y_i)_{i\in I}$ be two families of discrete random variables over a family $(S_i)_{i\in I}$ of countable sets. We say that $(X_i)_{i\in I}$ and $(Y_i)_{i\in I}$ are statistically indistinguishable if there exists a negligible function negl, and an index $i_0 \in I$ such that for all $i \geq i_0$, $\Delta(X_i, Y_i) \leq \mathsf{negl}(i)$.

This definition follows the asymptotic approach. In a concrete approach, we generally say that two distributions P, Q are statistically indisguishable if $\Delta(P, Q) \leq 2^{-\lambda}$ where λ is the concrete security parameter. The function $2^{-\lambda}$ can be replaced by any concretely negligible function of λ.

Computational Indistinguishability.

We can also define a notion of computational indistinguishability. Informally, two probability distributions are computationally indistinguishable if no efficient algorithm can distinguish between them.

Definition 1.7 (Computational Indistinguishability)

Let I be a (possibly infinite) set of indices. Let $(X_i)_{i\in I}$ and $(Y_i)_{i\in I}$ be two families of discrete random variables over a family $(S_i)_{i\in I}$ of countable sets. We say that $(X_i)_{i\in I}$ and $(Y_i)_{i\in I}$ are computationally indistinguishable if for all polynomial time distinguisher D, there exists a negligible function negl, and an index $i_0 \in I$ such that for all $i \geq i_0$,

$$\left|\mathbb{P}[D(1^i, X_i) = 1] - \mathbb{P}[D(1^i, Y_i) = 1]\right| \leq \mathsf{negl}(i)$$

Notice that the distinguisher is given i in unary encoding so that it knows it is allowed to run in time at most polynomial in i.

Rényi Divergence.

There are several other ways of measuring the closeness of two probability distributions. One measure that has been more and more frequently used in cryptography is called the Rényi divergence [Rén61, vEH14]. The Rényi divergence has been thoroughly studied for its use in cryptography as powerful alternative for the statistical distance.

Definition 1.8 (Rényi Divergence)

Let P and Q be two discrete probability distributions over a countable set S such that $\mathrm{Supp}(P) \subseteq \mathrm{Supp}(Q)$. Let α be a real such that $\alpha > 1$. The *Rényi Divergence of order* α is defined as

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

The Rényi divergence also benefits from some interesting properties such as probability preservation, multiplicativity and data processing, which we summarize in the following lemmata.

Lemma 1.5 (Probability Preservation)

Let P and Q be two discrete probability distributions over a countable set S such that $\mathrm{Supp}(P) \subseteq \mathrm{Supp}(Q)$. Let α be a real such that $\alpha > 1$, and $E \subseteq \mathrm{Supp}(Q)$ be an arbitrary event. Then it holds that

$$P(E)^{\frac{\alpha}{\alpha-1}} \leq \mathrm{RD}_\alpha(P\|Q) \cdot Q(E).$$

Lemma 1.6 (Rényi Divergence Properties)

Let X and Y be two discrete random variables over a countable set S such that $\mathrm{Supp}(X) \subseteq \mathrm{Supp}(Y)$, and $f : S \longrightarrow S'$ a (possibly randomized) function with S' a countable set. Let α

be a real such that $\alpha > 1$. It then holds that

$$\mathrm{RD}_\alpha(f(X)\|f(Y)) \leq \mathrm{RD}_\alpha(X\|Y).$$

Then, for $(P_i)_{i\in[\![1,n]\!]}, (Q_i)_{i\in[\![1,n]\!]}$ two families of independent discrete probability distributions over a family of countable sets $(S_i)_{i\in[\![1,n]\!]}$ and such that $\mathrm{Supp}(P_i) \subseteq \mathrm{Supp}(Q_i)$ for all i in $[\![1,n]\!]$, it holds that

$$\mathrm{RD}_\alpha(\otimes_{i\in[\![1,n]\!]}P_i\| \otimes_{i\in[\![1,n]\!]} Q_i) = \prod_{i\in[\![1,n]\!]} \mathrm{RD}_\alpha(P_i\|Q_i),$$

where $\otimes_{i\in[\![1,n]\!]}P_i$ denotes the joint probability distribution of the family $(P_i)_{i\in[\![1,n]\!]}$ and similarly for $(Q_i)_{i\in[\![1,n]\!]}$.

1.3 Security Models

Lattice-based cryptography is part of public-key cryptography. As such, we will focus on defining security models for public-key primitives. Security models are defined to capture real-life scenarios and to theoretically model the behavior of real-life attackers. If a security model is meaningless with respect to real-life situations, it should not be considered and overhauled. New cryptographic primitives arise as cryptography is a dynamic research area. As a result, new security models based on the use of such primitives are proposed. It is therefore hard to be exhaustive when it comes to all the possible security models. In this course, we focus on basic primitives such as public-key encryption and digital signatures.

A security model is the combination of (1) an attacker model, (2) an attack setting, and (3) an objective. The attacker model (1) defines the adversary's capabilities and resources in terms of computation time, computational power, memory availability, energy consumption, etc. The attack setting (2) establishes what the adversary has access to, e.g., only the public key, or the public key and intercepted ciphertexts, etc. Finally, the objective (3) fixes the goal of the adversary, e.g., recover the secret key, forge a signature, learn some information from the ciphertext, etc. In general, we only focus on (2) and (3) as we consider probabilistic polynomial-time (PPT) adversaries that can be modeled as Turing machines. By polynomial-time, we mean that the adversary has limited amount of time to perform the attack which is polynomial in the security parameter λ. We note that the attacker model has to be changed when evaluating quantum security. Indeed, the attacker may have access to quantum queries, quantum random access memory, and so on. We do not consider these quantum models in this course.

1.3.1 Public-Key Encryption

As introduced in the seminal paper of Diffie and Hellman [DH76], public-key cryptography is a fruitful area of cryptography. As opposed to symmetric cryptography, the users do not need to agree upon a shared secret key to communicate. Instead, they each possess a key-pair composed of public key pk, that they publish on a public-key infrastructure that everybody can access, and an associated secret key sk that they keep to themselves only. Both keys are usually linked mathematically, and recovering the secret key from the public key alone usually involves solving a hard mathematical problem. The first realization of public-key cryptography is the famous RSA [RSA78] encryption algorithm which relies on the factorization problem.

Definition 1.9 (Public-Key Encryption)

A public-key encryption (PKE) scheme is defined by three algorithms KeyGen, Enc and Dec which are described as follows.

- KeyGen: Takes a security parameter λ and outputs a public key pk and the associated secret key sk.
- Enc: Takes a public key pk, a plaintext m and outputs a ciphertext $c = \mathsf{Enc}(\mathsf{pk}, m)$.
- Dec: Takes a secret key sk, a ciphertext c and outputs a plaintext $m = \mathsf{Dec}(\mathsf{sk}, c)$.

The encryption scheme must be *correct*, i.e., verifying

$$\forall(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda), \forall m, \mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) = m$$

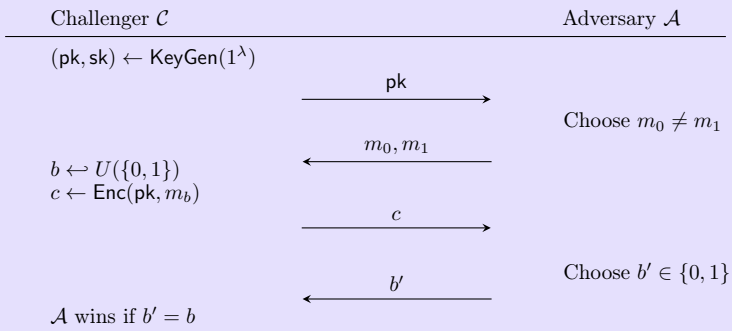
There are several possible security models for public-key encryption schemes. They usually are the association of a goal and attacker model, e.g., OW-CPA, IND-CPA, IND-CCA1. In this course, we focus on the security models that imply most (if not all) others. The easiest goal for an attacker is to know part or a function of the encrypted plaintext. The notion that formalizes this idea is the *indistinguishability of ciphertexts* or semantic security, as coined by Goldwasser and Micali in 1982 [GM82]. It captures the idea that an attacker that knows two messages and who is given the encryption of one of them cannot determine which message was encrypted.

IND-CPA Security.

We now need to combine this goal with an attack model. The first one we consider is that of *chosen plaintext attacks* (CPA). Knowing the public key allows the attacker to select plaintexts and encrypt them themselves, but not decrypting them.

Definition 1.10 (IND-CPA Security)

Let KeyGen, Enc and Dec define a public-key encryption scheme. We define the following experiment.



The advantage of the adversary $\mathcal{A}$ in the above game is defined as

$$\mathrm{Adv}_{IND-CPA}[\mathcal{A}] = \left|\mathbb{P}[b' = b] - \frac{1}{2}\right|.$$

We say that the public-key encryption scheme is (t, ε)-IND-CPA secure if for all probabilistic adversary $\mathcal{A}$ running in time at most t, it holds that $\mathrm{Adv}_{IND-CPA}[\mathcal{A}] \leq \varepsilon$.

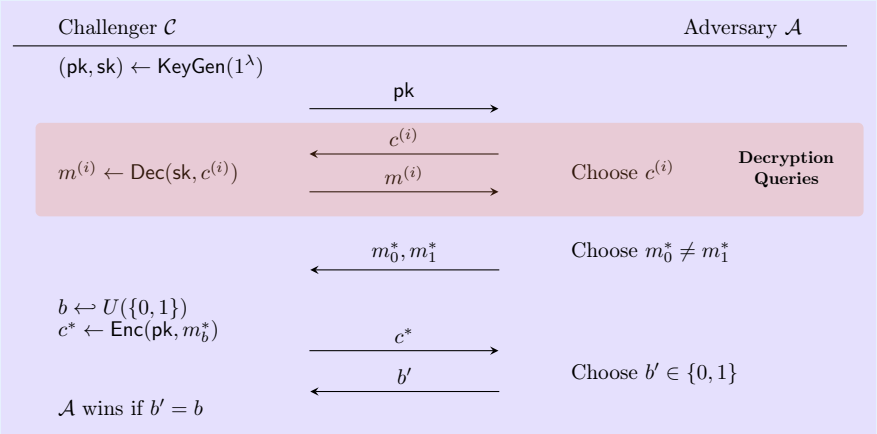
The advantage captures the fact that $\mathcal{A}$ can always make a random guess on which message has been encrypted. Then the probability of winning is $1/2$ which gives them an advantage of 0. The CPA attack models the fact that the attacker can know (plaintext,ciphertext) pairs which is usually true as they have access to the public key. Additionally, we see that the encryption scheme must be randomized to have a chance of meeting the IND-CPA security requirement. Indeed, assume the function Enc is deterministic. In the game, the adversary can compute $\mathsf{Enc}(\mathsf{pk}, m_0)$ and $\mathsf{Enc}(\mathsf{pk}, m_1)$ on their own and compare the result to c to determine the value of b. Randomizing Enc thwarts this attack as each computation of $\mathsf{Enc}(\mathsf{pk}, m_b)$ results in different ciphertexts. Finally, this notion implies that a single key pair can be used several times, which was the primary obstacle of the One Time Pad.

IND-CCA Security.

We can define a stronger security model that allows more freedom to the adversary than in the CPA attack model. We can allow the attacker to make decryption queries before (CCA1) and after (CCA2) having received the challenge ciphertext c (as long as the ciphertexts sent to the decryption queries are different from c). In that context, we talk about *chosen ciphertext attacks* (CCA).

Definition 1.11 (IND-CCA1 Security)

Let KeyGen, Enc and Dec define a public-key encryption scheme. We define the following experiment.



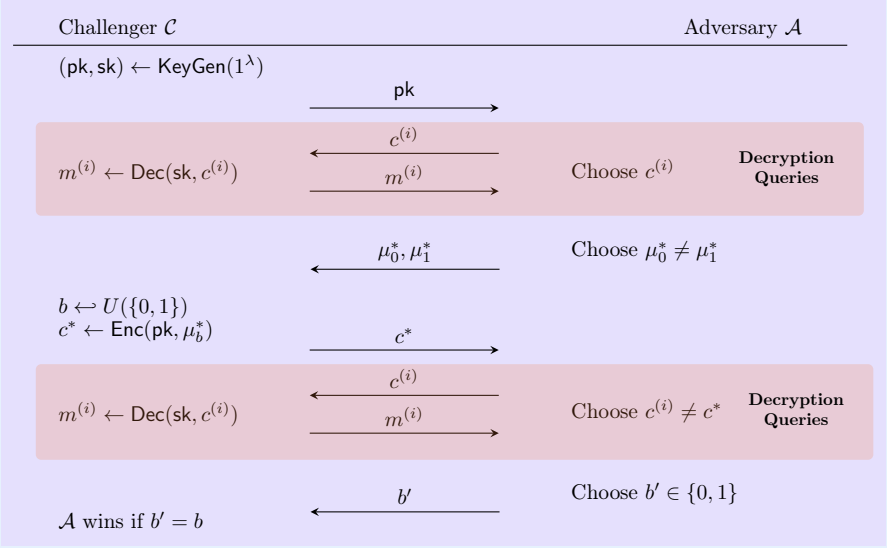
The advantage of the adversary $\mathcal{A}$ in the above game is defined as

$$\mathrm{Adv}_{IND-CCA1}[\mathcal{A}] = \left| \mathbb{P}[b' = b] - \frac{1}{2} \right|.$$

We say that the public-key encryption scheme is (t, ε)-IND-CCA1 secure if for all probabilistic adversary $\mathcal{A}$ running in time at most t, it holds that $\mathrm{Adv}_{IND-CCA1}[\mathcal{A}] \leq \varepsilon$.

Definition 1.12 (IND-CCA2 Security)

Let KeyGen, Enc and Dec define a public-key encryption scheme. We define the following experiment.



The advantage of the adversary $\mathcal{A}$ in the above game is defined as

$$\mathrm{Adv}_{IND-CCA2}[\mathcal{A}] = \left| \mathbb{P}[b' = b] - \frac{1}{2} \right|.$$

We say that the public-key encryption scheme is (t, ε)-IND-CCA2 secure if for all probabilistic adversary $\mathcal{A}$ running in time at most t, it holds that $\mathrm{Adv}_{IND-CCA2}[\mathcal{A}] \leq \varepsilon$..

1.3.2 Digital Signature

Although public-key encryption schemes ensure the confidentiality of messages, it does not guarantee the integrity or authenticity of data. This means the receiver has no guarantee that the ciphertext it received originated from the correct sender, nor does they know whether or not it was tampered with. For this very purpose, another type of primitive called *digital signature* was introduced, and which can be traced back to the seminal work of Diffie and Hellman [DH76]. They act as a certificate that the signed data is authentic, and they represent a digital version of handwritten signatures. Informally, a signature is produced on a message using the secret key so that only the owner of said key can certify data, and the signature can be verified using the message and the public key, making it verifiable by everybody.

Definition 1.13 (Digital Signature)

A digital signature scheme is defined by three algorithms KeyGen, Sign and Verify which are described as follows.

- KeyGen: Takes a security parameter λ and outputs a public key pk and the associated secret key sk.
- Sign: Takes a secret key sk, a message m and outputs a signature $\mathsf{sig} = \mathsf{Sign}(\mathsf{sk}, m)$.
- Verify: Takes a public key pk, a message m and a signature sig on said message, and outputs a bit $b = \mathsf{Verify}(\mathsf{pk}, m, \mathsf{sig})$ which is 1 if the signature is valid, and 0 otherwise.

The signature scheme must be *correct*, i.e., verifying

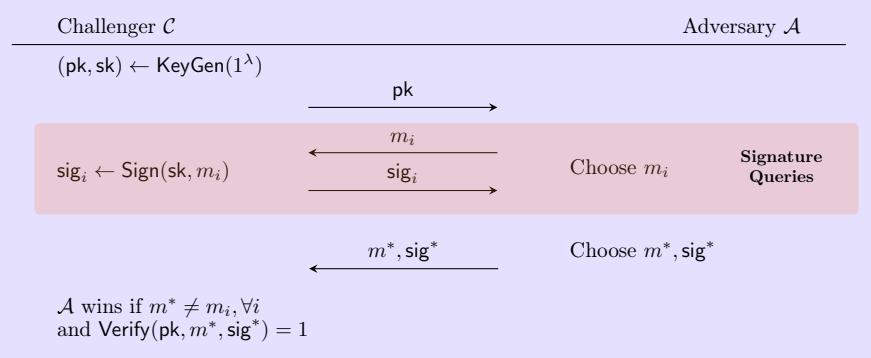
$$\forall(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda), \forall m, \mathsf{Verify}(\mathsf{pk}, m, \mathsf{Sign}(\mathsf{sk}, m)) = 1.$$

EUF-CMA Security.

The most widely used security model for digital signature schemes is the *existential unforgeability against chosen message attacks* (EUF-CMA). The security objective (EUF) is to produce a valid signature on a message of its choice, without having seen prior signature on said message. The attack context allows the adversary to query signatures on messages of its choice (CMA). This model then captures the idea that an adversary that only knows the public key is incapable of producing valid signatures on any message, even one of its choosing. It thus guarantees that nobody is able to usurp the identity of a signer and certify data in their name. The security model is formally defined in a three stage game.

Definition 1.14 (EUF-CMA Security)

Let KeyGen, Sign and Verify define a digital signature scheme. We define the following experiment.



The advantage of the adversary $\mathcal{A}$ in the above game is defined as

$$\mathrm{Adv}_{EUF-CMA}[\mathcal{A}] = \mathbb{P}[\mathsf{Verify}(\mathsf{pk}, m^*, \mathsf{sig}^*) = 1 \wedge \forall i, m^* \neq m_i].$$

We say that the public-key encryption scheme is (t, ε)-EUF-CMA secure if for all probabilistic adversary $\mathcal{A}$ running in time at most t, it holds that $\mathrm{Adv}_{EUF-CMA}[\mathcal{A}] \leq \varepsilon$.

We can define a stronger version of this model, providing further security guarantees. We then talk about *strong* EUF-CMA, also denoted sEUF-CMA. The only difference lies in the winning condition. We require that the forged signature is a valid signature on m^*, but we allow m^* to be one the messages m_i used in the signing queries as long as $\mathsf{sig}^* \neq \mathsf{sig}_i$. Said differently, the adversary is allowed to query a signature on m^*, but must produce a new one in the end. We summarize this condition saying that $\mathcal{A}$ wins the sEUF-CMA if $\mathsf{Verify}(\mathsf{pk}, m^*, \mathsf{sig}^*) = 1$ and $(m^*, \mathsf{sig}^*) \neq (m_i, \mathsf{sig}_i)$ for all the queries i (which implies that either m^* has never been queried as in the EUF-CMA game, or that it has been queried but the forged signature differs from that obtained in said query). The security sEUF-CMA is, as its name suggests, stronger than EUF-CMA security. More precisely, sEUF-CMA implies EUF-CMA.

1.3.3 Hash Functions

Along the same lines as commitment scheme, another family of tools omnipresent in cryptography is the hash functions. They are essential in many signature designs that we will see later on, especially when they are modeled as random oracles. The idea of a hash function is to compress an arbitrarily long string into a *digest* of fixed size. When we consider *cryptographic hash functions*, we also expect it to satisfy security requirements in addition to this compression feature. Typically, we want it to be *one-way*, meaning that given a digest, an adversary cannot find the input string used to compute it (uninvertibility), or any other input string that gives the given digest (second preimage resistance). We also usually expect it to be *collision-resistant*, i.e., it should be infeasible to find two different input strings that give the same digest. The most common application is the use of hash functions for password databases. Storing clear passwords in a database is prone to leakage by attacking the database. Instead, the database only contains the digest of the passwords. Since the function is one way, one cannot recover the underlying passwords. To authenticate, the server would compute the digest of the given password and compare it to the database.

Definition 1.15 (Hash Function)

A hash function is defined by two algorithms KeyGen, $\mathcal{H}$, which are described as follows.

- KeyGen: Takes a security parameter λ and outputs a key k.
- $\mathcal{H}$: There exists two polynomial functions ℓ, ℓ' with $\ell'(\lambda) > \ell(\lambda)$, such that given a key k and an input string $x \in \{0,1\}^{\ell'(\lambda)}$, the algorithm outputs $\mathcal{H}(k, x) \in \{0,1\}^{\ell(\lambda)}$.

In this case, the input string has a fixed length. For arbitrarily long strings, we can replace $\{0,1\}^{\ell'(\lambda)}$ by $\{0,1\}^*$.

The key k is usually public for universal hash functions and is part of the specification of the function. For example, for the SHA functions, the key can be thought as the constants used in the hash algorithm. In what follows, we will omit the key and simply refer to the hash function as $\mathcal{H}$, and to the digest of a string x as $\mathcal{H}(x)$.

Security Properties of Hash Functions

We now define the three security properties informally described above.

Definition 1.16 (Uninvertibility)

Let $\mathcal{H}$ be a hash function. We consider a distribution $\mathcal{X}$ over the input domain (usually uniform). The hash function $\mathcal{H}$ is said to be (t, ε)-uninvertible with respect to $\mathcal{X}$ if for all adversary $\mathcal{A}$ running in time at most t, it holds that

$$\mathbb{P}_{x \sim \mathcal{X}}[\mathcal{A}(\mathcal{H}, \mathcal{H}(x)) = x] \leq \varepsilon.$$

Definition 1.17 (Second Preimage Resistance)

Let $\mathcal{H}$ be a hash function. We consider a distribution $\mathcal{X}$ over the input domain (usually uniform). The hash function $\mathcal{H}$ is said to be (t, ε)-second preimage resistant with respect to

$\mathcal{X}$ if for all adversary $\mathcal{A}$ running in time at most t, it holds that

$$\mathbb{P}_{x\sim\mathcal{X}}[\mathcal{A}(\mathcal{H},\mathcal{H}(x)) \neq x \wedge \mathcal{H}(\mathcal{A}(\mathcal{H},\mathcal{H}(x))) = \mathcal{H}(x)] \leq \varepsilon.$$

The second preimage resistance does not necessarily assumes a distribution on x, in which case the probability only runs over the random coins of $\mathcal{A}$. Typically, we sometimes state it as follows: given x and $\mathcal{H}$, the adversary must find $x' \neq x$ such that $\mathcal{H}(x') = \mathcal{H}(x)$.

Definition 1.18 (One-Wayness)

Let $\mathcal{H}$ be a hash function. We consider a distribution $\mathcal{X}$ over the input domain (usually uniform). The hash function $\mathcal{H}$ is said to be (t,ε)-one-way with respect to $\mathcal{X}$ if for all adversary $\mathcal{A}$ running in time at most t, it holds that

$$\mathbb{P}_{x\sim\mathcal{X}}[\mathcal{H}(\mathcal{A}(\mathcal{H},\mathcal{H}(x))) = \mathcal{H}(x)] \leq \varepsilon.$$

Definition 1.19 (Collision Resistance)

Let $\mathcal{H}$ be a hash function. The hash function $\mathcal{H}$ is said to be (t,ε)-collision resistant if for all adversary $\mathcal{A}$ running in time at most t, it holds that

$$\mathbb{P}_{(x,x')\leftarrow\mathcal{A}(\mathcal{H})}[x \neq x' \wedge \mathcal{H}(x) = \mathcal{H}(x')] \leq \varepsilon.$$

These security properties are layered as we can show that certain properties or combination of them imply others. We summarize these claims in the following lemmas. These notions and results can be generalized for function families that are not necessarily hash functions, see e.g., [MP13].

Lemma 1.7 (Sufficient Condition of One-wayness)

Let $\mathcal{H}$ be a hash function, and $\mathcal{X}$ a distribution over the input domain. If $\mathcal{H}$ is (t,ε)-uninvertible with respect to $\mathcal{X}$ and $(t+t',\varepsilon')$-second preimage resistant with respect to $\mathcal{X}$, then it is $(t,\varepsilon+\varepsilon')$-one-way, where t' is the computation time of $\mathcal{H}$.

Proof (Lemma 1.7). Let $\mathcal{A}$ be an algorithm running in time at most t and attacking the one-wayness of $\mathcal{H}$. Let $x \hookleftarrow \mathcal{X}$ be a random inputand compute $y = \mathcal{A}(\mathcal{H},\mathcal{H}(x))$. We now bound the probability that $\mathcal{H}(x) = \mathcal{H}(y)$. If $y = x$, then $\mathcal{A}$ breaks the uninvertibility property, while if $y \neq x$, then it breaks the second preimage resistance. Hence, we have

$$\begin{aligned}\mathbb{P}_x[\mathcal{H}(x) = \mathcal{H}(y)] &= \mathbb{P}_x[x = y \wedge \mathcal{H}(x) = \mathcal{H}(y)] + \mathbb{P}_x[x \neq y \wedge \mathcal{H}(x) = \mathcal{H}(y)] \\ &= \mathbb{P}_x[x = \mathcal{A}(\mathcal{H},\mathcal{H}(x))] + \mathbb{P}_x[x \neq \mathcal{A}(\mathcal{H},\mathcal{H}(x)) \wedge \mathcal{H}(x) = \mathcal{H}(\mathcal{A}(\mathcal{H},\mathcal{H}(x)))] \\ &\leq \varepsilon + \varepsilon',\end{aligned}$$

as desired. For the second preimage resistance, one needs to verify that $\mathcal{H}(x) = \mathcal{H}(\mathcal{A}(\mathcal{H},\mathcal{H}(x)))$ which requires the computation of an extra hash digest, explaining the t'.

Lemma 1.8 (Sufficient Condition Second Preimage Resistance)

Let $\mathcal{H}$ be a hash function. If $\mathcal{H}$ is $(t+t',\varepsilon)$-collision resistant, then it is (t,ε)-second preimage resistant, where t' is the computation time of $\mathcal{H}$.

Proof (Lemma 1.8). Let $\mathcal{A}$ be an algorithm running in time at most t and attacking the second preimage resistance of $\mathcal{H}$ for arbitrary inputs. We construct an adversary $\mathcal{B}$ against the collision resistance as follows. $\mathcal{B}$ chooses x in the domain and computes $y = \mathcal{A}(\mathcal{H},\mathcal{H}(x))$. Then $x \neq y$ and $\mathcal{H}(x) = \mathcal{H}(y)$. It holds that $\mathcal{B}$ runs in time at most $t + t'$ as $\mathcal{A}$ runs in time

at most t. As a result

$$\varepsilon \geq \mathbb{P}_{(x,y)\leftarrow\mathcal{B}(\mathcal{H})}[x \neq y \wedge \mathcal{H}(x) = \mathcal{H}(x')] \geq \mathbb{P}_{\mathcal{A}}[\mathcal{A}(\mathcal{H}, \mathcal{H}(x)) \neq x \wedge \mathcal{H}(\mathcal{A}(\mathcal{H}, \mathcal{H}(x))) = \mathcal{H}(x)],$$

as desired.

Pseudorandomness and Random Oracle Model

Finally, it is sometimes expected that cryptographic hash functions produce close to random outputs. This property is typically used to argue the validity of the random oracle model of a hash functions. We describe here the pseudorandomness property of a hash function and its link to the random oracle model.

Definition 1.20 (Pseudorandomness)

Let $\mathcal{H}$ be a hash function with output space Y. We consider a distribution $\mathcal{X}$ over the input domain (usually uniform). The hash function $\mathcal{H}$ is said to be (t, ε)-pseudorandom with respect to $\mathcal{X}$ if for all distinguisher $\mathcal{A}$ running in time at most t, it holds that

$$\left|\mathbb{P}_{x\sim\mathcal{X}}[\mathcal{A}(\mathcal{H}, \mathcal{H}(x)) = 1] - \mathbb{P}_{y\sim U(Y)}[\mathcal{A}(\mathcal{H}, y) = 1]\right| \leq \varepsilon.$$

A random oracle is an oracle that replies to every unique query by a uniformly random samples. The random oracle should respond the same way on the same query.

Definition 1.21 (Random Oracle)

Let Y be an output space, and IOR an input-output register empty at the outset. We say that $\mathcal{RO}$ is a random oracle with output space Y and input-output register IOR if it behaves as follows. Given an input x

- **if** $(x, y) \in IOR$ for some $y \in Y$, then $\mathcal{RO}(x)$ outputs y.
- **else** sample $y \hookleftarrow U(Y)$, update $IOR \leftarrow IOR \cup \{(x, y)\}$, and $\mathcal{RO}(x)$ outputs y.

The random oracle model then states that every cryptographic hash function behaves as a random oracle. This essentially comes down to $\mathcal{H}$ being pseudorandom with consistent input-output pairs.

1.4 Provable Security

As we departed from the One Time Pad and Shannon's perfect security, a natural question is whether there is a chance to have proven security for cryptographic schemes. The answer is positive but with a caveat. The security models we introduced in Section 1.3 give a framework for the security proof of cryptographic primitives. However, although they model a real-life security requirement in a fairly simple and formal way, it may be complex to fully prove these security properties. In public-key cryptography, we instead reduce these security properties to a small set of mathematical assumptions. Said differently, we assume that a few simpler properties are verified, and prove that if they are indeed, they then imply the desired security property. These assumptions are generally linked to mathematical problems which are supposed to be difficult to solve (like the factorization problem).

1.4.1 Security Proofs

As another course is dedicated to security proofs, we will not enter the details and only recall a little vocabulary.

The purpose of security models is essentially to model real-life attack scenarios as mathematical problems. Once we have defined a proper security model for a given cryptographic construction, proving its security comes down to proving that the problem defined by the security model is a hard problem. To prove it hard, we can adopt either of the approaches from Section 1.2.1. In public-key cryptography, we mostly use the asymptotic or concrete approaches while making additional

mathematical assumptions that need to be reasonable. For example, the problem of factoring a number into a product of two large primes, i.e., the *factorization problem* is the assumption underlying the RSA [RSA78] cryptosystem. There are several viable ways of proving the hardness of the security problems. We cover the two main ones here, which we use later in this course.

Reduction.

A reduction is a way of establishing a hierarchy between the difficulty of mathematical problems. The goal is to prove that if a problem P_1 is hard to solve (for a notion of hardness which can be proven or assumed, and asymptotically or concretely), then the problem P_2 is also hard to solve. To do so, a reduction usually focuses on the contraposition of this statement and shows that if P_2 is easy, then so is P_1.

Definition 1.22 (Reduction)

Let P_1, P_2 denote two mathematical problems. We say that there is a reduction from P_1 to P_2 if there exists an algorithm which solves P_1 given a solver for P_2. In that case, we also say that P_2 is at least as hard as P_1.

Note that in this definition, there is no constraint on the reduction algorithm. However, in most cases we place limitations on this algorithm. Typically, we expect the reduction to be efficient, e.g., polynomial-time, and sometimes deterministic. In the literature, reductions are usually called *transformation reductions*. This means that the reduction algorithm takes an instance of P_1 and transforms it into an instance of P_2 which can be solved by the oracle. It has no impact in this course, but we decide to call reduction even those which would not be categorized as transformation reductions.

Game Hop.

Another method of proving the security of a cryptographic scheme is to use *game hops* methods. As can be seen in Sections 1.3.1 and 1.3.2, the security problems are formulated as games between a challenger and an adversary. The idea of a game hop is to slightly change the security game, while proving that the advantage of the adversary in this modified game is not significantly different from that in the original game. Let us take the exemple of the IND-CPA game for an public-key encryption scheme, which we note G_0. Assume we are able to define a similar game G_1 such that $\left|\mathrm{Adv}_{G_1}[\mathcal{A}] - \mathrm{Adv}_{G_0}[\mathcal{A}]\right| \leq \varepsilon$ for all adversary $\mathcal{A}$, and that in G_1, the challenge ciphertext c is independent of the random bit b. The latter property implies that $\mathrm{Adv}_{G_1}[\mathcal{A}] = 0$. Combined with the prior inequality between G_0 and G_1, it proves $\mathrm{Adv}_{G_0}[\mathcal{A}] \leq \varepsilon$. To show this kind of inequality, we look at the probability distribution of the view of the adversary. The view of the adversary corresponds to all the elements the adversary has access to that depend on the challenger. In our example, $\mathrm{View}(\mathcal{A}) = (\mathsf{pk}, c)$. We would thus prove the distribution of (pk, c) in G_1 is ε-indistinguishable from that of (pk, c) in G_0.

Remark 1.1

It is obviously possible to combine several games. For example, if we define $G_0, \ldots, G_N$ such that $\left|\mathrm{Adv}_{G_{i+1}}[\mathcal{A}] - \mathrm{Adv}_{G_i}[\mathcal{A}]\right| \leq \varepsilon_i$ and that we are able to bound $\mathrm{Adv}_{G_N}[\mathcal{A}] \leq \varepsilon$, this proves $\mathrm{Adv}_{G_0}[\mathcal{A}] \leq \varepsilon + \sum_{i=0}^{N-1} \varepsilon_i$ by the triangle inequality. If all the $\varepsilon, \varepsilon_i$ are negligible, it then shows $\mathrm{Adv}_{J_0}[\mathcal{A}]$ is negligible.
It is also possible to combine game hops with reductions. For example, we could show the games $G_0, \ldots, G_N$ verify $\left|\mathrm{Adv}_{G_{i+1}}[\mathcal{A}] - \mathrm{Adv}_{G_i}[\mathcal{A}]\right| \leq \varepsilon_i$ by game hops. Then, to bound $\mathrm{Adv}_{G_N}[\mathcal{A}] \leq \varepsilon$, we could show that the mathematical problem corresponding to G_N is at least as hard as a problem P. Then, the bound ε would be linked to the difficulty of solving P. Each game hop could also be argued using a reduction. For example, to show $\left|\mathrm{Adv}_{G_{i_0+1}}[\mathcal{A}] - \mathrm{Adv}_{G_{i_0}}[\mathcal{A}]\right| \leq \varepsilon_{i_0}$, one could argue that distinguishing the view of $\mathcal{A}$ in G_{i_0} from the view of $\mathcal{A}$ in G_{i_0+1} is at least as hard than a decisional problem P'. Then, ε_i would be linked to the difficulty of solving P'.

Bibliography

[DH76] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Trans. Inf. Theory*, 1976.

[GM82] S. Goldwasser and S. Micali. Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information. In *STOC*, 1982.

[MP13] D. Micciancio and C. Peikert. Hardness of SIS and LWE with Small Parameters. In *CRYPTO*, 2013.

[R61] A. Rényi. On Measures of Entropy and Information. In *Proc. 4th Berkeley Sympos. Math. Statist. and Prob., Vol. I*, 1961.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM*, 1978.

[vEH14] T. van Erven and P. Harremoës. Rényi Divergence and Kullback-Leibler Divergence. *IEEE Trans. Inf. Theory*, 2014.

Part II

Lattice Theory

This part focuses on the fundamentals of lattices and lattice problems.

2

Lattices

Lattices are mathematical objects that have been used for centuries in various area of mathematics and applied sciences. They are often referred to as Euclidean lattices but can only be traced back to Lagrange (1736 - 1813). Their properties and rich geometry offer very interesting applications in mechanics, physics, biology and many more. The algorithmic theory surrounding lattices that is now the base of lattice-based cryptography has emerged in the late 1900s with the work of Lenstra, Lenstra and Lovàsz [LJL82]. In this chapter, we present the mathematical definitions and properties of lattices needed for this course, as well as the algorithmic perspective that is not necessarily related to cryptographic, albeit very useful in lattice-based cryptanalysis.

Contents

2.1 Reminder in Linear Algebra

We start by fixing some notations and relevant materials from linear algebra as they will be ubiquitous throughout this course. Unless specified otherwise, the ambient vector space will be $\mathbb{R}^k$ for some positive integer k, where $(\mathbb{R}, +, \cdot)$ denotes the field of real numbers. The ring of integers is denoted by $\mathbb{Z}$.

Vectors and Matrices

Vectors are written in bold lowercase letters $\mathbf{x}$, and by convention they are column vectors. To specify the entries of a vector $\mathbf{x} \in \mathbb{R}^k$ in the canonical basis, we may write $\mathbf{x} = [x_i]_{i \in [\![1,k]\!]}$. Matrices are written in bold uppercase letters $\mathbf{A}$. For a ring R, and two positive integers m and k, we denote by $R^{m \times k}$ the vector space or module of matrices with m rows, k columns and entries in R. This space may be denoted by $\mathcal{M}_{m,d}(R)$ in other courses, and we thus insist on our change of notation. We may specify a matrix $\mathbf{A} \in R^{m \times k}$ by its columns as $\mathbf{A} = [\mathbf{a}_i]_{i \in [\![1,k]\!]}$, which means that the columns of $\mathbf{A}$ are the vectors $\mathbf{a}_1, \ldots, \mathbf{a}_k \in R^m$. We may also specify a matrix $\mathbf{A}$ by its entries as $\mathbf{A} = [a_{i,j}]_{i \in [\![1,m]\!], j \in [\![1,k]\!]}$, meaning that the entry at the i-th row and j-th column of $\mathbf{A}$ is $a_{i,j}$. The transpose operator is denoted by a right superscript T, i.e., if $\mathbf{A} = [a_{i,j}]_{i \in [\![1,m]\!], j \in [\![1,k]\!]}$,

then $\mathbf{A}^T = [a_{j,i}]_{j\in[\![1,k]\!],i\in[\![1,m]\!]}$. In a ring R where the complex conjugate makes sense, it is denote by a bar, i.e., $\overline{a}$. In such rings, the Hermitian operator is denoted by a right superscript H, i.e., if $\mathbf{A} = [a_{i,j}]_{i\in[\![1,m]\!],j\in[\![1,k]\!]}$, then $\mathbf{A}^H = \overline{\mathbf{A}}^T = \overline{\mathbf{A}^T} = [\overline{a_{j,i}}]_{j\in[\![1,k]\!],i\in[\![1,m]\!]}$. The k-dimensional vector with only zero entries (resp. 1 entries) is denoted by $\mathbf{0}_k$ (resp. $\mathbf{1}_k$) or simply $\mathbf{0}$ (resp. $\mathbf{1}$) if k is clear from the context. The identity matrix of dimension k is denoted by $\mathbf{I}_k$, and for a vector $\mathbf{x} \in R^k$, we denote by $\mathrm{diag}(\mathbf{x})$ the matrix of $R^{k\times k}$ whose diagonal entries are the entries of $\mathbf{x}$.

For four positive integers m, k, n, d, we may sometimes define a matrix by its blocks as $\mathbf{A} = [\mathbf{A}_{i,j}]_{i\in[\![1,m]\!],j\in[\![1,k]\!]} \in R^{mn\times kd}$ where each matrix $\mathbf{A}_{i,j}$ is in $R^{n\times d}$.

Quadratic Forms, Inner Products, and Norms

We recall that every positive definite quadratic form Q over a real vector space V uniquely corresponds to a bilinear form $\langle\cdot\,,\cdot\rangle$ that is symmetric and positive definite by

$$\forall(\mathbf{x},\mathbf{y}) \in V^2, \langle\mathbf{x}\,,\mathbf{y}\rangle = \frac{1}{2}(Q(\mathbf{x}+\mathbf{y}) - Q(\mathbf{x}) - Q(\mathbf{y})), \text{ and } \forall\mathbf{x}\in V, \langle\mathbf{x}\,,\mathbf{x}\rangle = Q(\mathbf{x})$$

The ambient space $\mathbb{R}^k$ is a Hilbert space gifted with the usual inner product. In this course, we denote it by $\langle\cdot\,,\cdot\rangle$ and it is defined as

$$\forall\mathbf{x},\mathbf{y}\in\mathbb{R}^k, \langle\mathbf{x}\,,\mathbf{y}\rangle = \mathbf{x}^T\mathbf{y}.$$

It is also equipped with the usual Euclidean norm $\|\cdot\|_2$ defined by

$$\forall\mathbf{x}\in\mathbb{R}^k, \|\mathbf{x}\|_2 = \sqrt{\mathbf{x}^T\mathbf{x}}.$$

More generally, we define the ℓ_p norm $\|\cdot\|_p$ of $\mathbb{R}^k$ for any positive integer p by

$$\forall\mathbf{x} = [x_i]_{i\in[\![1,k]\!]} \in \mathbb{R}^k, \|\mathbf{x}\|_p = \left(\sum_{i\in[\![1,k]\!]} |x_i|^p\right)^{1/p},$$

and the ℓ_∞ norm $\|\cdot\|_\infty$ by

$$\forall\mathbf{x} = [x_i]_{i\in[\![1,k]\!]} \in \mathbb{R}^k, \|\mathbf{x}\|_\infty = \max_{i\in[\![1,k]\!]} |x_i|\,.$$

Linear Independence, Spanning Set, Basis

Consider a ring R and M and R-module. For simplicity, take $M \subseteq R^k$ for a positive integer k. Let d be two positive integers and $(\mathbf{b}_i)_{i\in[\![1,d]\!]}$ be a family of vectors of M. We say that $(\mathbf{b}_i)_{i\in[\![1,d]\!]}$ is a family of R-linearly independent vectors if and only if for all $(\lambda_i)_{i\in[\![1,d]\!]} \in R^d$, if $\sum_{i\in[\![1,d]\!]} \lambda_i\mathbf{b}_i = \mathbf{0}$ then $\lambda_i = 0$ for all $i \in [\![1,d]\!]$. We say that $(\mathbf{b}_i)_{i\in[\![1,d]\!]}$ is a spanning set (or spanning family) of M if and only if $\mathrm{Span}_R(\mathbf{b}_1,\ldots,\mathbf{b}_d) = M$, where $\mathrm{Span}_R(\mathbf{b}_1,\ldots,\mathbf{b}_d) = \{\sum_{i\in[\![1,d]\!]} \lambda_i\mathbf{b}_i; (\lambda_i)_{i\in[\![1,d]\!]} \in R^d\}$. Finally, $(\mathbf{b}_i)_{i\in[\![1,d]\!]}$ is a basis of M if and only if it is a spanning family of M and is R-linearly independent. In that case, every $\mathbf{x}$ in M can be uniquely written along the basis by $\mathbf{x} = \sum_{i\in[\![1,d]\!]} x_i\mathbf{b}_i$, with $(x_i)_{i\in[\![1,d]\!]} \in R^d$. If $\mathbf{B} = [\mathbf{b}_i]_{i\in[\![1,d]\!]} \in R^{k\times d}$, we thus have

$$\mathbf{x} = \mathbf{B}\cdot\begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}.$$

Remark 2.1

Here the basis refer to basis of the ambient space or module M, that is with respect to scalars in R. Later, we also use the term *basis* to talk about the basis of a lattice. We use the same appellation which should be clear from the context.

2.2 Fundamentals of Lattices

We recall that $\mathbb{R}^k$ is equipped with the Euclidean norm $\|\cdot\|_2$ and inner product $\langle\cdot\,,\cdot\rangle$. We define the closed ℓ_p ball of radius $r \geq 0$ and center $\mathbf{c} \in \mathbb{R}^k$ by $\mathcal{B}_p(\mathbf{c},r) = \{\mathbf{x}\in\mathbb{R}^k : \|\mathbf{x}-\mathbf{c}\|_p \leq r\}$ for

any positive integer p. For the sake of completeness, we may also consider the hypercube of center $\mathbf{c} \in \mathbb{R}^k$ and half-side r by $\mathcal{B}_\infty(\mathbf{c}, r) = \{\mathbf{x} \in \mathbb{R}^k : \|\mathbf{x} - \mathbf{c}\|_\infty \leq r\}$. We use the right superscript o on a ball to specify that it is open. As a result, we have

$$\forall p \in \mathbb{N}^* \cup \{\infty\}, \forall \mathbf{c} \in \mathbb{R}^k, \forall r \geq 0, \mathcal{B}_p^o(\mathbf{c}, r) = \{\mathbf{x} \in \mathbb{R}^k : \|\mathbf{x} - \mathbf{c}\|_p < r\}$$

We recall the definition of a discrete set. It can be more generally defined in topological spaces but we only give the definition in a normed space.

Definition 2.1 (Discrete Set)

Let $(V, \|\cdot\|)$ be a normed space. Let S be an arbitrary subset of V. We say that S is a *discrete set* if and only if

$$\forall \mathbf{x} \in S, \exists r > 0, \mathcal{B}_{\|\cdot\|}^o(\mathbf{x}, r) \cap S = \{\mathbf{x}\}.$$

We are now able to define a lattice of $\mathbb{R}^k$. Once again, in all generality, there other ways to define lattices encompassing our definition. They involve Lie groups and measure theory. They are however generalizations that are unnecessary for this course.

Definition 2.2 (Lattice)

Let k be a positive integer, and $\mathcal{L} \subset \mathbb{R}^k$. The set $\mathcal{L}$ is called a (Euclidean) lattice if and only if $\mathcal{L}$ is a discrete subgroup of $(\mathbb{R}^k, +)$. A sublattice $\mathcal{L}' \subseteq \mathcal{L}$ is a discrete subgroup of the lattice $\mathcal{L}$.

Let $\mathcal{L} \subset \mathbb{R}^k$ be a lattice. We can show that there exists an $\mathbb{Z}$-linearly independent family $(\mathbf{b}_i)_{i \in [\![1,d]\!]}$ that is maximal in $\mathcal{L}$ such that $\mathcal{L} = \mathbb{Z}\mathbf{b}_1 \oplus \ldots \oplus \mathbb{Z}\mathbf{b}_d$. Since this family is linearly independent, it is a basis of the d-dimensional vector subspace $V = \oplus_{i \in [\![1,d]\!]} \mathbb{R}\mathbf{b}_i$. Additionally, since it is maximal in $\mathcal{L}$, i.e., adding a vector of $\mathcal{L}$ in the family makes it $\mathbb{Z}$-linearly dependent, it is called a *basis* of the lattice $\mathcal{L}$. The integer d is common to all the bases of $\mathcal{L}$ and is called the *rank* of the lattice $\mathcal{L}$, which we denote $d = \text{rank}(\mathcal{L})$. When $d = k$, we say that the lattice is *full-rank*.

Remark 2.2

Every basis $(\mathbf{b}_i)_{i \in [\![1,k]\!]}$ of $\mathbb{R}^k$ generates a lattice by limiting the scalars to $\mathbb{Z}$, i.e., $\text{Span}_{\mathbb{Z}}(\mathbf{b}_1, \ldots, \mathbf{b}_k)$ is a lattice. Conversely, every basis of a lattice of fixed rank d generates a d-dimensional vector space by extending the scalars to $\mathbb{R}$.

Given a linearly independent family $(\mathbf{b}_i)_{i \in [\![1,d]\!]}$ of $\mathbb{R}^k$, we denote by $\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_d) = \mathbb{Z}\mathbf{b}_1 \oplus \ldots \oplus \mathbb{Z}\mathbf{b}_d$ the lattice generated by this family. When using the canonical basis for $\mathbb{R}^k$, and $\mathbf{B} = [\mathbf{b}_i]_{i \in [\![1,d]\!]}$ the matrix of the $\mathbf{b}_i$, we also use $\mathcal{L}(\mathbf{B})$ or $\mathbf{B}\mathbb{Z}^d$ to denote this lattice.

Example 2.1 (Strict Sublattice)

We have the following inclusions of lattices over $\mathbb{R}$: $2\mathbb{Z} \subset \mathbb{Z} \subset \frac{1}{2}\mathbb{Z}$. This simple example highlights the fact that having the same rank and an inclusion is not sufficient to have the equality of lattices, as opposed to vector spaces. For all positive integer d, $\mathbb{Z}^d$ is a full-rank lattice of $\mathbb{R}^d$. The set S_d of the vectors of $\mathbb{Z}^d$ for which the sum of the entries is even is a strict sublattice of $\mathbb{Z}^d$ with rank d.

We can define two families of lattices that are very important in lattice-based cryptography. They are called *q-ary lattices* or *q-periodic lattices*. They essentially are lattice which are invariant by q-step translation. More precisely, we define them as follows.

Definition 2.3 (q-ary Lattices)

Let m, d be two positive integers, and q be an integer larger or equal than 2. Let $\mathbf{B}$ be in

$\mathbb{Z}^{d\times m}$. The following $\mathcal{L}_q(\mathbf{B})$ and $\mathcal{L}_q^\perp(\mathbf{B})$ are lattices of rank m, where

$$\mathcal{L}_q(\mathbf{B}) = \mathbf{B}^T\mathbb{Z}^d + q\mathbb{Z}^m$$
$$\mathcal{L}_q^\perp(\mathbf{B}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{B}\mathbf{x} = \mathbf{0} \bmod q\mathbb{Z}\}$$

They are called *q-ary lattices*.

Notice that the q-ary lattice $\mathcal{L}_q^\perp(\mathbf{B})$ is not defined by a basis. A basis is not necessary to define and describe a lattice. However, to use lattices algorithmically, we do not know how to do it without a basis. Hence every lattice used in algorithms should be describable by a basis that should be efficient to compute.

2.2.1 Lattice Bases

Determinant of Matrices

We now dive in a little more in the bases of lattices. In this chapter, we only need elementary properties of the determinant of matrices. The determinant is defined for *square* matrices. We recall that a matrix $\mathbf{A}$ of $\mathbb{R}^{d\times d}$ is invertible if and only if its columns form a basis of $\mathbb{R}^d$, or equivalently if its determinant $\det \mathbf{A}$ is non-zero. We insist on the fact that this is no longer true if one considers matrices over a ring that has zero divisors. In this course, for a ring R and a positive integer d, we denote by $GL_d(R)$ the space or module of invertible matrices of $R^{d\times d}$. In commutative rings, we have the following Cramer's formula.

Lemma 2.1 (Cramer's Formula)

Let R be a commutative ring, and d a positive integer. Then, for all matrices $\mathbf{A}$ in $R^{d\times d}$, it holds that

$$\mathbf{A} \cdot \mathrm{Com}(\mathbf{A})^T = \mathrm{Com}(\mathbf{A})^T \cdot \mathbf{A} = (\det \mathbf{A})\mathbf{I}_d,$$

where $\mathrm{Com}(\mathbf{A})$ is the comatrix of $\mathbf{A}$, i.e., the matrix of cofactors of $\mathbf{A}$.

As a consequence of Lemma 2.1, if $\det \mathbf{A}$ is a unit in R, then $\mathbf{A}$ is in $GL_d(R)$ and its inverse is $\mathbf{A}^{-1} = (\det \mathbf{A})^{-1}\mathrm{Com}(\mathbf{A})^T$. Additionally, the determinant verifies the following properties. For all matrices $\mathbf{A}, \mathbf{B}$ in $R^{d\times d}$, we have

$$\det(\mathbf{AB}) = (\det \mathbf{A})(\det \mathbf{B})$$
$$\det(\mathbf{A}^{-1}) = (\det \mathbf{A})^{-1}$$
$$\det(\mathbf{A}^T) = \det \mathbf{A}.$$

Instructive Examples

We give a few examples in dimension 2 to build the intuition that allows to characterize the bases of a lattice. Consider the example of $\mathbb{Z}^2$. The canonical basis $(\mathbf{e}_1, \mathbf{e}_2)$ of $\mathbb{R}^2$ is also a basis of the lattice $\mathbb{Z}^2$, represented by the identity matrix $\mathbf{I}_2$. Other bases are for example given by $(\mathbf{e}_1, \mathbf{u}_a)$ where $\mathbf{u}_a = [a|1]^T = a\mathbf{e}_1 + \mathbf{e}_2$ for some $a \in \mathbb{Z}$. If $\mathbf{U}$ is the matrix transforming the basis $(\mathbf{e}_1, \mathbf{e}_2)$ into $(\mathbf{e}_1, \mathbf{u}_a)$, we have

$$\mathbf{U} = \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix} \text{ and } \mathbf{U}^{-1} = \begin{bmatrix} 1 & -a \\ 0 & 1 \end{bmatrix}.$$

We notice that $\mathbf{U}$ and $\mathbf{U}^{-1}$ have integer entries, and that $\det \mathbf{U} = 1$ which is a unit of $\mathbb{Z}$. Hence $\mathbf{U}$ is in $GL_2(\mathbb{Z})$. Let us now take $\mathbf{v}_1 = [-1|3]^T$, $\mathbf{v}_2 = [-1|2]^T$ and $\mathcal{L} = \mathcal{L}(\mathbf{v}_1, \mathbf{v}_2)$. Even though the basis $(\mathbf{v}_1, \mathbf{v}_2)$ looks different geometrically, we can easily show that $\mathcal{L} = \mathbb{Z}^2$.

Indeed, $\mathbf{v}_1, \mathbf{v}_2$ have integer entries proving that $\mathcal{L} \subseteq \mathbb{Z}^2$, and $\mathbf{e}_2 = \mathbf{v}_1 - \mathbf{v}_2 \in \mathcal{L}$, and $\mathbf{e}_1 = 2\mathbf{e}_2 - \mathbf{v}_2 \in \mathcal{L}$ proving that $\mathbb{Z}^2 \subseteq \mathcal{L}$. The matrix $\mathbf{U}$ transforming $(\mathbf{e}_1, \mathbf{e}_2)$ into $(\mathbf{v}_1, \mathbf{v}_2)$ is

$$\mathbf{U} = \begin{bmatrix} -1 & -1 \\ 3 & 2 \end{bmatrix} \text{ and } \mathbf{U}^{-1} = \begin{bmatrix} 2 & 1 \\ -3 & -1 \end{bmatrix}.$$

Once again, the inverse of $\mathbf{U}$ has integer coefficients and $\det \mathbf{U} = -1$ which also a unit in $\mathbb{Z}$.

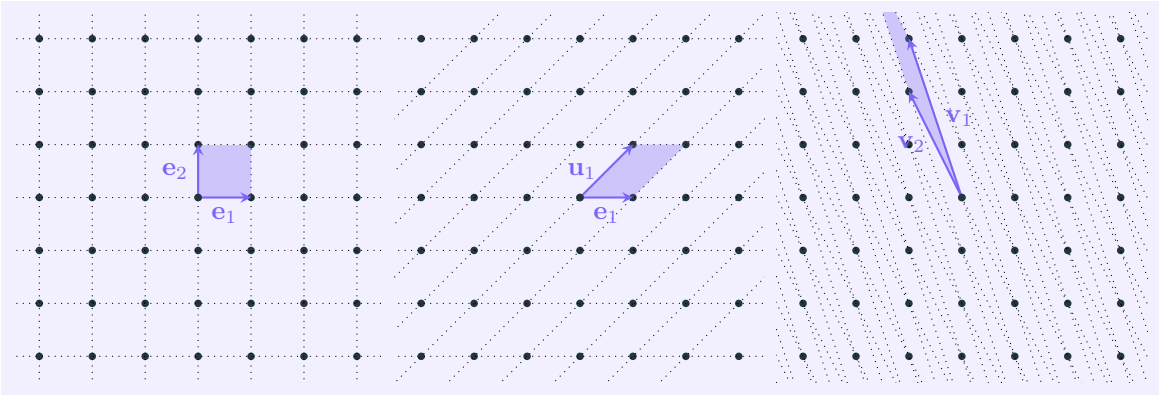


Figure 2.1: The lattice $\mathbb{Z}^2$ with different bases: $(\mathbf{e}_1, \mathbf{e}_2)$, $(\mathbf{e}_1, \mathbf{u}_1)$, $(\mathbf{v}_1, \mathbf{v}_2)$.

Let us now look at the lattice generated by $\mathbf{w}_1 = [1|-1]^T$ and $\mathbf{w}_2 = [1|2]^T$. It is clear that $\mathcal{L}(\mathbf{w}_1, \mathbf{w}_2) \subseteq \mathbb{Z}^2$ but the transfer matrices are

$$\mathbf{U} = \begin{bmatrix} 1 & 1 \\ -1 & 2 \end{bmatrix} \text{ and } \mathbf{U}^{-1} = \frac{1}{3}\begin{bmatrix} 2 & -1 \\ 1 & 1 \end{bmatrix}.$$

In particular, we cannot write $(\mathbf{e}_1, \mathbf{e}_2)$ as an *integer* linear combination of $\mathbf{w}_1$ and $\mathbf{w}_2$. Thence, $\mathcal{L}(\mathbf{w}_1, \mathbf{w}_2)$ is a strict sublattice of $\mathbb{Z}^2$. Geometrically, we can quickly see that the area of the parallelepiped described by $(\mathbf{w}_1, \mathbf{w}_2)$ is larger than that of the one generated by $(\mathbf{e}_1, \mathbf{e}_2)$. This leads us to the following characterization of the bases of a lattice.

Lemma 2.2 (Characterization of Bases)

Let d, k be two positive integers, and $(\mathbf{b}_i)_{i\in[\![1,d]\!]}$, $(\mathbf{b}'_i)_{i\in[\![1,d]\!]}$ two linearly-independent families of $\mathbb{R}^k$. We let $\mathbf{B} = [\mathbf{b}_i]_{i\in[\![1,d]\!]}$ and $\mathbf{B}' = [\mathbf{b}'_i]_{i\in[\![1,d]\!]}$ be the matrix representations of those bases. It holds that

$$\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{B}') \iff \exists \mathbf{U} \in GL_d(\mathbb{Z}), \mathbf{B}' = \mathbf{B}\mathbf{U}.$$

Proof (Lemma 2.2). Suppose $\mathbf{B}' = \mathbf{B}\mathbf{U}$ for some $\mathbf{U} \in GL_d(\mathbb{Z})$. Then for all $\mathbf{y}$ in $\mathbb{R}^k$, the following equivalences hold:

$$\begin{aligned}
\mathbf{y} \in \mathcal{L}(\mathbf{B}') &\iff \exists \mathbf{x} \in \mathbb{Z}^d, \mathbf{y} = \mathbf{B}'\mathbf{x} \\
&\iff \exists \mathbf{x} \in \mathbb{Z}^d, \mathbf{y} = \mathbf{B}\underbrace{\mathbf{U}\mathbf{x}}_{\in\mathbb{Z}^d} \\
&\iff \exists \mathbf{x}' \in \mathbb{Z}^d, \mathbf{y} = \mathbf{B}_1\mathbf{x}' \text{ (because } \mathbf{U} \in GL_d(\mathbb{Z})) \\
&\iff \mathbf{y} \in \mathcal{L}(\mathbf{B}),
\end{aligned}$$

which proves the desired equality.

Conversely, suppose that $\mathbf{B}$ and $\mathbf{B}'$ represent the same lattice. Let i be in $[\![1, d]\!]$. Then, $\mathbf{B}\mathbf{e}_i \in \mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{B}')$ so there exists $\mathbf{v}_i$ in $\mathbb{Z}^d$ such that $\mathbf{B}\mathbf{e}_i = \mathbf{B}'\mathbf{v}_i$. We then construct $\mathbf{V} = [\mathbf{v}_i]_{i\in[\![1,d]\!]}$, and by construction, $\mathbf{B}\mathbf{e}_i = \mathbf{B}'\mathbf{V}\mathbf{e}_i$, which yields $\mathbf{B} = \mathbf{B}'\mathbf{V}$ for some square integer matrix $\mathbf{V}$. Similarly we construct a square integer matrix $\mathbf{U}$ such that $\mathbf{B}' = \mathbf{B}\mathbf{U}$. Hence $\mathbf{B}' = \mathbf{B}'\mathbf{V}\mathbf{U}$, which means $\mathbf{V}\mathbf{U} = \mathbf{I}_d$ by linear independence of $\mathbf{B}'$, and similarly $\mathbf{U}\mathbf{V} = \mathbf{I}_d$. Therefore, $\mathbf{U}$ is in $GL_d(\mathbb{Z})$.

2.2.2 Fundamental Invariants

We now present several invariants of a lattice which arise at several occasions in cryptography and cryptanalysis. By invariants, we mean that these quantities do not depend on the given basis of the lattice. We note that certain bases may lead to the efficient computation of those quantities while other bases might make it more tedeous. We will see later in the course that this categorization of the different bases of a lattice is exactly what lattice-based cryptography relies on.

Volume of a Lattice

Before introducing the volume of a lattice, we give a few extra reminders of linear algebra. For a linearly-independent family $(\mathbf{b}_i)_{i\in[\![1,d]\!]}$ of $\mathbb{R}^k$, the Gram matrix of the family $(\mathbf{b}_i)_{i\in[\![1,d]\!]}$ is $\mathbf{G}((\mathbf{b}_i)_{i\in[\![1,d]\!]}) = [\langle \mathbf{b}_i , \mathbf{b}_j\rangle]_{i,j\in[\![1,d]\!]}$. If the family is represented as a matrix $\mathbf{B}$ of $\mathbb{R}^{k\times d}$, we have $\mathbf{G_B} := \mathbf{G}((\mathbf{b}_i)_{i\in[\![1,d]\!]}) = \mathbf{B}^T\mathbf{B}$. When clear from the context, we simply denote it as $\mathbf{G}$.

Definition 2.4 (Volume of a Lattice)

Let k, d be positive integers and $(\mathbf{b}_i)_{i\in[\![1,d]\!]}$ a linearly-independent family of $\mathbb{R}^k$. Let $\mathbf{B} = [\mathbf{b}_i]_{i\in[\![1,d]\!]}$ be the matrix representation of this basis. The *determinant* or *volume* of the lattice $\mathcal{L}(\mathbf{B})$ is defined as

$$\operatorname{Vol}\mathcal{L}(\mathbf{B}) = \det\mathcal{L}(\mathbf{B}) = \sqrt{\det\mathbf{G_B}} = \sqrt{\det\mathbf{B}^T\mathbf{B}}.$$

When $\mathcal{L}(\mathbf{B})$ is full-rank in $\mathbb{R}^k$, i.e., $d = k$, we have $\operatorname{Vol}\mathcal{L}(\mathbf{B}) = |\det\mathbf{B}|$.

We can now verify that the volume is indeed an invariant, meaning that it does not depend on the chosen basis.

Lemma 2.3 (Invariance of the Volume)

Let k be a positive integer and $\mathcal{L}$ a lattice of $\mathbb{R}^k$. The volume of the lattice $\mathcal{L}$ does not depend on the choice of the basis.

Proof (Lemma 2.3). We define $d = \operatorname{rank}\mathcal{L}$. Let $\mathbf{B}, \mathbf{B}'$ be two basis of the lattice $\mathcal{L}$, i.e., $\mathcal{L} = \mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{B}')$. By Lemma 2.2, it holds that there exists $\mathbf{U} \in GL_d(\mathbb{Z})$ such that $\mathbf{B}' = \mathbf{BU}$. We thus have

$$\det(\mathbf{B'}^T\mathbf{B}') = \det(\mathbf{U}^T\mathbf{B}^T\mathbf{BU}) = (\det\mathbf{U})^2\det(\mathbf{B}^T\mathbf{B}) = \det(\mathbf{B}^T\mathbf{B}),$$

where the last equality stems from the fact that $\det\mathbf{U} \in \mathbb{Z}^\times = \{-1, 1\}$.

In cryptography, the determinant of a lattice is generally called the *volume* in reference to its geometrical interpretation. Indeed, each basis defines a parallelepiped which has a volume that coincides with the determinant of the lattice.

Definition 2.5 (Fundamental Parallelepiped)

Let k, d be positive integers and $(\mathbf{b}_i)_{i\in[\![1,d]\!]}$ a linearly-independent family of $\mathbb{R}^k$. Let $\mathbf{B} = [\mathbf{b}_i]_{i\in[\![1,d]\!]}$ be the matrix representation of this basis. The *centered fundamental parallelepiped* associated to the basis $\mathbf{B}$ is

$$\mathcal{P}_\pm(\mathbf{B}) = \{\mathbf{Bx}; \mathbf{x} \in [-1/2, 1/2)^d\}.$$

The *fundamental parallelepiped* associated to the basis $\mathbf{B}$ is

$$\mathcal{P}(\mathbf{B}) = \{\mathbf{Bx}; \mathbf{x} \in [0, 1)^d\}.$$

Lemma 2.3 says that all the fundamental parallelepipeds of bases of the same lattice have the same volume. We could therefore avoid the choice of a basis and describe the volume directly from the lattice itself by studying the compact group $\mathbb{R}^k/\mathcal{L}$. This is however out of the scope of the course. However, the volume allows to give the missing condition for a sublattice of same rank to be equal to the surlattice.

Lemma 2.4 (Characterization by Volume)

Let $\mathcal{L}, \mathcal{L}'$ be two lattices such that $\mathcal{L}' \subseteq \mathcal{L}$ and $\operatorname{rank}\mathcal{L}' = \operatorname{rank}\mathcal{L}$. Then, it holds that $\operatorname{Vol}\mathcal{L} \leq \operatorname{Vol}\mathcal{L}'$. Additionally, we have that $\mathcal{L}' = \mathcal{L}$ if and only if $\operatorname{Vol}\mathcal{L}' = \operatorname{Vol}\mathcal{L}$.

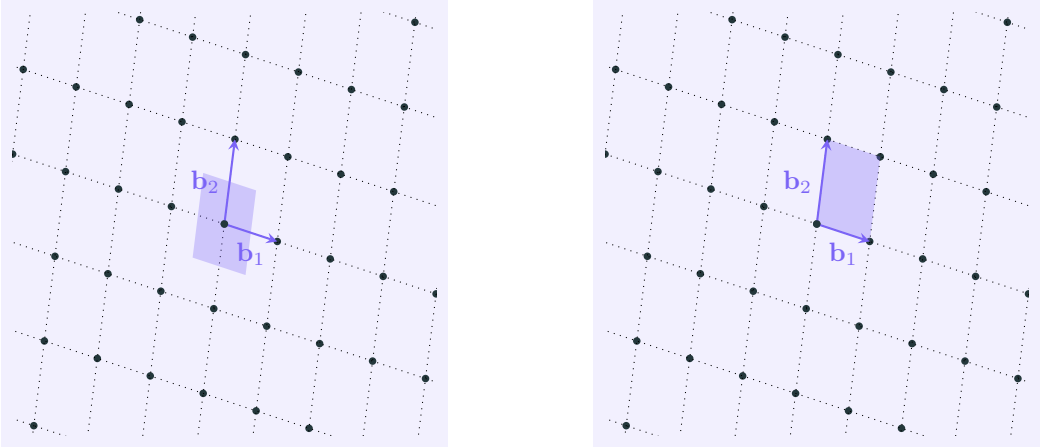


Figure 2.2: Centered (Left) and Uncentered (Right) Fundamental Parallelepipeds.

Proof (Lemma 2.4). We denote by k the dimension of the lattice $\mathcal{L}$, and by d its rank. Let $\mathbf{B}, \mathbf{B}'$ be the matrices for some basis of $\mathcal{L}$ and $\mathcal{L}'$ respectively. Let i be in $[\![1, d]\!]$. It holds that $\mathbf{B}'\mathbf{e}_i \in \mathcal{L}' \subseteq \mathcal{L}$. As a result, there exists $\mathbf{u}_i$ is $\mathbb{Z}^d$ such that $\mathbf{B}'\mathbf{e}_i = \mathbf{B}\mathbf{u}_i$. This therefore defines a matrix $\mathbf{U} \in \mathbb{Z}^{d\times d}$ such that $\mathbf{B}' = \mathbf{B}\mathbf{U}$. By definition of the volume, we have

$$\operatorname{Vol}\mathcal{L}' = |\det \mathbf{U}| \operatorname{Vol}\mathcal{L}.$$

Since $\mathbf{U}$ has integer entries, it holds that $\det \mathbf{U} \in \mathbb{Z}$. Additionally, because the columns of $\mathbf{B}'$ and that of $\mathbf{B}$ are linearly independent, we necessarily have that $\mathbf{U}$ is in $GL_d(\mathbb{R})$. As a result, $|\det \mathbf{U}| \geq 1$ which proves the first claim.

If $\mathcal{L} = \mathcal{L}'$, it directly holds by Lemma 2.3 that $\operatorname{Vol}\mathcal{L} = \operatorname{Vol}\mathcal{L}'$. Now assume that $\operatorname{Vol}\mathcal{L} = \operatorname{Vol}\mathcal{L}'$. This proves that $|\det \mathbf{U}| = 1$, thus proving that $\det \mathbf{U} \in \mathbb{Z}^\times$. As a result, since $\mathbf{U}$ has integer entries and unit determinant, it gives $\mathbf{U} \in GL_d(\mathbb{Z})$. Hence, $\mathbf{B}' = \mathbf{B}\mathbf{U}$ with $\mathbf{U} \in GL_d(\mathbb{Z})$ which, by Lemma 2.2, implies that $\mathcal{L} = \mathcal{L}'$.

Lemma 2.5 (Index of a Sublattice)

Let $\mathcal{L}, \mathcal{L}'$ be two lattices such that $\mathcal{L}' \subseteq \mathcal{L}$ and $\operatorname{rank}\mathcal{L}' = \operatorname{rank}\mathcal{L}$. Then, if $[\mathcal{L} : \mathcal{L}']$ is the index of $\mathcal{L}'$ in $\mathcal{L}$ as an abelian group, we have

$$[\mathcal{L} : \mathcal{L}'] = \frac{\operatorname{Vol}\mathcal{L}'}{\operatorname{Vol}\mathcal{L}}.$$

Example 2.2

We give a few examples to show that every assumption in the previous lemmas are important.

- $\mathbb{Z}^2$ and $\mathbb{Z}\mathbf{v}_1 \oplus \mathbb{Z}\mathbf{v}_2$, where $\mathbf{v}_1 = 2\mathbf{e}_1$ and $\mathbf{v}_2 = \frac{1}{2}\mathbf{e}_2$, have the same volume and the same rank, but are distinct.
- $\mathbb{Z}$ and $\mathbb{Z}^2$ have the same volume but not the same rank.
- $\mathcal{L}' = \mathbb{Z}\mathbf{e}_2$ is a sublattice of $\mathcal{L} = \mathbb{Z}\mathbf{v}_1 \oplus \mathbb{Z}\mathbf{v}_2$ with $\mathbf{v}_1 = 4\mathbf{e}_1$ and $\mathbf{v}_2 = \frac{1}{2}\mathbf{e}_2$, and verifies $\operatorname{Vol}\mathcal{L}' \leq \operatorname{Vol}\mathcal{L}$.

In particular, although the volume characterizes a lattice, it is not sufficient to order them.

Minima of a Lattice

Since a lattice is a discrete set, there necessarily exist non-zero vectors of the lattice which are the smallest possible with respect to the ambient norm. This allows to define quantities which are essential in lattice-based cryptography, and that also are invariants of the lattice.

Definition 2.6 (First Minimum of a Lattice)

Let k be a positive integer, and $\mathcal{L} \subset \mathbb{R}^k$ be a lattice. Let p be in $\mathbb{N}^* \cap \{\infty\}$. The first minimum of $\mathcal{L}$ with respect to the norm $\|\cdot\|_p$ is defined as

$$\begin{aligned}\lambda_1^p(\mathcal{L}) &= \min\{r > 0 : \left|\mathcal{B}_p(\mathbf{0}, r) \cap \mathcal{L}\right| > 1\} \\ &= \min_{\mathbf{x} \in \mathcal{L} \setminus \{\mathbf{0}\}} \|\mathbf{x}\|_p .\end{aligned}$$

We can then define the successive minima of a lattice. Intuitively, the i-th minimum is the size of the smallest ball that contains i linearly independent lattice vectors.

Definition 2.7 (Successive Minima of a Lattice)

Let k be a positive integer, and $\mathcal{L} \subset \mathbb{R}^k$ be a lattice, and $d = \text{rank}\mathcal{L}$. Let p be in $\mathbb{N}^* \cap \{\infty\}$. For all i in $[\![1, d]\!]$, the i-th minimum of $\mathcal{L}$ with respect to the norm $\|\cdot\|_p$ is defined as

$$\lambda_i^p(\mathcal{L}) = \min\{r > 0 : \dim\left(\text{Span}_{\mathbb{R}}(\mathcal{B}_p(\mathbf{0}, r) \cap \mathcal{L})\right) \geq i\}.$$

As the usual metric is the Euclidean norm, we omit the superscript p when $p = 2$. We note however that considering other values of p may lead to very different quantities and problems as a result.

2.2.3 Minkowski's Theorem

Although the two invariants introduced in Section 2.2.2 are not sufficient to fully characterize a lattice, they are linked by the following fundamental theorem.

Theorem 2.1 (Minkowski's First Theorem)

Let d be a positive integer and $\mathcal{L}$ a lattice of rank d. It holds that $\lambda_1(\mathcal{L}) \leq \sqrt{d}(\text{Vol}\,\mathcal{L})^{1/d}$.

The result is satisfying from a theoretical point of view. The exponent normalizes the volume of the lattice to a unidimensional quantity, and the theorem announces that this normalization seems to be a good approximation of the length of the shortest vector. There exist several proofs for it. In this course, we adopt a geometrical approach linked to the counting of lattice points in a ball. More precisely, we first prove the following theorem.

Theorem 2.2 (Convex Field Theorem, Minkowski)

Let d be a positive integer. Then, let $C \subseteq \mathbb{R}^d$ be a convex set that is symmetric and bounded, and let $\mathcal{L} \subset \mathbb{R}^d$ be a lattice of rank d. If $\text{Vol}\,C > 2^d \text{Vol}\,\mathcal{L}$, then C contains a non-zero vector of $\mathcal{L}$.

Proof (Theorem 2.2). First, we observe that without loss of generality we can assume $\mathcal{L} = \mathbb{Z}^d$. Indeed, consider $\mathbf{B} \in GL_d(\mathbb{R})$ a basis of $\mathcal{L}$, i.e., $\mathcal{L} = \mathbf{B}\mathbb{Z}^d$. Then $\text{Vol}\,(\mathbf{B}^{-1} \cdot C) = \text{Vol}\,C/\text{Vol}\,\mathcal{L}$. Because $\mathbf{B}^{-1} \cdot C$ is also convex, symmetric and bounded, it suffices to show that if $\text{Vol}\,C > 2^d$ then there exists a non-zero integer vector in C.

For that we consider the "half convex" $C' = \{\frac{1}{2}\mathbf{x}; \mathbf{x} \in C\}$. By assumption, it thus holds that $\text{Vol}\,C' = \text{Vol}\,C/2^d > 1$. We now show that there necessarily exists two distinct translations of C' by $\mathbb{Z}^d$ which are non disjoint. Assume towards contradiction that none of the translation intersect. For a positive integer $r > 0$, we consider the family $\mathcal{F}_r = \{C' + \mathbf{u}; \mathbf{u} \in [-r, r]^d\}$. We denote by D the diameter of C, and by K the hypercube $[-r - D, r + D]^d$ which by construction contains the entire family $\mathcal{F}_r$. We thus have

$$\text{Vol}\,\mathcal{F}_r = (2\lfloor r \rfloor + 1)^d \text{Vol}\,C' \leq \text{Vol}\,K = (2r + 2D)^d,$$

which can written as

$$\operatorname{Vol} C' \leq \left(1 + \frac{2D-1}{2r+1}\right)^d.$$

Since r is arbitrary, we thus have that $\operatorname{Vol} C' \leq \lim_{r \to +\infty} (1 + (2D-1)/(2r+1))^d = 1$, thus contradicting the assumption that $\operatorname{Vol} C' > 1$.

Thence, there exists two translations of C' that intersect, meaning that there exists $\mathbf{u}_1 \neq \mathbf{u}_2$ in $\mathbb{Z}^d$ such that $(C' + \mathbf{u}_1) \cap (C' + \mathbf{u}_2) \neq \varnothing$. As a result, there exists $\mathbf{x}' \in \mathbb{R}^d$ such that $\mathbf{x}' - \mathbf{u}_1 \in C'$ and $\mathbf{x}' - \mathbf{u}_2 \in C'$. Define $\mathbf{x} = \mathbf{x}' - \mathbf{u}_1$. Then $\mathbf{x} \in C'$ and $\mathbf{x} - \mathbf{u} \in C'$ for $\mathbf{u} = \mathbf{u}_2 - \mathbf{u}_1 \neq \mathbf{0}$. Since C' is symmetric, $\mathbf{u} - \mathbf{x}$ is also in C', and by convexity, the segment $[\![\mathbf{x}, \mathbf{u} - \mathbf{x}]\!]$ is included in C'. Henceforth, the middle of that segment $\mathbf{m} = \frac{1}{2}\mathbf{u}$ is also in C'.

All in all, this shows that $2\mathbf{m} = \mathbf{u}$ is in C. Yet $\mathbf{u}$ is in $\mathbb{Z}^d \setminus \{\mathbf{0}\}$ thus proving the theorem.

The proof of Minkowski's First Theorem is a natural consequence of the Convex Field Theorem. We provide the proof here.

Proof (Theorem 2.1). The open ball $C = \mathcal{B}_2^o(\mathbf{0}, \lambda_1(\mathcal{L}))$ is a convex set that is symmetric bounded, and by definition of $\lambda_1(\mathcal{L})$, it holds that $C \cap \mathcal{L} = \{\mathbf{0}\}$. From the previous theorem, its volume must therefore be less than or equal to $2^d \operatorname{Vol} \mathcal{L}$. Additionally, C contains the hypercube $H = \mathcal{B}_\infty^o(\mathbf{0}, \lambda_1(\mathcal{L})/\sqrt{d})$. Indeed, for all $\mathbf{x}$ in H, we have $\|\mathbf{x}\|_2 \leq \sqrt{d}\|\mathbf{x}\|_\infty < \sqrt{d}\lambda_1(\mathcal{L})/\sqrt{d} = \lambda_1(\mathcal{L})$, meaning that $\mathbf{x}$ belongs to C. As a result, $\operatorname{Vol} C \geq \operatorname{Vol} H = (2\lambda_1(\mathcal{L})/\sqrt{n})^d$. In the end, we get

$$\left(\frac{2\lambda_1(\mathcal{L})}{\sqrt{d}}\right)^d \leq \operatorname{Vol} C \leq 2^d \operatorname{Vol} \mathcal{L},$$

which can be rewritten as

$$\lambda_1(\mathcal{L}) \leq \sqrt{d}(\operatorname{Vol} \mathcal{L})^{1/d},$$

as desired.

We notice that in the proof, we lower bound the volume of the ball by the volume of the inscribed hypercube. We can instead use the volume of the ball directly, which is a little more complex. For that we use the following formula.

Lemma 2.6 (Volume of a Hyperball)

Let d be a positive integer, and r, p a positive real. It holds that

$$\operatorname{Vol} \mathcal{B}_p(\mathbf{0}, r) = \frac{(2r\Gamma(1/p+1))^d}{\Gamma(d/p+1)}$$

where Γ is the Gamma function defined by $\Gamma(x) = \int_0^{+\infty} e^{-t} t^{x-1} dt$ for all $x > 0$. In particular for $p = 2$, we have $\operatorname{Vol} \mathcal{B}_2(\mathbf{0}, r) = (\sqrt{\pi} r)^d / \Gamma(d/2+1)$.

Lemma 2.6 used in the proof of Theorem 2.1 above thus gives the following bound

$$\lambda_1(\mathcal{L}) \leq \frac{2}{\sqrt{\pi}} \cdot (\Gamma(d/2+1)\operatorname{Vol} \mathcal{L})^{1/d}.$$

Using upper bounds on the Gamma function, namely $\Gamma(x+1) < \sqrt{2\pi} x^x e^{-x} (x^2 + x/3 + 1/18)^{1/4}$, we get

$$\lambda_1(\mathcal{L}) \leq \sqrt{\frac{2}{e\pi}} \left(\sqrt{2\pi}\left(\frac{d^2}{4} + \frac{d}{6} + \frac{1}{18}\right)\right)^{1/4d} \cdot \sqrt{d}(\operatorname{Vol} \mathcal{L})^{1/d} \leq 0.67752 \cdot \sqrt{d}(\operatorname{Vol} \mathcal{L})^{1/d}$$

Example 2.3

It is easy to construct lattices that have a shortest vector which has length arbitrarily lower than this upper bound. For that, for all $0 < \varepsilon < 1$, consider the lattice $\mathcal{L} = \mathcal{L}(\varepsilon \mathbf{e}_1, \frac{1}{\varepsilon}\mathbf{e}_2, \mathbf{e}_3, \ldots, \mathbf{e}_d)$. Then, $\operatorname{Vol} \mathcal{L} = 1$ so the upper bound does not depend on ε.

However, $\lambda_1(\mathcal{L}) = \min(\varepsilon, 1/\varepsilon, 1) = \varepsilon$. We can therefore make ε go to 0, without changing the upper bound.

Remark 2.3

The upper bound is actually optimal up to a constant factor. More precisely, for all d, there exists a lattice $\mathcal{L}$ of rank d such that $\lambda_1(\mathcal{L}) = c\sqrt{d}(\mathrm{Vol}\,\mathcal{L})^{1/d}$ for a constant c. Theorem 2.1 shows that $c \leq 1$, while our finer analysis shows that $c \leq 1.4 \cdot \sqrt{2/(e\pi)} \leq 0.67752$.

It is also very natural to study the *Hermite constant* expressing the maximal ration between the length of the shortest vectors and the normalized volume. Let d be a positive integer. We denote by $\mathscr{L}_d$ the set of lattices of rank d. The *Hermite constant* of order d is defined by

$$\gamma_d = \sup_{\mathcal{L} \in \mathscr{L}_d} \frac{\lambda_1(\mathcal{L})^2}{(\mathrm{Vol}\,\mathcal{L})^{2/d}}.$$

We have the following associated theorem.

Theorem 2.3 (Hermite)

For all integer d larger or equal than 2, it holds that $\gamma_d \leq (\gamma_2)^{d-1}$, where $\gamma_2 = 4/3$.

2.2.4 Dual Lattice

Another fundamental notion of lattice theory is that we can define the dual of any lattice, and possibly work in this dual space. We caution here that the term *dual lattice* is not the same as the notion of a *dual space* for a vector space V, which would denote the space of all linear form of V.

Definition 2.8 (Dual Lattice)

Let $\mathcal{L}$ be a lattice. The dual lattice of $\mathcal{L}$, denoted by $\mathcal{L}^*$ or $\mathcal{L}^\vee$, is the lattice defined by

$$\mathcal{L}^* = \left\{\mathbf{y} \in \mathrm{Span}_{\mathbb{R}}(\mathcal{L}) : \forall \mathbf{x} \in \mathcal{L}, \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}\right\}.$$

Example 2.4

Let us first look at the simplest lattice, i.e., $\mathcal{L} = \mathbb{Z}^d$. Let $\mathbf{x}$ be in $\mathcal{L}^*$. Since the inner product with any vector of $\mathbb{Z}^d$ must be an integer, we can take the canonical basis vectors of $\mathbb{R}^d$, which are indeed in $\mathbb{Z}^d$. Hence, for all i in $[\![1, d]\!]$, we have $\langle \mathbf{x}, \mathbf{e}_i \rangle \in \mathbb{Z}$ which proves that $\mathbf{x} \in \mathbb{Z}^d$. Reciprocally, for all $\mathbf{x}, \mathbf{y}$ in $\mathbb{Z}^d$, $\langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}$. Thence, $\mathcal{L}^* = \mathcal{L} = \mathbb{Z}^d$. So the lattice $\mathbb{Z}^d$ is *self-dual*.
Let us now take a more interesting example, namely $\mathcal{L} = \alpha\mathbb{Z}^d$ for a non zero real α. Let $\mathbf{x}$ be in $\mathcal{L}^*$. Similarly as before, for all i in $[\![1, d]\!]$, we have $\langle \mathbf{x}, \alpha\mathbf{e}_i \rangle \in \mathbb{Z}$ which means that $\mathbf{x} \in \alpha^{-1}\mathbb{Z}^d$. Reciprocally, for all $(\mathbf{x}, \mathbf{y})$ in $\alpha^{-1}\mathbb{Z}^d \times \mathcal{L}$, $\langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}$. Thence, $\mathcal{L}^* = \alpha^{-1}\mathbb{Z}^d$.

We can generalize the result of Example 2.4 to arbitrary lattices and not just the integer lattice.

Lemma 2.7 (Dual Lattice Scaling)

Let k, d be two positive integers, and $\mathcal{L} \subset \mathbb{R}^k$ a lattice of rank d. For all $\alpha \neq 0$, it holds that $(\alpha\mathcal{L})^* = \alpha^{-1}\mathcal{L}^*$.

Proof (Lemma 2.7). First, let $\mathbf{x}$ be in $\alpha^{-1}\mathcal{L}^*$, i.e., $\mathbf{x} = \alpha^{-1}\widetilde{\mathbf{x}}$ with $\widetilde{\mathbf{x}} \in \mathcal{L}^*$. Let $\mathbf{y}$ be in $\alpha\mathcal{L}$, i.e., $\mathbf{y} = \alpha\widetilde{\mathbf{y}}$ for $\widetilde{\mathbf{y}} \in \mathcal{L}$. It then holds that $\langle \mathbf{x}, \mathbf{y} \rangle = \alpha^{-1}\alpha\langle \widetilde{\mathbf{x}}, \widetilde{\mathbf{y}} \rangle = \langle \widetilde{\mathbf{x}}, \widetilde{\mathbf{y}} \rangle \in \mathbb{Z}$ by definition of the dual. Additionally, it holds that

$$\mathbf{x} \in \alpha^{-1}\mathrm{Span}_{\mathbb{R}}(\mathcal{L}) = \mathrm{Span}_{\mathbb{R}}(\mathcal{L}) = \mathrm{Span}_{\mathbb{R}}(\alpha\mathcal{L}),$$

which concludes proving that $\mathbf{x} \in (\alpha\mathcal{L})^*$. Hence $\alpha^{-1}\mathcal{L}^* \subseteq (\alpha\mathcal{L})^*$.

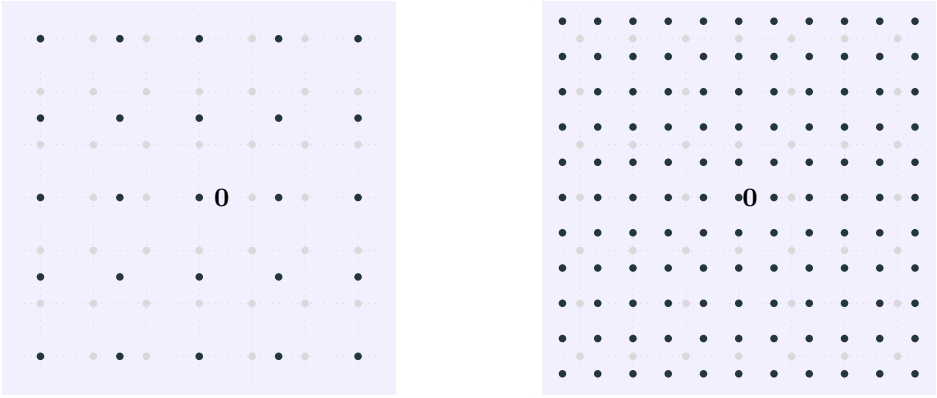


Figure 2.3: The scaled lattice $\mathcal{L} = \frac{3}{2}\mathbb{Z}^2$ and its dual $\mathcal{L}^* = \frac{2}{3}\mathbb{Z}^2$. The gray points represent the lattice $\mathbb{Z}^2$ in the background for reference.

Reciprocally, let $\mathbf{x}$ be in $(\alpha\mathcal{L})^*$ and define $\widetilde{\mathbf{x}} = \alpha\mathbf{x}$. Let $\mathbf{y} \in \mathcal{L}$. We have $\langle \widetilde{\mathbf{x}}\,, \mathbf{y}\rangle = \langle \alpha\mathbf{x}\,, \mathbf{y}\rangle = \langle \mathbf{x}\,, \alpha\mathbf{y}\rangle \in \mathbb{Z}$ by definition of $(\alpha\mathcal{L})^*$. Finally, we also have $\mathbf{x} \in \mathrm{Span}_{\mathbb{R}}(\alpha\mathcal{L}) = \alpha^{-1}\mathrm{Span}_{\mathbb{R}}(\mathcal{L})$ similarly as before. So $(\alpha\mathcal{L})^* \subseteq \alpha^{-1}\mathcal{L}^*$, thus proving equality.

Since we can associate a basis to any lattice, a natural question is whether we can make connections between a basis of the lattice $\mathcal{L}$ and a basis of its dual lattice $\mathcal{L}^*$. This leads us to the notion of *dual basis*.

Definition 2.9 (Dual Lattice)

Let k, d be two positive integers and $(\mathbf{b}_i)_{i\in[\![1,d]\!]}$ a linearly independent family of $\mathbb{R}^k$. Let $\mathbf{B} = [\mathbf{b}_i]_{i\in[\![1,d]\!]}$ be the matrix representation of this lattice basis. The *dual basis* of $(\mathbf{b}_i)_{i\in[\![1,d]\!]}$ is the unique basis $\mathbf{D} = [\mathbf{d}_i]_{i\in[\![1,d]\!]} \in \mathbb{R}^{k\times d}$ such that

- $\mathrm{Span}_{\mathbb{R}}(\mathbf{B}) = \mathrm{Span}_{\mathbb{R}}(\mathbf{D})$,
- $\mathbf{B}^T\mathbf{D} = \mathbf{I}_d$.

The dual basis is intrinsically link to the *primal* basis with the second condition. In particular, since $\mathbf{B}$ is full-rank, it holds that $\mathbf{B}^T\mathbf{B}$ is in $GL_d(\mathbb{R})$, and as a result, the dual basis $\mathbf{D}$ is the Moore-Penrose pseudo-inverse of $\mathbf{B}^T$, denoted by $(\mathbf{B}^T)^+ = \mathbf{B}(\mathbf{B}^T\mathbf{B})^{-1}$. We then have the following results.

Lemma 2.8 (Dual Lattice Properties)

Let k, d be two positive integers, and $\mathcal{L} \subset \mathbb{R}^k$ a lattice of rank d with basis $\mathbf{B} \in \mathbb{R}^{k\times d}$, and dual basis $\mathbf{D}$. It then holds that $\mathcal{L}^* = \mathcal{L}(\mathbf{D})$, that $(\mathcal{L}^*)^* = \mathcal{L}$ and that $\mathrm{Vol}\,\mathcal{L}^* = (\mathrm{Vol}\,\mathcal{L})^{-1}$.

Proof (Lemma 2.8). Let $\mathbf{y}$ be in $\mathcal{L}(\mathbf{D})$. There exists $\mathbf{x} \in \mathbb{Z}^d$ such that $\mathbf{y} = \mathbf{D}\mathbf{x}$. First, we have $\mathbf{y} \in \mathrm{Span}_{\mathbb{R}}(\mathbf{D}) = \mathrm{Span}_{\mathbb{R}}(\mathbf{B})$, and thus, for $\mathcal{L} = \mathcal{L}(\mathbf{B})$, it yields $\langle \mathcal{L}\,, \mathbf{y}\rangle = (\mathbb{Z}^d)^T\mathbf{B}^T\mathbf{D}\mathbf{x} = (\mathbb{Z}^d)^T\mathbf{x} \subseteq \mathbb{Z}$. This shows $\mathbf{y} \in \mathcal{L}^*$.

Conversely, let $\mathbf{y}$ be in $\mathcal{L}^*$. We thus have $\mathbf{y} \in \mathrm{Span}_{\mathbb{R}}(\mathbf{B}) = \mathrm{Span}_{\mathbb{R}}(\mathbf{D})$ and as a result there exists $\mathbf{x} \in \mathbb{R}^d$ such that $\mathbf{y} = \mathbf{D}\mathbf{x}$. By definition of the dual lattice, for all $\mathbf{t} \in \mathbb{Z}^d$, it holds that $\langle \mathbf{B}\mathbf{t}\,, \mathbf{y}\rangle \in \mathbb{Z}$. By taking $\mathbf{t}$ to be the vectors $\mathbf{e}_i$ of the canonical basis of $\mathbb{R}^d$, we get $\mathbf{B}^T\mathbf{D}\mathbf{x} \in \mathbb{Z}^d$. Since we also have the identity $\mathbf{B}^T\mathbf{D} = \mathbf{I}_d$, this proves that $\mathbf{x} \in \mathbb{Z}^d$. Hence, $\mathbf{y} \in \mathbf{D}\mathbb{Z}^d = \mathcal{L}(\mathbf{D})$. As a result we have $\mathcal{L}^* = \mathcal{L}(\mathbf{D})$.

Now, recalling that the dual basis $\mathbf{D}$ is $(\mathbf{B}^T)^+$, we have that the dual basis of $\mathbf{D}$ is $(\mathbf{D}^T)^+ = (((\mathbf{B}^T)^+)^T)^+$. Yet, the pseudo-inverse and transpose commute and are involutions. Hence, $(\mathbf{D}^T)^+ = \mathbf{B}$. We can also verify that using the closed-form expressions. Using the previous property, it yields $(\mathcal{L}^*)^* = (\mathcal{L}(\mathbf{D}))^* = \mathcal{L}((\mathbf{D}^T)^+) = \mathcal{L}(\mathbf{B}) = \mathcal{L}$.

For the last property, we simply have $\mathrm{Vol}\,\mathcal{L}^* = \sqrt{\det(\mathbf{D}^T\mathbf{D})} = \sqrt{\det((\mathbf{B}^T\mathbf{B})^{-1})} =$

$(\text{Vol}\,\mathcal{L})^{-1}$, as claimed.

We can now wonder about the other lattice invariants that we have introduced in Section 2.2.2. We have just proven how to link the volume of a lattice with the volume of its dual lattice. In particular, it shows that if the lattice is dense, then its dual is sparse. We can visualize this property in Figure 2.3. The other invariants that we have defined were the successive minima. The following results allow us to link the minima of a lattice with that of its dual.

Lemma 2.9 (Dual Lattice Minima Bound)

Let d be a positive integer and $\mathcal{L}$ a lattice of rank d. It then holds that

- $\lambda_1(\mathcal{L}) \cdot \lambda_1(\mathcal{L}^*) \leq d$,
- $\lambda_1(\mathcal{L}) \cdot \lambda_d(\mathcal{L}^*) \geq 1$.

Proof (Lemma 2.9). Using Minkowski's first theorem (Theorem 2.1) on both $\mathcal{L}$ and $\mathcal{L}^*$, we get that $\lambda_1(\mathcal{L}) \leq \sqrt{d}(\text{Vol}\,\mathcal{L})^{1/d}$ and $\lambda_1(\mathcal{L}^*) \leq \sqrt{d}(\text{Vol}\,\mathcal{L}^*)^{1/d}$. By multiplying both identities and using the third property of Lemma 2.8, we indeed get $\lambda_1(\mathcal{L}) \cdot \lambda_1(\mathcal{L}^*) \leq d$.

For the second inequality, we proceed as follows. Let $\mathbf{x}$ be in $\mathcal{L}$ such that $\|\mathbf{x}\|_2 = \lambda_1(\mathcal{L})$.

Suppose that: $\forall \mathbf{y} \in \mathcal{L}^*, \|\mathbf{y}\|_2 \leq \lambda_d(\mathcal{L}^*) \implies \langle \mathbf{x}\,,\mathbf{y}\rangle = 0$. Then, since there are at least d linearly independent vectors of $\mathcal{L}^*$ of norm less than or equal to $\lambda_d(\mathcal{L}^*)$ (by Definition 2.7), we have that $\langle \mathbf{x}\,, \text{Span}_{\mathbb{R}}(\mathcal{L}^*)\rangle = \{0\}$. Since $\text{Span}_{\mathbb{R}}(\mathcal{L}^*) = \text{Span}_{\mathbb{R}}(\mathbf{B})$ for a basis $\mathbf{B}$ of $\mathcal{L} = \mathcal{L}(\mathbf{B})$ and that $\mathbf{x}$ is also in $\text{Span}_{\mathbb{R}}(\mathbf{B})$, it yields that $\langle \mathbf{x}\,,\mathbf{x}\rangle = 0$ and thus $\mathbf{x} = \mathbf{0}$. Yet, $\mathbf{x}$ is non-zero by definition of λ_1.

This contradiction thus gives the existence of $\mathbf{y}$ in $\mathcal{L}^*$ of norm less than $\lambda_d(\mathcal{L}^*)$ and non-orthogonal to $\mathbf{x}$. By Cauchy-Schwarz, we have

$$\lambda_1(\mathcal{L})\lambda_d(\mathcal{L}^*) \geq \|\mathbf{x}\|_2\|\mathbf{y}\|_2 \geq \left|\langle \mathbf{x}\,,\mathbf{y}\rangle\right| \in \mathbb{R}^{+*} \cap \mathbb{Z}.$$

So $\lambda_1(\mathcal{L})\lambda_d(\mathcal{L}^*) \geq 1$.

In 1993, Banaszczyk [Ban93] proved a tighter upper bound than that of Lemma 2.9.

Theorem 2.4 (Banaszczyk's Theorem [Ban93])

Let d be a positive integer and $\mathcal{L}$ a lattice of rank d. It then holds that

$$1 \leq \lambda_1(\mathcal{L}) \cdot \lambda_d(\mathcal{L}^*) \leq d$$

2.3 Gram-Schmidt Orthogonalization

As we will see in Section 2.4, lattice reduction techniques heavily use the Gram-Schmidt Orthogonalization process. Intuitively, it is easier to combine long vectors to find short ones if those long vectors are close to orthogonal. Additionally, note that if a lattice $\mathcal{L}$ has a basis that is orthogonal, then one can find its shortest vectors directly. Indeed, if $\mathbf{B}$ is a basis of $\mathcal{L}$ such that its Gram matrix $\mathbf{B}^T\mathbf{B}$ is diagonal, then every orthogonal basis of $\mathcal{L}$ is a rotation of $\mathbf{B}$. This yields that $\lambda_1(\mathcal{L}) = \min_{i\in[\![1,d]\!]}\|\mathbf{b}_i\|_2$. However, not all lattices have an orthogonal basis. In this case, the idea is to find a basis that is as orthogonal as it can get, that is a basis $\mathbf{B}$ that minimizes all the cross-product $\left|\langle \mathbf{b}_i\,,\mathbf{b}_j\rangle\right|$ for $i \neq j$. To do so, the Gram-Schmidt Orthogonalization is paramount.

2.3.1 Orthogonality

We recall that in a euclidean space $(V, \langle\cdot\,,\cdot\rangle)$, two vectors $\mathbf{x}$ and $\mathbf{y}$ are said orthogonal if and only if $\langle \mathbf{x}\,,\mathbf{y}\rangle$. For all subspace W of V, the *orthogonal of* W is the subspace

$$W^{\perp} = \{\mathbf{y} \in V : \forall \mathbf{x} \in W, \langle \mathbf{x}\,,\mathbf{y}\rangle = 0\}.$$

We recall that we have the direct sum $V = W \oplus W^\perp$, which in particular yields $\dim W^\perp = \dim V - \dim W$ and $W \cap W^\perp = \{\mathbf{0}\}$. We define $k = \dim V$ and $d = \dim W$. A *projection* is a linear map P such that $P^2 = P$. This projection is called orthogonal projection if for all $(\mathbf{x}, \mathbf{y})$ in V^2, it holds that $\langle P(\mathbf{x}) \, , \mathbf{y}\rangle = \langle \mathbf{x} \, , P(\mathbf{y})\rangle$. In that case, there exists a unique projection $P^\perp$ such that $P + P^\perp = \mathrm{id}_V$. As expected, if the range of P is the subspace W, then the range of $P^\perp$ is $W^\perp$. Equivalently, it holds that $\ker P = W^\perp$ and $\ker P^\perp = W$.

Let $(\mathbf{b}_i)_{i\in[\![1,d]\!]}$ be a basis of W and $\mathbf{B}$ its matrix representation in the canonical basis of V. The matrix of P in the canonical basis of V is $\mathbf{P} = \mathbf{B}(\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T$, and that of $P^\perp$ is $\mathbf{I}_k - \mathbf{P}$. A family $(\mathbf{b}_i)_{i\in[\![1,d]\!]}$ is called *orthogonal* if for all $(i, j) \in [\![1, d]\!]^2$ such that $i \neq j$, then $\langle \mathbf{b}_i \, , \mathbf{b}_j\rangle = 0$. It holds that an orthogonal family is necessarily linearly independent. If $(\mathbf{b}_i)_{i\in[\![1,d]\!]}$ is an orthogonal family that is also a basis of W, then for all $\mathbf{x}$ in W, it holds

$$\mathbf{x} = \sum_{i\in[\![1,d]\!]} \frac{\langle \mathbf{x} \, , \mathbf{b}_i\rangle}{\|\mathbf{b}_i\|_2^2}\mathbf{b}_i.$$

In particular, if we denote by P_i the orthogonal projection on the subspace $\mathbb{R}\mathbf{b}_i$, we have $P_i(\mathbf{x}) = \frac{\langle \mathbf{x},\mathbf{b}_i\rangle}{\|\mathbf{b}_i\|_2^2}\mathbf{b}_i$, and that the linear form $\mathbf{x} \mapsto \langle \mathbf{x} \, , \mathbf{b}_i\rangle/\|\mathbf{b}_i\|_2^2$ gives the coordinate of $\mathbf{x}$ along $\mathbf{b}_i$.

A matrix $\mathbf{B} \in R^{d\times d}$ for $R \subseteq \mathbb{R}$ is called *orthogonal* if $\mathbf{B}^T\mathbf{B} = \mathbf{I}_d = \mathbf{B}\mathbf{B}^T$. The set of orthogonal matrices of dimension d is denoted by $\mathcal{O}_d(R)$.

2.3.2 Gram-Schmidt Process

It is known that there always exists orthogonal bases of euclidean spaces, but even better, the Gram-Schmidt Orthogonalization process gives an algorithm to compute them. This process is crucial when studying lattices from an algorithmic perspective. So let $(\mathbf{b}_i)_{i\in[\![1,d]\!]}$ be a linearly independent family of $\mathbb{R}^k$. We define the new family $(\mathbf{b}_i^*)_{i\in[\![1,d]\!]}$ iteratively with the following process.

Algorithm 2.1: GSO (Gram-Schmidt Orthogonalization)

Input: Linearly independent family $(\mathbf{b}_i)_{i\in[\![1,d]\!]}$ of $\mathbb{R}^k$.

1. $\mathbf{b}_1^* \leftarrow \mathbf{b}_1$
2. **for** $i = 2$ **to** d
   (a) $\mathbf{b}_i^* \leftarrow \mathbf{b}_i - \sum_{j\in[\![1,i-1]\!]} \frac{\langle \mathbf{b}_i \, , \mathbf{b}_j^*\rangle}{\left\|\mathbf{b}_j^*\right\|_2^2}\mathbf{b}_j^*$

Output: $(\mathbf{b}_i^*)_{i\in[\![1,d]\!]}$

Intuitively, each new vector is constructed by "orthogonally removing" the contribution of the vector space already spanned. Formally, $\mathbf{b}_2^* = \mathbf{b}_2 - P_1(\mathbf{b}_2) = P_1^\perp(\mathbf{b}_2)$, $\mathbf{b}_3^* = \mathbf{b}_3 - P_1(\mathbf{b}_3) - P_2(\mathbf{b}_3) = \mathbf{b}_3 - P_{\mathrm{Span}(\mathbf{b}_1,\mathbf{b}_2)}(\mathbf{b}_3) = P^\perp_{\mathrm{Span}(\mathbf{b}_1,\mathbf{b}_2)}(\mathbf{b}_3)$, and so on.

Lemma 2.10 (Span Preservation)

Let k, d be two positive integers, and $(\mathbf{b}_i)_{i\in[\![1,d]\!]}$ a linearly independent family of $\mathbb{R}^k$. We denote by $(\mathbf{b}_i^*)_{i\in[\![1,d]\!]} = \mathsf{GSO}((\mathbf{b}_i)_{i\in[\![1,d]\!]})$ obtained from Algorithm 2.1. Then $(\mathbf{b}_i^*)_{i\in[\![1,d]\!]}$ is an orthogonal family, and for all i in $[\![1, d]\!]$, we have $\mathrm{Span}_\mathbb{R}(\mathbf{b}_1, \ldots, \mathbf{b}_i) = \mathrm{Span}_\mathbb{R}(\mathbf{b}_1^*, \ldots, \mathbf{b}_i^*)$.

Proof (Lemma 2.10). For all i in $[\![1, d]\!]$, we define the predicate $\mathcal{P}(i)$: "$(\mathbf{b}_j^*)_{j\in[\![1,i]\!]}$ is an orthogonal family and $\mathrm{Span}_\mathbb{R}(\mathbf{b}_1, \ldots, \mathbf{b}_i) = \mathrm{Span}_\mathbb{R}(\mathbf{b}_1^*, \ldots, \mathbf{b}_i^*)$". We prove $\mathcal{P}(i)$ by induction. For clarity, we define $\mu_{i,j} = \langle \mathbf{b}_i \, , \mathbf{b}_j^*\rangle/\left\|\mathbf{b}_j^*\right\|_2^2$ for all $1 \leq j < i \leq d$.

Initialization. Since $\mathbf{b}_1^* = \mathbf{b}_1 \neq \mathbf{0}$, it is an orthogonal family, and trivially $\mathrm{Span}_\mathbb{R}(\mathbf{b}_1) = \mathrm{Span}_\mathbb{R}(\mathbf{b}_1^*)$. This proves $\mathcal{P}(1)$.

Induction. Let i be in $[d-1]$ and assume $\mathcal{P}(i)$. Let j be in $[\![1,i]\!]$. We have

$$\begin{aligned}\langle \mathbf{b}_{i+1}^* , \mathbf{b}_j^* \rangle &= \langle \mathbf{b}_{i+1} , \mathbf{b}_j^* \rangle - \sum_{\ell \in [\![1,i]\!]} \mu_{i+1,\ell} \langle \mathbf{b}_\ell^* , \mathbf{b}_j^* \rangle \\ &= \langle \mathbf{b}_{i+1} , \mathbf{b}_j^* \rangle - \mu_{i+1,j} \left\| \mathbf{b}_j^* \right\|_2^2 \text{ (by } \mathcal{P}(i)) \\ &= 0,\end{aligned}$$

which proves the orthogonality. Then, by construction, $\mathbf{b}_{i+1} = \mathbf{b}_{i+1}^* + \sum_{\ell \in [\![1,i]\!]} \mu_{i+1,\ell} \mathbf{b}_\ell^*$ and thus in the span $\mathrm{Span}_{\mathbb{R}}(\mathbf{b}_1^*, \ldots, \mathbf{b}_{i+1}^*)$. By $\mathcal{P}(i)$, it thus holds that $\mathrm{Span}_{\mathbb{R}}(\mathbf{b}_1, \ldots, \mathbf{b}_{i+1}) \subseteq \mathrm{Span}_{\mathbb{R}}(\mathbf{b}_1^*, \ldots, \mathbf{b}_{i+1}^*)$. Reciprocally, $\mathbf{b}_{i+1}^*$ is in $\mathrm{Span}_{\mathbb{R}}(\mathbf{b}_1^*, \ldots, \mathbf{b}_i^*, \mathbf{b}_{i+1})$. By $\mathcal{P}(i)$, we obtain $\mathrm{Span}_{\mathbb{R}}(\mathbf{b}_1^*, \ldots, \mathbf{b}_{i+1}^*) \subseteq \mathrm{Span}_{\mathbb{R}}(\mathbf{b}_1^*, \ldots, \mathbf{b}_i^*, \mathbf{b}_{i+1}) = \mathrm{Span}_{\mathbb{R}}(\mathbf{b}_1, \ldots, \mathbf{b}_{i+1})$. All in all, this proves $\mathcal{P}(i+1)$.

As a result, the first part of $\mathcal{P}(d)$ proves the first lemma statement, while $(\mathcal{P}(i))_{i \in [\![1,d]\!]}$ proves the second lemma statement.

In a basis of $\mathbb{R}^k$, we can describe the process in matrix form directly by the following.

$$\mathbf{B} = \mathbf{B}^* \mathbf{U}, \text{ with } \mathbf{U} = \begin{bmatrix} 1 & \mu_{2,1} & \cdots & \mu_{d,1} \\ 0 & 1 & & \vdots \\ & & \ddots & \mu_{d,d-1} \\ & & & 1 \end{bmatrix},$$

where $\mathbf{B} = [\mathbf{b}_i]_{i \in [\![1,d]\!]}$ and $\mathbf{B}^* = [\mathbf{b}_i^*]_{i \in [\![1,d]\!]}$. The entries $\mu_{i,j}$ of $\mathbf{U}$ are generally called the *Gram-Schmidt coordinates of* $\mathbf{b}_i$. We note that $\mathbf{U}$ has determinant 1 but the Gram-Schmidt coordinates are reals and not necessarily integers. It always possible to obtain an orthonormal basis by normalizing the $\mathbf{b}_i^*$. However, this also normalizes the volume to 1, thus loosing a property of the underlying lattice. In this course, we will not normalize the basis unless specified otherwise. In particular, the Gram-Schmidt Orthogonalization preserves the volume information of the lattice spanned by the $(\mathbf{b}_i)_{i \in [\![1,d]\!]}$.

Lemma 2.11

Let k, d be two positive integers, and $(\mathbf{b}_i)_{i \in [\![1,d]\!]}$ a linearly independent family of $\mathbb{R}^k$. Let $\mathbf{B} = [\mathbf{b}_i]_{i \in [\![1,d]\!]}$ be the matrix representation of this basis. We let $(\mathbf{b}_i^*)_{i \in [\![1,d]\!]} = \mathsf{GSO}((\mathbf{b}_i)_{i \in [\![1,d]\!]})$. It holds that

$$\mathrm{Vol}\, \mathcal{L}(\mathbf{B}) = \prod_{i \in [\![1,d]\!]} \|\mathbf{b}_i^*\|_2 .$$

Proof (Lemma 2.11). We define $\mathbf{B}^* = [\mathbf{b}_i^*]_{i \in [\![1,d]\!]}$. We thus have $\mathbf{B} = \mathbf{B}^* \mathbf{U}$ where $\mathbf{U}$ is the upper triangular matrix of the Gram-Schmidt coordinates. We then have

$$\mathrm{Vol}\, (\mathcal{L}(\mathbf{B}))^2 = \det(\mathbf{B}^T \mathbf{B}) = (\det \mathbf{U})^2 \det(\mathbf{B}^{*T} \mathbf{B}^*) = \det(\mathbf{B}^{*T} \mathbf{B}^*),$$

where the last equality comes from the fact that $\det \mathbf{U} = 1$. Finally, since $\mathbf{B}^*$ is orthogonal, its Gram matrix is $\mathbf{B}^{*T} \mathbf{B}^* = \mathrm{diag}(\|\mathbf{b}_1^*\|_2^2, \ldots, \|\mathbf{b}_d^*\|_2^2)$. As a result, we have

$$\mathrm{Vol}\, \mathcal{L}(\mathbf{B}) = \prod_{i \in [\![1,d]\!]} \|\mathbf{b}_i^*\|_2 .$$

Corollary 2.1 (Hadamard Inequality)

Let k, d be two positive integers, and $(\mathbf{b}_i)_{i \in [\![1,d]\!]}$ a linearly independent family of $\mathbb{R}^k$. Let

$\mathbf{B} = [\mathbf{b}_i]_{i\in[\![1,d]\!]}$ be the matrix representation of this basis. It holds that

$$\operatorname{Vol}\mathcal{L}(\mathbf{B}) \leq \prod_{i\in[\![1,d]\!]} \|\mathbf{b}_i\|_2 .$$

Proof (Corollary 2.1). We define $\mathbf{B}^* = [\mathbf{b}_i^*]_{i\in[\![1,d]\!]}$, where $(\mathbf{b}_i^*)_{i\in[\![1,d]\!]} = \mathsf{GSO}((\mathbf{b}_i)_{i\in[\![1,d]\!]})$. By the Pythagorean theorem, we have

$$\|\mathbf{b}_i\|_2^2 = \|\mathbf{b}_i^*\|_2^2 + \sum_{j\in[\![1,i-1]\!]} \mu_{i,j}^2 \left\|\mathbf{b}_j^*\right\|_2^2 ,$$

thus proving that $\|\mathbf{b}_i\|_2 \geq \left\|\mathbf{b}_i^*\right\|_2$. By Lemma 2.11, we get

$$\operatorname{Vol}\mathcal{L}(\mathbf{B}) = \prod_{i\in[\![1,d]\!]} \|\mathbf{b}_i^*\|_2 \leq \prod_{i\in[\![1,d]\!]} \|\mathbf{b}_i\|_2 ,$$

as desired.

Finally, we give a final result which comes in very handy to obtain guarantees of lattice reduction techniques later on in this course.

Lemma 2.12

Let k, d be two positive integers, and $(\mathbf{b}_i)_{i\in[\![1,d]\!]}$ a linearly independent family of $\mathbb{R}^k$. Let $\mathbf{B} = [\mathbf{b}_i]_{i\in[\![1,d]\!]}$ be the matrix representation of this basis. We let $(\mathbf{b}_i^*)_{i\in[\![1,d]\!]} = \mathsf{GSO}((\mathbf{b}_i)_{i\in[\![1,d]\!]})$. It holds that

$$\lambda_1(\mathcal{L}(\mathbf{B})) \geq \min_{i\in[\![1,d]\!]} \|\mathbf{b}_i^*\|_2 .$$

Proof (Lemma 2.12). Let $\mathbf{b}$ be in $\mathcal{L}(\mathbf{B}) \setminus \{\mathbf{0}\}$. There exists an integer $i_0 \in [\![1, n]\!]$ and $k_1, \ldots, k_{i_0}$ in $\mathbb{Z}$ such that $k_{i_0} \neq 0$ and $\mathbf{b} = \sum_{i\in[\![1,i_0]\!]} k_i\mathbf{b}_i$. By using the definition of the Gram-Schmidt vectors and Lemma 2.10, we get that there exists $k'_1, \ldots, k'_{i_0-1}$ in $\mathbb{R}$ such that $\mathbf{b} = k_{i_0}\mathbf{b}_{i_0}^* + \sum_{i\in[\![1,i_0-1]\!]} k'_i\mathbf{b}_i^*$. The Pythagorean theorem thus gives $\|\mathbf{b}\|_2^2 = |k_{i_0}|^2\left\|\mathbf{b}_{i_0}^*\right\|_2^2 + \sum_{i\in[\![1,i_0-1]\!]} |k'_i|^2\left\|\mathbf{b}_i^*\right\|_2^2$. Since the right-hand side sum is positive and since $k_{i_0} \in \mathbb{Z}$, we get $\|\mathbf{b}\|_2^2 \geq \left\|\mathbf{b}_{i_0}^*\right\|_2^2 \geq \min_{i\in[\![1,d]\!]}\left\|\mathbf{b}_i^*\right\|_2^2$. By choosing $\mathbf{b}$ such that $\|\mathbf{b}\|_2 = \lambda_1(\mathcal{L}(\mathbf{B}))$, we obtain the result.

From now on, when $\mathbf{B} = [\mathbf{b}_i]_{i\in[\![1,d]\!]}$, we will simply denote by the $\mathbf{B}^*$ the matrix whose columns are $\mathsf{GSO}((\mathbf{b}_i)_{i\in[\![1,d]\!]})$.

2.3.3 Gram-Schmidt Minimum

An interesting lattice quantity, that will be relevant when performing sampling over the lattice, is the *Gram-Schmidt minimum*. It is essentially the minimal Gram-Schmidt length over all the possible bases of a lattice. For a matrix $\mathbf{A} \in \mathbb{R}^{k\times d}$, we denote by $\|\mathbf{A}\|_\infty = \max_{i\in[\![1,d]\!]}\|\mathbf{A}\mathbf{e}_i\|_2$, where $\mathbf{e}_i$ is the i-th canonical basis vector.

Definition 2.10 (Gram-Schmidt Minimum)

Let k, d be two positive integers, and $\mathcal{L} \subset \mathbb{R}^k$ a lattice of rank d. Temporarily, we denote by $\mathcal{B}(\mathcal{L})$ the set of all basis of the lattice $\mathcal{L}$. The *Gram-Schmidt minimum* of $\mathcal{L}$ is defined by

$$\lambda_{\mathsf{GSO}}(\mathcal{L}) = \min_{\mathbf{B}\in\mathcal{B}(\mathcal{L})} \left\|\mathsf{GSO}(\mathbf{B})\right\|_\infty = \min_{\mathbf{B}\in\mathcal{B}(\mathcal{L})} \max_{i\in[\![1,d]\!]} \|\mathbf{b}_i^*\|_2 .$$

The Gram-Schmidt minimum was introduced by Gentry, Peikert and Vaikuntanathan in 2008 [GPV08]

and was related to the usual successive minima of a lattice in the following result.

Lemma 2.13 (Gram-Schmidt and Successive Minima)

Let k, d be two positive integers, and $\mathcal{L} \subset \mathbb{R}^k$ a lattice of rank d. It holds that

$$\lambda_1(\mathcal{L}) \leq \lambda_{\mathsf{GSO}}(\mathcal{L}) \leq \lambda_d(\mathcal{L}) \leq \sqrt{d} \cdot \lambda_{\mathsf{GSO}}(\mathcal{L}).$$

2.4 Lattice Reduction

We have seen that all the bases of a lattice differ from a integer transformation with determinant ± 1. The set of such transformation is known as the *unimodular group* $GL_d(\mathbb{Z})$. We can thus summarize this result by the group action

$$\begin{array}{rcl} GL_d(\mathbb{Z}) \times GL_d(\mathbb{R}) & \longrightarrow & GL_d(\mathbb{R}) \\ (\mathbf{U}, \mathbf{B}) & \longmapsto & \mathbf{BU}, \end{array}$$

and a lattice corresponds to an orbit of this action. From this standpoint, *lattice reduction* comes down to finding "good" representatives for each orbit, where the term "good" depends on the context. In algorithmic and cryptography, a good representative is a basis that is as orthogonal as it can get and composed of the shortest possible vectors. We also would like to find such representatives *constructively and efficiently*. The state-of-the-art suggests that this is a hard problem, as we will see in Chapter 3, and we must therefore find a compromise between the *quality* (length if the basis vectors) guaranteed by the algorithm and the execution time.

2.4.1 Gauss-Lagrange Reduction: Size-Reduced Basis

It is possible to get an intuitive idea of reduction mechanisms for lattices of dimension 2. Let us consider a lattice described by a "very bad basis" $(\mathbf{b}_1, \mathbf{b}_2)$, where without loss of generality $\|\mathbf{b}_1\|_2 \leq \|\mathbf{b}_2\|_2$. By very bad basis, we mean that the vectors are very long and $\langle \mathbf{b}_1 \, , \mathbf{b}_2 \rangle / \|\mathbf{b}_1\|_2 \|\mathbf{b}_2\|_2 = \cos(\mathbf{b}_1, \mathbf{b}_2)$ is close to ± 1. We denote by $(\mathbf{b}_1^*, \mathbf{b}_2^*)$ its Gram-Schmidt Orthogonalization, and see that $\|\mathbf{b}_2\|_2^2 = \|\mathbf{b}_2\|_2^2 \cdot (1 - \cos(\mathbf{b}_1, \mathbf{b}_2)^2)$. Said differently, it means that $\mathbf{b}_2^*$ is rather short. Although $\mathbf{b}_2^*$ is not a vector in the lattice, there are lattice points close to $\mathbf{b}_2^*$. For example, consider the vector

$$\mathbf{b}_2' = \mathbf{b}_2 - \left\lfloor \frac{\langle \mathbf{b}_2 \, , \mathbf{b}_1^* \rangle}{\|\mathbf{b}_1^*\|_2^2} \right\rceil \mathbf{b}_1.$$

We thus have a new basis for the lattice, and it is possible that $\mathbf{b}_2'$ is much smaller that $\mathbf{b}_2$. In that case, we made progress and we reiterate the process on the basis $(\mathbf{b}_2', \mathbf{b}_1)$. Each time we reduce the vectors as such, we decrease the quantity $\|\mathbf{b}_1\|_2 \|\mathbf{b}_2\|_2$. The transformations preserve the lattice and thus its volume. The Hadamard inequality from Corollary 2.1 thus says that we cannot reduce the size of the basis indefinitely. Assuming we gain a factor strictly larger than 1 at each step on the product, such an algorithm should terminate rather quickly. What we have described is called the *Gauss-Lagrange Algorithm*, which we describe in Algorithm 2.2.

Algorithm 2.2: GL (Gauss-Lagrange)

Input: Lattice basis $(\mathbf{b}_1, \mathbf{b}_2)$

1. $(\mathbf{v}_1, \mathbf{v}_2) \leftarrow (\mathbf{b}_1, \mathbf{b}_2)$
2. **Repeat**
   (a) **if** $\|\mathbf{v}_2\|_2 < \|\mathbf{v}_1\|_2$, **then** $(\mathbf{v}_1, \mathbf{v}_2) \leftarrow (\mathbf{v}_2, \mathbf{v}_1)$
   (b) $\mu \leftarrow \lfloor \langle \mathbf{v}_1 \, , \mathbf{v}_2 \rangle / \|\mathbf{v}_1\|_2^2 \rceil$
   (c) **if** $\mu = 0$, **then return** $(\mathbf{v}_1, \mathbf{v}_2)$
   (d) **else** $\mathbf{v}_2 \leftarrow \mathbf{v}_2 - \mu \mathbf{v}_1$.

Output: Reduced basis $(\mathbf{v}_1, \mathbf{v}_2)$

Underlying this algorithm is the fundamental idea of taking as the new basis the best integer approximation of the Gram-Schmidt Orthogonalization of the original basis. We now formalize this notion.

Definition 2.11 (Size-Reduced Basis)

Let k, d be two positive integers, and $(\mathbf{b}_i)_{i\in[\![1,d]\!]}$ be a linearly independent family of $\mathbb{R}^k$. We let $(\mathbf{b}_i^*)_{i\in[\![1,d]\!]} = \mathsf{GSO}((\mathbf{b}_i)_{i\in[\![1,d]\!]})$. The basis $(\mathbf{b}_i)_{i\in[\![1,d]\!]}$ is called *size-reduced* if

$$\max_{1\leq j<i\leq d} \left| \frac{\langle \mathbf{b}_i \, , \mathbf{b}_j^* \rangle}{\left\| \mathbf{b}_j^* \right\|_2^2} \right| \leq \frac{1}{2}.$$

These ideas also gave rise to the size reduction algorithm which we now give in Algorithm 2.3.

Algorithm 2.3: SizeReduce (Size-Reduction)

Input: Lattice basis $(\mathbf{b}_i)_{i\in[\![1,d]\!]}$

1. $(\mathbf{b}_i^*)_{i\in[\![1,d]\!]} = \mathsf{GSO}((\mathbf{b}_i)_{i\in[\![1,d]\!]})$ ▷ Algorithm 2.1
2. **for** $i = 2$ to d
   (a) **for** $j = i-1$ to 1
      i. $\mathbf{b}_i \leftarrow \mathbf{b}_i - \left\lfloor \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\left\| \mathbf{b}_j^* \right\|_2^2} \right\rceil \mathbf{b}_j$

Output: Size-reduced basis $(\mathbf{b}_i)_{i\in[\![1,d]\!]}$

Lemma 2.14 (Correctness)

Let k, d be two positive integers, and $(\mathbf{b}_i)_{i\in[\![1,d]\!]}$ be a linearly independent family of $\mathbb{R}^k$. Let $(\mathbf{b}'_i)_{i\in[\![1,d]\!]} = \mathsf{SizeReduce}((\mathbf{b}_i)_{i\in[\![1,d]\!]})$ from Algorithm 2.3. Then $(\mathbf{b}'_i)_{i\in[\![1,d]\!]}$ is a size-reduced basis of $\mathcal{L}((\mathbf{b}_i)_{i\in[\![1,d]\!]})$.

Proof (Lemma 2.14). The proof relies on the fact the loops do not modify the Gram-Schmidt $(\mathbf{b}_i^*)_{i\in[\![1,d]\!]}$. We denote by P_{i-1} the orthogonal projection onto $\mathrm{Span}_{\mathbb{R}}(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})^{\perp}$, so that by definition $\mathbf{b}_i^* = P_{i-1}(\mathbf{b}_i)$. By Lemma 2.10, we have $\mathrm{Span}_{\mathbb{R}}(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}) = \mathrm{Span}_{\mathbb{R}}(\mathbf{b}_1^*, \ldots, \mathbf{b}_{i-1}^*)$. We also have that for all j in $[\![1, i-1]\!]$ and α in $\mathbb{R}$, $P_{i-1}(\mathbf{b}_i - \alpha \mathbf{b}_j) = P_{i-1}(\mathbf{b}_i)$, which implies the invariance of the Gram-Schmidt vectors all along the algorithm. Now let $\mathbf{b}_i^{(j)}$ be the vector obtained after the subtraction of a multiple of $\mathbf{b}_j$ to $\mathbf{b}_i$ in the second loop. For j in $[\![1, i-2]\!]$, we have by definition

$$\langle \mathbf{b}_i^{(j)} \, , \mathbf{b}_j^* \rangle = \langle \mathbf{b}_i^{(j+1)} \, , \mathbf{b}_j^* \rangle - \left\lfloor \frac{\langle \mathbf{b}_i^{(j+1)} \, , \mathbf{b}_j^* \rangle}{\left\| \mathbf{b}_j^* \right\|_2^2} \right\rceil \langle \mathbf{b}_j \, , \mathbf{b}_j^* \rangle.$$

Even though the current $\mathbf{b}_i$ are not those from the start, the invariance of the Gram-Schmidt vectors gives $\langle \mathbf{b}_j \, , \mathbf{b}_j^* \rangle = \left\| \mathbf{b}_j^* \right\|_2^2$. Hence, the j-th coordinate of the Gram-Schmidt of $\mathbf{b}_i^{(j)}$ is $\mu'_{i,j} = \langle \mathbf{b}_i^{(j)} \, , \mathbf{b}_j^* \rangle / \left\| \mathbf{b}_j \right\|_2^2$. The equality above then gives $\left| \mu'_{i,j} \right| \leq 1/2$.

We then remark that, for example looking at the matrix $\mathbf{U}$, subtracting a multiple of $\mathbf{b}_j$ to a vector only impacts the first j coordinates in the Gram-Schmidt basis. The following steps in the second loop thus do not modify the $\mu_{i,j}$ that are already reduced.

We now only have to show that the lattice is preserved. It is not difficult to be convinced that the matrix corresponding to a transfer in the internal loop has integer coefficients and determinant 1. In fact, the transfer matrix in the iteration (i, j) of the inner loop is the following *shear matrix*

$$\mathbf{I}_d - \left\lfloor \frac{\langle \mathbf{b}_i \, , \mathbf{b}_j^* \rangle}{\left\| \mathbf{b}_j^* \right\|_2^2} \right\rceil \mathbf{E}_{j,i}.$$

2.4.2 The LLL Algorithm

Informally, the idea of the Gauss-Lagrange algorithm presented in Algorithm 2.2 is to chain size reductions, and swapping $\mathbf{b}_1$ and $\mathbf{b}_2$ whenever $\|\mathbf{b}_2\|_2$ is smaller than $\|\mathbf{b}_1\|_2$. This means we make "progress" in the reduction of the original basis. It is natural to try and extend this approach to higher dimensions, but the situation is more problematic. We need a quantitative criterion to measure such progress in order to know when to swap vectors. We also need this progress to allow a constant gain factor over the Hadamard inequality from Corollary 2.1 at each step if we hope to have only a polynomial number of size reductions. These thoughts lead Lenstra, Lenstra and Lovàsz to propose the LLL algorithm [LJL82] in 1982. In this course, we only present the algorithm. A detailed analysis giving the polynomial complexity is given in appendix. We start by giving the notion of an LLL-reduced basis.

Definition 2.12 (LLL-Reduced Basis)

Let k, d be two positive integers, and $(\mathbf{b}_i)_{i\in[\![1,d]\!]}$ be a linearly independent family of $\mathbb{R}^k$. We let $(\mathbf{b}_i^*)_{i\in[\![1,d]\!]} = \mathsf{GSO}((\mathbf{b}_i)_{i\in[\![1,d]\!]})$. The basis $(\mathbf{b}_i)_{i\in[\![1,d]\!]}$ is called *LLL-reduced* if

- $(\mathbf{b}_i)_{i\in[\![1,d]\!]}$ is size-reduced;
- For all i in $[\![1, d-1]\!]$, we have

$$\frac{3}{4}\|\mathbf{b}_i^*\|_2^2 \leq \left\| \frac{\langle \mathbf{b}_{i+1}, \mathbf{b}_i^* \rangle}{\|\mathbf{b}_i^*\|_2^2} \mathbf{b}_i^* + \mathbf{b}_{i+1}^* \right\|_2^2 \tag{2.1}$$

Equation (2.1) is known as the *Lovàsz Condition*. The motivation behing LLL-reduced bases is summarized in the following property: the Gram-Schmidt do not decrease too fast, and the first vector of the basis cannot be too long.

Lemma 2.15

Let k, d be two positive integers, and $(\mathbf{b}_i)_{i\in[\![1,d]\!]}$ be a linearly independent family of $\mathbb{R}^k$. Assume that the basis $(\mathbf{b}_i)_{i\in[\![1,d]\!]}$ is LLL-reduced. It holds that

- For all $i \in [\![1, d-1]\!]$, $\|\mathbf{b}_i^*\|_2 \leq \sqrt{2}\|\mathbf{b}_{i+1}^*\|_2$;
- $\|\mathbf{b}_1\|_2 \leq 2^{(d-1)/2}\lambda_1(\mathcal{L}((\mathbf{b}_i)_{i\in[\![1,d]\!]}))$.

Proof (Lemma 2.15). The first property comes from Equation (2.1) and the size-reduced criterion. Indeed, we have

$$\begin{aligned}\frac{3}{4}\|\mathbf{b}_i^*\|_2^2 &\leq \|\mu_{i+1,i}\mathbf{b}_i^* + \mathbf{b}_{i+1}^*\|_2^2 \\ &= |\mu_{i+1,i}|^2\|\mathbf{b}_i^*\|_2^2 + \|\mathbf{b}_{i+1}^*\|_2^2 \\ &\leq \frac{1}{4}\|\mathbf{b}_i^*\|_2^2 + \|\mathbf{b}_{i+1}^*\|_2^2,\end{aligned}$$

which gives the first statement. Then, by recursively using the first property, we get $\|\mathbf{b}_1\|_2 = \|\mathbf{b}_1^*\|_2 \leq 2^{(i-1)/2}\|\mathbf{b}_i^*\|_2$ for all i in $[\![1, d]\!]$. A fortiori, we get

$$\begin{aligned}\|\mathbf{b}_1\|_2 &\leq \min_{i\in[\![1,d]\!]} 2^{(i-1)/2}\|\mathbf{b}_i^*\|_2 \\ &\leq 2^{(d-1)/2} \min_{i\in[\![1,d]\!]} \|\mathbf{b}_i^*\|_2 \\ &\leq 2^{(d-1)/2}\lambda_1(\mathcal{L}((\mathbf{b}_i)_{i\in[\![1,d]\!]})),\end{aligned}$$

where the last inequality stems from Lemma 2.12.

If the Lovàsz condition is not satisfied, we can certainly make progress in the reduction by swapping $\mathbf{b}_i$ and $\mathbf{b}_{i+1}$. This leads to the following formulation for the LLL algorithm, which we

now present in Algorithm 2.4.

Algorithm 2.4: LLL

Input: Lattice basis $(\mathbf{b}_i)_{i\in[\![1,d]\!]}$

1. $(\mathbf{b}_i^*)_{i\in[\![1,d]\!]} = \mathsf{GSO}((\mathbf{b}_i)_{i\in[\![1,d]\!]})$ ▷ Algorithm 2.1
2. $(\mathbf{b}_i)_{i\in[\![1,d]\!]} = \mathsf{SizeReduce}((\mathbf{b}_i)_{i\in[\![1,d]\!]})$ ▷ Algorithm 2.3
3. **if** there exists $i \in [\![1, d-1]\!]$ such that $\frac{3}{4}\left\|\mathbf{b}_i^*\right\|_2^2 \leq \left\|\frac{\langle \mathbf{b}_{i+1}, \mathbf{b}_i^*\rangle}{\left\|\mathbf{b}_i^*\right\|_2^2}\mathbf{b}_i^* + \mathbf{b}_{i+1}^*\right\|_2^2$ **then**
   (a) $(\mathbf{b}_i, \mathbf{b}_{i+1}) \leftarrow (\mathbf{b}_{i+1}, \mathbf{b}_i)$
   (b) Go to step 1.

Output: LLL-reduced basis $(\mathbf{b}_i)_{i\in[\![1,d]\!]}$

Theorem 2.5 (LLL Analysis)

The LLL algorithm from Algorithm 2.4 terminates in a polynomial number of steps and works with numbers whose bit-size is polynomial in the size of the input basis.

Nguyen and Stehlé proposed in 2009 [NS09] a variant of LLL which has a complexity of

$$O(d^{4+\varepsilon}\log\|\mathbf{B}\|_{\max}(d+\log\|\mathbf{B}\|_{\max})),$$

where $\|\mathbf{B}\|_{\max} = \max_{i\in[\![1,k]\!],j\in[\![1,d]\!]}\left|b_{i,j}\right|$ is the max-norm of $\mathbf{B}$. Regardless, it is clear that if the algorithm terminates, the output basis is LLL-reduced. If we do not concern ourselves with the size of the numbers we work with, we can show that we perform a polynomial number of swaps with the same argument as that of Gauss-Lagrange: the product of the $\|\mathbf{b}_i\|_2^2$ decreases with each swap by a constant factor $(3/4)$, and we cannot go below the volume of the lattice.

Bibliography

[Ban93] W. Banaszczyk. New Bounds in Some Transference Theorems in the Geometry of Numbers. *Math. Ann.*, 1993.

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for Hard Lattices and New Cryptographic Constructions. In *STOC*, 2008.

[LJL82] A. K. Lenstra, H. W. Lenstra Jr., and L. Lovász. Factoring Polynomials with Rational Coefficients. *Math. Ann.*, 1982.

[NS09] P. Q. Nguyen and D. Stehlé. An LLL algorithm with quadratic complexity. *SIAM J. Comput.*, 2009.

3

Hard Problems on Lattices

The reason why lattices are so attractive from an algorithmic standpoint is because they offer a variety of mathematical problems that are computationally hard to solve. This makes it a very interesting research area in cryptology on three levels: (1) cryptanalysis which aims at finding better algorithms to solve such problems efficiently, (2) cryptography which aims at harnessing the hardness of such problems to construct secure primitives, and (3) mathematics and complexity theory which aims at theoretically proving a hierarchy among these problems and prove their hardness mathematically. In this chapter, we present the mathematical problems that are at the heart of lattice-based cryptography. We also take a sneak peek at their proven hardness. It is difficult to be exhaustive as there are many variants of these problems, and brand new problems altogether. We thus only give what we deem important to have a good grasp and comprehension of the stakes in lattice-based cryptography foundations.

Contents

3.1 Complexity Classes

Before diving into defining the lattice problems and their hardness, we need to make a quick reminder of complexity theory and complexity classes. We have defined in Section 1.2.1 what we mean by a polynomial-time algorithm. This concept is rooted in defining the complexity classes that we consider in cryptography.

First of all, mathematical problems are mostly divided into two families, namely *search problems* and *decision problems*. Search problems or computational problems essentially ask to compute a concrete solution of an instance of the problem, while decision problems ask to decide between two or more choices.

Example 3.1 (Diffie Hellman)

We take the example of the Diffie Hellman problem. We can define a search problem as follows.

Definition 3.1 (Computational Diffie Hellman)

Given a cyclic group $\mathbb{G}$ of order p and generator g, the Computational Diffie Hellman problem $\text{CDH}_{\mathbb{G},g,p}$ is as follows. Given (g^x, g^y) for $x, y \leftarrow\!\!\!\!\rightarrow U(\mathbb{Z}_p)$, compute g^{xy}.

Generally, for a problem which has both a search and a decision variant, the search variant is usually at least as hard as the decision variant. It is the case for $\text{CDH}_{\mathbb{G},g,p}$ and $\text{DDH}_{\mathbb{G},g,p}$. To prove that, we give a reduction (Definition 1.22) from $\text{DDH}_{\mathbb{G},g,p}$ to $\text{CDH}_{\mathbb{G},g,p}$. Assume we have an oracle $\mathcal{O}$ that solves $\text{CDH}_{\mathbb{G},g,p}$. Given an instance (g^x, g^y, g^z) of $\text{DDH}_{\mathbb{G},g,p}$, we call $\mathcal{O}(g^x, g^y)$ and get g^{xy}. We can then compare g^{xy} to g^z. If they match it means that $z = xy$ and otherwise z is chosen uniformly.

We now give the main complexity classes for decision problems. They are used to categorize the different decision problems, but without necessarily proving a hierarchy among each class.

Definition 3.2 (Complexity Classes)

Decision problems may fall in one or several of the following classes.

- Class P. This class is composed of all the decision problems that can be solved by a polynomial-time algorithm. It is also called the *polynomial class*.
- Class NP. This class is composed of all the decision problems that have a polynomial-time verifier, meaning that given a candidate solution, it is possible to verify in polynomial-time that it is indeed a solution. It is also called the class of *non-deterministic polynomial-time* problems.
- Class coNP. This class is the complementary class of NP, meaning it is composed of all the decision problems for which a proof verifiable in polynomial-time can prove that a candidate solution is not a valid solution.
- Class NP-hard. This class is composed of all the decision problems that are at least as hard as all NP problems.
- Class NP-complete. This class is composed of all the decision problems that are both in NP and NP-hard.

To show that a problem P is in NP-hard, we prove a reduction from a known problem in NP-hard to P. For example, we can consider the problems SAT, 3-SAT, Knapsack, Traveling Salesman, etc. The first NP-hard problem is the SAT problem. The theorem of Cook shows that SAT is the hardest problem of NP. Note that as a result SAT is in NP-complete. We naturally have the following hierarchy of classes.

$$\mathsf{P} \subseteq \mathsf{NP} \qquad \mathsf{NP\text{-}complete} \subseteq \mathsf{NP} \qquad \mathsf{NP\text{-}complete} \subseteq \mathsf{NP\text{-}hard}.$$

Since most of the problems we consider in cryptography are in NP, if the converse inclusion $\mathsf{NP} \subseteq \mathsf{P}$ (meaning $\mathsf{P} = \mathsf{NP}$), it would mean that every cryptographic construction could be broken in polynomial-time. This would break the hierarchy of the complexity classes. Similarly, if there exists a problem in $\mathsf{coNP} \cap \mathsf{NP\text{-}hard}$, it would entail $\mathsf{NP} \subseteq \mathsf{coNP}$ which would also break the polynomial hierarchy.

3.2 Shortest and Closest Vector Problems

We now define the two main problems known as the *Shortest Vector Problem* and *Closest Vector Problem*. We also define their variants which are the ones we consider in lattice-based cryptography.

3.2.1 Shortest Vector Problem and Variants

In Chapter 2, we presented lattice reduction techniques (Section 2.4) which aim at reducing the size of the given lattice basis. These algorithm are actually used in cryptanalysis in order to solve the shortest vector problem.

Exact Shortest Vector Problem

The shortest vector problem asks to find the shortest non-zero vector of a given lattice. Recalling the definitions introduced in Section 2.2, it means finding a vector $\mathbf{x}$ such that $\|\mathbf{x}\|_p = \lambda_1^p(\mathcal{L})$. It also offers a decision problem which we here state.

Definition 3.3 (Shortest Vector Problem)

Let k, d be two positive integers, and p in $\mathbb{N}^* \cup \{\infty\}$.

- Search Variant: The *Shortest Vector Problem* $\text{SVP}_{k,d}^p$ asks to find $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v}\|_p = \lambda_1^p(\mathcal{L})$ given the lattice $\mathcal{L}$ of dimension k and rank d.
- Decision Variant: The *Decision Shortest Vector Problem* $\text{dSVP}_{k,d}^p$ asks to decide whether $\lambda_1^p(\mathcal{L}) \leq r$ (YES) or not, given the lattice $\mathcal{L}$ of dimension k and rank d, and a real $r \in \mathbb{R}$.

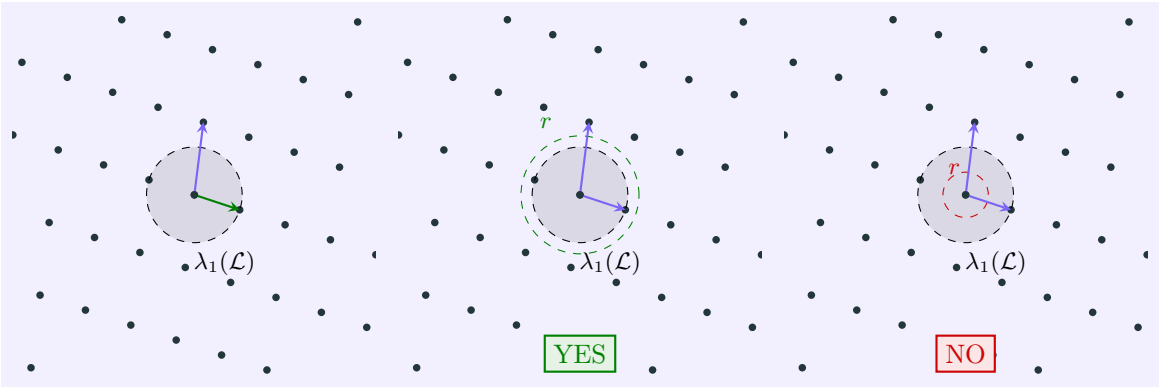


Figure 3.1: Search and decision versions of the *Shortest Vector Problem*. The left picture depicts SVP and a solution vector in green. The middle picture is a YES instance of dSVP, while the right picture is a NO instance.

Most often the lattice is full-rank, i.e., $k = d$, and the dimension is clear from the context, so we omit the subscripts k and d and simply write SVP^p and dSVP^p. Additionally, when $p = 2$, we omit the superscript as well. Note that an instance of $\text{SVP}_{k,d}^p$ is a lattice. To be used algorithmically, we must describe it by a basis $\mathbf{B} \in \mathbb{R}^{k \times d}$. However, using bases with real coefficients may cause problems to represent it in computers. As a result, we may limit ourselves to rational bases $\mathbf{B} \in \mathbb{Q}^{k \times d}$ or even integer bases.

Shortest Independent Vectors Problem

The SVP problem only focuses on the first minimum of a lattice. We could however generalize the problem and ask to find the shortest linearly independent lattice vectors. Although we could define the problem with respect to the i-th minimum for any i in $[\![1, d]\!]$, the formulation we use in cryptography is only for $i = d$.

Definition 3.4 (Shortest Independent Vectors Problem)

Let k, d be two positive integers, and p in $\mathbb{N}^* \cup \{\infty\}$. The *Shortest Independent Vectors Problem* $\text{SIVP}_{k,d}^p$ asks to find d linearly independent vectors $\mathbf{v}_1, \ldots, \mathbf{v}_d$ in $\mathcal{L}$ such that $\max_{i \in [\![1,d]\!]} \|\mathbf{v}_i\|_p = \lambda_d^p(\mathcal{L})$, given the lattice $\mathcal{L}$ of dimension k and rank d.

Approximate Variants of SVP

These two problems are a little too rigid, and are actually proven hard. It makes them impractical in large dimensions. In cryptography, we consider approximate versions of the problems which relax a little bit the problem to be practical and hard at the same time. Instead of finding the

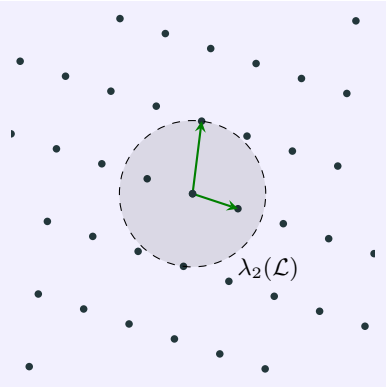


Figure 3.2: The *Shortest Independent Vectors Problem*. It depicts SIVP and a solution in green.

shortest vector, we ask for a vector whose norm is a small factor of the first minimum, for factor $\gamma \geq 1$ called the *approximation factor*.

Definition 3.5 (Approximate Shortest Vector Problem)

Let k, d be two positive integers, and p in $\mathbb{N}^* \cup \{\infty\}$. Let $\gamma \geq 1$ be a real approximation factor.

- Search Variant: The *Approximate Shortest Vector Problem* $\mathrm{SVP}^p_{k,d,\gamma}$ asks to find $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v}\|_p \leq \gamma \lambda^p_1(\mathcal{L})$ given the lattice $\mathcal{L}$ of dimension k and rank d.
- Decision Variant: The *Gap Shortest Vector Problem* $\mathrm{GapSVP}^p_{k,d,\gamma}$ asks to decide whether $\lambda^p_1(\mathcal{L}) \leq r$ (YES) or if $\lambda^p_1(\mathcal{L}) \geq \gamma r$ (NO), given the lattice $\mathcal{L}$ of dimension k and rank d, and a real $r \in \mathbb{R}$.

Notice that there is a difference in the decision variant. There is a gap between r and γr in which we accept either answer. So if the lattice $\mathcal{L}$ verifies $\lambda^p_1(\mathcal{L}) \in (r, \gamma r)$, the problem can be answered by either YES or NO. This never happens in practice. We always see GapSVP^p_γ promise problem, which means we are always in one of the two situations but not in the gap. When $\gamma = 1$, this coincides with SVP and dSVP. But we can see that the larger γ, the easier it gets as there are more solution vectors.

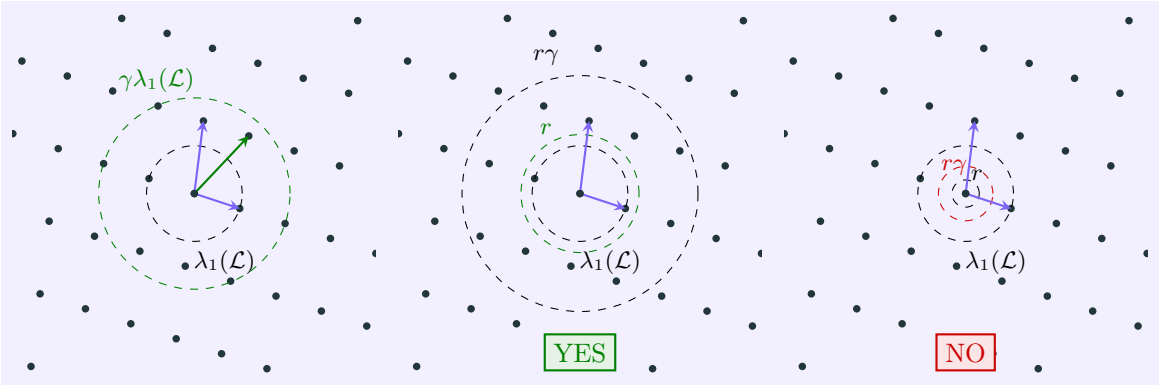


Figure 3.3: Search and decision versions of the *Approximate Shortest Vector Problem*. The left picture depicts SVP_γ for $\gamma = 2$ and a solution vector in green. The middle picture is a YES instance of GapSVP_γ, while the right picture is a NO instance.

We can also define an approximate version of the Shortest Independent Vectors Problem.

Definition 3.6 (Approximate Shortest Independent Vectors Problem)

Let k, d be two positive integers, and p in $\mathbb{N}^* \cup \{\infty\}$. Let $\gamma \geq 1$ be a real approximation factor. The *Approximate Shortest Independent Vectors Problem* $\mathrm{SIVP}^p_{k,d,\gamma}$ asks to find d linearly independent vectors $\mathbf{v}_1, \ldots, \mathbf{v}_d$ in $\mathcal{L}$ such that $\max_{i \in [\![1,d]\!]} \|\mathbf{v}_i\|_p \leq \gamma \lambda^p_d(\mathcal{L})$, given the

lattice $\mathcal{L}$ of dimension k and rank d.

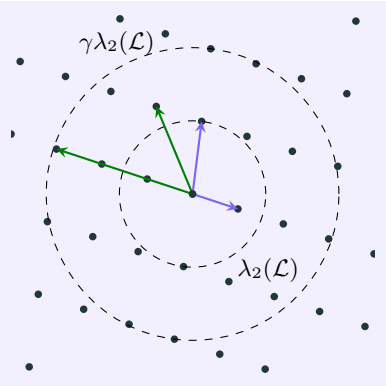


Figure 3.4: The *Approximate Shortest Independent Vectors Problem*. It depicts SIVP_γ for $\gamma = 2$ and a solution in green.

The solution highlighted in green is much bigger. In particular, this solution does not form a basis of the lattice. We can verify it for example by seeing that the fundamental parallelepiped it describes has a much bigger volume. Regardless, the SIVP and SIVP_γ problems do not ask for a basis of the lattice but for linearly independent lattice vectors, which is the case here. Knowing a set of linearly independent lattice vectors can be used to derive a basis from it that is smaller (or of the same length) than these vectors. But the difference is fundamental.

3.2.2 Closest Vector Problem and Variants

The Shortest Vector Problem essentially asks to find lattice points that are close to the origin. Although this is already a daunting task in high dimensions and given long and highly unorthogonal bases, another scenario often arises in lattice-based cryptography. In particular, we often have a vector in the ambient space and need to decode it to the closest lattice point. This constitutes another major lattice problem which can be used in cryptography.

Exact Closest Vector Problem

As described, the closest vector problem asks to find a lattice vector that is closest to a target in the ambient space. It also comes with a decision variant. For that, we recall the definition of a vector to a set in a normed space with respect to the ℓ_p norm. For a positive integer k, a set $S \subseteq \mathbb{R}^k$ and a target $\mathbf{t} \in \mathbb{R}^k$, we define the distance between $\mathbf{t}$ and S with respect the $\|\cdot\|_p$ for $p \in \mathbb{N}^* \cup \{\infty\}$ as

$$\text{dist}_p(\mathbf{t}, S) = \min_{\mathbf{s}\in S} \|\mathbf{t} - \mathbf{s}\|_p .$$

Definition 3.7 (Closest Vector Problem)

Let k, d be two positive integers, and p in $\mathbb{N}^* \cup \{\infty\}$.

- Search Variant: The *Closest Vector Problem* $\text{CVP}^p_{k,d}$ asks to find $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v} - \mathbf{t}\|_p = \text{dist}_p(\mathbf{t}, \mathcal{L})$ given the lattice $\mathcal{L}$ of dimension k and rank d, and a target $\mathbf{t} \in \mathbb{R}^k$.
- Decision Variant: The *Decision Closest Vector Problem* $\text{dCVP}^p_{k,d}$ asks to decide whether $\text{dist}_p(\mathbf{t}, \mathcal{L}) \leq r$ (YES) or not, given the lattice $\mathcal{L}$ of dimension k and rank d, a target $\mathbf{t} \in \mathbb{R}^k$, and a real $r \in \mathbb{R}$.

In situations where we need such decoding, we expect to uniquely recover the lattice point that was used during encoding. The encoding would start from a lattice vector $\mathbf{x} \in \mathcal{L}$, choose a small shift $\mathbf{e} \in \mathbb{R}^k$ and output the encoded lattice point as the target $\mathbf{t} = \mathbf{x} + \mathbf{e}$. When, decoding, we expect to recover $\mathbf{x}$. However, consider the pathological edge case where $\mathbf{e} = \frac{\lambda_1^p(\mathcal{L})}{2}\mathbf{x}^*$ where $\mathbf{x}^*$ is a shortest non-zero vector of $\mathcal{L}$. Then, geometrically, $\mathbf{t}$ would be the middle of the ℓ^p segment between $\mathbf{x}$ and $\mathbf{x} + \mathbf{x}^*$. The decoding would therefore decode either as $\mathbf{x}$ or $\mathbf{x} + \mathbf{x}^*$ with the same

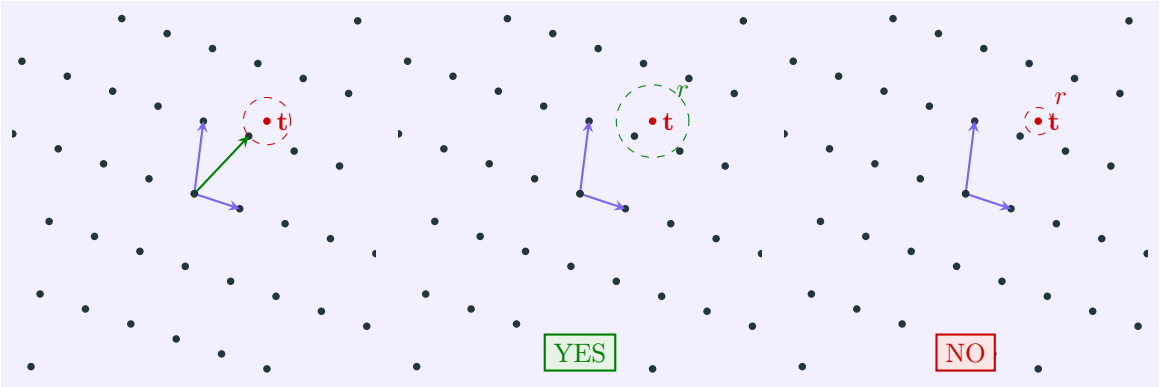


Figure 3.5: Search and decision versions of the *Closest Vector Problem*. The left picture depicts CVP and a solution vector in green. The middle picture is a YES instance of dCVP, while the right picture is a NO instance.

probability. To avoid such edge cases, we define a more practical variant of the closest vector problem called *Bounded Distance Decoding* problem. It is defined exactly as CVP but with the promise that the lattice point closest to the target is within a given distance δ. In particular, if $\delta < \lambda_1^p(\mathcal{L})/2$, there is only one lattice point that the target can be decoded to.

Definition 3.8 (Bounded Distance Decoding)

Let k, d be two positive integers, and p in $\mathbb{N}^* \cup \{\infty\}$. The *Bounded Distance Decoding* problem $\mathrm{BDD}^p_{k,d}$ asks to find $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v} - \mathbf{t}\|_p = \mathrm{dist}_p(\mathbf{t}, \mathcal{L})$ given the lattice $\mathcal{L}$ of dimension k and rank d, a target $\mathbf{t} \in \mathbb{R}^k$, and a bound $\delta > 0$ such that $\mathrm{dist}_p(\mathbf{t}, \mathcal{L}) \leq \delta$.

Approximate Variants of CVP

We can once again relax the closest vector problem and its variants by accepting solutions within an approximation factor $\gamma \geq 1$. The decision variant also gives rise to a promise gap problem, similarly to GapSVP.

Definition 3.9 (Approximate Closest Vector Problem)

Let k, d be two positive integers, and p in $\mathbb{N}^* \cup \{\infty\}$. Let $\gamma \geq 1$ be a real approximation factor.

- Search Variant: The *Approximate Closest Vector Problem* $\mathrm{CVP}^p_{k,d,\gamma}$ asks to find $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v} - \mathbf{t}\|_p \leq \gamma \mathrm{dist}_p(\mathbf{t}, \mathcal{L})$ given the lattice $\mathcal{L}$ of dimension k and rank d, and a target $\mathbf{t} \in \mathbb{R}^k$.
- Decision Variant: The *Gap Closest Vector Problem* $\mathrm{GapCVP}^p_{k,d,\gamma}$ asks to decide whether $\mathrm{dist}_p(, \mathbf{t}, \mathcal{L}) \leq r$ (YES) or if $\mathrm{dist}_p(\mathbf{t}, \mathcal{L}) \geq \gamma r$ (NO), given the lattice $\mathcal{L}$ of dimension k and rank d, a target $\mathbf{t} \in \mathbb{R}^k$, and a real $r \in \mathbb{R}$.

3.3 Hardness of SVP and CVP

Now that we have presented the main lattice problems, we need to assess if they are a good fit for cryptography. In particular, they need to be hard for a notion of hardness that is either proven or heuristic. In this course, we present the basic results and a reductions that start classifying the different problems. We first prove that dCVP is in NP-complete, that CVP and dCVP are equivalent, and that there exists a reduction from GapSVP_γ to GapCVP_γ. We then give the known hardness results for GapCVP_γ.

3.3.1 NP-Completeness of dCVP

As explained in Section 3.1, proving that a problem is in NP-complete means proving is in NP and that it is NP-hard. For the latter, we start from a problem that is known to be in NP-hard and

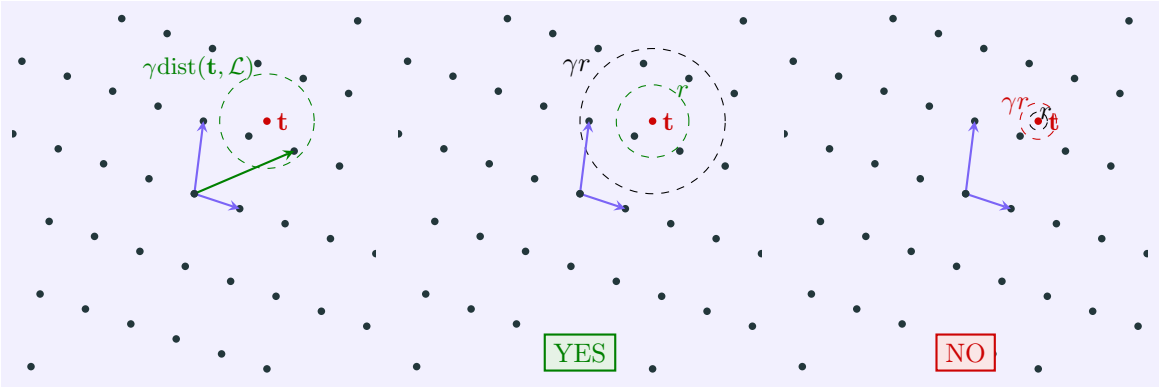


Figure 3.6: Search and decision versions of the *Approximate Closest Vector Problem*. The left picture depicts CVP_γ for $\gamma = 2$ and a solution vector in green. The middle picture is a YES instance of GapCVP_γ, while the right picture is a NO instance.

prove a reduction from that problem to dCVP. We then have to prove that dCVP is NP. To do so, we use the subset sum problem, which we now recall.

Definition 3.10 (Subset Sum Problem)

Let d be a positive integer. The Subset Sum Problem SSP_d is defined as follows. Given $d+1$ integers $a_1, \dots, a_d, S$, decide if there exists $A \subseteq \{1, \dots, d\}$ such that $\sum_{i \in A} a_i = S$.

We have the following theorem, proven in e.g. [CLRS09], states that the subset sum problem is NP-complete.

Theorem 3.1 (NP-Completeness of SSP)

The Subset Sum Problem SSP is in NP-complete.

We can now state our theorem of interest.

Theorem 3.2 (NP-Completeness of dCVP)

The decisional Closest Vector Problem dCVP is in NP-complete.

Proof (Theorem 3.2). Let us first prove that dCVP is in NP. Let $(\mathcal{L}, \mathbf{t}, r)$ be an instance of $\text{dCVP}^p_{k,d}$ such that $\text{dist}_p(\mathbf{t}, \mathcal{L}) \leq r$. Let $\mathbf{x}$ be in $\mathcal{L}$ such that $\|\mathbf{x} - \mathbf{t}\|_p \leq r$. Then $\mathbf{x}$ is the desired witness. Indeed, given this candidate witness $\mathbf{x}$, we can verify that $\mathbf{x}$ belongs to $\mathcal{L}$ and that $\|\mathbf{x} - \mathbf{t}\|_p \leq r$ in time polynomial in k, d. The latter norm condition implies that $\text{dist}_p(\mathbf{t}, \mathcal{L}) \leq r$. Hence $\text{dCVP}^p_{k,d} \in \text{NP}$.

Now we provide a reduction from SSP_d to $\text{dCVP}^p_{d+1,d}$. For that, we assume that we have access to an oracle $\mathcal{O}$ that solves $\text{dCVP}^p_{d+1,d}$. Let $(a_1, \dots, a_d, S)$ be an instance of SSP_d. We construct a basis $\mathbf{B} \in \mathbb{Z}^{d+1 \times d}$, a target $\mathbf{t} \in \mathbb{Z}^{d+1}$ and a real r as follows.

$$\mathbf{B} = \begin{bmatrix} a_1 & \dots & a_d \\ & 2\mathbf{I}_d & \end{bmatrix}, \text{ and } \mathbf{t} = \begin{bmatrix} S \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \text{ and } r = \sqrt[p]{d}.$$

We define $\mathcal{L} = \mathcal{L}(\mathbf{B})$. It is clearly a lattice of dimension $d+1$ and rank d. We then return $\mathcal{O}(\mathcal{L}, \mathbf{t}, r)$ as the solution to $\text{SSP}^p_{d+1,d}$.

Let us now analyze the reduction, that is the solution is correct, and the reduction is polynomial-time. Constructing the dCVP instance and sending it to the oracle is clearly polynomial-time. Now assume that the instance of SSP is a YES instance. As such, there exists $A \subseteq [\![1, d]\!]$ such that $\sum_{i \in A} a_i = S$. We then denote by $\mathbf{x} = [\mathbf{1}_A(i)]_{i \in [\![1,d]\!]} \in \{0,1\}^d$,

where $\mathbf{1}_A$ is the indicator function of A. It then follows that $\mathbf{y} = \mathbf{B}\mathbf{x} = [S, y_2, \ldots, y_{d+1}]^T$, where $y_i \in \{0, 2\}$ for all $i \in [\![2, d+1]\!]$. Hence, $\mathbf{y} - \mathbf{t} = [0, \pm 1, \ldots, \pm 1]^T$ which has ℓ_p-norm $\sqrt[p]{d} = r$. So we have found a lattice point $\mathbf{y}$ which is within ℓ_p-distance r of $\mathbf{t}$. As such, $\mathrm{dist}_p(\mathbf{t}, \mathcal{L}) \leq r$. The answer of $\mathcal{O}$ is thus YES as expected. Conversely, assume that $\mathcal{O}$ returns YES. Then, there exists $\mathbf{y} \in \mathcal{L}$ such that $\|\mathbf{y} - \mathbf{t}\|_p \leq \sqrt[p]{d}$. By construction, the last d coordinates of $\mathbf{y}$ must be even, and in turn the last d coordinates of $\mathbf{y} - \mathbf{t}$ must be odd. This shows that $\|\mathbf{y} - \mathbf{t}\|_p \geq \sqrt[p]{d}$. We thus obtain that $\|\mathbf{y} - \mathbf{t}\|_p = r$, which can only happen if the first coordinate is 0, and the others are ± 1. We define $A = \{i \in [\![1, d]\!] : [\mathbf{y} - \mathbf{t}]_i = 1\}$. It thus means that for $\mathbf{x} = [\mathbf{1}_A(i)]_{i \in [\![1,d]\!]}$, $\mathbf{B}\mathbf{x} = \mathbf{y}$ and therefore that $\sum_{i \in A} a_i = S$. So the answer to SSP_d should indeed be YES.

3.3.2 Equivalence of CVP and dCVP

We now show that the computational and decisional versions of CVP are actually equivalent. To prove equivalence, we first show in Lemma 3.1 a reduction from dCVP to CVP, and then in Lemma 3.2 the opposite reduction from CVP to dCVP.

Lemma 3.1 (Decision-to-Search CVP)

Let k, d be two positive integers and p in $\mathbb{N}^* \cup \{\infty\}$. There is a probabilistic polynomial-time reduction from $\mathrm{dCVP}^p_{k,d}$ to $\mathrm{CVP}^p_{k,d}$.

Proof (Lemma 3.1). Assume that there exists a PPT adversary $\mathcal{A}$ which can solve $\mathrm{CVP}^p_{k,d}$ with advantage ε. We construct a PPT adversary $\mathcal{B}$ solving $\mathrm{dCVP}^p_{k,d}$ with advantage ε. The algorithm $\mathcal{B}$ works as follows. Given an instance $(\mathcal{L}, \mathbf{t}, r)$ of $\mathrm{dCVP}^p_{k,d}$, it sends $(\mathcal{L}, \mathbf{t})$ to $\mathcal{A}$ and eventually receives $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v} - \mathbf{t}\|_p = \mathrm{dist}_p(\mathbf{t}, \mathcal{L})$ or $\perp$. If it received $\perp$, $\mathcal{B}$ outputs $\perp$. It then computes $u = \|\mathbf{v} - \mathbf{t}\|_p$ and output YES if $u \leq r$ and NO otherwise.

The algorithm $\mathcal{B}$ clearly runs in polynomial time as $\mathcal{A}$ runs in polynomial-time. We now analyze the advantage of $\mathcal{B}$. First, by assumption, we have $\mathbb{P}[\mathcal{B}(\mathcal{L}, \mathbf{t}, r) = \perp] = \varepsilon$. The advantage of $\mathcal{B}$ is defined as

$$\mathrm{Adv}[\mathcal{B}] = \left|\mathbb{P}[\mathcal{B}(\mathcal{L}, \mathbf{t}, r) = \mathrm{YES} | \mathrm{dist}_p(\mathbf{t}, \mathcal{L}) \leq r] - \mathbb{P}[\mathcal{B}(\mathcal{L}, \mathbf{t}, r) = \mathrm{YES} | \mathrm{dist}_p(\mathbf{t}, \mathcal{L}) > r]\right|.$$

For the first probability, we have

$$\begin{aligned}
&\mathbb{P}[\mathcal{B}(\mathcal{L}, \mathbf{t}, r) = \mathrm{YES} | \mathrm{dist}_p(\mathbf{t}, \mathcal{L}) \leq r] \\
&\qquad = \mathbb{P}[\mathcal{A}(\mathcal{L}, \mathbf{t}) = \perp] \cdot \mathbb{P}[\mathcal{B}(\mathcal{L}, \mathbf{t}, r) = \mathrm{YES} | \mathrm{dist}_p(\mathbf{t}, \mathcal{L}) \leq r \wedge \mathcal{A}(\mathcal{L}, \mathbf{t}) = \perp] \\
&\qquad\quad + \mathbb{P}[\mathcal{A}(\mathcal{L}, \mathbf{t}) \neq \perp] \cdot \mathbb{P}[\mathcal{B}(\mathcal{L}, \mathbf{t}, r) = \mathrm{YES} | \mathrm{dist}_p(\mathbf{t}, \mathcal{L}) \leq r \wedge \mathcal{A}(\mathcal{L}, \mathbf{t}) \neq \perp] \\
&\qquad = \mathbb{P}[\mathcal{A}(\mathcal{L}, \mathbf{t}) \neq \perp] \cdot \mathbb{P}[\mathcal{B}(\mathcal{L}, \mathbf{t}, r) = \mathrm{YES} | \mathrm{dist}_p(\mathbf{t}, \mathcal{L}) \leq r \wedge \mathcal{A}(\mathcal{L}, \mathbf{t}) \neq \perp] \\
&\qquad = \varepsilon \mathbb{P}[\left\|\mathcal{A}(\mathcal{L}, \mathbf{t}) - \mathbf{t}\right\|_p \leq r | \mathrm{dist}_p(\mathbf{t}, \mathcal{L}) \leq r] \\
&\qquad = \varepsilon \mathbb{P}[\mathrm{dist}_p(\mathbf{t}, \mathcal{L}) \leq r | \mathrm{dist}_p(\mathbf{t}, \mathcal{L}) \leq r] \\
&\qquad = \varepsilon.
\end{aligned}$$

For the second, we have

$$\begin{aligned}
&\mathbb{P}[\mathcal{B}(\mathcal{L}, \mathbf{t}, r) = \mathrm{YES} | \mathrm{dist}_p(\mathbf{t}, \mathcal{L}) > r] \\
&\qquad = \mathbb{P}[\mathcal{A}(\mathcal{L}, \mathbf{t}) = \perp] \cdot \mathbb{P}[\mathcal{B}(\mathcal{L}, \mathbf{t}, r) = \mathrm{YES} | \mathrm{dist}_p(\mathbf{t}, \mathcal{L}) > r \wedge \mathcal{A}(\mathcal{L}, \mathbf{t}) = \perp] \\
&\qquad\quad + \mathbb{P}[\mathcal{A}(\mathcal{L}, \mathbf{t}) \neq \perp] \cdot \mathbb{P}[\mathcal{B}(\mathcal{L}, \mathbf{t}, r) = \mathrm{YES} | \mathrm{dist}_p(\mathbf{t}, \mathcal{L}) > r \wedge \mathcal{A}(\mathcal{L}, \mathbf{t}) \neq \perp] \\
&\qquad = \mathbb{P}[\mathcal{A}(\mathcal{L}, \mathbf{t}) \neq \perp] \cdot \mathbb{P}[\mathcal{B}(\mathcal{L}, \mathbf{t}, r) = \mathrm{YES} | \mathrm{dist}_p(\mathbf{t}, \mathcal{L}) > r \wedge \mathcal{A}(\mathcal{L}, \mathbf{t}) \neq \perp] \\
&\qquad = \varepsilon \mathbb{P}[\left\|\mathcal{A}(\mathcal{L}, \mathbf{t}) - \mathbf{t}\right\|_p \leq r | \mathrm{dist}_p(\mathbf{t}, \mathcal{L}) > r] \\
&\qquad = \varepsilon \mathbb{P}[\mathrm{dist}_p(\mathbf{t}, \mathcal{L}) \leq r | \mathrm{dist}_p(\mathbf{t}, \mathcal{L}) > r] \\
&\qquad = 0.
\end{aligned}$$

As a result, $\mathrm{Adv}[\mathcal{B}] = \varepsilon$.

We now show the converse reduction which is a little bit more subtle. We have to use the decision oracle in order to construct a sparse lattice while keeping track of how the closest vector and target are modified. Once this sparse lattice is constructed, we leverage its sparsity and use an efficient solver of CVP in sparse lattices.

Lemma 3.2 (Search-to-Decision CVP)

Let k, d be two positive integers. There is a probabilistic polynomial-time reduction from $\mathrm{CVP}^2_{k,d}$ to $\mathrm{dCVP}^2_{k,d}$ for integer lattices.

Proof (Lemma 3.2). Assume that there exists an oracle can solve $\mathrm{dCVP}^2_{k,d}$. We construct a PPT adversary $\mathcal{B}$ solving $\mathrm{CVP}^2_{k,d}$. The algorithm $\mathcal{B}$ works as follows. It is given an integer lattice $\mathcal{L} \subset \mathbb{Z}^k$ and $\mathbf{t} \in \mathbb{R}^k$. We let $\mathbf{B} = [\mathbf{b}_1|\dots|\mathbf{b}_d] \in \mathbb{Z}^{k\times d}$ be a basis of $\mathcal{L}$.

Estimating $\mathrm{dist}(\mathbf{t}, \mathcal{L})$. The first step is to find the distance r from $\mathbf{t}$ to the lattice $\mathcal{L}$. For that we proceed by binary search as follows. There exists an (unknown) $\mathbf{x}$ in $\mathbb{Z}^n$ verifying $\|\mathbf{Bx} - \mathbf{t}\|_2 = r$. Then $r^2 = \|\mathbf{Bx} - \mathbf{t}\|_2^2 \in \mathbb{Z}$. We denote $R = \sum_{i\in[\![1,n]\!]} \|\mathbf{b}_i\|_2 /2$. First, notice that $0 < r \leq R$. Indeed, if $r = 0$ it means $\mathbf{t} \in \mathcal{L}(\mathbf{B})$ which can be verified efficiently. On the other hand, since $\mathbf{Bx}$ is the lattice point closest to $\mathbf{t}$, it means that $\mathbf{Bx} - \mathbf{t}$ is in $\mathcal{P}_{\pm}(\mathbf{B})$ and can be written as $\mathbf{Bx} - \mathbf{t} = \sum_{i\in[\![1,n]\!]} \alpha_i \mathbf{b}_i$ with $\alpha_i \in [-1/2, 1/2)$. By the triangle inequality, $\|\mathbf{Bx} - \mathbf{t}\|_2 \leq \sum_{i\in[\![1,n]\!]} \|\mathbf{b}_i\|_2 /2 = R$.

Then, by performing a binary search on $\{1, \dots, R^2\}$ we can find r^2 and thus r in $p \log_2 R$ steps. At each step, the search set is $\{a, \dots, b\}$. Call the dCVP oracle on $(\mathbf{B}, \mathbf{t}, (b-a)/2)$. If the oracle answers YES, then update the search set to $\{a, \dots, \lfloor (b-a)/2 \rfloor\}$ and otherwise update it to $\{\lceil (b-a)/2 \rceil, \dots, b\}$.

Iterative procedure. We then consider the following iterative procedure.

Input: $(\mathbf{B}', \mathbf{t}')$ such that $\mathcal{L}(\mathbf{B}') \subseteq \mathcal{L}(\mathbf{B})$ and $\mathbf{t}' = \mathbf{t} + \mathbf{v}$ with $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ and $\mathrm{dist}(\mathbf{t}', \mathcal{L}(\mathbf{B}')) = r$

Output: $(\mathbf{B}'', \mathbf{t}'')$

1. $\mathbf{B}'' \leftarrow [2\mathbf{b}'_1|\mathbf{b}'_2|\dots|\mathbf{b}'_d]$.
2. $b \leftarrow \mathcal{O}^{\mathrm{dCVP}}(\mathbf{B}'', \mathbf{t}', r)$ $\quad$ [$b = 1$ if $\mathrm{dist}(\mathbf{t}', \mathcal{L}(\mathbf{B}'')) \leq r$ and 0 otherwise]
3. Output $(\mathbf{B}'', \mathbf{t}'' = \mathbf{t}' - (1-b)\mathbf{b}'_1)$.

The output $(\mathbf{B}'', \mathbf{t}'')$ verifies the three following invariants (1) $\mathcal{L}(\mathbf{B}'') \subseteq \mathcal{L}(\mathbf{B})$, (2) $\mathbf{t}'' = \mathbf{t} + \mathbf{v}$ for some $\mathbf{v}$ in $\mathcal{L}(\mathbf{B})$ and (3) $r = \mathrm{dist}(\mathbf{t}'', \mathcal{L}(\mathbf{B}''))$. Indeed, we have the following.

(1) We have $\mathbf{B}'' = \mathbf{B}' + [\mathbf{b}'_1|\mathbf{0}|\dots|\mathbf{0}]$ so each $\mathbf{b}''_i$ is in $\mathbb{Z} \cdot \mathbf{b}'_i \subset \mathcal{L}(\mathbf{B}') \subseteq \mathcal{L}(\mathbf{B})$. Therefore, $\mathcal{L}(\mathbf{B}'') \subseteq \mathcal{L}(\mathbf{B})$.

(2) If the return value is $\mathbf{t}'' = \mathbf{t}'$, then $\mathbf{t}'' = \mathbf{t} + \mathbf{v}$ with $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ by construction of the input. If $\mathbf{t}'' = \mathbf{t}' - \mathbf{b}'_1$ then $\mathbf{t}'' = \mathbf{t} + (\mathbf{v} - \mathbf{b}'_1)$ and $\mathbf{v} - \mathbf{b}'_1 \in \mathcal{L}(\mathbf{B})$ because $\mathcal{L}(\mathbf{B}') \subseteq \mathcal{L}(\mathbf{B})$.

(3) If $\mathbf{t}'' = \mathbf{t}'$, then $\mathrm{dist}(\mathbf{t}'', \mathcal{L}(\mathbf{B}'')) = \mathrm{dist}(\mathbf{t}', \mathcal{L}(\mathbf{B}'')) \leq r$ (returned by the oracle). Since $\mathcal{L}(\mathbf{B}'') \subseteq \mathcal{L}(\mathbf{B}')$, we have $\mathrm{dist}(\mathbf{v}, \mathcal{L}(\mathbf{B}')) \leq \mathrm{dist}(\mathbf{v}, \mathcal{L}(\mathbf{B}''))$ for all $\mathbf{v}$. Therefore, it holds $\mathrm{dist}(\mathbf{t}'', \mathcal{L}(\mathbf{B}'')) \geq \mathrm{dist}(\mathbf{t}'', \mathcal{L}(\mathbf{B}')) = \mathrm{dist}(\mathbf{t}', \mathcal{L}(\mathbf{B}')) = r$, and as a result $\mathrm{dist}(\mathbf{t}'', \mathcal{L}(\mathbf{B}'')) = r$.
For the case $\mathbf{t}'' = \mathbf{t}' - \mathbf{b}'_1$, we first notice that $\mathcal{L}(\mathbf{B}') = \mathcal{L}(\mathbf{B}'') \cup (\mathcal{L}(\mathbf{B}'') + \mathbf{b}'_1)$. $\mathcal{L}(\mathbf{B}'')$ represents the vectors which have an even coordinate along $\mathbf{b}'_1$ and $(\mathcal{L}(\mathbf{B}'') + \mathbf{b}'_1)$ those with an odd coordinate along $\mathbf{b}'_1$. We thus trivially have this set equality. If $\mathbf{t}'' = \mathbf{t}' - \mathbf{b}'_1$, since the oracle returns $b = 0$, we get that $\mathrm{dist}(\mathbf{t}', \mathcal{L}(\mathbf{B}'')) > r$. Since $\mathrm{dist}(t', \mathcal{L}(\mathbf{B}')) = r$, we necessarily have $\mathrm{dist}(\mathbf{t}', \mathcal{L}(\mathbf{B}'') + \mathbf{b}'_1) = r$, which yields $\mathrm{dist}(\mathbf{t}'', \mathcal{L}(\mathbf{B}'')) = r$.

$\mathcal{B}$ run this procedure $\kappa = d + \log_2 r$ times, and does the same for the $d-1$ vectors of the basis (defining $\mathbf{B}'' = [\mathbf{b}'_1|2\mathbf{b}'_2|\mathbf{b}'_3|\dots|\mathbf{b}'_d]$ and so on). After these $d\kappa$ runs of the procedure, we get $(\mathbf{B}^*, \mathbf{t}^*)$ with $\mathbf{B}^* = [2^\kappa \mathbf{b}_1|\dots|2^\kappa \mathbf{b}_d]$, $\mathbf{t}^* = \mathbf{t} + \mathbf{v}$ for some $\mathbf{v}$ in $\mathcal{L}(\mathbf{B})$, and $\mathrm{dist}(\mathbf{t}^*, \mathcal{L}(\mathbf{B}^*)) = r$.
The coordinates of every vector of $\mathcal{L}(\mathbf{B}^*)$ are multiples of 2^κ so the distance between two distinct vectors is at least 2^κ. Hence, the second closest vector to $\mathbf{t}^*$ is at a distance $r' \geq 2^\kappa - r = 2^d 2^{\log_2 r} - r = r(2^d - 1)$. Using Babai's algorithm, we are able to find in polynomial time the closest vector $\mathbf{x}^* \in \mathcal{L}(\mathbf{B}^*)$ to $\mathbf{t}^*$ within an approximation factor $\gamma = 2^{d/2} < 2^d - 1$. The only vector of $\mathcal{L}(\mathbf{B}^*)$ at distance at most $r\gamma$ of $\mathbf{t}^*$ is the closest vector $\mathbf{x}^*$. Yet, since

$\text{dist}(\mathbf{t}^*, \mathcal{L}(\mathbf{B}^*)) = r$, we have $\|\mathbf{x}^* - \mathbf{t}^*\|_2 = r$. Moreover, $\mathbf{t}^* = \mathbf{t} + \mathbf{v}$ for some $\mathbf{v} \in \mathcal{L}(\mathbf{B})$. We thus define $\mathbf{x} = \mathbf{x}^* - \mathbf{v}$. Since $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ and that $\mathbf{x}^* \in \mathcal{L}(\mathbf{B}^*) \subseteq \mathcal{L}(\mathbf{B})$, we indeed get $\mathbf{x} \in \mathcal{L}(\mathbf{B})$. By construction, $\|\mathbf{x} - \mathbf{t}\|_2 = \|\mathbf{x}^* - \mathbf{t}^*\|_2 = r$. So $\mathbf{x}$ is indeed a desired solution of the $\text{CVP}_{k,d}$ instance.

3.3.3 Reduction from Approx-GapSVP to Approx-GapCVP

We now look at the hierarchy between GapSVP_γ and GapCVP_γ. More precisely, we show that GapCVP_γ is at least as hard as GapSVP_γ. This seems rather intuitive as CVP essentially asks to the shortest lattice vector with origin $\mathbf{t}$, whereas SVP asks to find the shortest lattice vector with origin $\mathbf{0}$. Naturally, one would want to call the CVP oracle with target $\mathbf{0}$. But since $\mathbf{0}$ is a lattice point, it would always return $\mathbf{0}$, which does not yield a solution for SVP.

The idea is thus to make a *hole* in the lattice around the target of CVP so that even if the target is on the lattice, the CVP oracle will not return it as a solution. However, making such a *hole* does not result in a lattice. So instead, we need to remove a subset of lattice points to preserve a lattice structure.

Theorem 3.3 (GapSVP_γ **to** GapCVP_γ)

Let k, d be two positive integers, p in $\mathbb{N}^* \cup \{\infty\}$, and $\gamma \geq 1$. There is a probabilistic polynomial-time reduction from $\text{GapSVP}^p_{k,d,\gamma}$ to $\text{GapCVP}^p_{k,d,\gamma}$.

Proof (Theorem 3.3). Assume we have access to an oracle for $\text{GapCVP}^p_{k,d,\gamma}$. We construct the solver for $\text{GapSVP}^p_{k,d,\gamma}$ as follows. On input $(\mathcal{L}, r)$ an instance of $\text{GapSVP}^p_{k,d,\gamma}$, we denote by $\mathbf{B} = [\mathbf{b}_i]_{i \in [\![1,d]\!]} \in \mathbb{R}^{k \times d}$ a basis of $\mathcal{L}$. For i ranging from 1 to n, we define:

$$\mathbf{B}_i = [\mathbf{b}_1 | \dots | \mathbf{b}_{i-1} | 2\mathbf{b}_i | \mathbf{b}_{i+1} | \dots | \mathbf{b}_n],$$

and the instance i for $\text{GapCVP}^p_{k,d,\gamma}$ is $(\mathbf{B}_i, \mathbf{b}_i, r)$. The algorithm calls the oracle for each instance. If one of the instance yields a YES, then the algorithm returns YES as answer to the instance $(\mathcal{L}, r)$, otherwise it returns NO.

The algorithm makes d calls to the $\text{GapCVP}^p_{k,d,\gamma}$ oracle and is therefore polynomial-time if the oracle also respond in polynomial time. We now have to prove its correctness.

First, suppose that $(\mathcal{L}, r)$ is a NO instance, which means $\lambda^p_1(\mathcal{L}) > \gamma r$. Let i be an intermediate instance for $\text{GapCVP}^p_{k,d,\gamma}$. Let $\mathbf{x} = k_1\mathbf{b}_1 + \dots + 2k_i\mathbf{b}_i + \dots k_n\mathbf{b}_n$ be in $\mathcal{L}(\mathbf{B}_i)$. Then $\mathbf{x} - \mathbf{b}_i$ has an odd coefficient along $\mathbf{b}_i$ which means that $\mathbf{x} - \mathbf{b}_i \neq \mathbf{0}$. Yet, $\mathbf{x} - \mathbf{b}_i \in \mathcal{L}$. So $\|\mathbf{x} - \mathbf{b}_i\|_p \geq \lambda^p_1(\mathcal{L}) > \gamma r$. By minimizing over $\mathbf{x}$, we get that $\text{dist}_p(\mathbf{b}_i, \mathcal{L}(\mathbf{B}_i)) > \gamma r$ so i is a NO instance of $\text{GapCVP}^p_{k,d,\gamma}$.

Conversely, suppose $(\mathcal{L}, r)$ to be a YES instance, meaning $\lambda^p_1(\mathcal{L}) \leq r$. Define $\mathbf{x} = \sum_{i \in [\![1,n]\!]} k_i\mathbf{b}_i$ to be the shortest non-zero vector of $\mathcal{L}$. Then there is i such that k_i is odd (otherwise $\mathbf{x}/2$ is still in the lattice and is stricly shorter). Now define $\mathbf{u} = \frac{k_i+1}{2} 2\mathbf{b}_i + \sum_{j \neq i} k_j\mathbf{b}_j$. Then $\mathbf{u}$ is in $\mathcal{L}(\mathbf{B}_i)$ and $\mathbf{u} - \mathbf{b}_i = \mathbf{x}$ so $\text{dist}_p(\mathbf{b}_i, \mathcal{L}(\mathbf{B}_i)) \leq \|\mathbf{u} - \mathbf{b}_i\|_p = \|\mathbf{x}\|_p = \lambda^p_1(\mathcal{L}) \leq r$. So i is a YES instance of $\text{GapCVP}^p_{k,d,\gamma}$.

We can see that this reduction preserves the dimension k, the rank d and the approximation factor γ of the problems. It is therefore tight in terms of parameters, but it still requires d calls to the $\text{GapCVP}^p_{k,d,\gamma}$ oracle. Additionally, we notice that the reduction goes through for all $\gamma \geq 1$, which means that it yields a reduction from dSVP to dCVP. It turns out that the converse reduction is much more difficult to establish and the dimension and rank of the lattice change throughout the reduction.

3.3.4 Hardness of Approx-GapCVP

We have seen that GapCVP_γ is no easier than GapSVP_γ, and that dCVP is in NP-complete. But how about the hardness of GapCVP_γ for $\gamma > 1$? In 2003, Dinur, Kindler, Raz and Safra [DKRS03] extended the NP-completeness to all γ that are "almost-polynomial". They actually proved it was in NP-hard, but since GapCVP_γ is also in NP for all γ, it proved NP-completeness.

Theorem 3.4 (NP-Completeness of GapCVP_γ [DKRS03])

Let k, d be two positive integers. Let $\gamma \leq d^{O(1/\log\log d)}$. Then, $\mathrm{GapCVP}^2_{k,d,\gamma}$ is in NP-complete.

Later, in 2003, Aharonov and Regev [AR04] proved that it was in $\mathsf{NP} \cap \mathsf{coNP}$ for $\gamma = \sqrt{d}$.

Theorem 3.5 (coNP of GapCVP_γ [AR04])

Let k, d be two positive integers. Then, $\mathrm{GapCVP}^2_{k,d,\sqrt{d}}$ is in $\mathsf{NP} \cap \mathsf{coNP}$.

It is very unlikely that a problem in $\mathsf{NP} \cap \mathsf{coNP}$ would also be in NP-complete. If this happened, it would imply a collapse of the polynomial hierarchy. Other problems from cryptography are known to be in $\mathsf{NP} \cap \mathsf{coNP}$ like $\mathrm{GapSVP}_{k,d,d}$ or the factorization problem.

Theorem 3.6 (coNP of GapSVP_d)

Let k, d be two positive integers. Then, $\mathrm{GapSVP}^2_{k,d,d}$ is in $\mathsf{NP} \cap \mathsf{coNP}$.

Proof (Theorem 3.6). We recall that a decision problem is in coNP if only NO instances can be verified in polynomial time, i.e., for NO instances there exists a witness that can be verified efficiently, but for YES instances, no witness can be verified.

Let $(\mathcal{L}, r)$ be a NO instance for $\mathrm{GapSVP}^2_{k,d,d}$, meaning that $\lambda_1(\mathcal{L}) > dr$. From the Banaszczyk theorem (Theorem 2.4), we get that $\lambda_d(\mathcal{L}^*) < \frac{1}{r}$. Hence, the witness would be d linearly independent vectors of $\mathcal{L}^*$ of length less than $\frac{1}{r}$.

Given the witness, it is easy to verify that these vectors belong to $\mathcal{L}^*$, that they are linearly independent, and that they have norm less than $\frac{1}{r}$. This allows us to verify that $\lambda_d(\mathcal{L}^*) < \frac{1}{r}$. We then know that $(\mathcal{L}, r)$ is a NO instance because if not we would have $\lambda_1(\mathcal{L}) \leq r$ and thus $\lambda_1(\mathcal{L})\lambda_d(\mathcal{L}^*) < 1$, which would contradict the Banaszczyk theorem.

Since all these verifications can be done in $\mathsf{poly}(d)$, then $\mathrm{GapSVP}^2_{k,d,d}$ is in coNP. Since it is also in NP, it gives the result.

The main conjecture that is done in lattice-based cryptography is therefore that $\mathrm{GapSVP}_{k,d,\gamma}$ and $\mathrm{GapCVP}_{k,d,\gamma}$ and other common variants are hard to solve even for $\gamma = \mathsf{poly}(d)$.

Conjecture 3.1 (Lattice Cryptography Conjecture)

Let k, d be two positive integers. We assume that common lattice problems in dimension k and rank d are hard to solve within $\gamma = \mathsf{poly}(d)$ approximation factors.

3.3.5 Some Lattice Reduction Solvers

We have seen certain lattice reduction techniques in Section 2.4 that may be used to solve these lattice problems. There are many other lattice reduction algorithms, but we do not cover them in this course. Nevertheless, we now give the performance of the most well known to see their impact on the hardness of lattice problems, and whether they endanger Conjecture 3.1.

LLL

We have seen that the LLL algorithm (Algorithm 2.4) returns an LLL-reduced basis by Theorem 2.5. By Lemma 2.15, it thus holds that the first vector of said basis has ℓ_2-norm smaller than $\gamma\lambda_1(\mathcal{L})$ where $\gamma = 2^{(d-1)/2}$. This means that LLL can solve $\mathrm{SVP}^2_{k,d,\gamma}$ in polynomial time for $\gamma \geq 2^{(d-1)/2}$.

It turns out that it can be adapted to solve the exact versions SVP and CVP, but it then requires a $2^{O(d^2)}$ time complexity, and a polynomial memory complexity.

Kannan

In 1983, Kannan [Kan83] improves on this variant to solve exact lattice problems with a reduced complexity. It shows that it is possible to solve SVP and CVP with polynomial memory complexity and $2^{O(d \log d)}$ time complexity.

Ajtai, Kumar, Sivakumar

Later on, Ajtai, Kumar and Sivakumar [AKS01] proposed a probabilistic sieving algorithm allowing to reduce the time complexity at the expense of the memory complexity. In particular, their algorithm has a complexity of $2^{O(d)}$ in both time and memory.

Micciancio, Voulgaris

Then, in 2010, Micciancio and Voulgaris [MV10] proposed a deterministic CVP solver based on Voronoi cell computations. It has time complexity 2^{2d} and memory complexity 2^d which improves on [AKS01].

Bibliography

[AKS01] M. Ajtai, R. Kumar, and D. Sivakumar. A Sieve Algorithm for the Shortest Lattice Vector Problem. In *STOC*, 2001.

[AR04] D. Aharonov and O. Regev. Lattice Problems in NP cap coNP. In *FOCS*, 2004.

[CLRS09] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009.

[DKRS03] I. Dinur, G. Kindler, R. Raz, and S. Safra. Approximating CVP to Within Almost-Polynomial Factors is NP-Hard. *Comb.*, 2003.

[Kan83] R. Kannan. Improved Algorithms for Integer Programming and Related Lattice Problems. In *STOC*, 1983.

[MV10] D. Micciancio and P. Voulgaris. A Deterministic Single Exponential Time Algorithm for Most Lattice Problems Based on Voronoi Cell Computations. In *STOC*, 2010.

Part III

Foundations of Lattice-Based Cryptography

This part presents the necessary tools as well as the fundamental assumptions underlying lattice-based cryptography.

4

Gaussian Distributions over Lattices

Gaussian probability distributions have very interesting probabilistic and geometric properties, and they appear everywhere in mathematics. They offer many possibilities in order to randomize certain processes, and typically instances of problems. Unfortunately, Gaussian distributions are by nature continuous over $\mathbb{R}^d$ which seems incompatible with the discrete aspect of lattices. There is however a way to discretize Gaussian distributions onto discrete sets such as lattices. The idea is essentially to take a continuous Gaussian distribution and condition the samples to be lattice points, thus needing normalization by the Gaussian mass of the lattice itself.

Albeit perfectly valid in theory, for it to be relevant, one needs a way to efficiently sample from such distributions. However, Gaussian distributions are known to be concentrated around their center, which means that if one is able to sample from very narrow Gaussian distributions centered around $\mathbf{0}$ (or around a target $\mathbf{t}$ in the case of CVP), they would be able to solve SVP, CVP and possibly other variants. As it turns out, sampling a Gaussian on a lattice and the quality of the samples highly depends on the size of the basis of the lattice that is used. As a result, a good basis with short and highly orthogonal vectors allows efficient and *qualitative* sampling, while a bad basis will result in long samples.

In this chapter, we thus define in all generality the notion of discrete Gaussian distributions, especially over lattices, and their properties and peculiarities compared to continuous ones. We then give more details on the results on discrete Gaussian sampling and the quality of the samples based on the quality of the basis.

We recall that for a finite set S, we denote by $|S|$ its cardinality and by $U(S)$ the discrete uniform distribution over S. If S is not finite but with bounded volume in a metric space, $U(S)$ can still be defined as the continuous uniform distribution over S, with probability density function $(\mathrm{Vol}\, S)^{-1}\mathbf{1}_S(\cdot)$. The action of sampling $x \in S$ from a distribution P is denoted by $x \hookleftarrow P$, whereas the notation $x \sim P$ means that the random variable x is distributed according to P.

Contents

4.1 Definition of Discrete Gaussians

4.1.1 Continuous Multidimensional Gaussian Distributions

We now make reminders on multidimensional Gaussian distributions. For that, we first define the Gaussian function, parameterized by a Gaussian parameter $s > 0$ (sometimes called width or standard deviation) and a center $\mathbf{c}$.

Definition 4.1 (Gaussian Function)

Let d be a positive integer. For a positive real $s > 0$ and a vector $\mathbf{c} \in \mathbb{R}^d$, we define the *Gaussian function of center* $\mathbf{c}$ *and width* s by

$$\forall \mathbf{x} \in \mathbb{R}^d, \rho_{s,\mathbf{c}}(\mathbf{x}) = \exp\left(-\frac{\pi}{s^2}\|\mathbf{x}-\mathbf{c}\|_2^2\right).$$

By normalizing it, we can then define the continuous multidimensional Gaussian distribution by its probability density function $D_{s,\mathbf{c}}$.

Definition 4.2 (ontinuous Gaussian Distribution)

Let d be a positive integer. For a positive real $s > 0$ and a vector $\mathbf{c}$ in $\mathbb{R}^d$, we define the *continuous Gaussian distribution of center* $\mathbf{c}$ *and width* s as the probability distribution whose probability density function is defined by

$$\forall \mathbf{x} \in \mathbb{R}^d, D_{s,\mathbf{c}}(\mathbf{x}) = \frac{1}{s^d}\rho_{s,\mathbf{c}}(\mathbf{x}) = \frac{1}{s^d}\exp\left(-\frac{\pi}{s^2}\|\mathbf{x}-\mathbf{c}\|_2^2\right).$$

For clarity, when the center is $\mathbf{c} = \mathbf{0}$, we omit it from the subscripts.

4.1.2 Discrete Gaussian Distributions

We can now define discrete Gaussian distributions. Although we use them over lattices or lattice cosets, it is possible to define them over arbitrary countable sets. To remain as general as possible, this is the choice we make.

Definition 4.3 (Discrete Gaussian Distribution)

Let d be a positive integer, and $S \subset \mathbb{R}^d$ be a countable set. Let $s > 0$ be a positive real and $\mathbf{c}$ a vector in $\mathbb{R}^d$. The *discrete Gaussian distribution over* S *of center* $\mathbf{c}$ *and width* s is the probability distribution whose probability density function is defined by

$$\forall \mathbf{x} \in S, \mathcal{D}_{S,s,\mathbf{c}}(\mathbf{x}) = \frac{\rho_{s,\mathbf{c}}(\mathbf{x})}{\rho_{s,\mathbf{c}}(S)} = \frac{\exp\left(-\frac{\pi}{s^2}\|\mathbf{x}-\mathbf{c}\|_2^2\right)}{\sum_{\mathbf{y}\in S}\exp\left(-\frac{\pi}{s^2}\|\mathbf{y}-\mathbf{c}\|_2^2\right)}.$$

Since lattices are countable sets, we will only focus on discrete Gaussian distributions for which the support is a lattice $\mathcal{L}$. We note that for all vector $\mathbf{a} \in \mathbb{R}^d$ and lattice $\mathcal{L} \subset \mathbb{R}^d$, $\mathbf{a} + \mathcal{L}$ is also countable. We will thus sometimes consider distributions of the form $\mathcal{D}_{\mathbf{a}+\mathcal{L},s,\mathbf{c}}$. We depict in Figure 4.1 the density functions of two discrete Gaussian distributions, one over the lattice $\mathbb{Z}$, and the second over a lattice of dimension 2.

4.2 Paramètre de Lissage

Although, the density of discrete Gaussian distributions rather look like that of continuous Gaussian discretized on a lattice, they should not be treated as continuous Gaussian distributions without care. In particular, a very important result about continuous Gaussians is that the sum of two independent Gaussian distributions with standard deviation s_1 and s_2 respectively is a Gaussian with width $\sqrt{s_1^2 + s_2^2}$. This fact does not hold true for discrete Gaussians, at least not without additional conditions. In 2004, Micciancio and Regev [MR07] introduced the notion of *smoothing parameter of a lattice* in order to capture the standard deviation threshold above which a discrete Gaussian behaves almost like a continuous one.

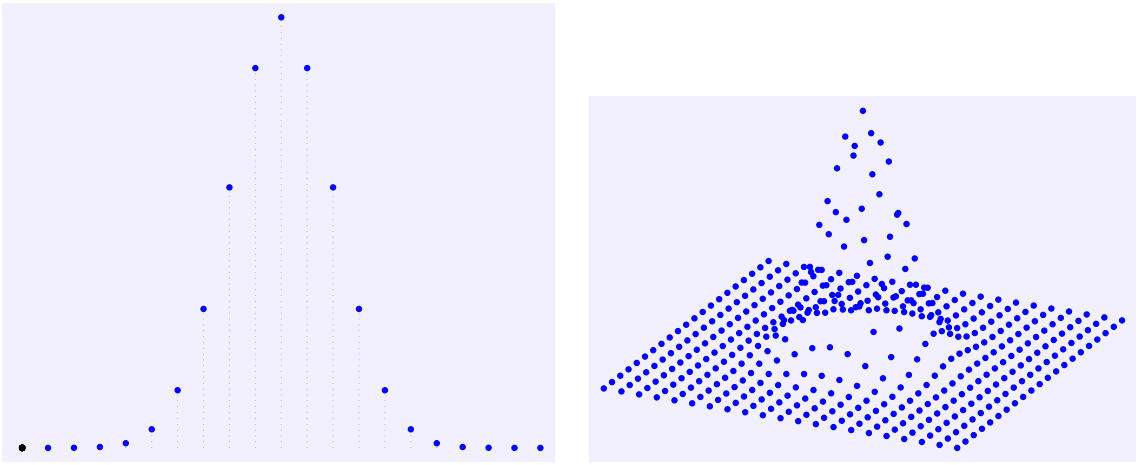


Figure 4.1: Probability density functions of $\mathcal{D}_{\mathbb{Z},s}$ and $\mathcal{D}_{\mathbb{Z}^2,s}$ for $s = 5$

4.2.1 Fourier Transform and Poisson Summation Formula

Before introducing the smoothing parameter, we make a few reminders of the tools needed to introduce it. In particular, certain results related to the smoothing parameter resort to the Fourier transform of the Gaussian function. The geometric properties of Gaussian distributions make them invariant under the Fourier transform, up to a rescaling of their standard deviation.

Definition 4.4 (Fourier Transform)

Let d, k be a positive integer, and $f : \mathbb{R}^d \longrightarrow \mathbb{R}^k$ a summable function. The *Fourier Transform* of the function f is the function $\widehat{f} : \mathbb{R}^d \longrightarrow \mathbb{C}^k$ defined by

$$\forall \omega \in \mathbb{R}^d, \widehat{f}(\omega) = \int_{\mathbb{R}^d} f(\mathbf{x}) \exp(-i \cdot 2\pi \langle \mathbf{x}\,, \omega \rangle) d\mathbf{x}.$$

We recall that a translation of a function results in a phase shift of the Fourier transform, i.e., for a summable function f and $\mathbf{v} \in \mathbb{R}^d$, the function $g : \mathbf{x} \mapsto f(\mathbf{x} + \mathbf{v})$ has Fourier transform $\widehat{g}(\cdot) = \widehat{f}(\cdot) \exp(2i\pi \langle \mathbf{v}\,, \cdot \rangle)$. As mentioned, the spherical Gaussian function is invariant under the Fourier transform up to scaling, that is $\widehat{\rho_s} = s^d \rho_{1/s}$. We can generalize it as follows.

Lemma 4.1 (Fourier Transform of Gaussian Function)

Let d be a positive integer, $s > 0$ and $\mathbf{c} \in \mathbb{R}^d$. It holds that

$$\forall \omega \in \mathbb{R}^d, \widehat{\rho_{s,\mathbf{c}}}(\omega) = s^d \cdot \exp(-i \cdot 2\pi \langle \mathbf{c}\,, \omega \rangle) \rho_{1/s}(\omega).$$

A very important and usual result in the study of lattices is the *Poisson Summation Formula*. It relates the mass of a lattice with respect to a function f, with the volume of its dual and the frequency mass of its dual.

Theorem 4.1 (Poisson Summation Formula)

Let k be a positive integer and $f : \mathbb{R}^k \longrightarrow \mathbb{C}$ a well-behaved function[a]. Let $\mathcal{L} \subset \mathbb{R}^k$ be a lattice. Then, it holds that

$$f(\mathcal{L}) = \mathrm{Vol}\, \mathcal{L}^* \cdot \widehat{f}(\mathcal{L}^*) = (\mathrm{Vol}\, \mathcal{L})^{-1} \widehat{f}(\mathcal{L}^*).$$

[a] The conditions are rather complex and specified in [Ebe02]. All the functions we consider in this course satisfy those conditions.

4.2.2 Regularity and Smoothing Parameter

We start by giving the following result of Micciancio and Regev [MR07] which is the natural starting point for introducing the smoothing parameter.

Lemma 4.2 ([MR07, Lem. 4.1])

Let d be a positive integer. Let $s > 0$, $\mathbf{c}$ in $\mathbb{R}^d$ and $\mathcal{L} \subset \mathbb{R}^d$ a full-rank lattice of rank d. We let $\mathbf{B} \in \mathbb{R}^{d\times d}$ be a basis of $\mathcal{L}$. It then holds that

$$\Delta(D_{s,\mathbf{c}} \bmod \mathcal{P}(\mathbf{B}), U(\mathcal{P}(\mathbf{B}))) \leq \frac{1}{2}\rho_{1/s}(\mathcal{L}^* \setminus \{\mathbf{0}\}),$$

where $\mathcal{P}(\mathbf{B})$ is the fundamental parallelepiped associated to the basis $\mathbf{B}$ of the lattice $\mathcal{L}$.

Proof (Lemma 4.2). We let f be the probability density function over $\mathcal{P}(\mathbf{B})$ defined by $D_{s,\mathbf{c}} \bmod \mathcal{P}(\mathbf{B})$. Thence, for all $\mathbf{x}$ in $\mathcal{P}(\mathbf{B})$, it holds that

$$f(\mathbf{x}) = \frac{1}{s^d} \cdot \sum_{\mathbf{y}\in\mathcal{L}} \rho_{s,\mathbf{c}}(\mathbf{x}+\mathbf{y}) = \frac{1}{s^d}\rho_{s,\mathbf{c}-\mathbf{x}}(\mathcal{L}).$$

By the Poisson summation formula of Theorem 4.1, and Lemma 4.1, it yields that

$$\begin{aligned} f(\mathbf{x}) &= \frac{1}{\mathrm{Vol}\,(\mathcal{L}) \cdot s^d}\widehat{\rho_{s,\mathbf{c}-\mathbf{x}}}(\mathcal{L}^*) \\ &= \frac{1}{\mathrm{Vol}\,(\mathcal{L}) \cdot s^d} s^d \sum_{\omega\in\mathcal{L}^*} \exp(-i2\pi\langle \mathbf{c}-\mathbf{x}\,,\omega\rangle)\rho_{1/s}(\omega) \\ &= \frac{1}{\mathrm{Vol}\,(\mathcal{L})}\left(1 + \sum_{\omega\in\mathcal{L}^*\setminus\{\mathbf{0}\}} \exp(-i2\pi\langle \mathbf{c}-\mathbf{x}\,,\omega\rangle)\rho_{1/s}(\omega)\right). \end{aligned}$$

The density function of $U(\mathcal{P}(\mathbf{B}))$ is $(\mathrm{Vol}\,\mathcal{P}(\mathbf{B}))^{-1}\mathbf{1}_{\mathcal{P}(\mathbf{B})} = (\mathrm{Vol}\,\mathcal{L})^{-1}\mathbf{1}_{\mathcal{P}(\mathbf{B})}$. Hence, we have the following inequalities.

$$\begin{aligned} \Delta(D_{s,\mathbf{c}} \bmod \mathcal{P}(\mathbf{B}), U(\mathcal{P}(\mathbf{B}))) &= \frac{1}{2}\int_{\mathbf{x}\in\mathcal{P}(\mathbf{B})} \left|f(\mathbf{x}) - \mathrm{Vol}\,(\mathcal{L})^{-1}\right| d\mathbf{x} \\ &\leq \frac{1}{2}\mathrm{Vol}\,(\mathcal{L}) \cdot \max_{\mathbf{x}\in\mathcal{P}(\mathbf{B})} \left|f(\mathbf{x}) - \mathrm{Vol}\,(\mathcal{L})^{-1}\right| \\ &= \frac{1}{2}\max_{\mathbf{x}\in\mathcal{P}(\mathbf{B})} \left|\left(1 + \sum_{\omega\in\mathcal{L}^*\setminus\{\mathbf{0}\}} \exp(-i2\pi\langle \mathbf{c}-\mathbf{x}\,,\omega\rangle)\rho_{1/s}(\omega)\right) - 1\right| \\ &\leq \frac{1}{2}\max_{\mathbf{x}\in\mathcal{P}(\mathbf{B})} \sum_{\omega\in\mathcal{L}^*\setminus\{\mathbf{0}\}} \rho_{1/s}(\omega) \\ &= \frac{1}{2}\rho_{1/s}(\mathcal{L}^* \setminus \{\mathbf{0}\}), \end{aligned}$$

as desired.

Lemma 4.2 then leads to the definition of the *smoothing parameter* which is essentially the smallest $\sqrt{\mathbf{S}}$ such that this statistical distance is bounded by $\varepsilon/2$ for some $\varepsilon > 0$.

Definition 4.5 (Smoothing Parameter of a Lattice)

Let k, d be two positive integers and $\mathcal{L} \subset \mathbb{R}^k$ a lattice of rank d. For any positive real ε, the *smoothing paramater* of $\mathcal{L}$ with respect to ε is defined by

$$\eta_\varepsilon(\mathcal{L}) = \min\{s > 0 : \rho_{1/s}(\mathcal{L}^*) \leq 1 + \varepsilon\}.$$

Byu definition of the smoothing parameter, whenever $s \geq \eta_\varepsilon(\mathcal{L})$, the result of Lemma 4.2

yields a statistical distance of at most $\varepsilon/2$. When ε is negligible, it means that the Gaussian distribution reduced to the fundamental volume of the lattice is indistinguishable from the uniform distribution over said volume. This is the reason for the appellation *smoothing*. We will see in the following sections some properties of discrete Gaussians that are satisfied when the width exceeds the smoothing parameter. However, it is imperative to be able to evaluate or estimate the value of said smoothing parameter in order to know how big we need the width to be. Several results were provided which bound the smoothing parameter by lattice quantities. In general, precisely estimating the smoothing parameter is difficult, and is still subject to active research today. We nevertheless give a few bound here, which are sufficient in most cases.

Lemma 4.3 (Smoothing Parameter Bounds)

Let d be a positive integer, ε a positive real, and p be in $\mathbb{N}^* \cup \{\infty\}$. Let $\mathcal{L} \subset \mathbb{R}^d$ be a full-rank lattice of rank d. The following bounds hold

$$\eta_\varepsilon(\mathcal{L}) \leq \sqrt{\frac{\ln(2d(1+\varepsilon^{-1}))}{\pi}} \cdot \lambda_d(\mathcal{L}) \qquad ([\text{MR07}])$$

$$\eta_\varepsilon(\mathcal{L}) \geq \sqrt{\frac{\ln(\varepsilon^{-1})}{\pi}} \cdot \frac{1}{\lambda_1(\mathcal{L}^*)} \geq \sqrt{\frac{\ln(\varepsilon^{-1})}{\pi}} \cdot \frac{\lambda_d(\mathcal{L})}{d} \qquad ([\text{Reg05}])$$

$$\eta_\varepsilon(\mathcal{L}) \leq \sqrt{\frac{\ln(2d(1+\varepsilon^{-1}))}{\pi}} \cdot \frac{1}{\lambda_1^\infty(\mathcal{L}^*)} \leq \sqrt{\frac{\ln(2d(1+\varepsilon^{-1}))}{\pi}} \cdot \frac{d^{1/p}}{\lambda_1^p(\mathcal{L}^*)} \qquad ([\text{Pei08}])$$

$$\eta_\varepsilon(\mathcal{L}) \leq \sqrt{\frac{\ln(2d(1+\varepsilon^{-1}))}{\pi}} \cdot \lambda_{\mathsf{GSO}}(\mathcal{L}) \qquad ([\text{GPV08}])$$

where $\lambda_{\mathsf{GSO}}(\mathcal{L}) = \inf_{\mathbf{B} \text{ basis of } \mathcal{L}} \|\mathsf{GSO}(\mathbf{B})\|_\infty = \inf_{\mathbf{B} \text{ basis of } \mathcal{L}} \max_{1 \leq i \leq d} \|\mathsf{GSO}(\mathbf{B})\mathbf{e}_i\|_2$.

4.3 Properties of Discrete Gaussians

We now look at some properties of discrete Gaussian distributions. In particular, we try to replicate the known results for continuous Gaussian distributions. In this course, we only look at the sum of two independent discrete Gaussians, the translation of a discrete Gaussian, and finally concentration bounds of discrete Gaussian samples which are essential for lattice cryptography.

4.3.1 Basic Properties

We start by a very important result which essentially states that when the standard deviation exceeds the smoothing parameter of the lattice, the Gaussian mass of the lattice does not really depend on the center.

Lemma 4.4 (Gaussian Mass of Lattice Cosets)

Let d be a positive integer and ε a positive real. Let $s > 0$, $\mathbf{c}$ in $\mathbb{R}^d$ and $\mathcal{L} \subset \mathbb{R}^d$ be a lattice of full rank d. If $s \geq \eta_\varepsilon(\mathcal{L})$, then we have

$$\rho_{s,\mathbf{c}}(\mathcal{L}) \in \left[\frac{1-\varepsilon}{1+\varepsilon}, 1\right] \rho_s(\mathcal{L}).$$

Proof (Lemma 4.4). See TD 1

From this result, we can adapt the result provided in Lemma 4.2 discrete Gaussian distributions reduced modulo a sublattice. This result is due to [GPV08, Cor. 2.8].

Lemma 4.5 ([GPV08, Cor. 2.8])

Let d be a positive integer. Let $s > 0$, $\mathbf{c}$ in $\mathbb{R}^d$ and $\mathcal{L}' \subseteq \mathcal{L} \subset \mathbb{R}^d$ two lattices of full rank d. Let $\varepsilon \in (0,1)$ and assume $s \geq \eta_\varepsilon(\mathcal{L}')$. We define the distributions $\mathcal{P}_0 = \mathcal{D}_{\mathcal{L},s,\mathbf{c}} \bmod \mathcal{L}'$ and $\mathcal{P}_1 = U(\mathcal{L} \bmod \mathcal{L}')$. We then have

$$\forall \mathbf{x} \in \mathcal{L} \bmod \mathcal{L}', \mathcal{P}_0(\mathbf{x}) \in \left[\frac{1-\varepsilon}{1+\varepsilon}, \frac{1+\varepsilon}{1-\varepsilon}\right] \cdot \mathcal{P}_1(\mathbf{x}).$$

In particular, we get $\Delta(\mathcal{D}_{\mathcal{L},s,\mathbf{c}} \bmod \mathcal{L}', U(\mathcal{L} \bmod \mathcal{L}')) \leq \varepsilon/(1-\varepsilon)$.

Proof (Lemma 4.5). Consider the marginal distribution of $(\mathbf{z} \bmod \mathcal{L}')$ where $\mathbf{z}$ is drawn from $\mathcal{D}_{\mathcal{L},s,\mathbf{c}}$. Then, for any coest $\mathbf{v} + \mathcal{L}'$ of $\mathcal{L}/\mathcal{L}'$, the probability that $\mathbf{z}$ belongs to $\mathbf{v} + \mathcal{L}'$ is proportional to

$$\rho_{s,\mathbf{c}}(\mathbf{v} + \mathcal{L}') = \rho_{s,\mathbf{c}-\mathbf{v}}(\mathcal{L}') \in \mathrm{Vol}\,(\mathcal{L}')^{-1} s^d [1-\varepsilon, 1+\varepsilon],$$

s in the proof of Lemma 4.4. Similarly, we have $\rho_{s,\mathbf{c}}(\mathcal{L}) \in \mathrm{Vol}\,(\mathcal{L})^{-1} s^d [1-\varepsilon, 1+\varepsilon]$ due to the fact that $\eta_\varepsilon(\mathcal{L}') \geq \eta_\varepsilon(\mathcal{L})$. As a result, we have

$$\mathbb{P}_{\mathbf{z}}[\mathbf{z} = \mathbf{v} \bmod \mathcal{L}'] = \frac{\rho_{s,\mathbf{c}}(\mathbf{v} + \mathcal{L}')}{\rho_{s,\mathbf{c}}(\mathcal{L})} \in \frac{\mathrm{Vol}\,(\mathcal{L})}{\mathrm{Vol}\,(\mathcal{L}')} \cdot \left[\frac{1-\varepsilon}{1+\varepsilon}, \frac{1+\varepsilon}{1-\varepsilon}\right] = \left|\mathcal{L}/\mathcal{L}'\right|^{-1} \cdot \left[\frac{1-\varepsilon}{1+\varepsilon}, \frac{1+\varepsilon}{1-\varepsilon}\right],$$

as claimed. The result on the statistical distance simply consists in using the first claim in the computation of the statistical distance, which gives

$$\Delta(\mathcal{D}_{\mathcal{L},s,\mathbf{c}} \bmod \mathcal{L}', U(\mathcal{L} \bmod \mathcal{L}') \leq \max\left(\varepsilon/(1-\varepsilon), \varepsilon/(1+\varepsilon)\right) = \varepsilon/(1-\varepsilon).$$

The Poisson summation formula allows for easily bounding the Gaussian mass of a lattice. A direct application is a lower bound on the min-entropy of a discrete Gaussian random variable. We recall the definition of min-entropy here.

Definition 4.6 (Min-Entropy)

Let S be a countable set, and x a random variable following a discrete distribution over S. The *min-entropy* of $\mathbf{x}$ (or of its distribution) is defined by

$$H_\infty(x) = -\log_2 \max_{x' \in S} \mathbb{P}[x = x'].$$

Lemma 4.6 (Entropy of Discrete Gaussian)

Let d be a positive integer and ε a positive real. Let $s > 0$, $\mathbf{c}$ in $\mathbb{R}^d$ and $\mathcal{L} \subset \mathbb{R}^d$ a lattice of full rank d. If $s \geq \eta_\varepsilon(\mathcal{L})$, it holds that

$$H_\infty(\mathcal{D}_{\mathcal{L},s,\mathbf{c}}) \geq d\log_2(s) - \log_2(\mathrm{Vol}\,(\mathcal{L})) + \log_2(1-\varepsilon).$$

Proof (Lemma 4.6). First of all, note that for all $\mathbf{x}$ in $\mathcal{L}$, $\rho_{s,\mathbf{c}}(\mathbf{x}) \leq 1$. It then suffices to bound the denominator $\rho_{s,\mathbf{c}}(\mathcal{L})$. In the proof of Lemma 4.4, we have seen that $\rho_{s,\mathbf{c}}(\mathcal{L}) \geq (1-\varepsilon)\mathrm{Vol}\,(\mathcal{L})^{-1} s^d$. We thus obtain that

$$H_\infty(\mathcal{D}_{\mathcal{L},\sqrt{\mathbf{S}},\mathbf{c}}) \geq \log_2 \rho_{s,\mathbf{c}}(\mathcal{L}) \geq d\log_2(s) - \log_2(\mathrm{Vol}\,(\mathcal{L})) + \log_2(1-\varepsilon),$$

as desired.

We now give the expected result that whenever the Gaussian width are sufficiently large with respect to the smoothing parameter, the sum of two independent discrete Gaussian distributions is statistically close to a discrete Gaussian distribution.

Lemma 4.7 (Sum of Independent Discrete Gaussians)

Let d be a positive integer, and ε in $(0, 1/2]$. Let s_1, s_2 be two positive reals, $\mathbf{c}_1, \mathbf{c}_2, \mathbf{a}_1, \mathbf{a}_2$ in $\mathbb{R}^d$ and $\mathcal{L} \subset \mathbb{R}^d$ a lattice of full rank d. We define $\mathbf{a} = \mathbf{a}_1 + \mathbf{a}_2$, $\mathbf{c} = \mathbf{c}_1 + \mathbf{c}_2$, and $s = \sqrt{s_1^2 + s_2^2}$. We also define $t = 1/\sqrt{s_1^{-2} + s_2^{-2}}$. If $t \geq \eta_\varepsilon(\mathcal{L})$, it then holds that

$$\Delta(\mathcal{D}_{\mathcal{L}+\mathbf{a}_1,s_1,\mathbf{c}_1} + \mathcal{D}_{\mathcal{L}+\mathbf{a}_2,s_2,\mathbf{c}_2}, \mathcal{D}_{\mathcal{L}+\mathbf{a},s,\mathbf{c}}) \leq \frac{\varepsilon}{1-\varepsilon} \underset{\varepsilon\to 0}{\sim} \varepsilon.$$

Proof (Lemma 4.7). We call f the probability density function of $\mathcal{D}_{\mathcal{L}+\mathbf{a}_1,s_1,\mathbf{c}_1} + \mathcal{D}_{\mathcal{L}+\mathbf{a}_2,s_2,\mathbf{c}_2}$. Its support is $\mathcal{L} + \mathbf{a}$. Let $\mathbf{x}$ be in $\mathcal{L} + \mathbf{a}$. We define $\mathbf{c}' = t^2(s_1^{-2}\mathbf{c}_1 + s_2^{-2}(\mathbf{x} - \mathbf{c}_2)) = (s_2/s)^2\mathbf{c}_1 + (s_1/s)^2(\mathbf{x} - \mathbf{c}_2)$. We then have

$$\begin{aligned}
f(\mathbf{x}) &= \sum_{\mathbf{y}\in\mathcal{L}+\mathbf{a}_1} \mathcal{D}_{\mathcal{L}+\mathbf{a}_1,s_1,\mathbf{c}_1}(\mathbf{y}) \cdot \mathcal{D}_{\mathcal{L}+\mathbf{a}_2,s_2,\mathbf{c}_2}(\mathbf{x}-\mathbf{y}) \\
&= \frac{1}{\rho_{s_1,\mathbf{c}_1}(\mathcal{L}+\mathbf{a}_1)\rho_{s_2,\mathbf{c}_2}(\mathcal{L}+\mathbf{a}_2)} \sum_{\mathbf{y}\in\mathcal{L}+\mathbf{a}_1} \rho_{s_1,\mathbf{c}_1}(\mathbf{y})\rho_{s_2,\mathbf{c}_2}(\mathbf{x}-\mathbf{y}) \\
&= \frac{1}{\rho_{s_1,\mathbf{c}_1-\mathbf{a}_1}(\mathcal{L})\rho_{s_2,\mathbf{c}_2-\mathbf{a}_2}(\mathcal{L})} \sum_{\mathbf{y}\in\mathcal{L}+\mathbf{a}_1} \rho_{s,\mathbf{c}}(\mathbf{x})\rho_{t,\mathbf{c}'}(\mathbf{y}) \qquad \text{[Pei10, } \textit{Fact } 2.1\text{]} \\
&= \mathcal{D}_{\mathcal{L}+\mathbf{a},s,\mathbf{c}}(\mathbf{x}) \cdot \frac{\rho_{s,\mathbf{c}}(\mathcal{L}+\mathbf{a})\rho_{t,\mathbf{c}'}(\mathcal{L}+\mathbf{a}_1)}{\rho_{s_1,\mathbf{c}_1-\mathbf{a}_1}(\mathcal{L})\rho_{s_2,\mathbf{c}_2-\mathbf{a}_2}(\mathcal{L})} \\
&= \mathcal{D}_{\mathcal{L}+\mathbf{a},s,\mathbf{c}}(\mathbf{x}) \cdot \frac{\rho_{s,\mathbf{c}-\mathbf{a}}(\mathcal{L})\rho_{t,\mathbf{c}'-\mathbf{a}_1}(\mathcal{L})}{\rho_{s_1,\mathbf{c}_1-\mathbf{a}_1}(\mathcal{L})\rho_{s_2,\mathbf{c}_2-\mathbf{a}_2}(\mathcal{L})}.
\end{aligned}$$

We now observe that f is proportional to $\mathcal{D}_{\mathcal{L}+\mathbf{a},s,\mathbf{c}}$, at the exception that $\mathbf{c}'$ depends on $\mathbf{x}$. Yet, we have seen that $\mathbf{c}'$ can be "smoothed out" when computing the Gaussian mass of the lattice if the Gaussian width exceeds the smoothing parameter. As $t \geq \eta_\varepsilon(\mathcal{L})$, Lemma 4.4 yields

$$\rho_{t,\mathbf{c}'-\mathbf{a}_1}(\mathcal{L}) \in \left[\frac{1-\varepsilon}{1+\varepsilon}, 1\right] \rho_t(\mathcal{L}) \tag{4.1}$$

We then get that

$$\begin{aligned}
f(\mathbf{x}) &\in \left[\frac{1-\varepsilon}{1+\varepsilon}, 1\right] \cdot \mathcal{D}_{\mathcal{L}+\mathbf{a},s,\mathbf{c}}(\mathbf{x}) \cdot \frac{\rho_{s,\mathbf{c}-\mathbf{a}}(\mathcal{L})\rho_t(\mathcal{L})}{\rho_{s_1,\mathbf{c}_1-\mathbf{a}_1}(\mathcal{L})\rho_{s_2,\mathbf{c}_2-\mathbf{a}_2}(\mathcal{L})} \\
&= \left[\frac{1-\varepsilon}{1+\varepsilon}, 1\right] \cdot \alpha \cdot \mathcal{D}_{\mathcal{L}+\mathbf{a},s,\mathbf{c}}(\mathbf{x})
\end{aligned}$$

for a constant α which does not depend on $\mathbf{x}$. Because the support of f is exactly that of $\mathcal{D}_{\mathcal{L}+\mathbf{a},s,\mathbf{c}}$ and that they are probability distributions, summing over all the $\mathbf{x} \in \mathcal{L} + \mathbf{a}$ then gives $\alpha(1-\varepsilon)/(1+\varepsilon) \leq 1 \leq \alpha$. We deduce that the constant of proportionality α is in the interval $[1, (1+\varepsilon)/(1-\varepsilon)]$. We then get that for all $\mathbf{x} \in \mathcal{L} + \mathbf{a}$

$$f(\mathbf{x}) \in \left[\frac{1-\varepsilon}{1+\varepsilon}, \frac{1+\varepsilon}{1-\varepsilon}\right] \cdot \mathcal{D}_{\mathcal{L}+\mathbf{a},s,\mathbf{c}}(\mathbf{x}),$$

and thus that

$$\left|f(\mathbf{x}) - \mathcal{D}_{\mathcal{L}+\mathbf{a},s,\mathbf{c}}(\mathbf{x})\right| \leq \max\left(\frac{2\varepsilon}{1+\varepsilon}, \frac{2\varepsilon}{1-\varepsilon}\right) = \frac{2\varepsilon}{1-\varepsilon}.$$

Summing over all $\mathbf{x}$, with the factor $1/2$ of the statistical distance, we get that

$$\Delta(\mathcal{D}_{\mathcal{L}+\mathbf{a}_1,s_1,\mathbf{c}_1} + \mathcal{D}_{\mathcal{L}+\mathbf{a}_2,s_2,\mathbf{c}_2}, \mathcal{D}_{\mathcal{L}+\mathbf{a},s,\mathbf{c}}) \leq \frac{\varepsilon}{1-\varepsilon} \underset{\varepsilon\to 0^+}{\sim} \varepsilon,$$

as claimed.

The smoothing condition can be simply met. For example, when we consider spherical Gaussians with width s_1, s_2, it suffices to have $s_1, s_2 \geq \sqrt{2}\eta_\varepsilon(\mathcal{L})$. This ensures that $t = (s_1^{-2} + s_2^{-2})^{-1} \geq \eta_\varepsilon(\mathcal{L})$.

4.3.2 Gaussian Tail Bounds

A very important property of Gaussian distribution is that there are concentrated around their center $\mathbf{c}$. This means that when sampling $\mathbf{x}$ according to a Gaussian distribution, it will be close to $\mathbf{c}$ with high probability (for a usual notion of distance like the Euclidean distance for example). These results are called *tail bounds* as they bound the probability that a Gaussian sample falls in the *tails* of the Gaussian. It capture the fact that the probability density is lower when you are far from the center than when you are close to the center. And the decrease is super-exponential. It turns out that discrete Gaussian distribution also verify such tail bounds, sometimes under smoothing conditions. We first give the original tail bound on the Euclidean norm by Banaszczyk [Ban93].

Lemma 4.8 (Banaszczyk Tail Bound [Ban93])

Let d be a positive integer. Let $\mathcal{L} \subset \mathbb{R}^d$ be a full-rank lattice, $\mathbf{c} \in \mathbb{R}^d$ and s a positive real. For all $c \geq 1/\sqrt{2\pi}$, it holds that

$$\mathbb{P}_{\mathbf{x}\sim\mathcal{D}_{\mathcal{L},s}}\left[\|\mathbf{x}\|_2 > c \cdot s\sqrt{d}\right] \leq \left(c\sqrt{2\pi e}e^{-\pi c^2}\right)^d$$
$$\mathbb{P}_{\mathbf{x}\sim\mathcal{D}_{\mathcal{L},s,\mathbf{c}}}\left[\|\mathbf{x}-\mathbf{c}\|_2 > c \cdot s\sqrt{d}\right] \leq 2\left(c\sqrt{2\pi e}e^{-\pi c^2}\right)^d \cdot \frac{\rho_s(\mathcal{L})}{\rho_{s,\mathbf{c}}(\mathcal{L})}$$

We can then derive the following tail bounds.

Lemma 4.9 (Euclidean Tail Bound)

Let d be a positive integer. Let $\mathcal{L} \subset \mathbb{R}^d$ be a full-rank lattice, $\mathbf{c} \in \mathbb{R}^d$ and s a positive real. It holds that

$$\mathbb{P}_{\mathbf{x}\sim\mathcal{D}_{\mathcal{L},s}}\left[\|\mathbf{x}\|_2 > s\sqrt{d}\right] \leq 2^{-2d}$$
$$\mathbb{P}_{\mathbf{x}\sim\mathcal{D}_{\mathcal{L},s,\mathbf{c}}}\left[\|\mathbf{x}-\mathbf{c}\|_2 > s\sqrt{d}\right] \leq \frac{1+\varepsilon}{1-\varepsilon}2^{-2d} \qquad \text{if } s \geq \eta_\varepsilon(\mathcal{L})$$

Proof (Lemma 4.9). The first bound comes directly from the first bound of Lemma 4.8 for $c = 1 \geq 1/\sqrt{2\pi}$. Indeed, we have $\sqrt{2\pi e}e^{-\pi} \leq 0.178 \leq 1/4$. For the second one, we also use Lemma 4.8 with $c = 1$ as well as Lemma 4.4 to conclude. Whenever $d \geq 3$, we have that $2(\sqrt{2\pi e}e^{-\pi})^d \leq 2^{-2d}$.

Another very interesting tail bound used in lattice-based cryptography is a bound on the ℓ_∞ norm. The following is generally attributed to Peikert [Pei08], but this simpler version was actually proven in [Ban95].

Lemma 4.10 (Infinity Tail Bound)

Let d be a positive integer. Let $\mathcal{L} \subset \mathbb{R}^d$ be a full-rank lattice, $\mathbf{c} \in \mathbb{R}^d$ and s a positive real. For all $t \geq 0$

$$\mathbb{P}_{\mathbf{x}\sim\mathcal{D}_{\mathcal{L},s}}\left[\|\mathbf{x}\|_\infty > t \cdot s\right] \leq 2d \cdot e^{-\pi t^2}$$
$$\mathbb{P}_{\mathbf{x}\sim\mathcal{D}_{\mathcal{L},s,\mathbf{c}}}\left[\|\mathbf{x}-\mathbf{c}\|_\infty > t \cdot s\right] \leq 2de \cdot e^{-\pi t^2} \qquad \text{if } s \geq \eta_\varepsilon(\mathcal{L}) \text{ for } \varepsilon \leq 1/3$$

4.4 Sampling Gaussians over Lattices

As presented, it is possible to theoretically define discrete Gaussian distributions over lattices. They benefit from nice geometric properties just like regular Gaussian distributions do. The properties that we enumerated in the previous sections are very interesting in cryptography. However, there remains one concerning point which is the link between discrete Gaussian distributions and the supposedly hard problems on lattices we introduced in Chapter 3 (SVP, CVP, etc.).

Let us consider the SVP_γ problem in full-rank lattices of dimension d, and let $\mathcal{L}$ be such a lattice. We let $s > \eta_\varepsilon(\mathcal{L})$ be a Gaussian parameter and we consider the distribution $\mathcal{D}_{\mathcal{L},s}$. Let $\mathbf{x}$

be a sample from this distribution. By Lemma 4.9, it holds that $\|\mathbf{x}\|_2 \leq s\sqrt{d}$ with overwhelming probability. Additionally, we have that $H_\infty(\mathcal{D}_{\mathcal{L},s}) \geq d\log_2 s - \log_2(\mathrm{Vol}\,\mathcal{L}) - \log_2(1-\varepsilon)$. Here, the volume of the lattice can be deduced from any basis of the lattice. Alternatively, we can use another result by Peikert and Rosen [PR06] which states that if $s > 2\eta_\varepsilon(\mathcal{L})$, then $H_\infty(\mathcal{D}_{\mathcal{L},s}) \geq d$, which is going to be sufficient for this argument. Indeed, it means that $\mathcal{D}_{\mathcal{L},s}(\mathbf{0}) \leq 2^{-H_\infty(\mathcal{D}_{\mathcal{L},s})} \leq 2^{-d}$. In other terms, it proves that the probability that the sample $\mathbf{x}$ is $\mathbf{0}$ is bounded above by 2^{-d}. Hence, $\mathbf{x}$ is non zero with overwhelming probability. This means $\mathbf{x}$ is a valid solution of SVP_γ if $\gamma\lambda_1(\mathcal{L}) \geq s\sqrt{d}$.

The question is now twofold.

Can we choose a Gaussian parameter s as small as $\gamma\lambda_1(\mathcal{L})/\sqrt{d}$? If so, can we sample $\mathbf{x}$ in polynomial time?

According to Conjecture 3.1, both should not be possible for $\gamma = \mathsf{poly}(d)$, otherwise the conjecture would not hold. It turns out that to be able to sample $\mathbf{x}$ according to a discrete Gaussian distribution over $\mathcal{L}$, we need an algorithmic representation of $\mathcal{L}$, that is a basis. However, we have seen that a basis can be composed of very long almost colinear vectors in which case it seems difficult to generate short Gaussian samples. On the contrary, if the basis is short and close to orthogonal, this task should be easy. Intuitively, the width s of the Gaussian we are able to sample should depend on the quality of the given lattice basis. This is indeed the case and it explains why our approach for solving SVP_γ does not work in practice. A hard instance of SVP_γ is giving a bad basis like we described, which means that we would only be able to sample Gaussian samples with a huge width that exceeds $\gamma\lambda_1(\mathcal{L})/\sqrt{d}$.

We note however that it is always possible to sample from a discrete Gaussian distribution with *any* width. The catch is that the narrower the Gaussian, the more time it takes if the basis is not sufficiently small to begin with. Typically, given a bad basis, it would take an exponential time complexity to sample from a discrete Gaussian that is significantly narrower than the size of the basis.

4.4.1 Klein Sampler

In 2000, Klein proposed a way to sample discrete Gaussians, which was later formalized and generalized in 2008 by Peikert, Gentry and Vaikuntanathan [GPV08]. This algorithm gives an efficient way to sample from a distribution that is statistically close to a discrete Gaussian on any lattice, provided that the Gaussian parameter is sufficiently larger than the Gram-Schmidt norm of the given basis. We describe the algorithm here, assuming an efficient integer sampler for the distribution $\mathcal{D}_{\mathbb{Z},s,c}$ for any $s \geq \eta_\varepsilon(\mathbb{Z})$ and $c \in \mathbb{R}$. The latter will be studied in TD 1.

Algorithm 4.1: Klein$(\mathbf{B}, s, \mathbf{c})$

Input: Basis $\mathbf{B} = [\mathbf{b}_i]_{i\in[\![1,d]\!]}$ of the lattice $\mathcal{L}$, Gaussian parameter $s > 0$, center $\mathbf{c} \in \mathbb{R}^d$.
Precomputation: Compute the Gram-Schmidt vectors $\mathsf{GSO}(\mathbf{B}) = [\mathbf{b}_i^*]_{i\in[\![1,d]\!]}$, and the intermediate widths $s_i = s/\|\mathbf{b}_i^*\|_2$.

1. $\mathbf{v}_d \leftarrow \mathbf{0}$
2. **for** $i = d, \ldots, 1$ **do**
3. $\quad d_i \leftarrow \langle \mathbf{c} - \mathbf{v}_i\,, \mathbf{b}_i^* \rangle / \|\mathbf{b}_i^*\|_2^2 \in \mathbb{R}$.
4. $\quad z_i \hookleftarrow \mathcal{D}_{\mathbb{Z},s_i,d_i}$.
5. $\quad \mathbf{v}_{i-1} \leftarrow \mathbf{v}_i + z_i\mathbf{b}_i$.

Output: $\mathbf{v}_0$. ▷ Statistically close to $\mathcal{D}_{\mathcal{L},s,\mathbf{c}}$.

The sampler was analyzed in depth by Prest [Pre17], giving a finer result on the distribution outputted by the sampler and its statistical distance with the ideal distribution.

Theorem 4.2 (Distribution of Klein's Sampler [GPV08, Pre17])

Let d be a positive integer. Let $\mathcal{L}$ be a lattice of full rank d, $\mathbf{c} \in \mathbb{R}^d$, ε in $(0, 1/4)$, and $s \geq \eta_\varepsilon(\mathbb{Z}^d) \cdot \left\|\mathsf{GSO}(\mathbf{B})\right\|_\infty$ where $\|\mathbf{A}\|_\infty = \max_{i\in[\![1,d]\!]}\|\mathbf{a}_i\|_2$. We then have

$$\Delta(\mathsf{Klein}(\mathbf{B}, s, \mathbf{c}), \mathcal{D}_{\mathcal{L},s,\mathbf{c}}) \leq \frac{1}{2}\left(\left(\frac{1+\varepsilon/d}{1-\varepsilon/d}\right)^d - 1\right) \approx \varepsilon.$$

Proof (Theorem 4.2). See TD 1

RECAP

- Discrete Gaussian are defined by the probability mass function $\mathcal{D}_{\mathcal{L},s,\mathbf{c}} = \rho_{s,\mathbf{c}}(\cdot)/\rho_{s,\mathbf{c}}(\mathcal{L})$ where $\rho_{s,\mathbf{c}}(\mathbf{x}) = \exp(-\pi\|\mathbf{x}-\mathbf{c}\|_2^2/s^2)$.
- The smoothing parameter of a lattice $\mathcal{L}$ is $\eta_\varepsilon(\mathcal{L}) = \min\{s > 0 : \rho_{1/s}(\mathcal{L}^*) \leq 1+\varepsilon\}$.
- When s is above the smoothing parameter of $\mathcal{L}$, the discrete Gaussian behaves like a continuous Gaussian. In particular, for $\mathbf{x} \hookleftarrow \mathcal{D}_{\mathcal{L},s,\mathbf{c}}$, the Gaussian tail bound gives $\|\mathbf{x}-\mathbf{c}\|_2 \leq s\sqrt{d}$ with overwhelming probability (provided that $s \geq \eta_\varepsilon(\mathcal{L})$).
- For an arbitrary lattice $\mathcal{L}$, Klein's algorithm allows for sampling a distribution that is statistically close to $\mathcal{D}_{\mathcal{L},s,\mathbf{c}}$ given a basis $\mathbf{B}$ of $\mathcal{L}$, as long as $s \geq \|\mathsf{GSO}(\mathbf{B})\|_\infty \cdot \eta_\varepsilon(\mathbb{Z}^d)$.

Bibliography

[Ban93] W. Banaszczyk. New Bounds in Some Transference Theorems in the Geometry of Numbers. *Math. Ann.*, 1993.

[Ban95] Wojciech Banaszczyk. Inequalites for convex bodies and polar reciprocal lattices in r^{n}. *Discret. Comput. Geom.*, 1995.

[Ebe02] W. Ebeling. *Lattices and Codes.* 2002.

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for Hard Lattices and New Cryptographic Constructions. In *STOC*, 2008.

[MR07] D. Micciancio and O. Regev. Worst-Case to Average-Case Reductions Based on Gaussian Measures. *SIAM J. Comput.*, 2007.

[Pei08] C. Peikert. Limits on the Hardness of Lattice Problems in l_p Norms. *Comput. Complex.*, 2008.

[Pei10] C. Peikert. An Efficient and Parallel Gaussian Sampler for Lattices. In *CRYPTO*, 2010.

[PR06] C. Peikert and A. Rosen. Efficient Collision-Resistant Hashing from Worst-Case Assumptions on Cyclic Lattices. In *TCC*, 2006.

[Pre17] T. Prest. Sharper Bounds in Lattice-Based Cryptography Using the Rényi Divergence. In *ASIACRYPT*, 2017.

[Reg05] O. Regev. On Lattices, Learning With Errors, Random Linear Codes, and Cryptography. In *STOC*, 2005.

5

Fundamental Problems: SIS and LWE

The lattice problems that we introduced in Chapter 3 form a solid base to prove the security of cryptographic systems. However, these problems are usually called *worst-case* problems which makes it difficult to directly base cryptographic designs on them. Indeed, for every dimension d, there exists lattices of dimension d for which it is easy to find the shortest vector (SVP) or the vector closest to a target (CVP). The hardness results on SVP, CVP, their approximate variants SVP_γ, CVP_γ, and many more, only concern the *worst* instances of these problems. In other words, these results only guarantee there exists lattices for which these problems are exponentially hard (either proven or conjectured). To obtain secure cryptographic constructions, one would need to have an efficient way of finding these hard instances, while finding secret information on them to allow signature or decryption for example. As this is no easy task, this motivated the introduction of more flexible fundamental problems called *average-case* problems for which the instances can be sampled (pseudo)randomly. The attractive feature of these problems, which is unique to lattice-based cryptography, is that they are proven to be at least as hard as the worst instance of the lattice problems defined in Chapter 3. They are thus easy to use, while proven at least as difficult to solve as hard problems on lattices. We present the two main average-case problems called *Short Integer Solution* (SIS) and *Learning With Errors* (LWE).

Contents

5.1 Short Integer Solution

We start by introducing the *Short Integer Solution* problem. The use of lattices in cryptography was really kick-started by the seminal work of Ajtai in 1996 [Ajt96]. He introduced the SIS problem as a versatile problem, while proving it was benefiting from a *worst-case to average-case reduction* from a variant of SVP. This means that if one is able to solve the SIS problem (on random instance with substantial probability), then one can solve the worst instance of GapSVP_γ. We start by defining the problem and how it links to lattice problems, before stating the hardness results of Ajtai. We then give the construction of Ajtai's hash function and Ajtai's commitment scheme whose security relies on the hardness of SIS.

5.1.1 Problem Definitions

The problem was introduced by Ajtai in 1996 [Ajt96], and later formalized by Micciancio and Regev in 2007 [MR07]. It relies on the observation mentioned above that hard lattices may be

impractical to find and use in cryptography. One would instead to find random instances of certain lattice problems, say SVP_γ. For that, one needs to essentially sample random lattices[1], for which the problems are expected to be hard. However, we want SVP_γ to be hard on such random lattices. We must then find families of lattices that are easy to sample randomly, while preserving some guarantee of hardness for the associated computational problems.

Thence, consider a matrix $\mathbf{A}$ in $\mathbb{Z}_q^{d\times m}$ for positive integers d, m and q such that $m \geq d$. Then one can consider the q-ary parity check lattice $\mathcal{L}_q^\perp(\mathbf{A})$ described in Definition 2.3, i.e.,

$$\mathcal{L}_q^\perp(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A}\mathbf{x} = \mathbf{0} \bmod q\mathbb{Z}\},$$

One can consider the SVP_γ problem on the lattice $\mathcal{L}_q^\perp(\mathbf{A})$ which is to find a non-zero integer vector $\mathbf{x}$ such that $\mathbf{A}\mathbf{x} = \mathbf{0} \bmod q\mathbb{Z}$ and $\|\mathbf{x}\|_2 \leq \gamma \cdot \lambda_1(\mathcal{L}_q^\perp(\mathbf{A})) =: \beta$. We note that the problem may be easy if one knows a good basis of $\mathcal{L}_q^\perp(\mathbf{A})$, i.e., composed of short vectors. In 1996, Ajtai showed that finding short vectors in $\mathcal{L}_q^\perp(\mathbf{A})$ (and in turn a short basis) for $\mathbf{A}$ uniformly chosen in $\mathbb{Z}_q^{d\times m}$ is at least as hard as the worst instances of GapSVP_γ for $\gamma = \mathsf{poly}(m)$. This result in the following definition for the SIS problem. We note it can be generalized to any ℓ_p-norm or even combinations of norms by imposing bounds on the ℓ_2 and ℓ_∞ norms for example.

Definition 5.1 (Short Integer Solution)

Let d, m, q be three positive integers, and β a positive real. The *Short Integer Solution* problem $\mathrm{SIS}_{d,m,q,\beta}$ asks to find a vector $\mathbf{x} \in \mathbb{Z}^m$ such that $\mathbf{A}\mathbf{x} = \mathbf{0} \bmod q\mathbb{Z}$ and $0 < \|\mathbf{x}\|_2 \leq \beta$, given a uniformly random matrix $\mathbf{A}$ in $\mathbb{Z}_q^{d\times m}$.

For an adversary $\mathcal{A}$, we define its advantage in solving $\mathrm{SIS}_{d,m,q,\beta}$ by

$$\mathrm{Adv}_{\mathrm{SIS}}[\mathcal{A}] = \mathbb{P}_{\mathbf{A}\sim U(\mathbb{Z}_q^{d\times m}),\mathcal{A}}[\mathbf{A}\cdot\mathcal{A}(\mathbf{A}) = \mathbf{0} \bmod q\mathbb{Z} \wedge 0 < \|\mathcal{A}(\mathbf{A})\| \leq \beta].$$

When the parameters are clear from the context, we define the hardness bound of $\mathrm{SIS}_{d,m,q,\beta}$ for a time complexity t by

$$\varepsilon_{\mathrm{SIS}}(t) = \sup_{\mathcal{A}\ t\text{-time}} \mathrm{Adv}_{\mathrm{SIS}}[\mathcal{A}].$$

The choice of the parameters d, m, q, β is crucial to obtain a problem that is hard enough, but that is not vacuously hard. Typically, one needs to set the parameters so that there exists such a solution, which is not guaranteed for every matrix $\mathbf{A}$ for any parameters. We give the following lemma yielding a sufficient condition on the parameters for the problem to admit a solution. This result is adapted from [MR07, Lem. 5.2].

Lemma 5.1 (Existence of SIS Solution)

Let d, m, q be three positive integers, and β a positive real. If $\beta \geq \sqrt{m}\cdot\lfloor q^{d/m}\rfloor$, then $\mathrm{SIS}_{d,m,q,\beta}$ admits at least one solution.

Proof (Lemma 5.1). The proof relies on the pidgeon-hole argument. Consider the domain $D = \{0, \ldots, \lfloor q^{d/m}\rfloor\}^m$. Let $\mathbf{A}$ be a matrix in $\mathbb{Z}_q^{d\times m}$ and consider the function $f_{\mathbf{A}} : \mathbf{x} \in D \mapsto \mathbf{A}\mathbf{x} \bmod q\mathbb{Z} \in \mathbb{Z}_q^d$. Since $|D| = (\lfloor q^{d/m}\rfloor + 1)^m > (q^{d/m})^m = q^d = \left|\mathbb{Z}_q^d\right|$, there must exist two distinct vectors $\mathbf{x}_1, \mathbf{x}_2$ in D such that $f_{\mathbf{A}}(\mathbf{x}_1) = f_{\mathbf{A}}(\mathbf{x}_2)$. Then, define $\mathbf{x} = \mathbf{x}_1 - \mathbf{x}_2$. By construction, it holds that $\mathbf{x} \neq \mathbf{0}$, and $\mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{x}_1 - \mathbf{A}\mathbf{x}_2 = \mathbf{0} \bmod q\mathbb{Z}$. Additionally, we have

$$\|\mathbf{x}\|_2 \leq \sqrt{m}\cdot\|\mathbf{x}\|_\infty \leq \sqrt{m}\cdot\lfloor q^{d/m}\rfloor \leq \beta,$$

which proves that $\mathbf{x}$ is a solution of $\mathrm{SIS}_{d,m,q,\beta}$. Hence, $\mathrm{SIS}_{d,m,q,\beta}$ admits a solution.

We note that the problem may be trivial if the norm constraints do not exclude vectors have coefficients of size q. Indeed, if the only norm constraint is $\beta \geq \|\mathbf{x}\|_2$ with $\beta \geq q$, then one could solve the problem with $\mathbf{x} = [q|0|\ldots|0]^T$. Hence, in cryptography, we always consider parameters for SIS such that $\beta < q$. Another method, which has become more and more usual, is to consider

[1] For example, we randomly generate a good basis (kept secret), and we publish its HNF (randomized) which represents the same lattice but which is a bad basis.

a second constraint in ℓ_∞ norm. More precisely, a solution must verify $\|\mathbf{x}\|_2 \leq \beta$ and $\|\mathbf{x}\|_\infty \leq \beta_\infty$. By choosing $\beta_\infty < q$, we could then allow for $\beta \geq q$ without necessarily making the problem easy. However, the concrete hardness of this variant is not yet well studied.

Another observation is that $\mathbf{A}$ defines a linear function. We can then efficiently find a non-zero vector that satisfy $\mathbf{Ax} = \mathbf{0} \bmod q\mathbb{Z}$ using the Hermite Normal Form algorithm (which is the generalization of the Gauss elimination algorithm for integer matrices). However, it is not guaranteed that $\mathbf{x}$ is bounded by β. Therefore, the norm constraint is paramount to hope for a hard problem.

Inhomogeneous Variant

We can also define an inhomogeneous version of the problem where instead of asking for a vector in the q-periodic kernel of $\mathbf{A}$, we ask for a preimage $\mathbf{x}$ of a random syndrome $\mathbf{y}$ by $\mathbf{A}$ that is short. Again, without a norm constraint, this problem would be easy by just solving an inhomogeneous system of linear equations.

Definition 5.2 (Inhomogeneous Short Integer Solution)

Let d, m, q be three positive integers, and β a positive real. The *Inhomogeneous Short Integer Solution* problem $\mathrm{ISIS}_{d,m,q,\beta}$ asks to find a vector $\mathbf{x} \in \mathbb{Z}^m$ such that $\mathbf{Ax} = \mathbf{y} \bmod q\mathbb{Z}$ and $\|\mathbf{x}\|_2 \leq \beta$, given a uniformly random matrix $\mathbf{A}$ in $\mathbb{Z}_q^{d\times m}$ and a uniform syndrome $\mathbf{y} \in \mathbb{Z}_q^d$.

Notice that we no longer constrain the vector $\mathbf{x}$ to be non zero. Indeed, if $\mathbf{x} = \mathbf{0}$ and $\mathbf{y} \neq \mathbf{0}$, then $\mathbf{x}$ is not a solution anyway. Also, we observe that if $\beta \geq q$, the ISIS problem is not necessarily trivial as SIS was. The argument we put forward was that q is non zero in $\mathbb{Z}$ but zero in $\mathbb{Z}_q$. We thus had trivial solutions $q\mathbf{e}_i$. Here, we do not ask for a short preimage of $\mathbf{0}$, which completely changes the context.

Nevertheless, we can identify conditions for which the problem is still easy. For that, assume we fix the representatives of equivalence classes of $\mathbb{Z}_q$ to be the integers in $Z_q = \mathbb{Z} \cap [-q/2, q/2)$. If $\beta \geq q\sqrt{m}/2$, the $\mathrm{ISIS}_{d,m,q,\beta}$ is trivially easy. Indeed, we can easily find $\mathbf{x} \in \mathbb{Z}_q^m$ such that $\mathbf{Ax} = \mathbf{y} \bmod q\mathbb{Z}$ (by solving the linear system without norm constraint). We consider $\mathbf{x}'$ to be the unique representative of $\mathbf{x}$ in Z_q^m. Then, $\mathbf{Ax}' = \mathbf{y} \bmod q\mathbb{Z}$, and $\|\mathbf{x}'\|_2 \leq q/2 \cdot \sqrt{m} \leq \beta$. Hence, we must have $\beta < q\sqrt{m}/2$.

We remark that $q\sqrt{m}/2$ is the maximal norm of a vector in $\mathbb{Z}_q^m$. It is possible that the found $\mathbf{x}'$ is smaller. In general, if $\mathbf{A}$ and $\mathbf{y}$ are chosen uniformly, a solution $\mathbf{x}'$ found by solving the linear system without constraint will be close to a uniform vector of $\mathbb{Z}_q^m$. Yet, the norm of a uniform vector is concentrated around $q\sqrt{m/12}$. So this naive method would solve ISIS as soon as $\beta \gtrsim q\sqrt{m/12}$. We can go even further by observing that the system defined by $\mathbf{A}$ is underdetermined due to the fact that $m \geq d$.If we assume $\mathbf{A}$ contains an invertible submatrix (let us assume it corresponds to the first d columns without loss of generality), we can choose $\mathbf{x}'_{:d}$ as $\mathbf{A}_{:d}^{-1}\mathbf{y} \bmod q\mathbb{Z}$ (where the inverse is computed in $\mathbb{Z}_q$) and define the rest of the entries of $\mathbf{x}'$ to be 0. As a result, $\mathbf{x}'$ would have a norm close to $q\sqrt{d/12}$.

To summarize, the problem is easy when $\beta \gtrsim q\sqrt{d/12}$. Recent works have shown that it is possible to solve it for slightly smaller bounds as well, but it is beyond the scope of this course.

5.1.2 Hardness of Short Integer Solution

We now provide one of the several result providing a worst-case to average-case reduction from lattice problems to SIS. In particular, the main reductions are given in [Ajt96, Ajt98, MR07, GPV08]. Here we provide the adapted statement and proof sketch of the reduction by Gentry et al. [GPV08].

Theorem 5.1 (Worst-Case Hardness of SIS)

Let d be a positive integers. There exists a polynomial-time worst-case to average-case reduction from SIVP_γ to $\mathrm{SIS}_{d,m,q,\beta}$ for all m, q, β, γ such that $\gamma \geq 2\beta\sqrt{d}$, $q \geq 2\beta\sqrt{d}$ and $m, \log_2 q \leq \mathsf{poly}(d)$.

Proof (Theorem 5.1). We only give a sketch proof which is simplified when $q = 2\beta\sqrt{d}$. We also assume that $\gamma \geq q$ to simplify the reduction, but this condition is not necessary (if $q > \gamma$, we later choose $s = q\|\mathbf{S}\|_\infty / \gamma$). For the full proof, we refer to [GPV08].

We assume the existence of an oracle $\mathcal{A}$ which solves $\mathrm{SIS}_{d,m,q,\beta}$ and aim at solving any instance of SIVP_γ. We thus consider an instance given by a lattice $\mathcal{L}$, and more precisely by a basis $\mathbf{B}$ of $\mathcal{L}$. For the reduction, we use a set (seen as a matrix) $\mathbf{S}$ of linearly independent vectors of the lattice, which is initialized to $\mathbf{S} = \mathbf{B}$. We choose a parameter $s = \|\mathbf{S}\|_\infty \geq q\eta_\varepsilon(\mathcal{L})$. Note that $\|\mathbf{S}\| = \max_{i\in[\![1,d]\!]}\|\mathbf{S}\mathbf{e}_i\|_2 \geq \|\mathbf{S}^*\|_\infty \geq \eta_\varepsilon(\mathcal{L})$, and if $\mathbf{S}$ is not a solution to SIVP_γ, then $\|\mathbf{S}\| \geq \gamma\lambda_d(\mathcal{L})$. The reduction proceeds as follows.

Sampling A : For i from 1 to m, sample $\mathbf{y}_i$ from $\mathcal{D}_{\mathcal{L},s}$ using $\mathbf{S}$ and set $\mathbf{a}_i = \mathbf{B}^{-1}(\mathbf{y}_i \bmod q\mathcal{L}) \bmod q\mathbb{Z} \in \mathbb{Z}_q^d$. We note here that $\mathbf{B}^{-1}$ simply allows us to get the vector of integer coefficients of a lattice point. We have seen that by Lemma 4.5 it holds $\Delta(\mathcal{D}_{\mathcal{L},s,\mathbf{c}} \bmod \mathcal{L}', U(\mathcal{L} \bmod \mathcal{L}')) \leq 2\varepsilon$ if $s \geq \eta_\varepsilon(\mathcal{L}')$. Since we have $s \geq q\eta_\varepsilon(\mathcal{L})$, we have $\Delta(\mathcal{D}_{\mathcal{L},s} \bmod q\mathcal{L}, U(\mathcal{L} \bmod q\mathcal{L})) \leq 2\varepsilon$, which implies that $\mathbf{a}_i$ is statistically close to the uniform over $\mathbb{Z}_q^d$. Additionally, since the $\mathbf{y}_i$ are independent, so are the $\mathbf{a}_i$. We can thus use the SIS oracle.

Solving SIS for A: We define $\mathbf{A} = [\mathbf{a}_i]_{i\in[\![1,m]\!]}$ and use $\mathcal{A}$ to solve $\mathrm{SIS}_{d,m,q,\beta}$ and get $\mathbf{x} \in \mathbb{Z}^m$ such that $\mathbf{A}\mathbf{x} = \mathbf{0} \bmod q\mathbb{Z}$ and $0 < \|\mathbf{x}\|_2 \leq \beta$.

Combining elements: We define $\mathbf{Y}$ the matrix of $\mathcal{L} \bmod q\mathcal{L}$ by $\mathbf{Y} = [\mathbf{y}_i \bmod q\mathcal{L}]_{i\in[\![1,m]\!]}$. We define $\mathbf{x}^* = \frac{1}{q}\mathbf{Y}\mathbf{x} \in \mathcal{L}$. We indeed want to show that $\mathbf{x}^*$ is in $\mathcal{L}$. It suffices to show that $\mathbf{B}^{-1}\mathbf{Y}\mathbf{x}$ is in $q\mathbb{Z}^d$ (equivalent to $\mathbf{Y}\mathbf{x} \in q\mathcal{L}$). By definition, we have $\mathbf{B}^{-1}\mathbf{Y} = \mathbf{A} \bmod q\mathbb{Z}$ so $\mathbf{B}^{-1}\mathbf{Y}\mathbf{x} = \mathbf{A}\mathbf{x} \bmod q\mathbb{Z} = \mathbf{0} \bmod q\mathbb{Z}$. So $\mathbf{B}^{-1}\mathbf{Y}\mathbf{x} \in q\mathbb{Z}^d$. We then have $\|\mathbf{Y}\mathbf{x}\|_2 \leq s\beta\sqrt{d}$ so $\|\mathbf{x}\|_2 \leq s/2$ which is shorter.

To solve SIVP_γ, we repeat this procedure to obtain a new set $\mathbf{S}'$ such that $\|\mathbf{S}'\|_\infty \leq \|\mathbf{S}\|_\infty / 2$. By repeating the procedure, we can reduce the size until we have a set of linearly independent vectors of norm at most $\gamma\lambda_d(\mathcal{L})$. The reduction works also for the ISIS problem by adding a well-chosen center to each of the $\mathbf{y}_i$.

5.1.3 Application: Ajtai Hash Function

Although it is less efficient than symmetric hash functions such as SHA-2/SHA-3, it is possible to construct hash functions whose security relies on the hardness of lattice problems. Typically, Ajtai [Ajt96] constructed a hash function whose collision resistance relies on the hardness of SIS, and that is pseudorandom (thus behaving as a random oracle) under strong parameter constraints. We now describe the algorithms KeyGen and $\mathcal{H}$ in Algorithms 5.1 and 5.2

Algorithm 5.1: KeyGen (Ajtai Hash Function)

Input: Integers d, m, q

1. $\mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{d\times m})$

Output: $k = \mathbf{A}$

Algorithm 5.2: $\mathcal{H}_\mathbf{A}$ (Ajtai Hash Function)

Input: Vector $\mathbf{x} \in D \subset \mathbb{Z}^m$

1. $\mathbf{y} \hookleftarrow \mathbf{A}\mathbf{x} \bmod q\mathbb{Z}$

Output: $\mathbf{y}$

We now give a statistical result due to Dodis et al. [DORS08, Lem. 2.1] which states that when q is prime and the input $\mathbf{x}$ has sufficient entropy, then $\mathcal{H}_\mathbf{A}$ is pseudorandom.

Lemma 5.2 (Leftover Hash Lemma)

Let d, m, q be positive integers with q prime. Let $\mathcal{X}$ be a distribution on D. Then, it holds

that

$$\Delta((\mathbf{A}, \mathcal{H}_{\mathbf{A}}(\mathbf{x})), (\mathbf{A}, \mathbf{y})) \leq \frac{1}{2}\sqrt{q^d \cdot 2^{-H_\infty(\mathcal{X})}},$$

where $\mathbf{A} \sim U(\mathbb{Z}_q^{d\times m})$, $\mathbf{x} \sim \mathcal{X}$, and $\mathbf{y} \sim U(\mathbb{Z}_q^d)$. In particular, if $|D| = B^m$ and $\mathcal{X} = U(D)$, then the statistical distance is $\frac{1}{2}2^{-\frac{1}{2}\cdot(m\log_2 B - d\log_2 q)}$. So if $m\log_2 B \geq d\log_2 q + 2(\lambda - 1)$, the statistical distance is bounded by $2^{-\lambda}$.

The leftover hash lemma (LHL) is an essential tool in cryptography. We have stated it in the case of Ajtai's hash function but it is much more general and can be used in various contexts. Here, it proves that $\mathcal{H}_{\mathbf{A}}$ is $(\infty, 2^{-\lambda})$-pseudorandom under the parameter constraints of Lemma 5.2. This means that $\mathcal{H}_{\mathbf{A}}$ is $2^{-\lambda}$-close to a behaving as a random oracle. We note that this lemma is statistical and there are no computational assumption underlying it. We now look at the collision resistance of $\mathcal{H}_{\mathbf{A}}$.

Lemma 5.3 (Collision Resistance of Ajtai Hash)

Let d, m, q, B be positive integers, and $D = \{0, \dots, B\}^m$. We then define $\beta = B\sqrt{m}$. Then, the Ajtai Hash Function is $(t, \varepsilon_{\mathrm{SIS}}(t + O(m)))$-collision resistant, where $\varepsilon_{\mathrm{SIS}}(t') = \sup_{\mathcal{A}\ t'\text{-time}} \mathrm{Adv}_{\mathrm{SIS}}[\mathcal{A}]$ is the hardness bound of $\mathrm{SIS}_{d,m,q,\beta}$.

Proof (Lemma 5.3). Let $\mathbf{A}$ be sampled uniformly in $U(\mathbb{Z}_q^{d\times m})$, and let $\mathcal{A}$ be an adversary against the collision resistance of $\mathcal{H}_{\mathbf{A}}$ running in time at most t. We denote by ε its probability of succeeding and we want to bound ε. We now construct $\mathcal{B}$ attacking $\mathrm{SIS}_{d,m,q,\beta}$ in time at most $t' = t + O(m)$. $\mathcal{B}$ calls $\mathcal{A}$ on $\mathbf{A}$ and gets a collision $(\mathbf{x}, \mathbf{x}') \in D^2$ such that $\mathbf{x} \neq \mathbf{x}'$ and $\mathcal{H}_{\mathbf{A}}(\mathbf{x}) = \mathcal{H}_{\mathbf{A}}(\mathbf{x}')$ with probability ε. It then computes $\mathbf{x}^* = \mathbf{x} - \mathbf{x}'$ and returns it as a solution of $\mathrm{SIS}_{d,m,q,\beta}$.

Firstly, $\mathcal{B}$ indeed runs in time at most $t + O(m)$ as it makes one call to $\mathcal{A}$ which takes time at most t and computes $\mathbf{x}^*$ which takes time at most $O(m)$. Now we verify that if $(\mathbf{x}, \mathbf{x}')$ breaks the collision resistance, then $\mathbf{x}^*$ is indeed a SIS solution. To see it, we first have that $\mathbf{A}\mathbf{x}^* = \mathcal{H}_{\mathbf{A}}(\mathbf{x}) - \mathcal{H}_{\mathbf{A}}(\mathbf{x}') = \mathbf{0} \bmod q\mathbb{Z}$. Additionally, it holds that $\mathbf{x}^* \neq 0$ as $\mathbf{x} \neq \mathbf{x}'$. Finally, we have that $\mathbf{x}^*$ is in $\{-B, \dots, B\}^m$. Thence

$$\|\mathbf{x}^*\|_2 \leq \|\mathbf{x}^*\|_\infty \sqrt{m} \leq B\sqrt{m} = \beta.$$

This proves the inclusion of events that we need to justify the following inequalities

$$\begin{aligned}
\varepsilon &= \mathbb{P}_{\mathbf{A}\sim U(\mathbb{Z}_q^{d\times m}),(\mathbf{x},\mathbf{x}')\leftarrow\mathcal{A}(\mathbf{A})}[\mathbf{x} \neq \mathbf{x}' \wedge \mathcal{H}_{\mathbf{A}}(\mathbf{x}) = \mathcal{H}_{\mathbf{A}}(\mathbf{x}')] \\
&\leq \mathbb{P}_{\mathbf{A}\sim U(\mathbb{Z}_q^{d\times m})}[\mathbf{A}\cdot\mathcal{B}(\mathbf{A}) = \mathbf{0} \bmod q\mathbb{Z} \wedge 0 < \left\|\mathcal{B}(\mathbf{A})\right\|_p \leq \beta] \\
&= \mathrm{Adv}_{\mathrm{SIS}}[\mathcal{B}] \\
&\leq \sup_{\mathcal{B}'\ t+O(m)\text{-time}} \mathrm{Adv}[\mathcal{B}'] \\
&= \varepsilon_{\mathrm{SIS}}(t + O(m)),
\end{aligned}$$

as claimed.

5.2 Learning With Errors

We now introduce a second fundamental problem called *Learning With Errors* (LWE). It was introduced by Regev in 2005 [Reg05] and was proven as versatile as the Short Integer Solution problem. In particular, the Learning With Errors problem benefits from worst-case to average-case reductions from hard lattice problems such as variants of the SVP problem like SVP_γ or GapSVP_γ. In the seminal paper of Regev, a simple bit encryption scheme is proposed and proven secure under the Learning With Errors assumption. We present this encryption scheme in Chapter 6. This construction, and the assumption more generally, gave rise to many other lattice-based cryptographic constructions. A famous example is the first realization of Fully-Homomorphic Encryption by Gentry [Gen09]. One of the perks of the LWE problem over the SIS problem is that it offers a decisional assumption. We start by introducing the problem and then establish results

on its hardness and on the equivalence between the search and decision variants.

5.2.1 Problem Definitions

Just like the Short Integer Solution problem, the purpose of the Learning With Errors problem is to provide a more flexible assumption to design cryptography upon. Nevertheless, it can still be interpreted as a lattice problem on the random lattices we introduced in Definition 2.3. More precisely, we consider a matrix $\mathbf{A}$ in $\mathbb{Z}_q^{m\times d}$ for positive integers d, m and q such that $m \geq d$. Notice that here the dimension m and d for the matrix are swapped compared to the case of SIS, which can be interpreted as a simple transpose of the matrix $\mathbf{A}$. Regardless, one can define the following lattice

$$\mathcal{L}_q(\mathbf{A}) = \{\mathbf{y} \in \mathbb{Z}^m : \exists \mathbf{s} \in \mathbb{Z}_q^d, \mathbf{A}\mathbf{s} = \mathbf{y} \bmod q\mathbb{Z}\}.$$

One can consider the CVP_γ or BDD_γ problems on the lattice $\mathcal{L}_q(\mathbf{A})$ which corresponds to finding the closest lattice point $\mathbf{A}\mathbf{s}$ to a target vector $\mathbf{t} \in \mathbb{R}^m$. By writing $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}$ with $\|\mathbf{e}\|_2 \leq \lambda_1(\mathcal{L}_q(\mathbf{A}))/2$, the problem becomes finding $\mathbf{e}$ (or $\mathbf{s}$) given $\mathbf{A}$ and $\mathbf{t}$.

In general, the LWE problem is not presented as a CVP instance but rather as a linear algebraic problem. Given $\mathbf{A}$ and $\mathbf{A}\mathbf{s} + \mathbf{e} \bmod q\mathbb{Z}$, the problem asks to solve this noisy linear system and recover $\mathbf{s}$. We note here that the problem is equivalent to finding $\mathbf{e}$ as it leads to knowing $\mathbf{A}\mathbf{s} \bmod q\mathbb{Z}$ which can then be solved using regular linear algebra, because $\mathbf{A}$ has more rows than columns. In particular, it means that the error $\mathbf{e}$ is crucial in making the problem hard. Without error, the problem would be solved using Gauss elimination. The decision variant consists in deciding whether the given vector $\mathbf{t}$ was indeed generated as $\mathbf{A}\mathbf{s} + \mathbf{e} \bmod q\mathbb{Z}$ for some secret $\mathbf{s}$ and error $\mathbf{e}$ or if it was sampled completely at random.

We now give the formal definition of LWE. We start by defining the LWE distribution. The distribution has a secret vector $\mathbf{s}$ hardcoded as well as an error distribution, and outputs what we call LWE samples. We then define the search and decision variants of Learning With Errors based on the latter distribution.

Definition 5.3 (Learning With Errors Distribution)

Let d, q be positive integers. Let $\mathbf{s}$ be in $\mathbb{Z}_q^d$, and $\mathscr{D}_e$ be a distribution over $\mathbb{Z}$. The LWE distribution denoted by $A_{\mathbf{s},\mathscr{D}_e}$ is defined by the following random process: sample $\mathbf{a} \hookleftarrow U(\mathbb{Z}_q^d)$, and $e \hookleftarrow \mathscr{D}_e$, and output $(\mathbf{a}, \mathbf{a}^T\mathbf{s} + e \bmod q\mathbb{Z})$.

A sample from this distribution is therefore one equation of the noisy linear system. Although it is common to limit the number of equations to m, and thus write the problem in matrix form $\mathbf{A}\mathbf{s} + \mathbf{e}$, several more theoretical works do not fix a priori the number of equations. We present here the two definitions for completeness.

Definition 5.4 ((Search) Learning With Errors)

Let d, q be positive integers. Let $\mathscr{D}_s$ be a secret distribution over $\mathbb{Z}_q^d$, and $\mathscr{D}_e$ be an error distribution over $\mathbb{Z}$. The *search Learning With Errors* problem $\mathrm{sLWE}_{d,q,\mathscr{D}_s,\mathscr{D}_e}$ is as follows. Let $\mathbf{s}$ be drawn from $\mathscr{D}_s$. Given arbitrarily many samples $(\mathbf{a}_i, \mathbf{a}_i^T\mathbf{s} + e_i \bmod q\mathbb{Z})$ drawn from $A_{\mathbf{s},\mathscr{D}_e}$, find $\mathbf{s}$.
When the number of available samples is limited to m, we write the problem as $\mathrm{sLWE}_{d,q,m,\mathscr{D}_s,\mathscr{D}_e}$, and we present it in matrix form as follows. Given $\mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{m\times d})$ and $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q\mathbb{Z}$ for some $\mathbf{s} \hookleftarrow \mathscr{D}_s$ and $\mathbf{e} \hookleftarrow \mathscr{D}_e^m$, find $\mathbf{s}$.

For an adversary $\mathcal{A}$, we define its advantage in solving $\mathrm{sLWE}_{d,q,\mathscr{D}_s,\mathscr{D}_e}$ as

$$\mathrm{Adv}_{\mathrm{sLWE}}[\mathcal{A}] = \mathbb{P}_{(\mathbf{a}_i,t_i)_i\sim A_{\mathbf{s},\mathscr{D}_e}}[\mathcal{A}((\mathbf{a}_i,t_i)_i) = \mathbf{s}].$$

When the maximal number of samples is fixed to m, we define it as

$$\mathrm{Adv}_{\mathrm{sLWE}}[\mathcal{A}] = \mathbb{P}_{\substack{\mathbf{A}\sim U(\mathbb{Z}_q^{d\times m})\\ \mathbf{s}\sim\mathscr{D}_s\\ \mathbf{e}\sim\mathscr{D}_e^m}}[\mathcal{A}(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q\mathbb{Z}) = \mathbf{s}].$$

When the parameters are not ambiguous, we define the hardness bound of $\mathrm{sLWE}_{d,q,\mathscr{D}_s,\mathscr{D}_s}$ and $\mathrm{sLWE}_{d,q,m,\mathscr{D}_s,\mathscr{D}_e}$ for a time complexity t as

$$\varepsilon_{\mathrm{sLWE}}(t) = \sup_{\mathcal{A}\ t\text{-time}} \mathrm{Adv}_{\mathrm{sLWE}}[\mathcal{A}].$$

Definition 5.5 ((Decision) Learning With Errors)

Let d, q be positive integers. Let $\mathscr{D}_s$ be a secret distribution over $\mathbb{Z}_q^d$, and $\mathscr{D}_e$ be an error distribution over $\mathbb{Z}$. The *decision Learning With Errors* problem $\mathrm{LWE}_{d,q,\mathscr{D}_s,\mathscr{D}_e}$ is as follows. Let $\mathbf{s}$ be drawn from $\mathscr{D}_s$, and let $\mathscr{D} \in \{A_{\mathbf{s},\mathscr{D}_e}, U(\mathbb{Z}_q^d \times \mathbb{Z}_q)\}$. Given arbitrarily many samples from $\mathscr{D}$, decide whether $\mathscr{D} = A_{\mathbf{s},\mathscr{D}_e}$ or if $\mathscr{D} = U(\mathbb{Z}_q^d \times \mathbb{Z}_q)$.
When the number of available samples is limited to m, we write the problem as $\mathrm{LWE}_{d,q,m,\mathscr{D}_s,\mathscr{D}_e}$, and we present it in matrix form as follows. Given $\mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{m\times d})$ and $\mathbf{t} \in \mathbb{Z}_q^m$, decide whether $\mathbf{t} = \mathbf{A}\mathbf{s}+\mathbf{e} \bmod q\mathbb{Z}$ for some $\mathbf{s} \hookleftarrow \mathscr{D}_s$ and $\mathbf{e} \hookleftarrow \mathscr{D}_e^m$, or if $\mathbf{t} \hookleftarrow U(\mathbb{Z}_q^m)$.

For an adversary $\mathcal{A}$, we define its advantage in solving $\mathrm{LWE}_{d,q,\mathscr{D}_s,\mathscr{D}_e}$ as

$$\mathrm{Adv}_{\mathrm{LWE}}[\mathcal{A}] = \left|\mathbb{P}_{(\mathbf{a}_i,t_i)_i \sim A_{\mathbf{s},\mathscr{D}_e}}[\mathcal{A}((\mathbf{a}_i,t_i)_i) = 1] - \mathbb{P}_{(\mathbf{a}_i,t_i)_i \sim U(\mathbb{Z}_q^{d+1})}[\mathcal{A}((\mathbf{a}_i,t_i)_i) = 1]\right|.$$

When the maximal number of samples is fixed to m, we define it as

$$\mathrm{Adv}_{\mathrm{LWE}}[\mathcal{A}] = \left|\mathbb{P}_{\substack{\mathbf{A}\sim U(\mathbb{Z}_q^{d\times m}) \\ \mathbf{s}\sim\mathscr{D}_s \\ \mathbf{e}\sim\mathscr{D}_e^m}}[\mathcal{A}(\mathbf{A}, \mathbf{A}\mathbf{s}+\mathbf{e} \bmod q\mathbb{Z}) = 1] - \mathbb{P}_{\substack{\mathbf{A}\sim U(\mathbb{Z}_q^{d\times m}) \\ \mathbf{t}\sim U(\mathbb{Z}_q^m)}}[\mathcal{A}(\mathbf{A},\mathbf{t}) = 1]\right|.$$

When the parameters are not ambiguous, we define the hardness bound of $\mathrm{LWE}_{d,q,\mathscr{D}_s,\mathscr{D}_s}$ and $\mathrm{LWE}_{d,q,m,\mathscr{D}_s,\mathscr{D}_e}$ for a time complexity t as

$$\varepsilon_{\mathrm{LWE}}(t) = \sup_{\mathcal{A}\ t\text{-time}} \mathrm{Adv}_{\mathrm{LWE}}[\mathcal{A}].$$

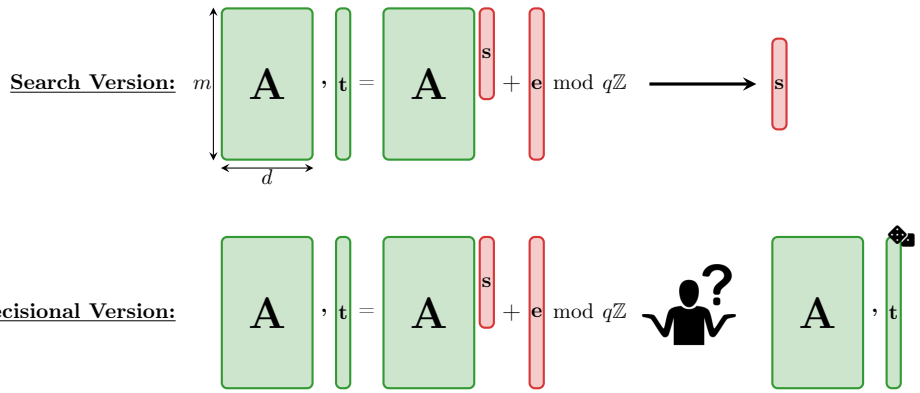


Figure 5.1: Version Calculatoire et Décisionnelle du problème *Learning With Errors*

Again, just like SIS, the hardness of LWE highly depends on the choice of the parameters d, q, m and the secret and error distributions $\mathscr{D}_s, \mathscr{D}_e$. Typically, if the error distribution returns 0 with probability 1 or negligibly close to 1, then the problem becomes vacuously easy. On the contrary, if $\mathscr{D}_e = U(\mathbb{Z}_q)$, then the decision variant becomes ill-defined while the search variant becomes statistically hard. Indeed, if $\mathbf{e}$ is uniform and independent of $\mathbf{A}\mathbf{s}$, then $\mathbf{A}\mathbf{s}+\mathbf{e} \bmod q\mathbb{Z}$ is perfectly uniform. There is then no distinction between the two distributions of the decision problem.

The original formulation by Regev [Reg05] chooses d as the dimension which drives the hardness, q as a prime number, and $\mathscr{D}_s = U(\mathbb{Z}_q^d)$ and $\mathscr{D}_e = \mathcal{D}_{\mathbb{Z},\alpha q}$ for a relative error rate $\alpha \in (1/q, 1)$. In this parameter regime, LWE can be proven as hard as standard lattice problems in the worst-case. For the original formulation, we change the subscript to $\mathrm{LWE}_{d,q,\alpha}$ for short.

There are however many other regimes that offer interesting perspectives such as $\mathscr{D}_s = U(\{0,1\}^d)$ and/or $\mathscr{D}_e = U(\{0,1\})$. They are much trickier to prove hard though. The class of problems for which $\mathscr{D}_s = \mathscr{D}_e^d$ are usually referred to as *Hermite Normal Form* Learning With Errors.

Remark 5.1 (Primal Attack and Unique SVP)

Although we introduced sLWE as a special CVP instance, the sLWE problem can be interpreted as a specific SVP instance instead. Let us consider the problem

$\text{sLWE}_{d,q,m,U(\{0,1\}^d),U(\{0,1\})}$, and a given instance $(\mathbf{A}, \mathbf{t})$ with $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q\mathbb{Z}$. The definition of $\mathbf{t}$ can be re-written as

$$[\mathbf{I}_m|\mathbf{A}|\mathbf{t}] \cdot \begin{bmatrix} \mathbf{e} \\ \mathbf{s} \\ -1 \end{bmatrix} = \mathbf{0} \bmod q\mathbb{Z}.$$

Defining $\mathbf{x} = [\mathbf{e}^T|\mathbf{s}^T|-1]^T$, it holds that $\mathbf{x}$ is a non-zero vector of norm at most $\sqrt{m+d+1}$ in the lattice $\mathcal{L}_q^\perp([\mathbf{I}_m|\mathbf{A}|\mathbf{t}])$. It can be shown that $\mathbf{x}$ actually verifies $\|\mathbf{x}\|_2 = \lambda_1(\mathcal{L}_q^\perp([\mathbf{I}_m|\mathbf{A}|\mathbf{t}]))$ and, as such, solving SVP on this lattice allows one to recover $\mathbf{s}$ (and $\mathbf{e}$). This solving method of sLWE is called the *primal attack*, which consists in interpreting the sLWE instance as an instance of *Unique*-SVP. The *Unique*-SVP problems corresponds to SVP where there is the extra assurance that there are only two non-zero vectors having the shortest norm.

We will see in TD 2 that under certain conditions on the parameters, there exists a reduction from the decisional version of LWE to SIS.

5.2.2 Computational-Decisional Equivalence

This is not often the case for cryptographic assumptions, but we can actually show that for well-chosen parameters the search and decision versions of LWE are equivalent. This result, also due to Regev [Reg05], proves that, up to a polynomial factor, the two problems are as hard as each other when $\mathscr{D}_s = U(\mathbb{Z}_q^d)$. We first show the most natural reduction from the decisional version to the search version. We prove it for the matrix form because it is the most usual in cryptography. Note however that these reductions also work for an unbounded number of LWE samples.

Lemma 5.4 (Decision to Search)

Let d, q, m be positive integers. Let $\mathscr{D}_s$ be a secret distribution over $\mathbb{Z}_q^d$, and $\mathscr{D}_e$ be an error distribution over $\mathbb{Z}$. There is a polynomial-time reduction from $\text{LWE}_{d,q,m,\mathscr{D}_s,\mathscr{D}_e}$ to $\text{sLWE}_{d,q,m,\mathscr{D}_s,\mathscr{D}_e}$.

Proof (Lemma 5.4). Let $\mathcal{A}$ be a polynomial-time adversary able to solve $\text{sLWE}_{d,q,m,\mathscr{D}_s,\mathscr{D}_e}$ with advantage ε. Let $(\mathbf{A}, \mathbf{t}) \in \mathbb{Z}_q^{m\times d} \times \mathbb{Z}_q^m$ be an instance of $\text{LWE}_{d,q,m,\mathscr{D}_s,\mathscr{D}_e}$. The reduction $\mathcal{B}$ simply calls $\mathcal{A}(\mathbf{A}, \mathbf{t})$. With probability ε, $\mathcal{A}$ returns a vector $\mathbf{s} \in \text{Supp}(\mathscr{D}_s)$ such that $\mathbf{t} - \mathbf{A}\mathbf{s} \bmod q\mathbb{Z}$ follows $\mathscr{D}_e^m$. If so, $\mathcal{B}$ returns 1 (LWE). On the other hand, if $\mathcal{A}$ fails (i.e., returns nothing) or if $\mathbf{s} \notin \text{Supp}(\mathscr{D}_s)$ or if $\mathbf{t} - \mathbf{A}\mathbf{s} \bmod q\mathbb{Z}$ does not follow $\mathscr{D}_e^m$, then $\mathcal{B}$ returns 0 (uniform). As there is only one call to $\mathcal{A}$, the reduction is indeed polynomial time. We also get

$$\text{Adv}_{\text{LWE}}[\mathcal{B}] \geq \text{Adv}_{\text{sLWE}}[\mathcal{A}].$$

Hence, if $\text{Adv}_{\text{sLWE}}[\mathcal{A}]$ is non-negligible, then $\mathcal{B}$ can solve LWE with non negligible advantage as well.

Lemma 5.5 (Search to Decision)

Let d, q, m be positive integers that are polynomial in the security parameter, with q prime. Let $\mathscr{D}_e$ be an error distribution over $\mathbb{Z}$. There is a polynomial-time reduction from $\text{sLWE}_{d,q,m,\mathscr{D}_s,\mathscr{D}_e}$ to $\text{LWE}_{d,q,m,\mathscr{D}_s,\mathscr{D}_e}$.

Proof (Lemma 5.5). Assume we have access to an oracle $\mathcal{O}$ able to solve the decision version $\text{LWE}_{d,q,m,\mathscr{D}_s,\mathscr{D}_e}$ with advantage ε. Let $(\mathbf{A}, \mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q\mathbb{Z})$ be an instance of $\text{sLWE}_{d,q,m,\mathscr{D}_s,\mathscr{D}_e}$. The reduction $\mathcal{B}$ works as follows.

$\mathcal{B}(\mathbf{A}, \mathbf{t})$:
0 **For** $i \in \{1, \dots, d\}$
1 **For** $s_i^\star \in \mathbb{Z}_q$
2 Sample $\mathbf{u} \hookleftarrow U(\mathbb{Z}_q^m)$
3 Construct $\mathbf{A}' \leftarrow \mathbf{A} + [\mathbf{0}|\dots|\mathbf{0}|\mathbf{u}|\mathbf{0}|\dots|\mathbf{0}]$ and $\mathbf{t}' \leftarrow \mathbf{t} + \mathbf{u} \cdot s_i^\star$
4 **If** $\mathcal{O}(\mathbf{A}', \mathbf{t}') = 1$, store $s_i^\star$ and go to the next i, **else** go to the next $s_i^\star$.
5 **Return** $\mathbf{s}^\star = [s_1^\star | \dots | s_d^\star]$.

The matrix $\mathbf{A}'$ is constructed by adding $\mathbf{u}$ at the i-th column of $\mathbf{A}$.

First, note that the procedure inside the two loops (steps 2 to 4) can be repeated k times to amplify the success probability according to the advantage of the oracle $\mathcal{O}$. Typically, if we assume ε is non negligible, we can take k of the order of $\lceil 1/\varepsilon \rceil$.

Let us now analyze the reduction. First, $\mathcal{B}$ calls $\mathcal{O}$ at most $d \cdot q$ times (or $d \cdot q \cdot k$ in case of amplification). Because d and q are polynomial, and $\mathcal{O}$ is polynomial time, $\mathcal{B}$ is indeed polynomial time itself.

The reduction tries all the possibilities coefficient per coefficient. Hence, instead of brute forcing the secret which would require $\left|\mathbb{Z}_q^d\right| = q^d$ tries, the oracle allows one to decompose the search coefficient-wise and only try qd possibilities. Consider the i-th iteration of the outer loop, and the guess $s_i^\star$ for the inner loop. First, because $\mathbf{A}$ is uniform and $\mathbf{u}$ is chosen uniformly and independently of $\mathbf{A}$, we get that $\mathbf{A}'$ is indeed uniform over $\mathbb{Z}_q^{m \times d}$. Let us now look at the distribution of $\mathbf{t}'$. We can rewrite $\mathbf{t}$ as $\mathbf{t} = \mathbf{A}'\mathbf{s} + \mathbf{e} - \mathbf{u} \cdot s_i \bmod q\mathbb{Z}$ where s_i is the i-th coefficient of $\mathbf{s}$. Thence, we have $\mathbf{t}' = \mathbf{A}'\mathbf{s} + \mathbf{e} + \mathbf{u}(s_i^\star - s_i) \bmod q\mathbb{Z}$. If $s_i^\star = s_i$, $\mathbf{t}' = \mathbf{A}'\mathbf{s} + \mathbf{e} \bmod q\mathbb{Z}$ which is of the form of LWE. As a result, the oracle must indeed return 1 in this case, indicating that the guess $s_i^\star$ is correct. If $s_i^\star = s_i$, we then have $(s_i^\star - s_i) \in \mathbb{Z}_q^\times$ because q is prime. Then, $\mathbf{u} \cdot (s_i^\star - s_i)$ is uniform in $\mathbb{Z}_q^m$ and independent of $\mathbf{A}'\mathbf{s} + \mathbf{e}$. We then have that $\mathbf{t}'$ is uniform in $\mathbb{Z}_q^m$. The oracle must then return 0 in that case, indicating the guess $s_i^\star$ is wrong.

Notice that we here look at the average-case version where $\mathbf{s}$ is drawn uniformly in $\mathbb{Z}_q^d$. We can prove the same reduction when $\mathbf{s}$ is arbitrarily chosen (worst-case). The idea is to re-randomize the secret and use the same reduction as for the average-case. For that, the reduction simply adds a first step consisting in sampling $\mathbf{s}' \hookleftarrow U(\mathbb{Z}_q^d)$ and computing $\mathbf{t}'' = \mathbf{t} + \mathbf{A}\mathbf{s}' \bmod q\mathbb{Z}$ before calling $\mathcal{B}(\mathbf{A}, \mathbf{t}'')$. The recovered secret is then $\mathbf{s} + \mathbf{s}'$ which is uniform in $\mathbb{Z}_q^d$ as in the first case described above. The reduction can then unblind the real secret by substracting $\mathbf{s}'$ (which is known by the reduction) and recover $\mathbf{s}$.

5.2.3 Hardness of Learning With Errors

As mentioned in Remark 5.1, one way to solve sLWE is by solving uSVP on a specific lattice defined by the sLWE instance. However, uSVP is an extremely hard problem, even with an approximation factor, as discussed in Chapter 3. There are other ways of approaching solving sLWE but they (mostly) all are linked to solving hard lattice problems. The best attacks typically using BKZ have a complexity of $\exp(O(d \log_2 q / \log_2^2 \alpha))$ for the original formulation $\mathrm{sLWE}_{d,q,\alpha}$. One of the explanation for such a high complexity is that there exists a worst-case to average-case reduction from GapSVP_γ to $\mathrm{sLWE}_{d,q,\alpha}$. More precisely, Regev proposed a quantum reduction in [Reg05] which was later made classical (with different parameters and proof techniques) by Brakerski et al. [BLP+13]. We summarize these results in the following theorem.

Theorem 5.2 (Worst-Case Hardness of LWE)

Let d, q be positive integers such that $q \geq 2$, and let $\alpha \in (0, 1)$ be such that $\alpha q \geq 2\sqrt{d}$. If q is prime and polynomial in d, there exists a polynomial-time quantum reduction from $\mathrm{GapSVP}_{d,\gamma}$ or $\mathrm{SIVP}_{d,\gamma}$ to $\mathrm{sLWE}_{d,q,\alpha}$ for $\gamma = \widetilde{O}(d/\alpha)$. For any q, there exists a polynomial-time classical reduction from $\mathrm{GapSVP}_{\Theta(\sqrt{d}),\gamma}$ to $\mathrm{sLWE}_{d,q,\alpha}$ where $\gamma = \widetilde{O}(d^2/\alpha)$.

In the quantum reduction, the dimension is preserved between the GapSVP and the sLWE problems. However, the de-quantization of the reduction weakens the starting assumption as the dimension is now lowered to around $\sqrt{d}$ in the GapSVP assumption, and the approximation factor is larger by a factor d. We now sketch the proof structure of Regev's reduction [Reg05]. The details of the reduction are out of scope for this course.

Regev's Proof Structure

Regev uses an intermediate problem called the *Discrete Gaussian Sampling* (DGS) problem. The latter is linked to the ability of sampling discrete Gaussian on arbitrary lattices with very small Gaussian parameters, as seen in Chapter 4. He then shows that DGS is at least as hard as SIVP_γ (or GapSVP_γ), and then shows a quantum iterative reduction from DGS to sLWE. Let us first present the DGS problem.

Definition 5.6 (Discrete Gaussian Sampling Problem)

Let k, d be positive integers, and ϕ be a function from the set of all k-dimensional rank-d lattices to $\mathbb{R}^{+*}$. The *Discrete Gaussian Sampling* problem $\mathrm{DGS}_{k,d,\phi}$ asks to return a sample from $\mathcal{D}_{\mathcal{L},s}$ given a lattice $\mathcal{L}$ of dimension k and rank d, and a positive real $s > \phi(\mathcal{L})$.

Note that the difficulty of DGS highly depends on the lattice basis (see Klein's algorithm and Theorem 4.2), and also on how small s can be. The smaller s is, the harder it gets.

Reduction from $\mathrm{SIVP}_{2\sqrt{d}\phi(\mathcal{L})}$ to DGS_ϕ if $\phi(\mathcal{L}) \geq \sqrt{2}\eta_\varepsilon(\mathcal{L})$. Given a lattice $\mathcal{L}$, we start be applying LLL (Algorithm 2.4) to obtain a set S of d linearly independent vectors of $\mathcal{L}$ that have norm at most $2^d\lambda_d(\mathcal{L})$. We call L_d the largest norm among those vectors. By construction, we have $\lambda_d(\mathcal{L}) \leq L_d \leq 2^d\lambda_d(\mathcal{L})$. Then, for $i \in \{0, \dots, 2d\}$, call d^2 times the DGS_ϕ oracle on the instance $(\mathcal{L}, r_i = L_d 2^{-i})$ to get a set S_i of d^2 vectors. This requires to check that $r_i > \phi(\mathcal{L})$ of course. For each S_i, select (the shortest) d linearly independent vectors, and then select the shortest set among all these sets.

Let us now briefly explain why this reduction works. First, if $\phi(\mathcal{L}) \geq L_d$, then S is already short enough as all the elements have norm at most $L_d \leq 2\sqrt{d}\phi(\mathcal{L})$. Otherwise, define $i = \lceil \log_2(L_d/\phi(\mathcal{L})) \rceil$. We now check that $i \in \{0, \dots, 2d\}$ and that $\phi(\mathcal{L}) \leq r_i < 2\phi(\mathcal{L})$. First, $L_d \geq \phi(\mathcal{L})$ by assumption, so $i \geq 0$. Also, we have

$$2^{2d}\phi(\mathcal{L}) \geq 2^{2d} \cdot 2\eta_\varepsilon(\mathcal{L}) \geq 2^{2d} \cdot 2 \cdot \lambda_d(\mathcal{L})/d > 2^d\lambda_d(\mathcal{L}) \geq L_d.$$

Then, $L_d/\phi(\mathcal{L}) \leq 2^{2d}$ which shows $i \leq 2d$. Finally, by definition of i and $\lceil \cdot \rceil$, we have $\log_2(L_d/\phi(\mathcal{L})) - 1 < i \leq \log_2(L_d/\phi(\mathcal{L}))$ which implies $\phi(\mathcal{L}) \leq L_d/2^i < 2\phi(\mathcal{L})$.

Then, because S_i contains d^2 vectors, with overwhelming probability, there exists d of such vectors that are linearly independent. Additionally, by Lemma 4.9, all these vectors have norm at most $r_i\sqrt{d} \leq 2\sqrt{d}\phi(\mathcal{L})$.

Reduction from DGS_ϕ to $\mathrm{sLWE}_{d,q,\alpha}$ for $\phi(\mathcal{L}) = 2\sqrt{d}\eta_\varepsilon(\mathcal{L})/\alpha$ with $\alpha q \geq 2\sqrt{d}$. Let $\mathcal{L}$ be a lattice and r be a real such that $r > 2\sqrt{d}\eta_\varepsilon(\mathcal{L})/\alpha$. For all i, we define $r_i = r(\alpha q/\sqrt{d})^i$. The reduction starts by producing d^c samples from $\mathcal{D}_{\mathcal{L},r_{3d}}$. This is possible because $r_{3d} > 2^{3d}r > 2^{2d}\lambda_d(\mathcal{L})$. Then we iterate to obtain samples with width r_i for i going from $3d$ to 0.

Classical. Given d^c samples from $\mathcal{D}_{\mathcal{L},r_i}$ and an oracle for $\mathrm{sLWE}_{d,q,\alpha}$, construct an oracle for $\mathrm{BDD}_{\mathcal{L}^*,\alpha q/r_i}$.

Quantum. Given the oracle for $\mathrm{BDD}_{\mathcal{L}^*,\alpha q/r_i}$, produce d^c samples from $\mathcal{D}_{\mathcal{L},r_{i-1}}$.

In the end, we have d^c samples from $\mathcal{D}_{\mathcal{L},r_0} = \mathcal{D}_{\mathcal{L},r}$ as desired.

A Word on Quantum Reductions

The proof we presented contains a quantum step. This means that if one is able to solve $\mathrm{sLWE}_{d,q,\alpha}$, then we can obtain a quantum algorithm for solving SIVP_γ. This is more restrictive than a purely classical reduction. As stated in Theorem 5.2, there also exists classical reduction. More precisely, Peikert showed in [Pei09] that when $q \geq 2^{d/2}$, there is a classical reduction from $\mathrm{GapSVP}_{\Theta(\sqrt{d}),\gamma}$ to $\mathrm{sLWE}_{d,q,\alpha}$. It is then completed by [BLP+13] with a reduction from $\mathrm{sLWE}_{d,q,\alpha}$ to $\mathrm{sLWE}_{d,q',\alpha'}$ for any q' (and where the noise α' is linked to α, q, and q'). The classical reduction states that if one is able to break sLWE classically (resp. quantumly), then one is also able to break $\mathrm{GapSVP}_{\Theta(\sqrt{d}),\gamma}$ classically (resp. quantumly) in the worst-case, which is a much tighter statement.

Hardness of Variants

This reduction only holds for the standard formulation of LWE, namely with uniform secret and discrete Gaussian error. Several works have also studied the hardness of LWE with other distributions. The most common result is that if $\mathrm{LWE}_{d,q,m+m',\mathscr{D}_s,\mathscr{D}_e}$ is hard and $m = \Omega(d^2)$, then

$\mathrm{LWE}_{d,q,m',\mathscr{D}_e^d,\mathscr{D}_e}$ is also hard [ACPS09]. The same holds true for the search variant. This means that the Hermite Normal Form is harder than its regular counterpart, at the expense of reducing the number of samples. We will leave this reduction for the exercise sessions.

Another popular line of results is the study of LWE with short distributions. Goldwasser et al. [GKPV10] were the first to provide a hardness result for LWE with $\mathscr{D}_s = U(\{0,1\}^d)$ and $\mathscr{D}_e = \mathcal{D}_{\mathbb{Z},\alpha q}$, but it was at the expense of a q that is super-polynomial in the dimension d. It was later improved by Brakerski et al. [BLP+13]. As for the error part, it was shown by Micciancio and Peikert [MP13] that one could prove the hardness of LWE with $\mathscr{D}_s = U(\mathbb{Z}_q^d)$ and $\mathscr{D}_e = U(I)$ for I a small interval of integers, e.g., $I = \{0, \ldots, \eta - 1\}$ for $\eta \ll q$, provided that the number of samples m was at most $d(1 + O(\log_\eta d))$. For example, for $\eta = 2$, this yield $m \geq d(1 + O(\log_2 d))$. In particular, whenever $m \geq \binom{d}{\eta}$, there are possible polynomial-time algebraic attacks.

RECAP

- The $\mathrm{SIS}_{d,m,q,\beta}$ problem asks to find $\mathbf{x} \in \mathbb{Z}^m$ such that $\mathbf{Ax} = \mathbf{0} \bmod q\mathbb{Z}$ and $0 < \|\mathbf{x}\|_2 \leq \beta$ given $\mathbf{A} \in \mathbb{Z}_q^{d \times m}$ uniform.
- The $\mathrm{ISIS}_{d,m,q,\beta}$ problem asks to find $\mathbf{x} \in \mathbb{Z}^m$ such that $\mathbf{Ax} = \mathbf{u} \bmod q\mathbb{Z}$ and $\|\mathbf{x}\|_2 \leq \beta$ given $\mathbf{A} \in \mathbb{Z}_q^{d \times m}$ and $\mathbf{u} \in \mathbb{Z}_q^d$ uniform.
- If $m \geq d \log_2 q + 2\lambda$, the leftover hash lemma gives $\Delta((\mathbf{A}, \mathbf{Ax} \bmod q\mathbb{Z}), (\mathbf{A}, \mathbf{u})) \leq 2^{-\lambda}$ where $\mathbf{A} \in \mathbb{Z}_q^{d \times m}$, $\mathbf{x} \in \{0,1\}^m$ and $\mathbf{u} \in \mathbb{Z}_q^d$ are uniformly sampled in their respective spaces.
- The $\mathrm{sLWE}_{d,q,m,\mathscr{D}_s,\mathscr{D}_e}$ problem asks to recover the secret $\mathbf{s}$, sampled from $\mathscr{D}_s$, given $\mathbf{A} \in \mathbb{Z}_q^{m \times d}$ uniform, and $\mathbf{t} = \mathbf{As} + \mathbf{e} \bmod q\mathbb{Z}$ where $\mathbf{e} \hookleftarrow \mathscr{D}_e^m$. The decisional version $\mathrm{LWE}_{d,q,m,\mathscr{D}_s,\mathscr{D}_e}$ asks to distinguish such a $\mathbf{t}$ from a perfectly uniform vector over $\mathbb{Z}_q^m$.
- The SIS, ISIS, sLWE, LWE problems enjoy worst-case to average-case reductions from SIVP_γ in dimension d. The reduction for sLWE/LWE is quantum but can be made classical by changing the parameters. The problems sLWE and LWE are equivalent under certain constraints if q is polynomial.

Bibliography

[ACPS09] B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems. In *CRYPTO*, 2009.

[Ajt96] M. Ajtai. Generating Hard Instances of Lattice Problems (Extended Abstract). In *STOC*, 1996.

[Ajt98] M. Ajtai. The Shortest Vector Problem in L_2 is *NP*-hard for Randomized Reductions (Extended Abstract). In *STOC*, 1998.

[BLP+13] Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé. Classical Hardness of Learning With Errors. In *STOC*, 2013.

[DORS08] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. D. Smith. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. *SIAM J. Comput.*, 2008.

[Gen09] C. Gentry. Fully Homomorphic Encryption using Ideal Lattices. In *STOC*, 2009.

[GKPV10] S. Goldwasser, Y. Tauman Kalai, C. Peikert, and V. Vaikuntanathan. Robustness of the Learning with Errors Assumption. In *ICS*, 2010.

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for Hard Lattices and New Cryptographic Constructions. In *STOC*, 2008.

[MP13] D. Micciancio and C. Peikert. Hardness of SIS and LWE with Small Parameters. In *CRYPTO*, 2013.

[MR07] D. Micciancio and O. Regev. Worst-Case to Average-Case Reductions Based on Gaussian Measures. *SIAM J. Comput.*, 2007.

[Pei09] C. Peikert. Public-key Cryptosystems from the Worst-case Shortest Vector Problem: Extended Abstract. In *STOC*, 2009.

[Reg05] O. Regev. On Lattices, Learning With Errors, Random Linear Codes, and Cryptography. In *STOC*, 2005.

Part IV

Constructions

In this part, we present the elementary cryptographic constructions on lattices. In particular, we look at the design of two public key encryption schemes, as well as the two lattice-based digital signature design paradigms.

6

Public-Key Encryption from LWE

The two problems we have introduced in Chapter 5, namely the Short Integer Solution (SIS) problem and the Learning With Errors (LWE) problem, are fundamental in designing lattice-based cryptosystems. The latter is most often used in the design of encryption schemes as we present in the present chapter. More precisely, we will see two designs of public-key encryption schemes whose semantic security (Definition 1.10) is proven under the assumption that LWE is a hard problem. As we have seen in Section 5.2.3, in some parameter regimes, this assumption is proven at least as hard as worst-case lattice problems. We present the Regev encryption scheme whose keys are pseudorandom under LWE and whose ciphertexts look somewhat like decision-ISIS instances. We then present the dual Regev encryption where the roles are essentially swapped.

Contents

6.1 Regev Encryption Scheme

The first public-key encryption scheme whose security relies on the hardness of the LWE problem was proposed by Regev [Reg05] and is called Regev encryption. The idea is to generate a key pair which is an sLWE instance $(\mathbf{A}, \mathbf{t})$ for the public key and the underlying secret $\mathbf{s}$ is the secret key used for decryption. The encryption and decryption then has similarities with the El Gamal encryption scheme over classical groups, but presents key differences due to the peculiarities of lattices. In particular, lattice problems usually rely on the hardness of finding short vectors. This is why we choose the error distribution of the LWE problem to output short vectors. This error however introduces potential decryption failures, which needs to be accounted for during the parameter setting.

6.1.1 Description

We now describe the Regev encryption scheme at high level before presenting it formally. We consider a public key composed of sLWE samples of the form $(\mathbf{a}_i, t_i = \mathbf{a}_i^T\mathbf{s} + e_i \bmod q\mathbb{Z})_i$ for the underlying secret key $\mathbf{s}$. To encrypt a plaintext $M \in \{0,1\}$, we choose a random bit-string $\mathbf{r} \in \{0,1\}^m$ which consists in selecting a random subset of the samples $(\mathbf{a}_i, t_i)_i$. We sum the samples of this subset and add $\frac{q}{2}M$ to the sum of the t_i. To decrypt, one uses $\mathbf{s}$ to substract the part of $\sum_{i\in S} t_i + \frac{q}{2}M$ that depends on $\mathbf{s}$, which then should be the message part perturbed by a small error. As the message encoding is chosen so that the difference of encoding of two distinct messages (here 0 and 1) is large $(q/2)$, the error stays sufficiently small to distinguish these two cases and thus decrypt correctly.

Let us now present the scheme in the three algorithms KeyGen, Enc, Dec from Algorithms 6.1, 6.2 and 6.3.

Algorithm 6.1: KeyGen (Regev)

Input: Integers d, q, m, real α, which are functions of the security parameter λ.

1. $\mathbf{s} \hookleftarrow U(\mathbb{Z}_q^d)$
2. $\mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{m \times d})$
3. $\mathbf{e} \hookleftarrow \mathcal{D}_{\mathbb{Z}^m, \alpha q}$
4. $\mathbf{t} \leftarrow \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q\mathbb{Z}$

Output: $\mathsf{pk} = (\mathbf{A}, \mathbf{t})$, $\mathsf{sk} = \mathbf{s}$

Algorithm 6.2: Enc (Regev)

Input: Public parameters, Public Key $\mathsf{pk} = (\mathbf{A}, \mathbf{t}) \in \mathbb{Z}_q^{m \times (d+1)}$, Message $M \in \{0, 1\}$.

1. $\mathbf{r} \hookleftarrow U(\{0,1\}^m)$
2. $(\mathbf{c}_1, c_2) \leftarrow (\mathbf{A}^T\mathbf{r} \bmod q\mathbb{Z}, \mathbf{r}^T\mathbf{t} + \lfloor q/2 \rceil M \bmod q\mathbb{Z}) \in \mathbb{Z}_q^d \times \mathbb{Z}_q$

Output: $(\mathbf{c}_1, c_2)$

Algorithm 6.3: Dec (Regev)

Input: Public parameters, Secret Key $\mathsf{sk} \in \mathbb{Z}_q^d$, Ciphertext $(\mathbf{c}_1, c_2) \in \mathbb{Z}_q^{d+1}$.

1. $u \leftarrow c_2 - \mathbf{c}_1^T\mathbf{s} \bmod q\mathbb{Z}$
2. **If** $|u - 0| < |u \pm \lfloor q/2 \rceil|$, then $M \leftarrow 0$, **else** $M \leftarrow 1$.

Output: M

The last check relies on the fact that u is the representative of the equivalence class $c_2 - \mathbf{c}_1^T\mathbf{s} \bmod q\mathbb{Z}$ that lies in $(-q/2, q/2]$. We note that the scheme here can only be used to encrypt one bit at a time. One can iterate over a bit-string $\mathbf{m}$ to encrypt arbitrary long bit-string, but this would result in quite large ciphertexts. Another possibility is to use a matrix variant of LWE and design the keys to be $(\mathbf{A}, \mathbf{T} = \mathbf{A}\mathbf{S} + \mathbf{E} \bmod q\mathbb{Z})$ where $\mathbf{S}, \mathbf{E}$ would have k columns in order to encrypt k-bitstrings directly.

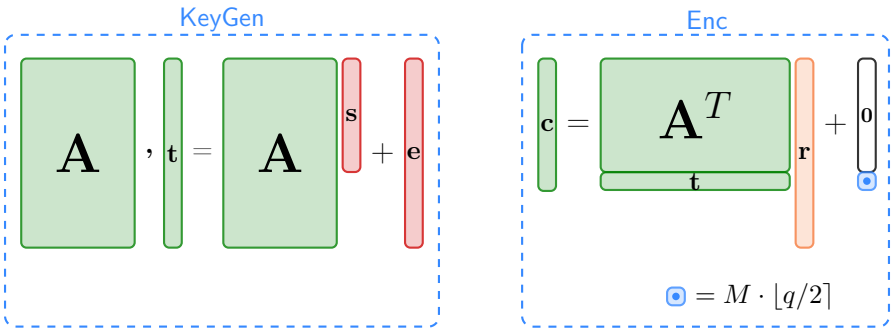


Figure 6.1: Regev public-key encryption scheme

6.1.2 Security Analysis

We now need to ensure that Regev encryption satisfies the security requirements that we expect from a public-key encryption scheme. As seen in the course "Security Proofs (PRS)" and recalled in Section 1.3.1, there exists several security models for public key encryption schemes. In our case, we will show correctness, and semantic security or IND-CPA security. Let us first look at the correctness. The correctness actually places restriction on the size of the error $\mathbf{e}$ in the public key to make sure that one can recover the correct message during decryption.

Lemma 6.1 (Regev Correctness)

Let d, q, m be positive integers, and $\alpha \in (0, 1/(4m))$. Then, for $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(d, q, m, \alpha)$ and $M \in \{0, 1\}$, it holds that $\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, M)) = M$ except with probability at most 2^{-2m}, proving that the encryption scheme is correct, with a decryption failure probability of at most 2^{-2m}.

Proof (Lemma 6.1). Let pk, sk are honestly generated keys and $(\mathbf{c}_1, c_2) \leftarrow \mathsf{Enc}(\mathsf{pk}, M)$ be an honestly generated ciphertext. Then, there exists $\mathbf{r} \in \{0, 1\}^m$ such that $\mathbf{c}_1 = \mathbf{A}^T\mathbf{r} \bmod q\mathbb{Z}$ and $c_2 = \mathbf{r}^T(\mathbf{A}\mathbf{s} + \mathbf{e}) + \lfloor q/2 \rceil M \bmod q\mathbb{Z}$. As a result, we have

$$\begin{aligned} u = c_2 - \mathbf{s}^T\mathbf{c}_1 \bmod q\mathbb{Z} &= (\mathbf{A}\mathbf{s} + \mathbf{e})^T\mathbf{r} + \lfloor q/2 \rceil M - \mathbf{s}^T\mathbf{A}^T\mathbf{r} \bmod q\mathbb{Z} \\ &= \mathbf{e}^T\mathbf{r} + \lfloor q/2 \rceil M \bmod q\mathbb{Z}. \end{aligned}$$

By Cauchy-Schwarz inequality, it holds that $\left|\mathbf{r}^T\mathbf{e}\right| \leq \|\mathbf{e}\|_2\|\mathbf{r}\|_2 \leq \|\mathbf{e}\|_2\sqrt{m}$ as $\mathbf{r}$ is binary. Then, since $\mathbf{e}$ is distributed according $\mathcal{D}_{\mathbb{Z}^m, \alpha q}$, it holds that $\|\mathbf{e}\|_2 \leq \alpha q\sqrt{m}$ except with probability 2^{-2m} by Lemma 4.9. We now condition on the event $\|\mathbf{e}\|_2 \leq \alpha q\sqrt{m}$. Then, we have $\left|\mathbf{r}^T\mathbf{e}\right| \leq \alpha q m \leq q/4$.

When $M = 0$, we have $u = \mathbf{r}^T\mathbf{e}$ and thus $|u| \leq q/4 < \left|u \pm \lfloor q/2 \rceil\right|$ so the retrieved message during decryption is indeed $\mathsf{Dec}(\mathsf{sk}, (\mathbf{c}_1, c_2)) = 0$ as expected. On the contrary, when $M = 1$, we have $u \in [-q/2, -\lfloor q/2 \rceil + q/4] \cup [\lfloor q/2 \rceil - q/4, q/2]$ and thus $\mathsf{Dec}(\mathsf{sk}, (\mathbf{c}_1, c_2)) = 1$ as desired. As a result, we have

$$\begin{aligned} &\mathbb{P}[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, M)) = M] \\ &\quad = \mathbb{P}[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, M)) = M | \|\mathbf{e}\|_2 \leq \alpha q\sqrt{m}]\mathbb{P}_{\mathbf{e}\sim\mathcal{D}_{\mathbb{Z}^m,\alpha q}}[\|\mathbf{e}\|_2 \leq \alpha q\sqrt{m}] \\ &\quad\quad + \mathbb{P}[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, M)) = M | \|\mathbf{e}\|_2 > \alpha q\sqrt{m}]\mathbb{P}_{\mathbf{e}\sim\mathcal{D}_{\mathbb{Z}^m,\alpha q}}[\|\mathbf{e}\|_2 > \alpha q\sqrt{m}] \\ &\quad \geq 1 \cdot (1 - 2^{-2m}) + 0 \\ &\quad = 1 - 2^{-2m}, \end{aligned}$$

as claimed.

Note that the decryption error of 2^{-2m} is solely due to the Gaussian tail bound of the secret error term. Hence, one could sample $\mathbf{e}$ from a Gaussian conditioned on this bound being verified and thus enforce the bound during key generation. This would result in a perfectly correct encryption scheme, but the error distribution would now be a *truncated* discrete Gaussian distribution.

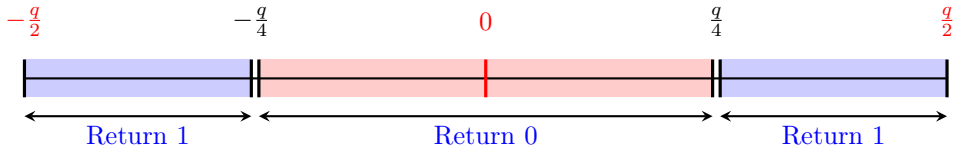


Figure 6.2: Value of u during decryption

e now prove the semantic security of the encryption scheme. The proof relies on the $\mathrm{LWE}_{d,q,m,\alpha}$ assumption as well as the leftover hash lemma from Lemma 5.2. We clearly see that the link between the secret key and the public key is an instance of $\mathrm{sLWE}_{d,q,m,\alpha}$. Under the assumption that the latter is a hard problem, it should be infeasible to recover the secret key from the public key. However, the IND-CPA security is a little more delicate to prove. At a high level, we need to show that $\mathbf{c}_1$ and c_2 do not reveal any information on M. This can be interpreted as requiring that $\mathbf{r}^T[\mathbf{A}|\mathbf{t}]$ correctly masks the vector $[\mathbf{0}_d|\lfloor q/2 \rceil M]$.

Lemma 6.2 (Regev Semantic Security)

Let λ, d, q, m be positive integers such that q is prime and that $m \geq (d + 1)\log_2 q + 2\lambda$. Let $\alpha \in (0, 1/(4m))$. Then Regev encryption scheme is (t, ε)-IND-CPA secure where $\varepsilon = 2^{-\lambda-1} + \varepsilon_{\mathrm{LWE}}(t)$ with $\varepsilon_{\mathrm{LWE}}(t)$ the hardness bound of $\mathrm{LWE}_{d,q,m,\alpha}$. In particular, if $\varepsilon_{\mathrm{LWE}}(t) \leq 2^{-\lambda-1}$,

then the scheme is $(t, 2^{-\lambda})$-IND-CPA secure.

Proof (Lemma 6.2). See TD 2

We have proven the semantic security of Regev's encryption scheme. There however exists other security models which may be more relevant. In particular, one might wonder if Regev's encryption is also secure in the IND-CCA models which offer stronger security guarantees. Unfortunately, Regev's encryption scheme is not secure in the IND-CCA2 nor IND-CCA1 model. We can indeed use the algebraic nature of the encryption to modify the ciphertexts in a controlled way while predicting the behavior of the decryption. This will be studied in TD 6 .

6.2 Dual Regev Encryption Scheme

An alternative to Regev encryption scheme was proposed by Gentry, Peikert and Vaikuntanathan [GPV08] in 2008, where they designed a dual version of the scheme called Dual Regev. The idea is essentially to swap the $\mathbf{r}^T\mathbf{A}$ and the $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}$ from Regev encryption scheme. The key would then be pseudorandom based on Lemma 5.2, and the ciphertexts would then be skewed LWE instances. The overall structure is very similar, but we provide it in this course nonetheless.

6.2.1 Description

As the underlying structure and the idea behind Dual Regev are much similar to Regev encryption scheme, we directly jump to the formal description of the scheme. The main difference in presentation stems from the fact that all the user now share the same matrix $\mathbf{A}$. We note however, that this could also be enforced in Regev encryption. We describe the three algorithms KeyGen, Enc, Dec from Algorithms 6.4, 6.5 and 6.6.

Algorithm 6.4: KeyGen (Dual Regev)

Input: Integers d, q, m, real α, which are functions of the security parameter λ, matrix $\mathbf{A} \in \mathbb{Z}_q^{m\times d}$.

1. $\mathbf{r} \hookleftarrow U(\{0,1\}^m)$
2. $\mathbf{y} \leftarrow \mathbf{A}^T\mathbf{r} \bmod q\mathbb{Z}$

Output: $\mathsf{pk} = \mathbf{y}$, $\mathsf{sk} = \mathbf{r}$

Algorithm 6.5: Enc (Dual Regev)

Input: Public parameters, Public Key $\mathsf{pk} = \mathbf{y} \in \mathbb{Z}_q^d$, Message $M \in \{0,1\}$.

1. $\mathbf{s} \hookleftarrow U(\mathbb{Z}_q^d)$
2. $\mathbf{e} \hookleftarrow \mathcal{D}_{\mathbb{Z}^m,\alpha q}$
3. $e' \hookleftarrow \mathcal{D}_{\mathbb{Z},\alpha q}$
4. $(\mathbf{c}_1, c_2) \leftarrow (\mathbf{A}\mathbf{s} + \mathbf{e} \bmod q\mathbb{Z}, \mathbf{y}^T\mathbf{s} + e' + \lfloor q/2 \rceil M \bmod q\mathbb{Z}) \in \mathbb{Z}_q^m \times \mathbb{Z}_q$

Output: $(\mathbf{c}_1, c_2)$

Algorithm 6.6: Dec (Dual Regev)

Input: Public parameters, Secret Key $\mathsf{sk} \in \mathbb{Z}_q^d$, Ciphertext $(\mathbf{c}_1, c_2) \in \mathbb{Z}_q^{m+1}$.

1. $u \leftarrow c_2 - \mathbf{c}_1^T\mathbf{r} \bmod q\mathbb{Z}$
2. **If** $|u - 0| < |u \pm \lfloor q/2 \rceil|$, then $M \leftarrow 0$, **else** $M \leftarrow 1$.

Output: M

Once again, we insist that the last check relies on the fact that u is the representative of the equivalence class $c_2 - \mathbf{c}_1^T\mathbf{s} \bmod q\mathbb{Z}$ that lies in $(-q/2, q/2]$.

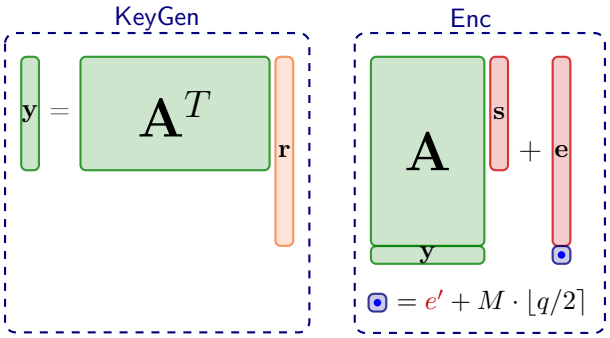


Figure 6.3: Dual Regev public-key encryption scheme

6.2.2 Security Analysis

We now need to ensure that these modification to Regev encryption do not hinder the security of the scheme. In particular, we need to derive new conditions for the correctness and the semantic security to hold.

Lemma 6.3 (Dual Regev Correctness)

Let d, q, m be positive integers, and $\alpha \in (0, 1/(4(m+1)))$. Then, for $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(d, q, m, \alpha, \mathbf{A})$ and $M \in \{0,1\}$, it holds that $\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, M)) = M$ except with probability at most $2^{-2(m+1)}$, proving that the encryption scheme is correct.

Proof (Lemma 6.3). The proof follows the same structure than that of Lemma 6.1. Let pk, sk are honestly generated keys and $(\mathbf{c}_1, c_2) \leftarrow \mathsf{Enc}(\mathsf{pk}, M)$ be an honestly generated ciphertext. Then, there exists $\mathbf{s} \in \mathbb{Z}_q^d$ and $[\mathbf{e}^T | e']^T \hookleftarrow \mathcal{D}_{\mathbb{Z}^{m+1}, \alpha q}$ such that $\mathbf{c}_1 = \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q\mathbb{Z}$ and $c_2 = \mathbf{y}^T\mathbf{s} + e' + \lfloor q/2 \rceil M \bmod q\mathbb{Z}$. As a result, we have

$$\begin{aligned} u = c_2 - \mathbf{r}^T\mathbf{c}_1 \bmod q\mathbb{Z} &= \mathbf{y}^T\mathbf{s} + e' + \lfloor q/2 \rceil M - \mathbf{r}^T(\mathbf{A}\mathbf{s} + \mathbf{e}) \bmod q\mathbb{Z} \\ &= -\mathbf{r}^T\mathbf{e} + e' + \lfloor q/2 \rceil M \bmod q\mathbb{Z}. \end{aligned}$$

We denote by $\mathbf{e}^* = [\mathbf{e}^T | e']^T$ and by $\mathbf{r}^* = [\mathbf{r}^T | 1]^T$. By Cauchy-Schwarz inequality, it holds that $\left|\mathbf{r}^{*T}\mathbf{e}^*\right| \leq \|\mathbf{e}^*\|_2 \|\mathbf{r}^*\|_2 \leq \|\mathbf{e}^*\|_2 \sqrt{m+1}$ as $\mathbf{r}$ is binary. Then, since $\mathbf{e}^*$ is distributed according $\mathcal{D}_{\mathbb{Z}^{m+1}, \alpha q}$, it holds that $\|\mathbf{e}^*\|_2 \leq \alpha q\sqrt{m+1}$ except with probability $2^{-2(m+1)}$ by Lemma 4.9. We now condition on the event $\|\mathbf{e}^*\|_2 \leq \alpha q\sqrt{m+1}$. Then, we have $\left|\mathbf{r}^T\mathbf{e}\right| \leq \alpha q(m+1) \leq q/4$.

When $M = 0$, we have $u = \mathbf{r}^{*T}\mathbf{e}^*$ and thus $|u| \leq q/4 < \left|u \pm \lfloor q/2 \rceil\right|$ so the retrieved message during decryption is indeed $\mathsf{Dec}(\mathsf{sk}, (\mathbf{c}_1, c_2)) = 0$ as expected. On the contrary, when $M = 1$, we have $u \in [-q/2, -\lfloor q/2 \rceil + q/4] \cup [\lfloor q/2 \rceil - q/4, q/2]$ and thus $\mathsf{Dec}(\mathsf{sk}, (\mathbf{c}_1, c_2)) = 1$ as desired. As a result, we have

$$\begin{aligned} &\mathbb{P}[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, M)) = M] \\ &\quad = \mathbb{P}[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, M)) = M | \|\mathbf{e}^*\|_2 \leq \alpha q\sqrt{m+1}] \mathbb{P}_{\mathbf{e}^* \sim \mathcal{D}_{\mathbb{Z}^{m+1}, \alpha q}}[\|\mathbf{e}^*\|_2 \leq \alpha q\sqrt{m+1}] \\ &\qquad + \mathbb{P}[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, M)) = M | \|\mathbf{e}^*\|_2 > \alpha q\sqrt{m+1}] \mathbb{P}_{\mathbf{e}^* \sim \mathcal{D}_{\mathbb{Z}^{m+1}, \alpha q}}[\|\mathbf{e}^*\|_2 > \alpha q\sqrt{m+1}] \\ &\quad \geq 1 \cdot (1 - 2^{-2(m+1)}) + 0 \\ &\quad = 1 - 2^{-2(m+1)}, \end{aligned}$$

as claimed.

The semantic security proof follows the same idea as for Regev semantic security, but with reversed arguments. It now relies on the $\mathrm{LWE}_{d,q,m+1,\alpha}$ assumption as well as the leftover hash lemma from Lemma 5.2.

Lemma 6.4 (dual Regev Semantic Security)

Let λ, d, q, m be positive integers such that q is prime and that $m \geq d \log_2 q + 2\lambda$. Let $\alpha \in (0, 1/(4(m+1)))$. Then Dual Regev encryption scheme is (t, ε)-IND-CPA secure where $\varepsilon = 2^{-\lambda-1} + \varepsilon_{\mathrm{LWE}}(t)$ with $\varepsilon_{\mathrm{LWE}}(t)$ the hardness bound of $\mathrm{LWE}_{d,q,m+1,\alpha}$. In particular, if $\varepsilon_{\mathrm{LWE}}(t) \leq 2^{-\lambda-1}$, then the scheme is $(t, 2^{-\lambda})$-IND-CPA secure.

Proof (Lemma 6.4). Again, the proof structure is exactly the same but changing the order of the arguments. We write it fully for completeness.

Game G_0:

Challenger $\mathcal{C}$		Adversary $\mathcal{A}$
$(\mathsf{pk}, \mathsf{sk}) = (\mathbf{y} = \mathbf{A}^T\mathbf{r} \bmod q\mathbb{Z}), \mathbf{r}$ $\mathbf{r} \hookleftarrow U(\{0,1\}^m)$		
	$\xrightarrow{\mathsf{pk}}$	
		Choose $M_0 \neq M_1$
	$\xleftarrow{M_0, M_1}$	
$b \hookleftarrow U(\{0,1\})$ $\mathbf{s} \hookleftarrow U(\mathbb{Z}_q^d), (\mathbf{e}, e') \hookleftarrow \mathcal{D}_{\mathbb{Z}^{m+1}, \alpha q}$ $\mathbf{c}_1 \leftarrow \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q\mathbb{Z}$ $c_2 \leftarrow \mathbf{y}^T\mathbf{s} + e' + \lfloor q/2 \rceil M_b \bmod q\mathbb{Z}$		
	$\xrightarrow{\mathbf{c}_1, c_2}$	
		Choose $b' \in \{0,1\}$
$\mathcal{A}$ wins if $b' = b$	$\xleftarrow{b'}$	

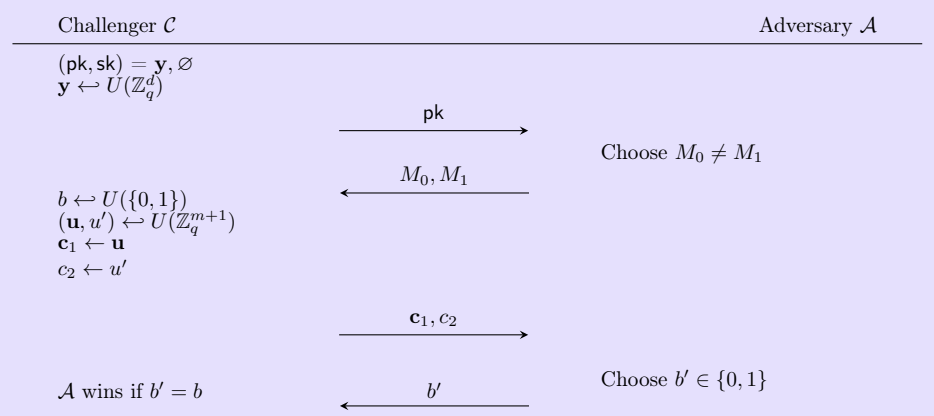
Game G_1:

Challenger $\mathcal{C}$		Adversary $\mathcal{A}$
$(\mathsf{pk}, \mathsf{sk}) = \mathbf{y}, \varnothing$ $\mathbf{y} \hookleftarrow U(\mathbb{Z}_q^d)$		
	$\xrightarrow{\mathsf{pk}}$	
		Choose $M_0 \neq M_1$
	$\xleftarrow{M_0, M_1}$	
$b \hookleftarrow U(\{0,1\})$ $\mathbf{s} \hookleftarrow U(\mathbb{Z}_q^d), (\mathbf{e}, e') \hookleftarrow \mathcal{D}_{\mathbb{Z}^{m+1}, \alpha q}$ $\mathbf{c}_1 \leftarrow \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q\mathbb{Z}$ $c_2 \leftarrow \mathbf{y}^T\mathbf{s} + e' + \lfloor q/2 \rceil M_b \bmod q\mathbb{Z}$		
	$\xrightarrow{\mathbf{c}_1, c_2}$	
		Choose $b' \in \{0,1\}$
$\mathcal{A}$ wins if $b' = b$	$\xleftarrow{b'}$	

Game G_2:

Challenger $\mathcal{C}$		Adversary $\mathcal{A}$
$(\mathsf{pk}, \mathsf{sk}) = \mathbf{y}, \varnothing$ $\mathbf{y} \hookleftarrow U(\mathbb{Z}_q^d)$		
	$\xrightarrow{\mathsf{pk}}$	
		Choose $M_0 \neq M_1$
	$\xleftarrow{M_0, M_1}$	
$b \hookleftarrow U(\{0,1\})$ $(\mathbf{u}, u') \hookleftarrow U(\mathbb{Z}_q^{m+1})$ $\mathbf{c}_1 \leftarrow \mathbf{u}$ $c_2 \leftarrow u' + \lfloor q/2 \rceil M_b \bmod q\mathbb{Z}$		
	$\xrightarrow{\mathbf{c}_1, c_2}$	
		Choose $b' \in \{0,1\}$
$\mathcal{A}$ wins if $b' = b$	$\xleftarrow{b'}$	

Game G_3:



We now prove that the games are indistinguishable from one another.
G_0 - G_1: First, G_0 and G_1 are statistically close from the leftover hash lemma of Lemma 5.2 because $m \geq d \log_2 q + 2\lambda$. In particular, this proves that for a ∞-time adversary $\mathcal{A}$ against the IND-CPA game, we would have

$$\left|\mathrm{Adv}_{G_0}[\mathcal{A}] - \mathrm{Adv}_{G_1}[\mathcal{A}]\right| \leq \Delta\left(\mathbf{A}, \mathbf{A}^T\mathbf{r}), (\mathbf{A}, \mathbf{y})\right) \leq 2^{-\lambda-1},$$

which thus holds true for t-time adversaries as well.

G_1 - G_2: Then, the games G_1 et G_2 are computationally indistinguishable under the $\mathrm{LWE}_{n,q,m+1,\alpha}$ assumption. Indeed, if we consider a distinguisher between the views of the adversary in G_1 and G_2, we can construct a distinguisher for $\mathrm{LWE}_{n,q,m+1,\alpha}$. Given an instance $((\mathbf{A}, \mathbf{c}_1'), (\mathbf{y}, c_2'))$ of the problem, the reduction defines $\mathsf{pk} = \mathbf{y}$, $\mathbf{c}_1' = \mathbf{c}_1$ and $c_2 = c_2' + \lfloor q/2 \rceil M_b \bmod q\mathbb{Z}$ and proceeds normally for the rest of the game. If $\mathbf{c}_1'$ is of the form $\mathbf{As} + \mathbf{e} \bmod q\mathbb{Z}$ and $c_2' = \mathbf{y}^T\mathbf{s} + e'$, the distribution of $(\mathbf{c}_1, c_2)$ is identical as the one in G_1. If they are uniform, the distribution of $(\mathbf{c}_1, c_2)$ is identical to the one in G_2. Hence, one call to the distinguisher between G_1 and G_2 allows one to solve $\mathrm{LWE}_{n,q,m+1,\alpha}$. In particular, it proves that for all time-t adversaries $\mathcal{A}$, we have

$$\left|\mathrm{Adv}_{G_1}[\mathcal{A}] - \mathrm{Adv}_{G_2}[\mathcal{A}]\right| \leq \varepsilon_{\mathrm{LWE}}(t)$$

G_2 - G_3: Then, since u' is uniform and independent of M_b in G_2, the distribution of $[\mathbf{c}_1^T | c_2]^T$ is exactly the uniform distribution of $\mathbb{Z}_q^{m+1}$. Hence G_2 and G_3 are identically distributed.

To conclude, in G_3, the advantage of $\mathcal{A}$ is 0 because its view is independent of b. The only possible strategy is guessing. Then, by the triangle inequality, the advantage of $\mathcal{A}$ in G_0 is at most

$$\mathrm{Adv}_{G_0}[\mathcal{A}] \leq \varepsilon_{\mathrm{LWE}}(t) + 2^{-\lambda-1},$$

as claimed.

RECAP

- Regev's public key encryption scheme allows for encrypting one bit $M \in \{0,1\}$. The keys form an instance of LWE with $\mathsf{pk} = (\mathbf{A}, \mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e})$ and $\mathsf{sk} = \mathbf{s}$. The encryption samples $\mathbf{r}$ in $\{0,1\}^m$ and computes the two-part ciphertext $\mathbf{c}_1 = \mathbf{A}^T\mathbf{r}$ and $c_2 = \mathbf{t}^T\mathbf{r} + \lfloor q/2 \rceil M$. To decrypt, we compute $c_2 - \mathbf{c}_1^T\mathbf{s}$ and return 0 if it is closer to 0 than $q/2$, and 1 otherwise.
- The Dual-Regev's public key encryption scheme swaps the roles of $\mathbf{c}_1$ and $\mathbf{t}$. The keys are $\mathsf{pk} = \mathbf{y} = \mathbf{A}^T\mathbf{r}$ and $\mathsf{sk} = \mathbf{r}$. To encrypt, one samples $\mathbf{s} \in \mathbb{Z}_q^d$ and $(\mathbf{e}, e')$ following $\mathcal{D}_{\mathbb{Z}^{m+1},\alpha q}$, and computes $\mathbf{c}_1 = \mathbf{A}\mathbf{s} + \mathbf{e}$ and $c_2 = \mathbf{y}^T\mathbf{s} + e' + \lfloor q/2 \rceil M$. The decryption computes $c_2 - \mathbf{c}_1^T\mathbf{r}$ and decodes as in Regev's decryption.
- The Regev and Dual-Regev encryption schemes are IND-CPA based on the LWE assumption and the leftover hash lemma. They are not IND-CCA.

Bibliography

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for Hard Lattices and New Cryptographic Constructions. In *STOC*, 2008.

[Reg05] O. Regev. On Lattices, Learning With Errors, Random Linear Codes, and Cryptography. In *STOC*, 2005.

7

Signature Schemes

In this chapter we introduce lattice-based signature designs whose security relies on the standard lattice assumptions that we introduced. Most lattice-based signature schemes follow one of two main paradigms. The first one is called *Hash-and-Sign* and was instantiated in [GPV08] using lattice preimage sampleable trapdoor functions. The idea is to hash the message to the range of a trapdoor function and use said trapdoor to obtain a short preimage of this hash. An alternative design paradigm, called *Fiat-Shamir with Aborts* (FSwA) and proposed by Lyubashevsky [Lyu12], builds signature on Schnorr-like proofs made non-interactive with the Fiat-Shamir transform. However, certain peculiarities of lattices requires the addition of a rejection sampling step to preserve security. In this chapter, we first introduce the FSwA paradigm, before defining the notion of trapdoor and the Hash-and-Sign paradigm.

Contents

7.1 Fiat-Shamir with Aborts Paradigm

The Fiat-Shamir paradigm is another popular signature design which is not specific to lattice constructions. The name stems from the work of Fiat and Shamir [FS86] who gave a generic transform to turn a identification Sigma protocol into a signature scheme in the random oracle model. A very popular framework on which is inspired EdDSA [BDL+11] are Schnorr Sigma protocol which can then be turned into signature using the Fiat-Shamir transform. One can use a lattice equivalent of Schnorr's Sigma protocol on lattices but the peculiarities of lattices make it insecure without care. A solution proposed by Lyubashevsky [Lyu12] was to use a key tool from probability theory called rejection sampling which allows to patch the security holes in lattice Schnorr's protocol. This is at the expense of rejection, as the name suggests, which is why the lattice name of the Fiat-Shamir paradigm is *Fiat-Shamir with Aborts*. We start by presenting the high level idea of identification Sigma protocols and the Fiat-Shamir transform before presenting the signature scheme of [Lyu12].

7.1.1 From Identification to Signature

An identification Sigma protocol is a protocol in 3 exchanges between a prover $\mathcal{P}$ trying to identify and a verifier $\mathcal{V}$. The stages of such protocols are usually referred to as *Commitment*, *Challenge*

and *Response*. The prover holds a secret key sk and aims at convincing $\mathcal{V}$ of its identity who holds the corresponding public key pk. The prover $\mathcal{P}$ starts by sending a commitment CMT to the verifier, allowing them to commit to their identity. The verifier then sends a random challenge CH so that only the person with the correct identity should be able to reply correctly. In the third stage, $\mathcal{P}$ computes the correct response RSP to the commitment CMT and challenge CH. If the triplet (CMT, CH, RSP) verifies a certain property which is specific to each identification protocol, then $\mathcal{V}$ accepts the proof of identification. We summarize it in Figure 7.1.

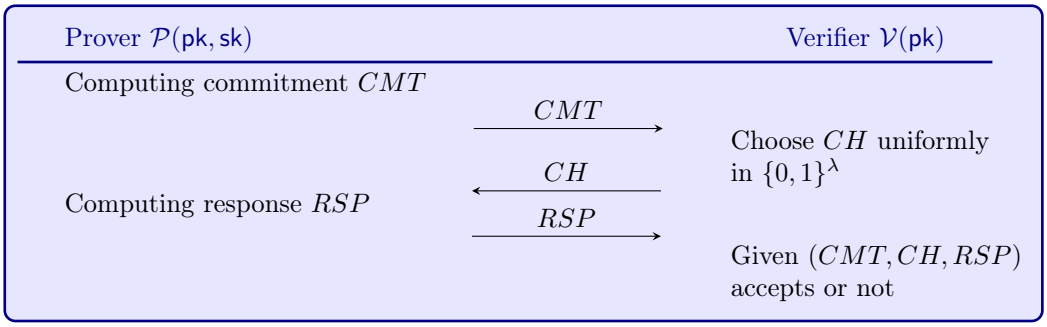


Figure 7.1: Generic Identification Sigma Protocol

In 1986, Fiat and Shamir [FS86] proposed a transform to turn such identification Sigma protocols into a signature scheme. The signature actually corresponds to what the prover $\mathcal{P}$ executes, and by choosing the challenge CH as the hash output of CMT and the message M, i.e., $CH \leftarrow \mathcal{H}(CMT \| M)$. When $\mathcal{H}$ is modeled as random oracle, the challenge is indeed uniformly random. The signature on the message M is then (CMT, RSP). To verify the signature, one can compute $CH = \mathcal{H}(CMT \| M)$ and verify that (CMT, CH, RSP) is a valid proof of identification. The security of the signature then directly relies on the security of the identification protocol and on the random oracle model. The advantage of such signature schemes is that they do not rely on complex trapdoor functions.

7.1.2 Description

Rejection Sampling

Before describing the scheme, we do a quick reminder on rejection sampling. Let $\mathscr{D}_s$ and $\mathscr{D}_t$ the density functions of two probability distributions called the *source* and *target* distributions. The objective is to use a sampler for the source distribution to produce samples for the target distribution. In a context of signature, the source distribution may for example depend on the secret key which is undesirable. We then use it to produce elements following the target distribution which is independent of all secret data.

For that, assume that there exists $M > 1$ such that for all x, $\mathscr{D}_t(x) \leq M \cdot \mathscr{D}_s(x)$, which can be written as $\mathrm{RD}_\infty(\mathscr{D}_t \| \mathscr{D}_s) \leq M$. Then, the distribution obtained by sampling x from $\mathscr{D}_s$ and returning it with probability $\mathscr{D}_t(x)/(M \cdot \mathscr{D}_s(x))$ corresponds exactly to the distribution $\mathscr{D}_t$. Hence, by sampling from $\mathscr{D}_s$ and rejecting with the correct probability, we can ensure that the output follows $\mathscr{D}_t$. The parameter M is called the repetition rate which somewhat quantifies the number of iterations needed before actually outputting a sample.

We can also perform *inexact rejection* in order to relax the constraint on $\mathscr{D}_s, \mathscr{D}_t$. Assuming that there exists $M > 1$ and $\varepsilon \in (0, 1)$ such that

$$\mathbb{P}_{x \sim \mathscr{D}_t}[\mathscr{D}_t(x) \leq M \mathscr{D}_s(x)] \geq 1 - \varepsilon.$$

Then, by rejecting samples with probability $\min(1, \mathscr{D}_t(x)/(M \cdot \mathscr{D}_s(x)))$, one can show, e.g., [Lyu12], that the output will be statistically close to $\mathscr{D}_t$ up to a loss linked to ε.

Lyubashevsky's Signature Scheme

We describe the signature scheme with the KeyGen, Sign, and Verify algorithms in Algorithms 7.1, 7.2 and 7.3. We let $\mathcal{H}$ be a hash function from $\mathbb{Z}_q^d \times \{0,1\}^*$ to $\{-1, 0, 1\}^k$.

Algorithm 7.1: KeyGen (Fiat-Shamir with Aborts)

Input: Integers d, q, m, k, and $s > 0$ a Gaussian parameter.

1. $\mathbf{S} \hookleftarrow U(\{-1, 0, 1\}^{m \times k})$
2. $\mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{d \times m})$
3. $\mathbf{B} = \mathbf{AS} \bmod q\mathbb{Z}$

Output: $\mathsf{pk} = (\mathbf{A}, \mathbf{B})$, $\mathsf{sk} = \mathbf{S}$

Algorithm 7.2: Sign (Fiat-Shamir with Aborts)

Input: Public parameters, Public Key $\mathsf{pk} = (\mathbf{A}, \mathbf{B}) \in \mathbb{Z}_q^{d \times m} \times \mathbb{Z}_q^{d \times k}$, Secret Key $\mathsf{sk} = \mathbf{S} \in \mathbb{Z}^{m \times k}$, Message $M \in \{0,1\}^*$.

1. $\mathbf{y} \hookleftarrow \mathcal{D}_{\mathbb{Z}^m, s}$.
2. $\mathbf{c} \leftarrow \mathcal{H}(\mathsf{pk} \| \mathbf{Ay} \bmod q\mathbb{Z} \| M) \in \{-1, 0, 1\}^k$.
3. $\mathbf{z} = \mathbf{Sc} + \mathbf{y}$
4. Accepts $\mathbf{z}$ with probability $P(\mathbf{z})$, otherwise restart.

Output: $\mathsf{sig} = (\mathbf{z}, \mathbf{c})$

Algorithm 7.3: Verify (Fiat-Shamir with Aborts)

Input: Public parameters, Public Key $\mathsf{pk} = (\mathbf{A}, \mathbf{B}) \in \mathbb{Z}_q^{d \times m} \times \mathbb{Z}_q^{d \times k}$, Signature $\mathsf{sig} = (\mathbf{z}, \mathbf{c}) \in \mathbb{Z}^m \times \mathbb{Z}^k$, Message $M \in \{0,1\}^*$.

1. $b \leftarrow (\mathbf{c} = \mathcal{H}(\mathsf{pk} \| \mathbf{Az} - \mathbf{Bc} \bmod q\mathbb{Z} \| M)) \wedge (\|\mathbf{z}\|_2 \leq s\sqrt{m})$

Output: $b \in \{0,1\}$

where the probability $P(\mathbf{z})$ is here to ensure that $\mathbf{z}$ is statistically close to $\mathcal{D}_{\mathbb{Z}^m, s}$, that is

$$P(\mathbf{z}) = \min\left(1, \frac{\mathcal{D}_{\mathbb{Z}^m, s}(\mathbf{z})}{M \cdot \mathcal{D}_{\mathbb{Z}^m, s, \mathbf{Sc}}(\mathbf{z})}\right)$$

Indeed, the source distribution corresponds to $\mathbf{Sc} + \mathcal{D}_{\mathbb{Z}^m, s}$ which by Lemma 7.5 is $\mathcal{D}_{\mathbb{Z}^m + \mathbf{Sc}, s, \mathbf{Sc}}$. Yet, since $\mathbf{Sc} \in \mathbb{Z}^m$, the coset $\mathbb{Z}^m + \mathbf{Sc}$ is exactly $\mathbb{Z}^m$ itself. So the source distribution is indeed $\mathcal{D}_{\mathbb{Z}^m, s, \mathbf{Sc}}$. The correctness holds due to the fact that $\mathbf{z}$ is then close to $\mathcal{D}_{\mathbb{Z}^m, s}$, and because $\mathbf{Az} - \mathbf{Bc} = \mathbf{Ay} - \mathbf{ASc} - \mathbf{Bc} = \mathbf{Ay} \bmod q\mathbb{Z}$. Hence, $\mathbf{c} = \mathcal{H}(\mathsf{pk} \| \mathbf{Az} - \mathbf{Bc} \bmod q\mathbb{Z} \| M)$ for honestly generated signatures.

Remark 7.1 (Security of Fiat-Shamir with Aborts Signature)

We do not provide the EUF-CMA security proof of the signature scheme as it involves rather technical tools such as the generalized forking lemma. Nevertheless, this scheme can be proven secure in the random oracle model based on the hardness of $\mathrm{SIS}_{d,m,q,\beta}$ and the inexact rejection sampling argument.

The presence of rejection sampling seems artificial without the details of the security proof. And yet, this step is essential. Indeed, we will show in TD 3 that without rejection, it is easy to produce valid signature without knowledge of the secret key or to recover the secret key from sufficiently many signatures.

7.2 Lattice Trapdoors

The first method for constructing digital signatures over lattices uses the notion of trapdoor functions. A trapdoor function f is a function that is efficiently computable, but that is difficult to invert unless given a secret *trapdoor information* s. More precisely, the signature design relies on the possibility of sampling preimages using the trapdoor s, meaning that for a syndrome y the user can use s to sample from a distribution over $f^{-1}\langle\{y\}\rangle$, thus introducing randomness in the preimage finding process. The preimage x then corresponds to the signature of y, which can be verified by checking that $f(x) = y$ and that x looks like the correct distribution. This approach is rather generic and not related to lattices. We now present how to construct preimage sampleable trapdoor functions from lattices.

7.2.1 Short Bases and Trapdoor Construction

We start with a generic approach and identify its limitations. A first idea to achieve the functionalities we are aiming for is to have a lattice $\mathcal{L}$ along with a short basis $\mathbf{B}$ of this lattice. Given an element $\mathbf{y}$ in the ambient space $\mathrm{Span}_{\mathbb{R}}(\mathcal{L})$, one could sample $\mathbf{x}$ from the discrete Gaussian distribution $\mathcal{D}_{\mathcal{L},s,\mathbf{y}}$ using the basis $\mathbf{B}$ and output $\mathbf{x}$ as the preimage. For that, we can use Klein's sampler (Algorithm 4.1) which, according to Theorem 4.2, produces samples statistically close to $\mathcal{D}_{\mathcal{L},s,\mathbf{y}}$ if $s \geq \eta_\varepsilon(\mathbb{Z}^d)\left\|\mathsf{GSO}(\mathbf{B})\right\|_\infty$. Then, one could easily check if $\mathbf{x}$ is in $\mathcal{L}$ given a bad basis of $\mathcal{L}$, and check that $\|\mathbf{x}-\mathbf{y}\|_2$ is short. The security would then rely on the hardness of CVP_γ. This approach perfectly satisfies the requirements we placed at the outset at one exception. We need to efficiently generate a strong lattice along with a short basis. In particular, what we aim for in lattice cryptography is to use random lattices. A good candidate is a q-ary lattice like $\mathcal{L}_q^\perp(\mathbf{A})$ described in Definition 2.3.

Let us then consider a lattice $\mathcal{L} = \mathcal{L}_q^\perp(\mathbf{A})$ for a matrix $\mathbf{A} \in \mathbb{Z}_q^{d\times m}$. If we are able to produce a short basis of $\mathcal{L}_q^\perp(\mathbf{A})$, then we would be able to produce discrete Gaussian samples over this lattice. Unfortunately, given a random matrix $\mathbf{A}$, finding a short basis, e.g., whose vectors have norm less than β, is exactly solving (many times) the $\mathrm{SIS}_{d,m,q,\beta}$ problem, which is assumed to be infeasible. In 2009, Alwen and Peikert [AP09] proposed a way to conjointly generate the matrix $\mathbf{A}$ and a short basis $\mathbf{B_A}$ of $\mathcal{L}_q^\perp(\mathbf{A})$ so that the resulting $\mathbf{A}$ is statistically close to uniform.

Lemma 7.1 (Short Bases for Random Lattices)

There exists a probabilistic polynomial-time algorithm $\mathsf{TrapGen}$ taking as input d, m, q such that $q \geq 2$ and $m \geq \Omega(d\log_2 q)$ and that outputs $\mathbf{A} \in \mathbb{Z}_q^{d\times m}$ and a basis $\mathbf{B_A}$ of $\mathcal{L}_q^\perp(\mathbf{A})$ such that $\Delta(\mathbf{A}, U(\mathbb{Z}_q^{d\times m})) \leq 2^{-\Omega(d)}$ and $\left\|\mathsf{GSO}(\mathbf{B_A})\right\|_\infty \leq O(\sqrt{d\log_2 q})$.

We present here the simplified description of the algorithm $\mathsf{TrapGen}$. The idea is to use the leftover hash lemma to generate the columns of $\mathbf{A}$ along with the rows (and columns) of $\mathbf{B_A}$. Assume that we have constructed $\mathbf{A} \in \mathbb{Z}_q^{d\times m}$ and $\mathbf{B_A} \in \mathbb{Z}^{m\times m}$ such that $\mathbf{AB_A} = \mathbf{0} \bmod q\mathbb{Z}$. We now add a column to $\mathbf{A}$ and complete the basis accordingly.

1. Sample a random vector $\mathbf{r} \hookleftarrow U(\{0,1\}^m)$.
2. Update the matrix $\mathbf{A'} = [\mathbf{A}|\mathbf{Ar}] \bmod q\mathbb{Z}$
3. Update the trapdoor $\mathbf{B}_{\mathbf{A}'} = \begin{bmatrix} \mathbf{B_A} & \mathbf{r} \\ \mathbf{0} & -1 \end{bmatrix}$.

By Lemma 5.2, it holds that $\mathbf{A'}$ is statistically close to uniform. Then, by construction, the new column of $\mathbf{B}_{\mathbf{A}'}$ is linearly independent from the others and we maintain the equality $\mathbf{A'B}_{\mathbf{A}'} = \mathbf{0} \bmod q\mathbb{Z}$. It is then possible to re-randomize the trapdoor $\mathbf{B_A}$ while preserving the quality of the basis with a random unimodular matrix. It is also possible to change the basis in a way that it is independent of the matrix $\mathbf{A}$. For that, we can use $\mathbf{B_A}$ to sample sufficiently many Gaussian vectors so as to find a set of linearly independent ones that can be used to construct a new basis (albeit larger than the original one).

There are other ways of generating a close-to-uniform matrix $\mathbf{A}$ along with a short basis of $\mathcal{L}_q^\perp(\mathbf{A})$. A more efficient way was proposed in 2012 by Micciancio and Peikert [MP12]. We will see this method in TD 4 .

7.2.2 Trapdoor Usage

The motivation behind lattice trapdoors was to design signature schemes following the framework of [GPV08], which we present in Section 7.3. Trapdoor functions are however much broader than their use in signature schemes. We have seen for example that thanks to Klein's sampler, it is possible to sample discrete Gaussian vectors. These vectors can be used as side information allowing one to solve the LWE, SIS and ISIS problems, which remain hard without knowledge of the this trapdoor. Trapdoor functions have also been used in the design of advanced forms of signatures like group or blind signatures, but they can also be found in other constructions like vector commitment schemes, etc. Here, we only focus on the use of the trapdoor in solving LWE, SIS and ISIS.

Solving SIS

Assume that we are given a matrix $\mathbf{A} \in \mathbb{Z}_q^{d\times m}$ that is statistically close to a uniform matrix, and a trapdoor $\mathbf{B_A} \in \mathbb{Z}^{m\times m}$, i.e., short basis of $\mathcal{L}_q^\perp(\mathbf{A})$. Let $s \geq \left\|\mathsf{GSO}(\mathbf{B_A})\right\|_\infty \times \eta_\varepsilon(\mathbb{Z}^m)$, where $\eta_\varepsilon(\mathbb{Z}^m)$ is the smoothing parameter of $\mathbb{Z}^m$. Then, it holds that s is a large enough to use Theorem 4.2. By defining $\beta = s\sqrt{m}$, a challenger knowing $\mathbf{B_A}$ can efficiently solve $\mathrm{SIS}_{d,m,q,\beta}$ on the matrix $\mathbf{A}$.

Lemma 7.2 (Solving SIS with Trapdoors)

Let d, q, m be positive integers with q prime and $m \geq d$. Let $\mathbf{A} \in \mathbb{Z}_q^{d\times m}$ be primitive (i.e., surjective or such that $\mathbf{A}\mathbb{Z}_q^m = \mathbb{Z}_q^d$) and $\mathbf{B_A}$ be a basis of $\mathcal{L}_q^\perp(\mathbf{A})$. We let $s = \left\|\mathsf{GSO}(\mathbf{B_A})\right\|_\infty \cdot \sqrt{\ln(2m(1+\varepsilon^{-1})/\pi)}$ for some (negligible) $\varepsilon \in (0,1)$. Then, there exists an adversary $\mathcal{A}$ taking $\mathbf{A}$ and $\mathbf{B_A}$ which solves $\mathrm{SIS}_{d,m,q,\beta}$ with

$$\mathrm{Adv}_{\mathrm{SIS}}[\mathcal{A}] \geq 1 - \left(\frac{q^d}{(1-\varepsilon)s^m} + 2^{-2m}\frac{1}{2}\left(\left(\frac{1+\varepsilon/m}{1-\varepsilon/m}\right)^m - 1\right)\right),$$

and where $\beta = s\sqrt{m}$. In particular, when $m \geq (d\log_2 q + \lambda)/\log_2 s$, the advantage is overwhelming when ε is negligible.

Proof (Lemma 7.2). We construct the adversary $\mathcal{A}$ as follows. The adversary simply runs $\mathbf{x} \leftarrow \mathsf{Klein}(\mathbf{B_A}, s, \mathbf{0})$, and outputs $\mathbf{x}$. First, by Lemma 4.3, it holds that $s \geq \eta_\varepsilon(\mathbb{Z}^m)\left\|\mathsf{GSO}(\mathbf{B_A})\right\|_\infty$. Hence, by Theorem 4.2, the adversary is indeed polynomial time. We now have to check that $\mathbf{x}$ is indeed a valid solution. For that we will use the properties of the statistical distance (for example Lemma 1.4), as well as standard results on Gaussian distributions. We define $\delta = \Delta(\mathsf{Klein}(\mathbf{B_A}, s, \mathbf{0}), \mathcal{D}_{\mathcal{L}_q^\perp(\mathbf{A}),s})$, which is bounded by $(((1+\varepsilon/m)/(1-\varepsilon/m))^m - 1)/2$ according to Theorem 4.2. First, we naturally have $\mathbf{x} \in \mathcal{L}_q^\perp(\mathbf{A})$ which shows $\mathbf{Ax} = \mathbf{0} \bmod q\mathbb{Z}$.

Non-triviality. We first check that $\mathbf{x}$ is non-zero. We have the following inequalities. Let us first assume that $\mathbf{x}$ follows $\mathcal{D}_{\mathcal{L}_q^\perp(\mathbf{A}),s}$. We can now use Lemma 4.6 if we are able to show that $s \geq \eta_{\varepsilon'}(\mathcal{L})$ for some $\varepsilon' \in (0,1)$ and if we are able to compute $\mathrm{Vol}\,\mathcal{L}_q^\perp(\mathbf{A})$.

We have assumed that q is prime and that $\mathbf{A}$ is primitive, i.e., $\mathbf{A}\mathbb{Z}_q^m = \mathbb{Z}_q^d$. We note that the matrices $\mathbf{A}$ generated by $\mathsf{TrapGen}$ in Lemma 7.1 are primitive. In this case, this is enough to argue that $\mathrm{Vol}\,\mathcal{L}_q^\perp(\mathbf{A}) = q^d$. Then since $\left\|\mathsf{GSO}(\mathbf{B_A})\right\|_\infty \geq \lambda_{\mathsf{GSO}}(\mathcal{L}_q^\perp(\mathbf{A}))$, Lemma 4.3 gives that $s \geq \eta_\varepsilon(\mathcal{L}_q^\perp(\mathbf{A}))$. As a result, we have by Lemma 4.6 that

$$2^{-H_\infty(\mathcal{D}_{\mathcal{L}_q^\perp(\mathbf{A}),s})} \leq \frac{q^d}{(1-\varepsilon)s^m}.$$

If q is not prime or $\mathbf{A}$ not primitive, we can use [PR06, Lem. 2.11] instead of Lemma 4.6 which gives $H_\infty(\mathcal{D}_{\mathcal{L}_q^\perp(\mathbf{A}),s}) \geq m - \log_2 \frac{1+\varepsilon}{1-\varepsilon}$ if $s \geq 2\eta_\varepsilon(\mathcal{L}_q^\perp(\mathbf{A}))$. In any case, this reasoning on the min-entropy shows the Gaussian sample is zero with probability at most $\frac{q^d}{(1-\varepsilon)s^m}$.

Norm Bounds. We now verify the bounds. It holds that

$$\mathbb{P}_{\mathbf{x}\sim\mathcal{D}_{\mathcal{L}_q^\perp(\mathbf{A}),s}}[\|\mathbf{x}\|_2 > \beta] \leq 2^{-2m}$$

by Lemma 4.9.

Advantage. We can then bound the advantage of $\mathcal{A}$. It holds that

$$\begin{aligned}\mathbb{P}_{\mathbf{x}\sim\mathsf{Klein}(\mathbf{B_A},s,\mathbf{0})}[\mathbf{x} = \mathbf{0} \vee \|\mathbf{x}\|_2 > \beta] &\leq \mathbb{P}_{\mathbf{x}\sim\mathcal{D}_{\mathcal{L}_q^\perp(\mathbf{A}),s}}[\mathbf{x} = \mathbf{0} \vee \|\mathbf{x}\|_2 > \beta] + \delta \\ &\leq \mathbb{P}_{\mathbf{x}\sim\mathcal{D}_{\mathcal{L}_q^\perp(\mathbf{A}),s}}[\mathbf{x} = \mathbf{0}] + \mathbb{P}_{\mathbf{x}\sim\mathcal{D}_{\mathcal{L}_q^\perp(\mathbf{A}),s}}[\|\mathbf{x}\|_2 > \beta] + \delta \\ &\leq \frac{q^d}{(1-\varepsilon)s^m} + 2^{-2m} + \delta.\end{aligned}$$

As a result, we directly obtain that

$$\mathrm{Adv}_{\mathrm{SIS}}[\mathcal{A}] \geq 1 - \left(\frac{q^d}{(1-\varepsilon)s^m} + 2^{-2m} + \delta\right).$$

The term q^d/s^m is less than $1 - 1/\mathsf{poly}(d)$ if and only if $m \geq (\log_s q)/(1 - 1/\mathsf{poly}(d))$.

Solving LWE

We now see that one can efficiently solve the decisional variant of $\mathrm{LWE}_{d,q,m,\mathscr{D}_s,\mathscr{D}_e}$ on the matrix $\mathbf{A}^T$, provided that $\mathscr{D}_e$ is sufficiently small to enable distinguishing between $\mathcal{D}^T_{\mathcal{L}_q^\perp(\mathbf{A}),s}\mathscr{D}_e^m$ from uniform. The idea of this solver actually rely on the following reduction from $\mathrm{LWE}_{d,q,m,\mathscr{D}_s,\mathscr{D}_e}$ to $\mathrm{SIS}_{d,m,q,\beta}$.

Lemma 7.3 (Reduction from LWE to SIS)

Let d, q, m be positive integers, $\mathscr{D}_s$ a distribution over $\mathbb{Z}^d$, $\mathscr{D}_e$ a distribution over $\mathbb{Z}^m$, and β a positive real. Assume that $\mathscr{D}_e$ and m are such that $\mathbb{P}_{\mathbf{e}\sim\mathscr{D}_e^m}[\|\mathbf{e}\|_2 > \gamma] \leq \varepsilon$, for $\gamma \in [0, q/(4\beta)]$ and $\varepsilon \in [0, 1 - 1/\mathsf{poly}(d))$. Then, there exists a polynomial-time reduction from $\mathrm{LWE}_{d,q,m,\mathscr{D}_s,\mathscr{D}_e}$ to $\mathrm{SIS}_{d,m,q,\beta}$.

Proof (Lemma 7.3). See TD 2

Trapdoors can also be used to solve the search variant of LWE. If one is given a short enough basis of $\mathcal{L}_q^\perp(\mathbf{A})$, one can recover the error vector using linear algebra, and then solve sLWE.

Lemma 7.4 (Solving LWE with Trapdoors)

Let d, q, m be positive integers, $\mathscr{D}_s$ a distribution over $\mathbb{Z}^d$, $\mathscr{D}_e$ a distribution over $\mathbb{Z}^m$. Assume that $\mathscr{D}_e$ and m are such that $\mathbb{P}_{\mathbf{e}\sim\mathscr{D}_e^m}[\|\mathbf{e}\|_2 > \gamma] \leq \varepsilon$, for $\gamma > 0$ and ε negligible. Let $\mathbf{A} \in \mathbb{Z}_q^{d\times m}$ and $\mathbf{B_A}$ be a basis of $\mathcal{L}_q^\perp(\mathbf{A})$ such that $\|\mathbf{B_A}\|_\infty < q/(2\gamma)$. Then, there exists an adversary taking $\mathbf{A}$ and $\mathbf{B_A}$ and $\mathbf{t} = \mathbf{A}^T\mathbf{s} + \mathbf{e} \bmod q\mathbb{Z}$ for $\mathbf{s} \sim \mathscr{D}_s$ and $\mathbf{e} \sim \mathscr{D}_e^m$, which outputs $\mathbf{s}$ in polynomial-time.

Proof (Lemma 7.4). The adversary $\mathcal{A}$, first computes $\mathbf{c} = \mathbf{B}_\mathbf{A}^T\mathbf{t} \bmod q\mathbb{Z}$, where $\mathbf{c}$ is then the representative of $\mathbf{B}_\mathbf{A}^T\mathbf{t} \bmod q\mathbb{Z}$ with coefficients in $(-q/2, q/2)$. It then computes $\mathbf{e}^* = (\mathbf{B_A})^{-T}\mathbf{c}$ in the real field. It then recovers $\mathbf{s}^*$ by using linear algebra over $\mathbb{Z}_q$ to solve the system $\mathbf{A}^T\mathbf{s}^* = \mathbf{t} - \mathbf{e}^* \bmod q\mathbb{Z}$. It sends $\mathbf{s}^*$ as the output.

Let us now analyze the reduction. We have $\mathbf{c} = \mathbf{B}_\mathbf{A}^T\mathbf{A}^T\mathbf{s} + \mathbf{B}_\mathbf{A}^T\mathbf{e} \bmod q\mathbb{Z} = \mathbf{B}_\mathbf{A}^T\mathbf{e} \bmod q\mathbb{Z}$. Now, we need to find the unique representative of $\mathbf{B}_\mathbf{A}^T\mathbf{e} \bmod q\mathbb{Z}$ in $(-q/2, q/2)^m$. We show that this unique representative is exactly $\mathbf{B}_\mathbf{A}^T\mathbf{e}$ itself (over $\mathbb{Z}$), meaning that there is no reduction wrap-around modulo q. For that, let us look at $\left\|\mathbf{B}_\mathbf{A}^T\mathbf{e}\right\|_\infty$. Let i be in $[\![1, m]\!]$. It holds that

$$\left|[\mathbf{B}_\mathbf{A}^T\mathbf{e}]_i\right| = \left|[\mathbf{B_A}]_i^T\mathbf{e}\right| \leq \left\|[\mathbf{B_A}]_i\right\|_2 \|\mathbf{e}\|_2 ,$$

where $[\mathbf{B_A}]_i$ is the i-th column of $\mathbf{B_A}$, and where the inequality stems from Cauchy-Schwarz. Then, $\left\|[\mathbf{B_A}]_i\right\|_2$ is by definition bounded by $\|\mathbf{B_A}\|_\infty < q/(2\gamma)$. Then, with overwhelming probability of at least $1-\varepsilon$, it holds that $\|\mathbf{e}\|_2 \leq \gamma$. Hence, with probability at least $1-\varepsilon$, one has $\left\|\mathbf{B}_\mathbf{A}^T\mathbf{e}\right\|_\infty < q/2$ and thus that $\mathbf{c} = \mathbf{B}_\mathbf{A}^T\mathbf{e}$ over $\mathbb{Z}$.

Then, $\mathbf{e}^* = \mathbf{e}$, which means that $\mathbf{t} - \mathbf{e}^* \bmod q\mathbb{Z} = \mathbf{A}^T\mathbf{s} \bmod q\mathbb{Z}$ and as there is only one such $\mathbf{s}$, $\mathbf{s}^* = \mathbf{s}$.

One can also solve sLWE on the parity check matrix of $\mathbf{A}$ by using the duality results from [MM11] using the solver for ISIS that we will see just now. Other methods are however possible depending on the form of the trapdoor, e.g., [MP12].

Résoudre ISIS

The idea is very similar to that of solving SIS but by adjusting the Gaussian sampling step. The inhomogeneous version ISIS corresponds to finding a short vector in $\mathcal{L}_q^\mathbf{u}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{Ax} = \mathbf{u} \bmod q\mathbb{Z}\}$. This set is actually a coset of $\mathcal{L}_q^\perp(\mathbf{A})$, meaning that $\mathcal{L}_q^\mathbf{u}(\mathbf{A}) = \mathbf{x}^* + \mathcal{L}_q^\perp(\mathbf{A})$ for any $\mathbf{x}^*$ verifying $\mathbf{Ax}^* = \mathbf{u} \bmod q\mathbb{Z}$. This situation corresponds to the exhaustive solving of linear systems or differential equations. Indeed, $\mathbf{x}^*$ represents a *particular solution*, to which we add the set of *homogeneous solutions* ($\mathcal{L}_q^\perp(\mathbf{A})$). As a result, we just need to be able to sample a discrete Gaussian

over any coset to be able to solve ISIS for any $\mathbf{u} \in \mathbb{Z}_q^d$. To do so, we just make an observation on discrete Gaussian distributions, which we formalize in the following lemma.

Lemma 7.5 (Shifted Discrete Gaussian)

Let d be a positive integer, and $S \subset \mathbb{R}^d$ a countable set. Let $s > 0$ and $\mathbf{c}, \mathbf{a} \in \mathbb{R}^d$. It holds that $\mathbf{a} + \mathcal{D}_{S,s,\mathbf{c}-\mathbf{a}} = \mathcal{D}_{S+\mathbf{a},s,\mathbf{c}}$.

Proof (Lemma 7.5). We denote by P the distribution $\mathbf{a} + \mathcal{D}_{S,s,\mathbf{c}-\mathbf{a}}$. For all $\mathbf{x}^* \in \mathbb{R}^d$, it holds that

$$\mathbb{P}_{\mathbf{x}\sim P}[\mathbf{x} = \mathbf{x}^*] = \mathbb{P}_{\mathbf{y}\sim\mathcal{D}_{S,s,\mathbf{c}-\mathbf{a}}}[\mathbf{y} = \mathbf{x}^* - \mathbf{a}] = \mathbf{1}_S(\mathbf{x}^* - \mathbf{a})\frac{\rho_{s,\mathbf{c}-\mathbf{a}}(\mathbf{x}^* - \mathbf{a})}{\rho_{s,\mathbf{c}-\mathbf{a}}(S)}.$$

Yet, we also have $\rho_{s,\mathbf{c}-\mathbf{a}}(\mathbf{x} - \mathbf{a}) = \rho_{s,\mathbf{c}}(\mathbf{x} - \mathbf{a} + \mathbf{a}) = \rho_{s,\mathbf{c}}(\mathbf{x})$ pour tout $\mathbf{x} \in \mathbb{R}^d$, et $\rho_{s,\mathbf{c}-\mathbf{a}}(S) = \rho_{s,\mathbf{c}}(S + \mathbf{a})$. Cela prouve donc que

$$\mathbb{P}_{\mathbf{x}\sim P}[\mathbf{x} = \mathbf{x}^*] = \mathbf{1}_S(\mathbf{x}^* - \mathbf{a})\frac{\rho_{s,\mathbf{c}}(\mathbf{x}^*)}{\rho_{s,\mathbf{c}}(S + \mathbf{a})} = \mathbf{1}_{S+\mathbf{a}}(\mathbf{x}^*)\frac{\rho_{s,\mathbf{c}}(\mathbf{x}^*)}{\rho_{s,\mathbf{c}}(S + \mathbf{a})} = \mathbb{P}_{\mathbf{x}\sim\mathcal{D}_{S+\mathbf{a},s,\mathbf{c}}}[\mathbf{x} = \mathbf{x}^*].$$

On en conclue donc que $P = \mathcal{D}_{S+\mathbf{a},s,\mathbf{c}}$.

The procedure to solve ISIS is then as follows. First, compute $\mathbf{x}_0 \in \mathbb{Z}^m$ such that $\mathbf{A}\mathbf{x}_0 = \mathbf{u} \bmod q\mathbb{Z}$ using linear algebra (without constraints on the size of $\mathbf{x}_0$). Then, sample $\mathbf{x}_1$ using $\mathsf{Klein}(\mathbf{B}_\mathbf{A}, s, -\mathbf{x}_0)$ and compute $\mathbf{x} = \mathbf{x}_0 + \mathbf{x}_1$. Since $\mathbf{x}_1$ is statistically close to $\mathcal{D}_{\mathcal{L}_q^\perp(\mathbf{A}),s,-\mathbf{x}_0}$, Lemma 7.5 directly gives that $\mathbf{x}$ is statistically close to $\mathcal{D}_{\mathcal{L}_q^\mathbf{u}(\mathbf{A}),s}$. In particular, we have $\mathbf{A}\mathbf{x} = \mathbf{u} \bmod q\mathbb{Z}$, and by Lemma 4.9, since $s \geq \eta_\varepsilon(\mathcal{L}_q^\perp(\mathbf{A}))$, we get $\|\mathbf{x}\|_2 = \|\mathbf{x}_1 - (-\mathbf{x}_0)\|_2 \leq s\sqrt{m}$.

Remark 7.2

An immediate corollary of Lemma 7.5 is that the conditional probability distribution of $\mathbf{x} \sim \mathcal{D}_{\mathbb{Z}^m,s}$ given $\mathbf{A}\mathbf{x} = \mathbf{u} \bmod q\mathbb{Z}$ is exactly $\mathcal{D}_{\mathcal{L}_q^\mathbf{u}(\mathbf{A}),s}$. Formally:

$$\forall \mathbf{x}^* \in \mathbb{R}^m, \mathbb{P}_{\mathbf{x}\sim\mathcal{D}_{\mathbb{Z}^m,s}}[\mathbf{x} = \mathbf{x}^* | \mathbf{A}\mathbf{x} = \mathbf{u} \bmod q\mathbb{Z}] = \mathbb{P}_{\mathbf{x}\sim\mathcal{D}_{\mathcal{L}_q^\mathbf{u}(\mathbf{A}),s}}[\mathbf{x} = \mathbf{x}^*]$$

This result is for example stated in [GPV08, Lem. 5.2].

7.3 GPV Hash-and-Sign Paradigm

The *Hash-and-Sign* signature paradigm is not specific to lattices. What it describes is the family of signature schemes for which the signing procedure essentially consists in hashing the message to a particular hash space, and applying (more or less in black box) a signature procedure on this message digest. The is for example how the RSA [RSA78] signature scheme works in PKCS#1.5. The version from PKCS#2.1, called RSA-PSS (for RSA Probabilistic Signature Scheme), can also be seen as a hash-and-sign signature, albeit slightly more complex in essence. The ECDSA [JMV01] is also considered hash-and-sign signature schemes. Over lattices, the first hash-and-sign signature that was proposed was called NTRUSign (or PASS) [HS01] which was broken a year later. In 2008, Gentry, Peikert and Vaikuntanathan [GPV08] proposed a framework to design sEUF-CMA signatures in the random oracle model based on preimage sampleable trapdoor functions, and instantiated this framework over lattices. This is to date one of main signature paradigm in lattice-based cryptography. We now describe an instantiation of the GPV framework with trapdoor functions over lattices.

7.3.1 Description

We describe the signature scheme using the algorithm $\mathsf{TrapGen}$, which results in quite large parameters. There has been more efficient instantiations of the GPV framework, but they all follow more or less the same structure by just changing the trapdoor generation mechanism (and the accompanying preimage sampler). We describe the $\mathsf{KeyGen}, \mathsf{Sign}$, and Verify algorithms in Algorithms 7.4, 7.5 and 7.6.

Algorithm 7.4: KeyGen (GPV)

Input: Integers d, q, m, with q prime and $m = \Omega(d \log_2 q)$, and $s > 0$ a Gaussian parameter.

1. $(\mathbf{A}, \mathbf{B_A}) \leftarrow \mathsf{TrapGen}(d, m, q) \in \mathbb{Z}_q^{d \times m} \times \mathbb{Z}^{m \times m}$

Output: $\mathsf{pk} = \mathbf{A}$, $\mathsf{sk} = \mathbf{B_A}$

Algorithm 7.5: Sign (GPV)

Input: Public parameters, Public Key $\mathsf{pk} = \mathbf{A} \in \mathbb{Z}_q^{d \times m}$, Secret Key $\mathsf{sk} = \mathbf{B_A} \in \mathbb{Z}^{m \times m}$, Message $M \in \{0,1\}^*$.

1. **If** M was already queried, let $\mathbf{x} \in \mathbb{Z}^m$ be the corresponding signature
2. **Else** Solve the equation $\mathbf{Ac} = \mathcal{H}(M) \bmod q\mathbb{Z}$ for $\mathbf{c}$, and set $\mathbf{x} \leftarrow \mathbf{c} + \mathsf{Klein}(\mathbf{B_A}, s, -\mathbf{c})$

Output: $\mathsf{sig} = \mathbf{x}$

Algorithm 7.6: Verify (GPV)

Input: Public parameters, Public Key $\mathsf{pk} = \mathbf{A} \in \mathbb{Z}_q^{d \times m}$, Signature $\mathsf{sig} = \mathbf{x} \in \mathbb{Z}^m$, Message $M \in \{0,1\}^*$.

1. $b \leftarrow (\mathbf{Ax} = \mathcal{H}(M) \bmod q\mathbb{Z}) \wedge (\|\mathbf{x}\|_2 \leq s\sqrt{m})$

Output: $b \in \{0,1\}$

The idea is to hash the message with a cryptographic hash function, and compute a short Gaussian preimage of $\mathcal{H}(M)$ by $\mathbf{A}$. In this presentation of the scheme, the signature is called *stateful*. This means that the signer must keep track of a *state* among all the emitted signatures. Here the state essentially keeps track of all the messages that have been queried for signature. The reason for it is to avoid emitting two different signature on the same message, which would allow a user to solve SIS on the public key matrix $\mathbf{A}$. Indeed, assume $\mathcal{A}$ has obtained two signatures $\mathbf{x} \neq \mathbf{x}'$ on a message M such that $\mathsf{Verify}(\mathbf{A}, \mathbf{x}, M) = \mathsf{Verify}(\mathbf{A}, \mathbf{x}', M) = 1$. Then, one would obtain $\mathbf{A}(\mathbf{x} - \mathbf{x}') = \mathbf{0} \bmod q\mathbb{Z}$ and $\|\mathbf{x} - \mathbf{x}'\|_2 \leq 2s\sqrt{m}$ for $\mathbf{x} - \mathbf{x}' \neq \mathbf{0}$. This stateful variant can be turned into a *stateless* variant, where there is no longer a need to keep track of state, by simply adding salt in the hash. One would then sample $\mathbf{x}$ for the syndrome $\mathcal{H}(r\|M)$ where r is a uniformly random bitstring in $\{0,1\}^k$. The value of k is used to adjust the probability of finding collisions on the hash function, i.e., finding $r \neq r'$ such that $\mathcal{H}(r\|M) = \mathcal{H}(r'\|M)$. Typically, a conservative choice would be $k = 2\lambda$ where λ is the security parameter. Additionally, in this version, the salt r must be provided in addition to the preimage to allow for verification. Hence, the signature would be $\mathsf{sig} = (r, \mathbf{x})$, which is only k bits larger (typically 256 bits).

For clarity, we define the algorithm $\mathsf{SamplePre}$ taking as input $\mathbf{B_A}, \mathbf{A}, s, \mathbf{y}$ and that performs the following: solve the equation $\mathbf{Ax}_0 = \mathbf{y} \bmod q\mathbb{Z}$, sample $\mathbf{x}_1 = \mathsf{Klein}(\mathbf{B_A}, s, -\mathbf{x}_0)$ and output $\mathbf{x} = \mathbf{x}_0 + \mathbf{x}_1$. This algorithm thus samples a preimage of $\mathbf{y}$ for the matrix $\mathbf{A}$ following a distribution close to $\mathcal{D}_{\mathcal{L}_q^{\mathbf{y}}(\mathbf{A}),s}$.

7.3.2 Security Analysis

Let us now prove the security requirements for the signature scheme. We need to verify that the scheme is both correct and EUF-CMA secure. It turns out that the scheme can be proven to be *strong* EUF-CMA secure. The correctness directly relies on the use of trapdoors to solve ISIS. Here, we use the secret trapdoor $\mathbf{B_A}$ to solve the ISIS instance $(\mathbf{A}, \mathcal{H}(M))$. The signatures are thus statistically close to $\mathcal{D}_{\mathcal{L}_q^{\mathcal{H}(M)}(\mathbf{A}),s}$ proving that $b = 1$ for honestly generated signatures.

We focus on proving the sEUF-CMA security. For that we will need the following technical lemma which argues that the signatures do not leak information on the trapdoor $\mathbf{B_A}$. This is called the *simulation result* of the preimage sampler, and it is necessary in ensuring that the scheme is secure. If signatures leak an ever so slight amount of information on $\mathbf{B_A}$, we have to consider the fact that an adversary may see a lot of signatures. Typically, NIST requirements state that a signature scheme should be secure for at least $Q = 2^{64}$ emitted signatures. This is a very conservative choice but it should be considered when deriving the parameters, as we will see in the security proof.

Lemma 7.6 (Simulating Signatures)

Let d, q, m be positive integers with q prime and $m = \Omega(d \log_2 q)$. Let $s > O(\sqrt{d \log_2 q}) \cdot \sqrt{\ln(2m(1+\varepsilon^{-1})/\pi)}$ a Gaussian parameter for some $\varepsilon \in (0,1)$. Let $(\mathbf{A}, \mathbf{B_A}) \leftarrow \mathsf{TrapGen}(d, q, m)$. We define the following distributions.

$\mathcal{P}_0$**:** $\mathbf{u} \hookleftarrow U(\mathbb{Z}_q^d)$, and $\mathbf{x} \leftarrow \mathsf{SamplePre}(\mathbf{B_A}, \mathbf{A}, s, \mathbf{u})$. Output $(\mathbf{x}, \mathbf{u})$.

$\mathcal{P}_1$**:** $\mathbf{x} \hookleftarrow \mathcal{D}_{\mathbb{Z}^m, s}$, $\mathbf{u} \leftarrow \mathbf{A}\mathbf{x} \bmod q\mathbb{Z}$. Output $(\mathbf{x}, \mathbf{u})$.

Then, it holds that for all $(\mathbf{x}, \mathbf{u}) \in \mathrm{Supp}(\mathcal{P}_0) = \mathrm{Supp}(\mathcal{P}_1)$, we have $\mathcal{P}_0(\mathbf{x}, \mathbf{u}) \in [\delta_1, \delta_2]\mathcal{P}_1(\mathbf{x}, \mathbf{u})$, with

$$\delta_1 = \frac{1-\varepsilon}{1+\varepsilon} \underset{\varepsilon \to 0^+}{\sim} 1 - 2\varepsilon \quad \text{and} \quad \delta_2 = \left(\frac{1+\varepsilon/m}{1-\varepsilon/m}\right)^m \frac{1+\varepsilon}{1-\varepsilon} \underset{\varepsilon \to 0^+}{\sim} 1 + 4\varepsilon$$

In particular, it gives $\Delta(\mathcal{P}_0, \mathcal{P}_1) \leq (\delta_2 - 1)/2 \approx 2\varepsilon$.

Proof (Lemma 7.6). First of all, because of the condition on s, we can combine Lemmas 4.3 and 7.1 to obtain that $s \geq \eta_\varepsilon(\mathbb{Z}^m) \left\| \mathsf{GSO}(\mathbf{B_A}) \right\|_\infty \geq \eta_\varepsilon(\mathcal{L}_q^\perp(\mathbf{A}))$. In $\mathcal{P}_0$, if $\mathbf{x}_0$ denotes a solution of $\mathbf{A}\mathbf{x}_0 = \mathbf{u} \bmod q\mathbb{Z}$ computed in $\mathsf{SamplePre}$ for a fixed $\mathbf{u}$, then $\mathbf{x}_1 = \mathsf{Klein}(\mathbf{B_A}, s, -\mathbf{x}_0)$ verifies

$$\mathbb{P}_{\mathbf{x}_1 \sim \mathsf{Klein}(\mathbf{B_A}, s, -\mathbf{x}_0)}[\mathbf{x}_1 = \mathbf{x}_1^\star] \in \left[1, \left(\frac{1+\varepsilon/m}{1-\varepsilon/m}\right)^m\right] \cdot \mathcal{D}_{\mathcal{L}_q^\perp(\mathbf{A}), s, -\mathbf{x}_0}(\mathbf{x}_1^\star),$$

which is a stronger version of Theorem 4.2. Indeed, we here obtain a relation on the marginal distributions and not simply the statistical distance. Then, we have $\mathsf{SamplePre}(\mathbf{B_A}, \mathbf{A}, s, \mathbf{u})(\mathbf{x}^\star) \in [1, \delta]\mathcal{D}_{\mathcal{L}_q^\mathbf{u}(\mathbf{A}), s}(\mathbf{x}^\star)$ by Lemma 7.5, where $\delta = (1+\varepsilon/m)^m/(1-\varepsilon/m)^m$. Note that this is conditioned on a specific value for $\mathbf{u}$. Considering the distribution of $\mathbf{u}$, we get

$$\begin{aligned}
\mathbb{P}_{(\mathbf{x},\mathbf{u}) \sim \mathcal{P}_0}[(\mathbf{x}, \mathbf{u}) = (\mathbf{x}^\star, \mathbf{u}^\star)] &= \mathbb{P}_{\mathbf{u} \sim U(\mathbb{Z}_q^d)}[\mathbf{u} = \mathbf{u}^\star] \cdot \mathbb{P}_{\mathbf{u}, \mathbf{x} \sim \mathsf{SamplePre}(\mathbf{B_A}, \mathbf{A}, s, \mathbf{u})}[\mathbf{x} = \mathbf{x}^\star | \mathbf{u} = \mathbf{u}^\star] \\
&= q^{-d} \cdot \mathsf{SamplePre}(\mathbf{B_A}, \mathbf{A}, s, \mathbf{u}^\star)(\mathbf{x}^\star) \\
&\in [1, 1+\varepsilon'] q^{-d} \cdot \mathcal{D}_{\mathcal{L}_q^{\mathbf{u}^\star}(\mathbf{A}), s}(\mathbf{x}^\star).
\end{aligned}$$

We now look at $\mathcal{P}_1$. We have the following equalities.

$$\begin{aligned}
\mathbb{P}_{(\mathbf{x},\mathbf{u}) \sim \mathcal{P}_1}[(\mathbf{x}, \mathbf{u}) = (\mathbf{x}^\star, \mathbf{u}^\star)] &= \mathbb{P}_{\mathbf{x} \sim \mathcal{D}_{\mathbb{Z}^m, s}}[\mathbf{A}\mathbf{x} = \mathbf{u}^\star \bmod q\mathbb{Z}] \cdot \mathbb{P}_{\mathbf{x} \sim \mathcal{D}_{\mathbb{Z}^m, s}}[\mathbf{x} = \mathbf{x}^\star | \mathbf{A}\mathbf{x} = \mathbf{u}^\star \bmod q\mathbb{Z}] \\
&= \mathbb{P}_{\mathbf{x} \sim \mathcal{D}_{\mathbb{Z}^m, s}}[\mathbf{A}\mathbf{x} = \mathbf{u}^\star \bmod q\mathbb{Z}] \cdot \mathbb{P}_{\mathbf{x} \sim \mathcal{D}_{\mathcal{L}_q^{\mathbf{u}^\star}(\mathbf{A}), s}}[\mathbf{x} = \mathbf{x}^\star] \\
&= \mathbb{P}_{\mathbf{x} \sim \mathcal{D}_{\mathbb{Z}^m, s}}[\mathbf{A}\mathbf{x} = \mathbf{u}^\star \bmod q\mathbb{Z}] \cdot \mathcal{D}_{\mathcal{L}_q^{\mathbf{u}^\star}(\mathbf{A}), s}(\mathbf{x}^\star),
\end{aligned}$$

where the second equality holds from Remark 7.2. We then have to evaluate $\mathbb{P}_{\mathbf{x} \sim \mathcal{D}_{\mathbb{Z}^m, s}}[\mathbf{A}\mathbf{x} = \mathbf{u}^* \bmod q\mathbb{Z}]$. Since $\mathbf{A}$ is generated from $\mathsf{TrapGen}$, it holds that $\mathbf{A}$ is primitive, i.e., $\mathbf{A}\mathbb{Z}_q^m = \mathbb{Z}_q^d$. Using Lemma 4.5 on $\mathcal{L} = \mathbb{Z}^m$ and $\mathcal{L}' = \mathcal{L}_q^\perp(\mathbf{A})$, and noticing that $\mathbb{Z}^m / \mathcal{L}_q^\perp(\mathbf{A})$ is isomorphic to $\mathbb{Z}_q^d$ by the mapping $\mathbf{e} + \mathcal{L}_q^\perp(\mathbf{A}) \mapsto \mathbf{A}\mathbf{e} \bmod q\mathbb{Z}$, we have that

$$\mathbb{P}_{\mathbf{x} \sim \mathcal{D}_{\mathbb{Z}^m, s}}[\mathbf{A}\mathbf{x} = \mathbf{u}^\star \bmod q\mathbb{Z}] \in \left[\frac{1-\varepsilon}{1+\varepsilon}, \frac{1+\varepsilon}{1-\varepsilon}\right] \cdot q^{-d}.$$

It directly yields

$$\mathbb{P}_{(\mathbf{x},\mathbf{u}) \sim \mathcal{P}_1}[(\mathbf{x}, \mathbf{u}) = (\mathbf{x}^\star, \mathbf{u}^\star)] \in \left[\frac{1-\varepsilon}{1+\varepsilon}, \frac{1+\varepsilon}{1-\varepsilon}\right] q^{-d} \cdot \mathcal{D}_{\mathcal{L}_q^{\mathbf{u}^\star}(\mathbf{A}), s}(\mathbf{x}^\star).$$

Combining both yields $\mathcal{P}_0(\mathbf{x}, \mathbf{u}) \in [\delta_1, \delta_2]\mathcal{P}_1(\mathbf{x}, \mathbf{u})$ for

$$\delta_1 = \frac{1-\varepsilon}{1+\varepsilon} \quad \text{and} \quad \delta_2 = \left(\frac{1+\varepsilon/m}{1-\varepsilon/m}\right)^m \frac{1+\varepsilon}{1-\varepsilon}$$

The statistical distance directly follows by definition.

We can now prove the sEUF-CMA security of the GPV signature scheme (with this instantiation of trapdoors and preimage sampler). We refer to [GPV08] for the more general proof.

Theorem 7.1 (GPV Signature sEUF-CMA Security)

Let d, q, m be positive integers with q prime and $m = \Omega(d \log_2 q)$. Let $s > O(\sqrt{d \log_2 q}) \cdot \sqrt{\ln(2m(1+\varepsilon^{-1})/\pi)}$ a Gaussian parameter for some $\varepsilon \in (0,1)$. Let Q be the maximal number of signatures per key pair. Then, the GPV signature scheme is (t, δ)-sEUF-CMA secure in the Random Oracle Model where

$$\delta \leq \delta_2^Q \cdot \left(2^{-\Omega(d)} + 2\frac{q^d}{s^m} + \frac{\varepsilon_{\mathrm{SIS}}(t)}{1 - 2^{-2m}} \right),$$

with $\varepsilon_{\mathrm{SIS}}(t)$ the hardness bound of $\mathrm{SIS}_{d,m,q,\beta}$ for $\beta = 2s\sqrt{m}$ and

$$\delta_2 = \left(\frac{1+\varepsilon/m}{1-\varepsilon/m} \right)^m \frac{1+\varepsilon}{1-\varepsilon} \underset{\varepsilon \to 0^+}{\sim} 1 + 4\varepsilon$$

Proof (Theorem 7.1). Let $\mathcal{A}$ be a t-time adversary against the sEUF-CMA security of the signature scheme. We denote by $\mathcal{B}$ the challenger which answers both the signing queries and random oracle queries. Additionally, we assume that $\mathcal{A}$ has made a random oracle query on the message $M^\star$ that it outputs a forgery on. We first proceed by game hops to modify the sEUF-CMA game, and in the last game we use SIS to bound the advantage of the adversary. We define the following games between $\mathcal{A}$ and $\mathcal{B}$ by only looking at the setup and queries stages. The challenger $\mathcal{B}$ maintains two tables, one $T_{\mathcal{RO}}$ for the random oracle queries and one for the signature queries T_S.

Game G_0. We recall the setup and queries in the original sEUF-CMA game.

- *Setup.* $\mathcal{B}$ runs $(\mathbf{A}, \mathbf{B_A}) \leftarrow \mathsf{KeyGen}(d, q, m, s)$ and sends $\mathsf{pk} = \mathbf{A}$ to $\mathcal{A}$.
- *Random Oracle Queries.* On input $M \in \{0,1\}^*$, $\mathcal{B}$ checks if M appears in $T_{\mathcal{RO}}$. If so, it directly outputs the syndrome $\mathbf{u}$ from $T_{\mathcal{RO}}$ corresponding to M. If not it samples $\mathbf{u} \hookleftarrow U(\mathbb{Z}_q^d)$ and stores the key-value pair $(M, \mathbf{u})$ in $T_{\mathcal{RO}}$ before sending $\mathbf{u}$ to $\mathcal{A}$.
- *Signing Queries.* On input $M \in \{0,1\}^*$, $\mathcal{B}$ first checks if it appears in T_S. If so it outputs the $\mathbf{x}$ from T_S corresponding to M. If not it proceeds as follows. If M appears in $T_{\mathcal{RO}}$, it gets the corresponding hash $\mathbf{u}$, and if not it samples $\mathbf{u} \hookleftarrow U(\mathbb{Z}_q^d)$ and stores $(M, \mathbf{u})$ in $T_{\mathcal{RO}}$. Then it runs the rest of the signing algorithm by computing $\mathbf{x} \leftarrow \mathsf{SamplePre}(\mathbf{B_A}, \mathbf{A}, s, \mathbf{u})$. It then stores $(M, \mathbf{x})$ in T_S and sends $\mathbf{x}$ to $\mathcal{A}$.

Game G_1. We change the original game by simulating the signatures and random oracle queries, by relying on Lemma 7.6. The setup stage is identical to that of G_0.

- *Random Oracle Queries.* On input $M \in \{0,1\}^*$, $\mathcal{B}$ checks if M appears in $T_{\mathcal{RO}}$. If so, it directly outputs the syndrome $\mathbf{u}$ from $T_{\mathcal{RO}}$ corresponding to M. If not it samples $\mathbf{x} \hookleftarrow \mathcal{D}_{\mathbb{Z}^m, s}$, and computes $\mathbf{u} = \mathbf{A}\mathbf{x} \bmod q\mathbb{Z}$. It then stores $(M, \mathbf{u})$ in $T_{\mathcal{RO}}$, and $(M, \mathbf{x})$ in T_S before sending $\mathbf{u}$ to $\mathcal{A}$.
- *Signing Queries.* On input $M \in \{0,1\}^*$, $\mathcal{B}$ first checks if it appears in T_S. If so it outputs the $\mathbf{x}$ from T_S corresponding to M. If not, this means that M does not appear in $T_{\mathcal{RO}}$ either. It thus samples $\mathbf{x} \hookleftarrow \mathcal{D}_{\mathbb{Z}^m, s}$, and computes $\mathbf{u} = \mathbf{A}\mathbf{x} \bmod q\mathbb{Z}$. It then stores $(M, \mathbf{u})$ in $T_{\mathcal{RO}}$, and $(M, \mathbf{x})$ in T_S before sending $\mathbf{x}$ to $\mathcal{A}$.

Game G_2. We finish by simulating the public key $\mathbf{A}$. Concretely, as the trapdoor is no longer used to generate signatures, we can sample the public key $\mathbf{A}$ uniformly without needing a trapdoor.

- *Setup.* $\mathcal{B}$ samples $\mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{d \times m})$ and sets $\mathsf{pk} = \mathbf{A}$ before sending it to $\mathcal{A}$.

We now argue why these games are indistinguishable to the adversary. This comes down to studying the distribution of the view of the adversary which is composed of pk, the syndromes obtained from random oracle queries, and the queried signatures.

G_0-G_1. The view of $\mathcal{A}$ only changes on the syndromes and preimages. In particular, the pairs $(\mathbf{x}, \mathbf{u})$ follow $\mathcal{P}_0$ in G_0 and $\mathcal{P}_1$ in G_1, where distributions $\mathcal{P}_0, \mathcal{P}_1$ are that of Lemma 7.6. The latter lemma gives

$$\mathrm{Adv}^{G_0}_{sEUF-CMA}[\mathcal{A}] \leq \delta_2^Q \mathrm{Adv}^{G_1}_{sEUF-CMA}[\mathcal{A}],$$

where $\delta_2 = \left(\frac{1+\varepsilon/m}{1-\varepsilon/m}\right)^m \frac{1+\varepsilon}{1-\varepsilon} \underset{\varepsilon \to 0^+}{\sim} 1 + 4\varepsilon$.

G_1-G_2. The view of $\mathcal{A}$ only changes on the public key. By Lemma 7.1, it directly holds that $\mathbf{A}$ is statistically close to uniform and thus that

$$\mathrm{Adv}^{G_1}_{sEUF-CMA}[\mathcal{A}] \leq \mathrm{Adv}^{G_2}_{sEUF-CMA}[\mathcal{A}] + 2^{-\Omega(d)}$$

Bounding advantage in G_2. We now bound the advantage of $\mathcal{A}$ in G_2 by exploiting the forgery to produce a SIS solution on the matrix $\mathbf{A}$. More precisely, after having queried at most Q signatures, $\mathcal{A}$ outputs $(M^\star, \mathbf{x}^\star)$ such that $\mathsf{Verify}(\mathbf{A}, \mathbf{x}^\star, M^\star) = 1$ and $(M^\star, \mathbf{x}^\star) \neq (M_i, \mathbf{x}_i)$ for all the signature queries $i \in [\![1, Q]\!]$. Because $M^\star$ has been queried to the random oracle, the must exist pairs $(M^\star, \mathbf{u}')$ and $(M^\star, \mathbf{x}')$ in $T_{\mathcal{RO}}$ and T_S respectively. Then, $\mathcal{B}$ computes $\mathbf{x} = \mathbf{x}^\star - \mathbf{x}'$ and outputs it as a solution to $\mathrm{SIS}_{d,q,m,\beta}$ for the instance $\mathbf{A}$ (which is indeed uniform as needed per Definition 5.1).

If $M^\star$ was queried to the signature oracle, there must exist $i \in [\![1, Q]\!]$ such that $M_i = M^\star$ and $\mathbf{x}_i = \mathbf{x}'$. Because $(M^\star, \mathbf{x}^\star) \neq (M_i, \mathbf{x}_i)$, we directly have $\mathbf{x}^\star \neq \mathbf{x}'$ and thus $\mathbf{x} \neq \mathbf{0}$. Then, because $\mathbf{x}^\star$ is a valid signature on $M^\star$, we have $\|\mathbf{x}^\star\|_2 \leq s\sqrt{m}$. Moreover, Lemmma 4.9 yields $\|\mathbf{x}'\|_2 \leq s\sqrt{m}$ with probability at least $1 - 2^{-2m}$. Conditioned on this event, we have $\|\mathbf{x}\|_2 \leq 2s\sqrt{m} = \beta$. Finally, it holds that $\mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{x}^\star - \mathbf{u}' \bmod q\mathbb{Z} = \mathbf{0} \bmod q\mathbb{Z}$. Hence $\mathbf{x}$ is a valid solution of $\mathrm{SIS}_{d,q,m,\beta}$.

If $M^\star$ was never queried to the signature oracle, then $\mathcal{A}$ never received the corresponding $\mathbf{x}'$. However, the entropy of $\mathbf{x}'$ knowing $\mathbf{u}' = \mathbf{A}\mathbf{x}' \bmod q\mathbb{Z}$ is at least $H_\infty(\mathbf{x}') - d\log_2 q \geq m\log_2 s - d\log_2 q - 1$ by Lemma 4.6. So the probability that $\mathbf{x}' = \mathbf{x}^\star$ is at most $2q^d/s^m$. The other conditions are verified the same way as above.

We thus have

$$\begin{aligned}
\mathrm{Adv}_{\mathrm{SIS}}[\mathcal{B}] &= \mathbb{P}_{\mathbf{x}'}\left[\|\mathbf{x}'\|_2 \leq s\sqrt{m}\right]\mathbb{P}_{\mathbf{x}'}\left[\mathbf{x} \text{ solution} \middle| \|\mathbf{x}'\|_2 \leq s\sqrt{m}\right] \\
&\quad + \mathbb{P}_{\mathbf{x}'}\left[\|\mathbf{x}'\|_2 > s\sqrt{m}\right]\mathbb{P}_{\mathbf{x}'}\left[\mathbf{x} \text{ solution} \middle| \|\mathbf{x}'\|_2 > s\sqrt{m}\right] \\
&\geq (1 - 2^{-2m})(\mathrm{Adv}^{G_2}_{sEUF-CMA}[\mathcal{A}] - 2q^d/s^m)
\end{aligned}$$

which yields

$$\mathrm{Adv}^{G_2}_{sEUF-CMA}[\mathcal{A}] \leq 2\frac{q^d}{s^m} + \varepsilon_{\mathrm{SIS}}(t)\frac{1}{1 - 2^{-2m}}.$$

Combining everything gives

$$\mathrm{Adv}^{G_0}_{sEUF-CMA}[\mathcal{A}] \leq \delta_2^Q \cdot \left(2^{-\Omega(d)} + 2\frac{q^d}{s^m} + \frac{\varepsilon_{\mathrm{SIS}}(t)}{1 - 2^{-2m}}\right),$$

as claimed. Choosing $\varepsilon \leq 1/4Q$, is sufficient ot make δ_2^Q constant in the security parameter.

7.4 Standard Model Signatures

The signatures that we have presented in Sections 7.1 and 7.3 are using hash functions which must be modeled as random oracles for the security proof to go through. Several works have proposed alternative constructions to remove the need for the random oracle model in the security proof, but it usually results in less efficient constructions. We can for example cite [Boy10] or [CHKP10] which proposed lattice-based signatures in the standard model and whose security rely on SIS using rather elegant and sophisticated arguments. However, they are outside of the scope of this course, and we leave them to explore in the exercise sessions.

RECAP

- There exists two main lattice signature design paradigms: Fiat-Shamir with Aborts (Lyubashevsky), and Hash-and-Sign (GPV).
- Lyubashevsky's signature scheme is a variant of Schnorr's signature by adding a rejection sampling step necessary for security. The keys are $\mathsf{pk} = (\mathbf{A}, \mathbf{AS})$ and $\mathsf{sk} = \mathbf{S} \in \{0, \pm 1\}^{m \times k}$. To sign, we sample $\mathbf{y} \hookleftarrow \mathcal{D}_{\mathbb{Z}^m, s}$, compute $\mathbf{c} = \mathcal{H}(\mathsf{pk} \| \mathbf{Ay} \| M)$ and then $\mathbf{z} = \mathbf{y} + \mathbf{Sc}$. If we reject $\mathbf{z}$ with probability $P(\mathbf{z})$, we restart. To verify the signature, we check that $\|\mathbf{z}\|_2 \leq s\sqrt{m}$ and $\mathbf{c} = \mathcal{H}(\mathsf{pk} \| \mathbf{Az} - \mathbf{Bc} \| M)$.
- Lyubashevsky's signature is EUF-CMA in the random oracle model under the SIS assumption.
- A trapdoor function is easy to compute but hard to invert without knowledge of the trapdoor. In lattice cryptography, a trapdoor is generally a short basis of the considered lattice. The function $\mathbf{x} \mapsto \mathbf{Ax} \bmod q\mathbb{Z}$ is a trapdoor function, where the trapdoors are short bases of $\mathcal{L}_q^\perp(\mathbf{A}) = \{\mathbf{x} : \mathbf{Ax} = \mathbf{0} \bmod q\mathbb{Z}\}$. In that case, we can use the trapdoor to solve SIS, ISIS, sLWE, LWE for the matrix $\mathbf{A}$.
- A trapdoor allows for sampling short preimages with Klein's algorithm.
- The GPV signature scheme has keys of the form $\mathsf{pk} = \mathbf{A}$ close to uniform, and $\mathsf{sk} = \mathbf{B_A}$ a trapdoor on $\mathbf{A}$. To sign, we hash the message M into a syndrome $\mathbf{u} = \mathcal{H}(M)$, and then use Klein's sampler to sample a Gaussian vector $\mathbf{x}$ such that $\mathbf{Ax} = \mathbf{u} \bmod q\mathbb{Z}$. To verify the signature, we check that $\|\mathbf{x}\|_2 \leq s\sqrt{m}$ and $\mathbf{Ax} = \mathcal{H}(M) \bmod q\mathbb{Z}$.
- The GPV signature is EUF-CMA (and even sEUF-CMA) in the random oracle model under the SIS.

Bibliography

[AP09] J. Alwen and C. Peikert. Generating Shorter Bases for Hard Random Lattices. In *STACS*, 2009.

[BDL+11] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang. High-speed high-security signatures. In *CHES*, 2011.

[Boy10] X. Boyen. Lattice Mixing and Vanishing Trapdoors: A Framework for Fully Secure Short Signatures and More. In *PKC*, 2010.

[CHKP10] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai Trees, or How to Delegate a Lattice Basis. In *EUROCRYPT*, 2010.

[FS86] A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *CRYPTO*, 1986.

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for Hard Lattices and New Cryptographic Constructions. In *STOC*, 2008.

[HS01] J. Hoffstein and J. H. Silverman. Polynomial Rings and Efficient Public Key Authentication II. In *Cryptography and Computational Number Theory*. 2001.

[JMV01] D. Johnson, A. Menezes, and S. Vanstone. The elliptic curve digital signature algorithm (ECDSA). *Certicom*, 2001. https://www.cs.miami.edu/home/burt/learning/Csc609.142/ecdsa-cert.pdf.

[Lyu12] V. Lyubashevsky. Lattice Signatures without Trapdoors. In *EUROCRYPT*, 2012.

[MM11] D. Micciancio and P. Mol. Pseudorandom Knapsacks and the Sample Complexity of LWE Search-to-Decision Reductions. In *CRYPTO*, 2011.

[MP12] D. Micciancio and C. Peikert. Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller. In *EUROCRYPT*, 2012.

[PR06] C. Peikert and A. Rosen. Efficient Collision-Resistant Hashing from Worst-Case Assumptions on Cyclic Lattices. In *TCC*, 2006.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM*, 1978.

Part V

Efficient Constructions and Standards

In order to improve the efficiency of cryptographic schemes, many optimizations were proposed. One of the most fundamental ones is the use of algebraic integer rings, defining structured variants of the schemes and underlying assumptions. We then present the newly published post-quantum cryptography standards based on lattices, namely ML-KEM and ML-DSA, which are optimized versions of Regev's encryption scheme and Lyubashevsky's signature scheme respectively.

8

Algebraic Lattices and Structured Problems

In this chapter, we present the computational assumptions that underlie the security of most of the efficient lattice-based constructions today. Although the fundamental problems introduced in Chapter 5 enable the simple design of encryption and signature schemes, as seen in Chapters 6 and 7, the latter suffer from innefficiencies. The parameters that are needed to provide an acceptable security level are too large so that the resulting schemes can be used in concrete applications. A line of work initiated in 2004 aims at improving the efficiency of said schemes by adding an algebraic structure into the underlying lattices. This was presented in the first part of this module through ideal lattices. We recall here a few notions on algebraic number theory, before introducing the structured problems that leads to more efficient designs. These problems underlie the security of the new post-quantum cryptography standards published by NIST.

Contents

8.1 Algebraic Number Theory

This section is dedicated to giving the necessary background to define structured lattices and the computational problems on them. The extensive use of algebraic number theory in lattice-based cryptography started in 2004 and later formalized by Lyubashevsky, Peikert and Regev [LPR10] followed by Langlois and Stehlé [LS15]. Here, we only present the necessary concepts for this chapter. The idea is rather simple and consists in packing integer vectors of $\mathbb{Z}^n$ into a single element from a ring R of degree n. Performing ring operations would then translate to the relevant operations on $\mathbb{Z}^n$ but could be performed more efficiently.

8.1.1 Intuition

Instead of considering $\mathbb{Z}$, let us take the more general ring of integer polynomials $\mathbb{Z}[x]$. The addition of polynomials is performed coefficient-wise. However, the multiplication of two polynomials can result in a polynomial of higher degree. Since the purpose of this structure is to use such polynomials in algorithms, it is necessary to be able to represent them in a computer, which implies that the degree of said polynomials must be bounded above by some value n. Additionally, $\mathbb{Z}[x]$ does not provide a rich algebraic structure which would thwart the efficiency losses of previous schemes. We thus consider a slightly different ring where the multiplication is carried so as to always keep a bounded degree. More precisely we consider the ring $R = \mathbb{Z}[x]/\langle f \rangle$ corresponding to the quotient

ring of $\mathbb{Z}[x]$ by the principal ideal $\langle f \rangle$, where $f \in \mathbb{Z}[x]$ is monic and irreducible in $\mathbb{Q}[x]$[1]. Denoting by n the degree of f, the ring R is in bijection with the set of polynomials of $\mathbb{Z}[x]$ of degree less than n and where the multiplication of two elements is performed as follows.

- Let $r = \sum_{i=0}^{n-1} r_i x^i$ and $s = \sum_{i=0}^{n-1} s_i x^i$ be two ring elements with $r_0, \dots, r_{n-1}, s_0, \dots s_{n-1}$ in $\mathbb{Z}$.
- Compute the polynomial product $t' = rs$ in $\mathbb{Z}[x]$, i.e., $t' = \sum_{i,j=0}^{n-1} r_i s_j x^{i+j}$
- Compute t as the remainder of the euclidean division of t' by f. The element t defines the product of r and s in the ring.

The same way we consider integers modulo q, we can reduce all the coefficients modulo q. For a positive integer q, we thus define $R_q = \mathbb{Z}_q[x]/\langle f \rangle \equiv \mathbb{Z}_[x]/\langle q, f \rangle$. This only adds the extra step of reducing each coefficient modulo q in the multiplication procedure above.

8.1.2 How to Choose the Defining Polynomial f?

The polynomial f has several constraints on it but for any degree n, there are many choices for a suitable f. The form of f will change the multiplication procedure which can lead to better or worse efficiency. A *bad* polynomial f would be one that increases the coefficients of $t = rs$ when performing the euclidean division. In general, we choose f to be a *cyclotomic polynomial* as defined here.

Definition 8.1 (Cyclotomic Polynomial)

Let ℓ be a positive integer. The ℓ-th cyclotomic polynomial Φ_ℓ is defined by

$$\Phi_\ell = \prod_{\substack{0 \le k < \ell \\ k \wedge \ell = 1}} \left(x - e^{i \frac{2k\pi}{\ell}} \right).$$

Cyclotomic polynomials have several interesting properties, and are well studied in terms of algebraic features as well as geometry. Although it features complex exponentials, we can show that these polynomials are monic, irreducible in $\mathbb{Q}[x]$, and have integer coefficients. The degree of Φ_ℓ is also given by $\varphi(\ell) = \left| \mathbb{Z}_\ell^\times \right|$ where φ is the Euler totient function.

Lemma 8.1 (Cyclotomic Polynomial Properties)

Let ℓ be a positive integer. The ℓ-th cyclotomic polynomial Φ_ℓ is monic, has integer coefficients, is irreducible in $\mathbb{Q}[x]$ and has degree $n = \varphi(\ell) = \left| \mathbb{Z}_\ell^\times \right|$.

The most popular choice for ℓ that leads to the most efficient constructions is $\ell = 2^{k+1}$ for a non-negative integer k. In this case, $\Phi_\ell = x^n + 1$ where $n = \varphi(2^{k+1}) = 2^k$. An advantage of this polynomial is that the euclidean division does not increase the magnitude of the coefficients, because in the ring we would have $x^n = -1$ both sides having coefficients of magnitude 1. Throughout the rest of the chapter, the ring R will be $R = \mathbb{Z}[x]/\langle x^n + 1 \rangle$ for $n = 2^k$ unless specified otherwise. The problems from Section 8.2 can be generalized to other polynomials f and thus other rings R, but the constructions we will see later use a power-of-two cyclotomic polynomial.

8.1.3 Coefficient Embedding and Multiplication Matrix

The ring R defined as above is isomorphic to $\mathbb{Z}^n$ because each element of R (resp. $R_q = R/qR$) can be represented uniquely by a vector of $\mathbb{Z}^n$ (resp. $\mathbb{Z}_q^n$) corresponding to its vector of coefficients. This isomorphism is called to *coefficient embedding* and it is often used implicitly as an alternative representation of ring elements. In our case, we use τ to denote this isomorphism, and thus for an element $r = \sum_{0 \le i < n} r_i x^i$, $\tau(r) = [r_0 | \dots | r_{n-1}]^T \in \mathbb{Z}^n$. We may also use the notation $\tau_i(r) = r_i$ for each $i \in \{0, \dots, n-1\}$. By mapping ring elements to the euclidean space, it means we can consider norms and distance over R. Typically, we can define the usual ℓ_p-norm over R by $\|r\|_p := \left\|\tau(r)\right\|_p$.

[1] There are actually other requirements, e.g., that the corresponding number field should be monogenic, but it is outside of the scope of this course.

Using this embedding and its inverse, we can also sample discrete Gaussian distribution over R by first sampling over $\mathbb{Z}^n$ and mapping it back to R via τ^{-1}.

A natural question however is how to perform the equivalent of ring multiplication directly in the embedded space? The most common multiplication operation over $\mathbb{R}^n$ is the Hadamard product $\odot$ or coefficient-wise product. Unfortunately, in general we have $\tau(rs) \neq \tau(r) \odot \tau(s)$. To see how to express the coefficient embedding of rs, we need to go back to how multiplication is defined. Having $\tau(rs) = \tau(r) \odot \tau(s)$ would mean that multiplication in the ring essentially comes down to pairwise multiplication of the coefficients, which is obviously not true. Indeed, the reduction modulo $f(x)$ scrambles the coefficients in a certain way, but linearly. In particular, $\tau(rs)$ can be expressed as $M_\tau(r)\tau(s)$, where $M_\tau(r) \in \mathbb{Z}^{n\times n}$ only depends on r. Taking the simple example of $f(x) = x^n + 1$ for $n = 2^k$, it holds that

$$M_\tau(r) = \begin{bmatrix} r_0 & -r_{n-1} & \dots & -r_1 \\ r_1 & r_0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & -r_{n-1} \\ r_{n-1} & r_{n-2} & \dots & r_0 \end{bmatrix}.$$

We can see that the entries of $M_\tau(r)$ in this case are only one coefficient multiplied by ± 1. This is not true for all choice of f, but $M_\tau(r)$ is always fully determined by r itself and therefore $\tau(r)$. Hence the entire matrix $M_\tau(r)$ can be stored by only storing n integers instead of n^2. To summarize, it holds that

$$\forall (r,s) \in R^2, \tau(rs) = M_\tau(r)\tau(s) = \begin{bmatrix} r_0 & -r_{n-1} & \dots & -r_1 \\ r_1 & r_0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & -r_{n-1} \\ r_{n-1} & r_{n-2} & \dots & r_0 \end{bmatrix} \begin{bmatrix} s_0 \\ \vdots \\ \vdots \\ s_{n-1} \end{bmatrix}.$$

It is possible to extend the coefficient embedding to vectors of $R^{d'}$ by simply concatenating the coefficient embeddings of each entry. Similarly, one can extend the multiplication matrix map M_τ to matrices of $R^{m'\times d'}$ relying on block matrices. We thus have that for all $\mathbf{A} \in R^{m'\times d'}$ and $\mathbf{x} \in R^{d'}$,

$$\tau(\mathbf{Ax}) = \begin{bmatrix} M_\tau([\mathbf{A}]_{1,1}) & \dots & M_\tau([\mathbf{A}]_{1,d'}) \\ \vdots & & \vdots \\ M_\tau([\mathbf{A}]_{m',1}) & \dots & M_\tau([\mathbf{A}]_{m',d'}) \end{bmatrix} \begin{bmatrix} \tau([\mathbf{x}]_1) \\ \vdots \\ \tau([\mathbf{x}]_{d'}) \end{bmatrix} = M_\tau(\mathbf{A})\tau(\mathbf{x}).$$

Lattice-based cryptography relies heavily on matrices, vectors, and products among those. A matrix of $\mathbb{Z}^{m\times d}$ without any specific structure requires the storage of md integers. Here, for $d' = d/n$ and $m' = m/n$, the matrix $M_\tau(\mathbf{A})$ is also in $\mathbb{Z}^{m\times d}$. However, since it is composed of structured blocks, it only requires $d'm' \cdot n$ integers, that is dm/n. For equivalent dimensions, we thus gain a factor of n in the storage of matrices. This is extremely interesting for cryptographic applications to reduce the size of cryptographic keys which are usually matrices.

This ring structure also turns out to be relevant for computational efficiency and computation speed. At first glance, if multiplying r and s is performed by doing $\tau^{-1}(M_\tau(r)\tau(s))$, the overall complexity would be $O(n^2)$, which does not provide any improvement. However, the nega-circulant structure of $M_\tau(r)$ already helps to perform the matrix-vector product more efficiently. This structure is not innocent as it corresponds to the multiplication in R. The latter can actually be carried out using extremely efficient algorithms such as FFT (*Fast Fourier Transform*). For multiplications in $R_q = R/qR$, we use a variant called NTT (*Number Theoretic Transform*). These algorithms compute the product rs with a complexity of $O(n \log n)$ instead of $O(n^2)$. Hence, the product $\mathbf{Ax}$ can be computed in $O(m'd'n \log n) = O(md \log(n)/n)$ instead of $O(md)$. We will see it in TD 7 .

8.2 Structured Problems

Just like we can update our cryptographic toolbox to this new algebraic setting, we also need to account for the peculiarities we might have introduced as well. In particular, the underlying security assumptions need to be changed to feature this ring R, thus leading to structured problems. Via the coefficient embedding and multiplication matrix map, we are essentially able to map it back

to $\mathbb{Z}$ and interpret it as an unstructured problem where we forget the nega-circulant behavior of M_τ. This simple thought experiment means that for equivalent dimensions and parameters, adding structure can only decrease the hardness of the problem and in turn the security of cryptographic primitives. An active area of research is to determine whether this algebraic structure makes the lattice problems and fundamental problems we introduced significantly easier. We start by formulating the fundamental problems over rings in Section 8.2.1 before giving their generalizations to modules[2] in Section 8.2.2. Although there are similar results as those from Chapter 5 on the hardness of these structured problems, this falls outside of the scope of this course.

8.2.1 Fundamental Problems over Algebraic Rings

Ring Short Integer Solution

The ring version of the Short Integer Solution problem was introduced in 2007 [Mic07] to demonstrate the advantage of what was called *quasi-cyclic* lattices. They turned to be a specific case of *ideal lattices* corresponding to the ring $\mathbb{Z}[x]/\langle x^n - 1\rangle$ (Careful, here we have $x^n - 1$ with n prime, which differs from power-of-two cyclotomics). The idea is the same as the standard SIS problem but by adding structure to the matrix (and thus interpreting it directly in R). More precisely, consider a vector $\mathbf{a} \in R_q^m$ for a positive integer m. One can consider the ideal lattice

$$\mathcal{L}_q^\perp(\mathbf{a}^T) = \{\mathbf{x} \in R^m : \mathbf{a}^T\mathbf{x} = \mathbf{0} \bmod qR\},$$

Applying the coefficient embedding, and defining $\mathbf{A} = M_\tau(\mathbf{a}^T) = [M_\tau(a_1)|\dots|M_\tau(a_m)] \in \mathbb{Z}_q^{n\times nm}$, the lattice $\mathcal{L}_q^\perp(\mathbf{a}^T)$ is isomorpic to

$$\mathcal{L}_q^\perp(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^{nm} : \mathbf{A}\mathbf{x} = \mathbf{0} \bmod q\mathbb{Z}\}.$$

We now give the formal definition which is very similar to Definition 5.1.

Definition 8.2 (Ring Short Integer Solution)

Let n be a power-of-two and $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$. Let m, q be two positive integers, and β a positive real. The *Ring Short Integer Solution* problem R-SIS$_{n,m,q,\beta}$ asks to find a vector $\mathbf{x} \in R^m$ such that $\mathbf{a}^T\mathbf{x} = \mathbf{0} \bmod qR$ and $0 < \|\mathbf{x}\|_2 \leq \beta$, given a uniformly random vector $\mathbf{a}$ in R_q^m.

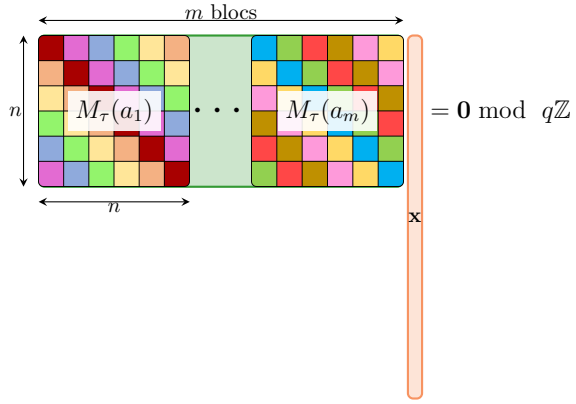


Figure 8.1: Ring Short Integer Solution problem embedded into the integers via τ and M_τ.

Ring Learning With Errors

Similarly, we can define a ring version of the Learning With Errors problem. It was first introduced in 2009 and 2010 by Lyubashevsky, Peikert and Regev [LPR10]. We can once again consider a

[2] A module over a ring R is the equivalent of a vector space over a field K. It is an algebraic structure consider addition of module elements, as well as scalar multiplication by ring elements. We note that most of the results commonly known in linear algebra over vector spaces *do not* carry straightforwardly to modules.

vector $\mathbf{a} \in R_q^m$ and the lattice

$$\mathcal{L}_q(\mathbf{a}) = \{\mathbf{y} \in R^m : \exists s \in R_q, \mathbf{a}s = \mathbf{y} \bmod qR\}.$$

The CVP_γ problem then defines the ring version of LWE: given $\mathbf{a}$ and $\mathbf{a}s+\mathbf{e}$, find $\mathbf{s}$. The coefficient embedding links it to the integers as follows. Given $\mathbf{A} = M_\tau(\mathbf{a})$ and $\mathbf{b} = M_\tau(\mathbf{a})\tau(s) + \tau(\mathbf{e})$, find $\tau(s)$. Here are the formal definitions.

Definition 8.3 (Ring Learning With Errors Distribution)

Let n be a power-of-two and $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$. Let q be a positive integer. Let s be in R_q, and $\mathscr{D}_e$ be a distribution over R. The R-LWE distribution denoted by $A^R_{s,\mathscr{D}_e}$ is defined by the following random process: sample $a \hookleftarrow U(R_q)$, and $e \hookleftarrow \mathscr{D}_e$, and output $(a, as + e \bmod qR)$.

Definition 8.4 ((Search) Ring Learning With Errors)

Let n be a power-of-two and $R = \mathbb{Z}[x]/\langle x^n+1\rangle$. Let q be a positive integer. Let $\mathscr{D}_s$ be a secret distribution over R_q, and $\mathscr{D}_e$ be an error distribution over R. The *search Ring Learning With Errors* problem $\text{sR-LWE}_{n,q,\mathscr{D}_s,\mathscr{D}_e}$ is as follows. Let s be drawn from $\mathscr{D}_s$. Given arbitrarily many samples $(a_i, a_i s + e_i \bmod qR)$ drawn from $A^R_{s,\mathscr{D}_e}$, find s.
When the number of available samples is limited to m, we write the problem as $\text{sLWE}_{n,q,m,\mathscr{D}_s,\mathscr{D}_e}$, and we present it in vector form as follows. Given $\mathbf{a} \hookleftarrow U(R_q^m)$ and $\mathbf{t} = \mathbf{a}s + \mathbf{e} \bmod qR$ for some $s \hookleftarrow \mathscr{D}_s$ and $\mathbf{e} \hookleftarrow \mathscr{D}_e^m$, find s.

Definition 8.5 ((Decision) Ring Learning With Errors)

Let n be a power-of-two and $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$. Let q be a positive integer. Let $\mathscr{D}_s$ be a secret distribution over R_q, and $\mathscr{D}_e$ be an error distribution over R. The *decision Ring Learning With Errors* problem $\text{R-LWE}_{n,q,\mathscr{D}_s,\mathscr{D}_e}$ is as follows. Let s be drawn from $\mathscr{D}_s$, and let $\mathscr{D} \in \{A^R_{s,\mathscr{D}_e}, U(R_q \times R_q)\}$. Given arbitrarily many samples from $\mathscr{D}$, decide whether $\mathscr{D} = A^R_{s,\mathscr{D}_e}$ or if $\mathscr{D} = U(R_q \times R_q)$.
When the number of available samples is limited to m, we write the problem as $\text{R-LWE}_{n,q,m,\mathscr{D}_s,\mathscr{D}_e}$, and we present it in vector form as follows. Given $\mathbf{a} \hookleftarrow U(R_q^m)$ and $\mathbf{t} \in R_q^m$, decide whether $\mathbf{t} = \mathbf{a}s+\mathbf{e} \bmod qR$ for some $s \hookleftarrow \mathscr{D}_s$ and $\mathbf{e} \hookleftarrow \mathscr{D}_e^m$, or if $\mathbf{t} \hookleftarrow U(R_q^m)$.

Another Fundamental Problem: NTRU

This algebraic structure allows one to define other algorithmic problems such as the NTRU problem [HPS98]. NTRU was proposed by Hoffstein, Pipher and Silverman in 1998 with the goal of the designing an algebraic encryption scheme. This scheme is part of lattice-based cryptography and is therefore one of the first encryption scheme over lattices. Underlying the security of this primitive is another computational assumption, just like SIS or LWE, which is formulated over $R = \mathbb{Z}[x]/\langle f\rangle$. As opposed to SIS and LWE, NTRU does not have an *unstructured* variant, and did not benefit from worst-case to average-case reductions from variants of SVP. This last aspect has been the subject of recent research by Pellet–Mary and Stehlé [PS21] giving the first reductions for NTRU.

Definition 8.6 ((Search) NTRU)

Let n be a power-of-two and $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$. Let q be a positive integer. Let $\mathscr{D}_f$ and $\mathscr{D}_g$ be two distributions on $R_q^\times$ and R_q respectively, where $R_q^\times$ is the group of invertible elements of R_q. The *search NTRU* problem $\text{sNTRU}_{n,q,\mathscr{D}_f,\mathscr{D}_g}$ is as follows. Let f be drawn from $\mathscr{D}_f$, and g from $\mathscr{D}_g$. Given $h = gf^{-1} \bmod qR$, find f and g.

Definition 8.7 ((Decision) NTRU)

Let n be a power-of-two and $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$. Let q be a positive integer. Let $\mathscr{D}_f$ and $\mathscr{D}_g$ be two distributions on $R_q^\times$ and R_q respectively, where $R_q^\times$ is the group of invertible elements

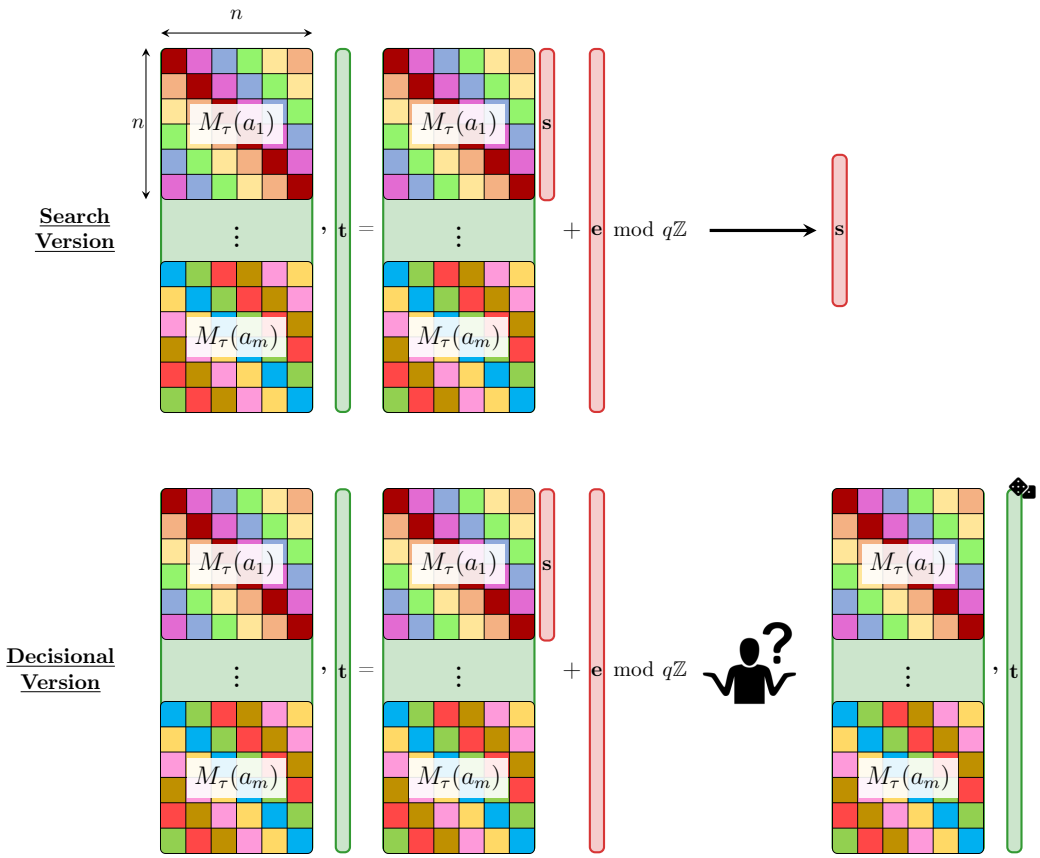


Figure 8.2: Ring Learning With Errors problem embedded into the integers via τ and M_τ.

of R_q. The *decision NTRU* problem $\text{NTRU}_{n,q,\mathscr{D}_f,\mathscr{D}_g}$ is as follows. Given $h \in R_q$, decide whether $h = gf^{-1} \bmod qR$ for f drawn from $\mathscr{D}_f$ and g from $\mathscr{D}_g$, or $h \hookleftarrow U(R_q)$.

The distributions $\mathscr{D}_f$ and $\mathscr{D}_g$ are chosen so that f and g are short elements, that is $\|\tau(f)\|_2$ and $\|\tau(g)\|_2$ are small. The problem thus comes down to finding the numerator and denominator of a quotient of short polynomials, or to distinguish such a quotient from a uniformly random element. We can see this problem once again as a variant of SVP over a specific lattice. By rewriting the equation $h = gf^{-1} \bmod qR$ as $g + h(-f) = 0 \bmod qR$, it holds that the vector $[g|-f]^T$ is a short vector of the lattice

$$\mathcal{L}_q^\perp([1|h]) = \{\mathbf{x} \in R^2 : [1|h]\mathbf{x} = 0 \bmod qR\}.$$

8.2.2 Generalization to Modules

In the problems we presented in Chapter 5, the dimension of the problem d, m, namely that of the matrix $\mathbf{A}$ involved, were set somewhat independently. Here, when embedding the problem into the integers, one dimension is n while the other is nm. In particular, the dimension driving the security is n. As a result, increasing the hardness of the problem requires to increase n, but it in turn drastically increases the second dimension which may be undesirable. Langlois and Stehlé [LS15] thus formalized a generalization of the ring problems, and proved their hardness as well, to allow for an improved flexibility in the parameter choices. These generalizations are called *Module Short Integer Solution* and *Module Learning With Errors* and encompass both SIS, LWE and R-SIS, R-LWE. The idea is to start from the original problems SIS and LWE and simply replace the ring of integers $\mathbb{Z}$ by the new algebraic ring R. We thus end up with matrices over R of dimensions m and d, i.e., dimensions nm and nd when embedded into $\mathbb{Z}$. Thence, one can tweak m and d to change the security of the scheme without having to change the underlying ring R (and its degree n). We give the formal definitions only.

Definition 8.8 (Module Short Integer Solution)

Let n be a power-of-two and $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$. Let d, m, q be three positive integers, and β a positive real. The *Module Short Integer Solution* problem M-SIS$^p_{n,d,m,q,\beta}$ asks to find a vector $\mathbf{x} \in R^m$ such that $\mathbf{Ax} = \mathbf{0} \bmod qR$ and $0 < \|\mathbf{x}\|_2 \leq \beta$, given a uniformly random matrix $\mathbf{A}$ in $R_q^{d\times m}$.

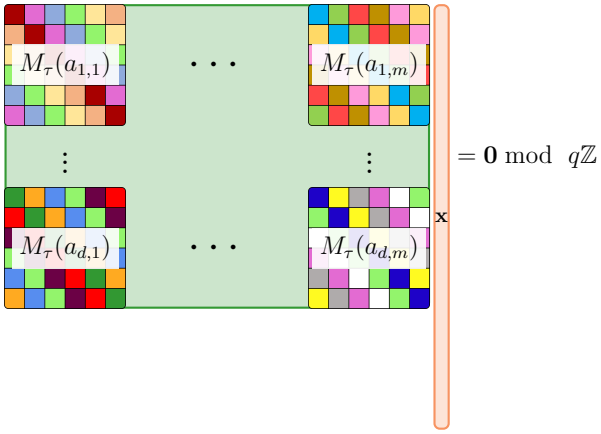


Figure 8.3: Module Short Integer Solution problem embedded into the integers via τ and M_τ.

Definition 8.9 (Module Learning With Errors Distribution)

Let n be a power-of-two and $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$. Let d, q be positive integers. Let $\mathbf{s}$ be in R_q^d, and $\mathscr{D}_e$ be a distribution over R. The R-LWE distribution denoted by $A^M_{\mathbf{s},\mathscr{D}_e}$ is defined by the following random process: sample $\mathbf{a} \hookleftarrow U(R_q^d)$, and $e \hookleftarrow \mathscr{D}_e$, and output $(\mathbf{a}, \mathbf{a}^T\mathbf{s} + e \bmod qR)$.

Definition 8.10 ((Search) Module Learning With Errors)

Let n be a power-of-two and $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$. Let d, q be a positive integer. Let $\mathscr{D}_s$ be a secret distribution over R_q^d, and $\mathscr{D}_e$ be an error distribution over R. The *search Module Learning With Errors* problem sM-LWE$_{n,d,q,\mathscr{D}_s,\mathscr{D}_e}$ is as follows. Let $\mathbf{s}$ be drawn from $\mathscr{D}_s$. Given arbitrarily many samples $(\mathbf{a}_i, \mathbf{a}_i^T\mathbf{s} + e_i \bmod qR)$ drawn from $A^M_{\mathbf{s},\mathscr{D}_e}$, find $\mathbf{s}$.
When the number of available samples is limited to m, we write the problem as sM-LWE$_{n,d,q,m,\mathscr{D}_s,\mathscr{D}_e}$, and we present it in matrix form as follows. Given $\mathbf{A} \hookleftarrow U(R_q^{m\times d})$ and $\mathbf{t} = \mathbf{As} + \mathbf{e} \bmod qR$ for some $\mathbf{s} \hookleftarrow \mathscr{D}_s$ and $\mathbf{e} \hookleftarrow \mathscr{D}_e^m$, find $\mathbf{s}$.

Definition 8.11 ((Decision) Module Learning With Errors)

Let n be a power-of-two and $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$. Let d, q be positive integers. Let $\mathscr{D}_s$ be a secret distribution over R_q^d, and $\mathscr{D}_e$ be an error distribution over R. The *decision Module Learning With Errors* problem M-LWE$_{n,d,q,\mathscr{D}_s,\mathscr{D}_e}$ is as follows. Let $\mathbf{s}$ be drawn from $\mathscr{D}_s$, and let $\mathscr{D} \in \{A^M_{\mathbf{s},\mathscr{D}_e}, U(R_q^d \times R_q)\}$. Given arbitrarily many samples from $\mathscr{D}$, decide whether $\mathscr{D} = A^M_{\mathbf{s},\mathscr{D}_e}$ or if $\mathscr{D} = U(R_q^d \times R_q)$.
When the number of available samples is limited to m, we write the problem as M-LWE$_{n,d,q,m,\mathscr{D}_s,\mathscr{D}_e}$, and we present it in matrix form as follows. Given $\mathbf{A} \hookleftarrow U(R_q^{m\times d})$ and $\mathbf{t} \in R_q^m$, decide whether $\mathbf{t} = \mathbf{As} + \mathbf{e} \bmod qR$ for some $\mathbf{s} \hookleftarrow \mathscr{D}_s$ and $\mathbf{e} \hookleftarrow \mathscr{D}_e^m$, or if $\mathbf{t} \hookleftarrow U(R_q^m)$.

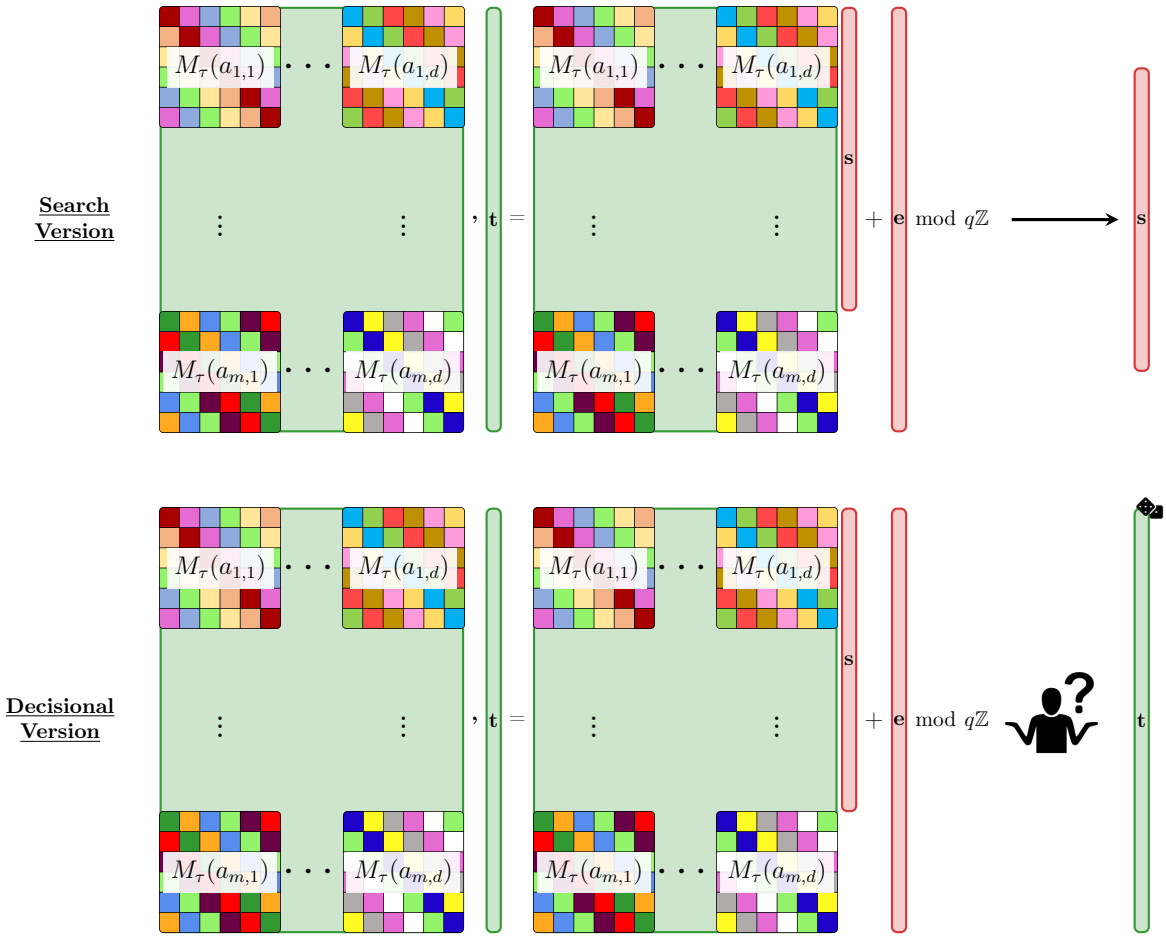


Figure 8.4: Module Learning With Errors problem embedded into the integers via τ and M_τ.

8.3 Future Post-Quantum Cryptography Standards from Lattices

The anticipate the threat of quantum computing and deploy quantum-resistant cryptographic algorithms and protocols, the US National Institute of Standards and Technology (NIST) launched a post-quantum standardization effort in 2016 [NIS]. The goal was to motivate the cryptologic research community to come up with public-key encryption schemes (or key encapsulation mechanisms) as well as digital signatures that would withstand quantum attacks. The proposals were open for scrutiny and the community was encouraged to assess their efficiency but mostly their security by either providing security reductions to hard problems, or provide attacks to break or decrease certain security claims. A total of 82 submissions were featured in the first round from different families of assumptions such as lattices, codes, isogenies, multivariate, or even symmetric ones. Among them, 4 lattice-based signatures and 24 lattice-based KEM (Key Encapsulation Mechanism) were proposed, many of which made it to the second and third rounds. In July 2020, the third round started with four finalists for encryption and three finalists for signatures which are mentioned in Table 8.1.

Table 8.1 (Third round finalists of the NIST competition)

KEM	Signature	
Crystals-Kyber	**Crystals-Dilithium**	Lattices
Saber		
NTRU	**Falcon**	
Classic McEliece	**Rainbow** ⚠	
Codes	**Multivariate** (broken)	

Then, in July 2022, four algorithms were selected as the first post-quantum cryptography standards. Three of them are lattice-based: Crystals-Kyber (KEM), Crystals-Dilithium (Signature) and Falcon (Signature). The fourth (SPHINCS+) is a hash-based signature which was selected among the alternate finalists of the third round in order to have a variety of assumptions among the standards. Hash-based signatures are very robust, but suffer from large signature sizes. The standards FIPS 203 (ML-KEM for Crystals-Kyber), FIPS 204 (ML-DSA for Crystals-Dilithium) and FIPS 205 (SLH-DSA for SPHINCS+) are recently been officially published by NIST on August 13th 2024. The standard FIPS 206 (FN-DSA for Falcon) is still pending and expect within 2025. Although the first standards are now published, the optimizations of these algorithms or the proposal of new ones is still an active research area. Optimizations have been proposed during and after the competition, some of which led to other algorithms which fed into other standardization efforts (ISO, KPQC, etc.). In the next chapters, we present only ML-KEM (Kyber) and ML-DSA (Dilithium).

RECAP

- For improved efficiency, we consider an algebraic setting by replacing $\mathbb{Z}$ with a ring $R = \mathbb{Z}[x]/\langle x^n+1\rangle$ where n is a power of 2. This structure allows for faster computations using the FFT (or NTT for multiplications in $R_q = R/qR$), and allows for a better storage of matrices. The norme of elements in R is defined by the norm of their coefficient vector when seen as polynomials.
- The SIS, ISIS, LWE problems underlying the security of the constructions must be adapted to this new algebraic setting. We talk about module variants.
- The M-SIS$_{n,d,m,q,\beta}$ problem asks to find $\mathbf{x} \in R^m$ such that $\mathbf{Ax} = \mathbf{0} \bmod qR$ and $0 < \|\mathbf{x}\|_2 \leq \beta$ given $\mathbf{A} \in R_q^{d\times m}$ uniform.
- The sM-LWE$_{n,d,q,m,\mathscr{D}_s,\mathscr{D}_e}$ problem asks to find the secret $\mathbf{s}$, sampled from $\mathscr{D}_s$, given $\mathbf{A} \in R_q^{m\times d}$ uniform, and $\mathbf{t} = \mathbf{As} + \mathbf{e} \bmod qR$ where $\mathbf{e} \hookleftarrow \mathscr{D}_e^m$. The decisional version M-LWE$_{n,d,q,m,\mathscr{D}_s,\mathscr{D}_e}$ asks to distinguish such a $\mathbf{t}$ from a perfectly uniform vector over R_q^m.
- The ring variants R-SIS, sR-LWE, R-LWE correspond to the module problems in the specific case of $d = 1$. The unstructured problems SIS, sLWE, LWE correspond to the specific case of $n = 1$.
- The hardness of this problems is proven under variants of SVP restricted to structured lattices (ideals or modules).

Bibliography

[HPS98] J. Hoffstein, J. Pipher, and J. H. Silverman. NTRU: A Ring-Based Public Key Cryptosystem. In *ANTS*, 1998.

[LPR10] V. Lyubashevsky, C. Peikert, and O. Regev. On Ideal Lattices and Learning with Errors over Rings. In *EUROCRYPT*, 2010.

[LS15] A. Langlois and D. Stehlé. Worst-case to Average-case Reductions for Module Lattices. *DCC*, 2015.

[Mic07] D. Micciancio. Generalized Compact Knapsacks, Cyclic Lattices, and Efficient One-Way Functions. *Comput. Complex.*, 2007.

[NIS] NIST. Post-Quantum Cryptography Standardization. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization.

[PS21] A. Pellet-Mary and D. Stehlé. On the Hardness of the NTRU Problem. In *ASIACRYPT*, 2021.

9

ML-KEM : Crystals-Kyber

In this chapter, we present in detail the algorithm Crystals-Kyber [BDK+18] which is now the new key encapsulation mechanism standard (ML-KEM) published by NIST in the FIPS 203 document (FIPS means *Federal Information Processing Standard*). We present this algorithm step by step. First of all, we start from a variant of Regev's encryption scheme which we optimize to obtain a public key encryption secure in the IND-CPA model. The parameters of this scheme are however chosen to only encrypt elements of at most 256 bits, which corresponds to the size of a symmetric key. Then, we briefly present the transformation from an IND-CPA-secure public key encryption to an IND-CCA-secure KEM. The transformation used in the case of Kyber is a variant of the generic transform due to Fujisaki and Okamoto [FO99]. This chapter is heavily inspired by the Kyber specification and the tutorial of Lyubashevsky [Lyu24].

Contents

9.1 Crystals-Kyber: IND-CPA Public Key Encryption

We start the description of Kyber by our starting point in terms of lattice-based encryption, Regev's encryption scheme. As we have seen in TD 2 , the IND-CPA security requires two elements: simulating the public key under the LWE assumption, and simulating the ciphertexts with the leftover hash lemma. The latter is however very constraining on the parameters as it requires $m \geq (d+1)\log_2 q + 2\lambda$. This drastically deteriorates the performance of the scheme. Fortunately, in TD 3 , we have seen how to bypass the lefotver hash lemma by using a second LWE assumption in the simulation of ciphertexts. Typically, this allows for choosing $m = d$, leading to much more compact parameters. To be coherent with the notations in the ML-KEM standard, we use k instead of $m = d$. At a high level, the algorithm is described as follows.

KeyGen: Sample $\mathbf{s} \hookleftarrow \mathcal{D}_{\mathbb{Z}^k,\alpha q}$ and $\mathbf{e} \hookleftarrow \mathcal{D}_{\mathbb{Z}^k,\alpha q}$. Sample $\mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{k\times k})$, and compute $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q\mathbb{Z}$. Output $\mathsf{pk} = (\mathbf{A}, \mathbf{t})$ and $\mathsf{sk} = \mathbf{s}$.

Enc: Given a message $M \in \{0,1\}$, sample $\mathbf{r} \hookleftarrow U(\{0,1\}^k)$ and $\mathbf{e}_1 \hookleftarrow \mathcal{D}_{\mathbb{Z}^k,\gamma q}, e_2 \hookleftarrow \mathcal{D}_{\mathbb{Z},\gamma q}$ and output $\mathbf{c} = [\mathbf{c}_1^T | c_2]^T$ where

$$\mathbf{c}_1 = \mathbf{A}^T\mathbf{r} + \mathbf{e}_1 \bmod q\mathbb{Z}, \quad \text{and} \quad c_2 = \mathbf{t}^T\mathbf{r} + e_2 + \lfloor q/2 \rceil M \bmod q\mathbb{Z},$$

Dec: Given a ciphertext $[\mathbf{c}_1^T | c_2]^T \in \mathbb{Z}_q^{k+1}$, we compute $u = c_2 - \mathbf{c}_1^T\mathbf{s} \bmod q\mathbb{Z}$. Output 0 if u is closer to 0 than $\pm\lfloor q/2 \rceil$, and 1 otherwise.

We now start from this algorithm and add several optimizations to end up with the Kyber encryption scheme.

9.1.1 Algebraic Structure

We observe that, as in Regev's encryption, this variant only allows for encrypting one bit at a time which is far from optimal. Also, as explained in Chapter 8, the schemes based on integers and modular integers can be made more efficient by considering a richer algebraic structure. This is the case for Kyber which works over a power-of-two cyclotomic ring $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$ with n a power of two 2. Just like we have seen in the previous chapter, it is possible to modify the underlying assumptions and consider M-LWE instead of LWE. The key then becomes $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e} \bmod qR$, where the entries of each matrix and vectors are now in R. For the ciphertext, we can proceed in a similar way and compute $\mathbf{c}_1 = \mathbf{A}^T\mathbf{r} + \mathbf{e}_1 \bmod qR$, and $c_2 = \mathbf{t}^T\mathbf{r} + \lfloor q/2 \rceil M \bmod qR$.

Here, the element M corresponding to the message must be an element of R. Because 0 and 1 are indeed elements of R (seen as constant polynomials), we could keep $\{0, 1\}$ as the message space. However, this would ignore all the higher degree monomials we have at our disposal. Instead, we consider messages as polynomials for which *all* the coefficients are binary in $\{0, 1\}$. The message space becomes

$$\mathcal{M} = \tau^{-1}(\{0, 1\}^n).$$

In particular, we can encode one message bit in each of the coefficients, and thus n bits in a single element of R. Our goal being to encrypt a symmetric key of at most 256 bits, it is then sufficient to choose $n \geq 256$. During decryption, we decode each coefficient individually. We obtain a new version of our scheme as follows. Note that the effective lattice dimension (after embedding with τ and M_τ) is nk. To preserve the same dimension as the unstructured version, we can reduce the value of k by a factor of n. For $n = 256$, k will essentially be a small constant between 2 and 4 depending on the targeted security level.

KeyGen: Sample $\mathbf{s} \hookleftarrow \mathcal{D}_{R^k,\alpha q}$ and $\mathbf{e} \hookleftarrow \mathcal{D}_{R^k,\alpha q}$. Sample $\mathbf{A} \hookleftarrow U(R_q^{k\times k})$, and compute $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e} \bmod qR$. Output $\mathsf{pk} = (\mathbf{A}, \mathbf{t})$ and $\mathsf{sk} = \mathbf{s}$.

Enc: Given a message $M \in \mathcal{M}$, sample $\mathbf{r} \hookleftarrow U(\mathcal{M}^k)$ and $\mathbf{e}_1 \hookleftarrow \mathcal{D}_{R^k,\gamma q}, e_2 \hookleftarrow \mathcal{D}_{R,\gamma q}$ and output $\mathbf{c} = [\mathbf{c}_1^T | c_2]^T$ where

$$\mathbf{c}_1 = \mathbf{A}^T\mathbf{r} + \mathbf{e}_1 \bmod qR, \quad \text{and} \quad c_2 = \mathbf{t}^T\mathbf{r} + e_2 + \lfloor q/2 \rceil M \bmod qR,$$

Dec: Given a ciphertext $[\mathbf{c}_1^T | c_2]^T \in R_q^{k+1}$, we compute $u = c_2 - \mathbf{c}_1^T\mathbf{s} \bmod qR$. We define the decrypted message $M \in \mathcal{M}$ as follows. For i from 0 to $n - 1$, we define $\tau_i(M) = 0$ if $\tau_i(u)$ is closer to 0 than $\pm\lfloor q/2 \rceil$, and $\tau_i(M) = 1$ otherwise.

9.1.2 Error Distributions

The scheme we presented uses discrete Gaussian distributions over R. In most case, discrete Gaussians are to be avoided from an implementation perspective because they can be fairly complex to sample from securely. Even in the simple case where sampling over R comes down to sampling over the trivial lattice $\mathbb{Z}^n$, the procedure requires either the use of precomputed tables, floating-point arithmetic, etc. Also, the encryption procedure should use the smallest possible errors to avoid decryption failure (although not too small to preserve security).

In the Kyber encryption scheme, the authors chose to use centered binomial distributions which we present here. The advantage is that these distributions are very easy to sample from, do not require a lot of randomness (for small binomial parameters), and generally give smaller elements. This thus improves the computational efficiency, while choosing smaller parameters (modulus q) due to a smaller decryption error.

Centered Binomial Distribution

Binomial distributions frequently arise in probability theory through the repetition of Bernoulli trials. In particular, if $x_1, \ldots, x_N$ are independent random variables taking the value 1 with probability p and 0 otherwise, the distribution of $X = \sum_{i=1}^{N} x_i$ is called the binomial distribution of parameters (N, p). This distribution has a mean of Np and is therefore not centered. We can define the corresponding centered distribution by that of $X - Np$. This gives rise to the centered binomial distribution of parameters (N, p). In this course, we only consider the case of $p = 1/2$.

Definition 9.1 (Centered Binomial Distribution)

Let η be a positive integer. We define the centered binomial distribution of parameter η, denoted ψ_η, as the distribution of $(\sum_{i=1}^{2\eta} x_i) - \eta$, where $x_1, \dots, x_{2\eta}$ are independent random variables following $U(\{0,1\})$.
The distribution ψ_η can also be defined by the distribution of $\sum_{i=1}^{\eta}(x_i - x'_i)$ where $x_1, x'_1, \dots, x_\eta, x'_\eta$ are independent random variables following $U(\{0,1\})$.

Sampling from this distribution thus comes down to summing 2η random bits (for example generated by a cryptographic hash function or a PRF (pseudrandom function)) and then substracting η, which is very easy to do. This distribution ψ_η produces elements of $[\![-\eta, \eta]\!]$ following a binomial distribution. In our case, we must produce elements of R and we thus define the distribution $\mathcal{B}_\eta = \tau^{-1}(\psi_\eta^n)$. This means each coefficient of the ring element is sampled from ψ_η independently of the others. The outputted elements are then in $\tau^{-1}([\![-\eta, \eta]\!]^n)$, set that we usually denote by S_η.

Different Parameters for a Finer Security

In the previous scheme, we use different error distributions $\mathcal{D}_{R^k,\alpha q}, U(\mathcal{M}^k), \mathcal{D}_{R^k,\gamma q}$, etc. The use of different parameters for each distribution allows for a finer security. For example, the simulation of the public key relies on M-LWE$_{n,k,k,q,\mathcal{D}_{R^k,\alpha q},\mathcal{D}_{R^k,\alpha q}}$, while the ciphertext simulation relies on M-LWE$_{n,k,k+1,q,U(\mathcal{M}^k),\mathcal{D}_{R^{k+1},\gamma q}}$. We see that one of the dimensions goes from k to $k+1$ which legitimates the change from α to γ.

Kyber's encryption also uses a similar granularity to obtain a finer security estimation, and thus choose the most optimal parameters so as not to overshoot the security target. In their case, they use centered binomial distributions with two different parameters η_1 and η_2. The parameter η_1 is used in the public key ($\mathbf{s}$ and $\mathbf{e}$) *as well as* the randomness $\mathbf{r}$. The parameter η_2 is used only to fix the ciphertext errors $(\mathbf{e}_1, e_2)$. We then get the following version of our ongoing construction of Kyber.

KeyGen: Sample $\mathbf{s} \hookleftarrow \mathcal{B}_{\eta_1}^k$ and $\mathbf{e} \hookleftarrow \mathcal{B}_{\eta_1}^k$. Sample $\mathbf{A} \hookleftarrow U(R_q^{k\times k})$, and compute $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e} \bmod qR$. Output $\mathsf{pk} = (\mathbf{A}, \mathbf{t})$ and $\mathsf{sk} = \mathbf{s}$.

Enc: Given a message $M \in \mathcal{M}$, sample $\mathbf{r} \hookleftarrow \mathcal{B}_{\eta_1}^k$ and $\mathbf{e}_1 \hookleftarrow \mathcal{B}_{\eta_2}^k, e_2 \hookleftarrow \mathcal{B}_{\eta_2}$ and output $\mathbf{c} = [\mathbf{c}_1^T | c_2]^T$ where

$$\mathbf{c}_1 = \mathbf{A}^T\mathbf{r} + \mathbf{e}_1 \bmod qR, \quad \text{and} \quad c_2 = \mathbf{t}^T\mathbf{r} + e_2 + \lfloor q/2 \rceil M \bmod qR,$$

Dec: Given a ciphertext $[\mathbf{c}_1^T | c_2]^T \in R_q^{k+1}$, we compute $u = c_2 - \mathbf{c}_1^T\mathbf{s} \bmod qR$. We define the decrypted message $M \in \mathcal{M}$ as follows. For i from 0 to $n-1$, we define $\tau_i(M) = 0$ if $\tau_i(u)$ is closer to 0 than $\pm\lfloor q/2 \rceil$, and $\tau_i(M) = 1$ otherwise.

9.1.3 Compression

The optimizations we introduce before getting to the Kyber encryption are about compression. To understand the motivations behind that, let us estimate the size of the keys and ciphertexts. The public key pk contains $\mathbf{A}$ and $\mathbf{t}$ which are uniform over R_q (recall $\mathbf{t}$ is uniform under M-LWE). Each coefficient of a polynomial of R_q is then an element of $\mathbb{Z}_q$ which must then be stored with $\lceil \log_2 q \rceil$ bits. The total storage of pk then requires $k^2 n \lceil \log_2 q \rceil + kn \lceil \log_2 q \rceil = k(k+1)n \lceil \log_2 q \rceil$ bits. Similarly, each ciphertext $\mathbf{c}$ is a vector uniform in R_q^{k+1} which requires $(k+1)n\lceil \log_2 q \rceil$ bits of storage. For parameters giving the smallest acceptable security level, this corresponds to 2300 bytes for the public key and 1150 bytes for the ciphertext. As a comparison, classical key exchange mechanisms like Diffie-Hellman only require sending 64 bytes in total. It would thus be preferable to reduce the sizes as much as possible, albeit without significiantly hindering security. For that, it is possible to use compression methods.

Public Key Compression

Let us first look at the public key. We can see that the matrix $\mathbf{A}$ is perfectly uniform, which means no secret information is hidden inside it. We can therefore sample it pseudo-randomly from a seed ρ of 256 bits. Hence, the storage of the matrix $\mathbf{A}$ would be reduced to storing ρ. The only drawback is that we will need to expand the seed ρ to recover $\mathbf{A}$ at each encryption procedure. This is nonetheless perfectly acceptable because it allows one to reduce the public key size to

$256 + kn\lceil \log_2 q \rceil$ bits (so 800 bytes for the same estimations as before). More precisely, the matrix $\mathbf{A}$ is computed from ρ using an *extendable output function* (XOF) such as SHAKE (based on the SHA-3)

For the second part of pk, the vector $\mathbf{t}$ depends on secret information and thus cannot be recomputed from public information. It must then be included in the public key. Overall, the public key is now $(\rho, \mathbf{t})$ which is represented by $256 + kn\lceil \log_2 q \rceil$ bits.

Ciphertext Compression and Rounding Errors

Let us now spend some time on the ciphertext compression. For that, we look at the coefficients individually. Each element of $\mathbb{Z}_q$ requires $\lceil \log_2 q \rceil$ bits. Reducing this size to say d bits implies compressing the set $\mathbb{Z}_q$ to a set S of size 2^d. However, this compression is irreversible and induces a loss of information on the initial ciphertext. It means the compression introduces an error which affects the decryption as we no longer have access to the full ciphertext. Because of the decryption, we must be careful as to how we choose this set S. In particular, we want to define $S \subset \mathbb{Z}_q$ so that the maximal distance between two points of S (measured by the number of elements of $\mathbb{Z}_q$ in between those points) is the smallest possible. With this constraint, we know that the aforementioned distance is at least $q/2^d$. In the case of Kyber, we choose

$$S = \{\lfloor i \cdot q/2^d \rceil ; 0 \leq i < 2^d\}$$

Now that we have the intuition for it, let us formally define the compression and decompression functions used in Kyber.

Definition 9.2 (Compression and Decompression Functions)

Let q be a positive integer. For all $d < \lceil \log_2 q \rceil$, we define the functions $\mathsf{Compress}_d : \mathbb{Z}_q \to \mathbb{Z}_{2^d}$ and $\mathsf{Decompress}_d : \mathbb{Z}_{2^d} \to \mathbb{Z}_q$ as follows (q is implicit in the notation).

$$\begin{aligned} \mathsf{Compress}_d : \mathbb{Z}_q &\to \mathbb{Z}_{2^d} \\ x &\mapsto \lfloor x \cdot 2^d/q \rceil \mod^+ 2^d \end{aligned}$$

$$\begin{aligned} \mathsf{Decompress}_d : \mathbb{Z}_{2^d} &\to \mathbb{Z}_q \\ x &\mapsto \lfloor x \cdot q/2^d \rceil \end{aligned}$$

where $r \mod^+ 2^d$ is the unique representative of $r \in \mathbb{Z}_{2^d}$ in $\{0, \ldots, 2^d - 1\}$.

The end of the encryption procedure would then perform a compression of $\mathbf{c}_1$ and c_2, while the decryption procedure would start by decompressing these elements. As mentioned above, the compression induces an information loss in general, which means that $\mathsf{Decompress}_d(\mathsf{Compress}_d(x)) \neq x$ in the general case. It turns out that we keep the equality if $x \in \mathbb{Z}_{2^d}$. In other cases, the compression error can be bounded, which is crucial in the security evaluation of Kyber as well as the estimation of the decryption failure probability. We indeed have the following lemma.

Lemma 9.1 (Erreur de Compression)

Let q, d be positive integers such that $2^d < q$. Then for all x in $\mathbb{Z}_q$, we have

$$\mathsf{Decompress}_d\left(\mathsf{Compress}_d(x)\right) = x + e' \in \mathbb{Z}_q$$

for some $e' \in \mathbb{Z}$ such that $\left|e'\right| \leq q/2^{d+1} + 1/2$.

This error e' introduced by the compression must then be taken into account in the decryption. In particular, because the decryption procedure multiplies $\mathbf{c}_1$ with $\mathbf{s}$, the compression error of $\mathbf{c}_1$ will be multiplied by $\mathbf{s}$ as well. It is therefore this term that will impact the decryption failure probability the most. The compression will then need to be smaller for $\mathbf{c}_1$ than for c_2. Hence, $\mathbf{c}_1$ and c_2 having very distinct roles, we use two different compression parameters d_u and d_v. Moreover, to avoid any confusion, we note $\mathbf{u}, v$ the uncompressed versions of $\mathbf{c}_1, c_2$ (those in R_q), and keep the notations $\mathbf{c}_1, c_2$ for the compressed versions (thos in $R_{2^{d_u}}$ and $R_{2^{d_v}}$).

Increased Security with Compression

Additionally, we remark that the error stemming from the compression adds to the binomial errors of the M-LWE problem. As it increases the error, it seems like the compression improves the security of the system. It is indeed the case, but we can no longer rely solely on M-LWE for that. The compression error depends, in a deterministic way, on the value to be compressed. Yet, this value to be compressed is the M-LWE instance. Hence, the compression error depends on the M-LWE instance and we cannot argue indistinguishability from uniform directly. In practice, this corresponds to a new assumption which is somewhat hybrid between M-LWE and *Module Learning With Rounding* M-LWR.

The M-LWR problem can be seen as a variant of M-LWE where we add a deterministic rounding error instead of a random one. Concretely, the goal would be to recover $\mathbf{s}$ given $\mathsf{Decompress}_d\left(\mathsf{Compress}_d(\mathbf{As} \bmod qR)\right)$. This problem, at least in its unstructured version LWR, benefits from reductions from LWE in specific situations. In most concrete applications, its hardness is less studied than LWE or M-LWE. Nevertheless, here we have, as mentioned above, a hybrid version between M-LWE and M-LWR. Indeed, the (decompressed) ciphertexts are of the form $\mathsf{Decompress}_d\left(\mathsf{Compress}_d(\mathbf{As}+\mathbf{e} \bmod qR)\right) = \mathbf{As}+\mathbf{e}+\mathbf{e}' \bmod qR$. The combination of these two assumptions seems to reinforce the heuristic security of the system. We will not go further into the details of the security evaluation.

Message Encoding

The encryption algorithm requires encoding each coefficient of the message with respect to q so that the messages 0 and 1 are sufficiently distant to allow for correct decryption. This is why we multiply the message by $\lfloor q/2 \rceil$. This also requires the decoding of u during decryption, which corresponds to 0 when it is closer to 0 than $\lfloor q/2 \rceil$ and 1 otherwise. Thanks to the compression and decompression functions we just introduced, we can rewrite this two steps in a simpler way. In particular, when M has binary coefficients, we indeed have $\lfloor q/2 \rceil M = \mathsf{Decompress}_1(M)$ where the decompression parameter is $d = 1$. On the other end, the decoding step at the end of the decryption can be rewritten with $\mathsf{Compress}_1$ as well. Indeed, for all x in $\mathbb{Z}_q$, $\mathsf{Compress}_1(x)$ is exactly 0 if x is closer to 0 than $\lfloor q/2 \rceil$ and 1 otherwise.

9.1.4 Description de Kyber

We can finally describe the three algorithms KeyGen, Enc and Dec defining the Kyber public key encryption scheme in Algorithms 9.1, 9.2, and 9.3.

Algorithm 9.1: KeyGen (Kyber)

Input: Ring R of degree n, integers k, q, η_1.

1. $\rho \hookleftarrow U(\{0,1\}^{256})$
2. $\mathbf{A} \leftarrow XOF(\rho)$ ▷ $\mathbf{A} \in R_q^{k \times k}$
3. $\mathbf{s} \hookleftarrow \mathcal{B}_{\eta_1}^k$
4. $\mathbf{e} \hookleftarrow \mathcal{B}_{\eta_1}^k$
5. $\mathbf{t} \leftarrow \mathbf{As}+\mathbf{e} \bmod qR$

Output: $\mathsf{pk} = (\rho, \mathbf{t})$, $\mathsf{sk} = \mathbf{s}$

Algorithm 9.2: Enc (Kyber)

Input: Public parameters, public key $\mathsf{pk} = (\rho, \mathbf{t}) \in \{0,1\}^{256} \times R_q^k$, message $M \in \mathcal{M}$.

1. $\mathbf{A} \leftarrow XOF(\rho)$
2. $\mathbf{r} \hookleftarrow \mathcal{B}_{\eta_1}^k$
3. $\mathbf{e}_1 \hookleftarrow \mathcal{B}_{\eta_2}^k$
4. $e_2 \hookleftarrow \mathcal{B}_{\eta_2}$
5. $\mathbf{u} \leftarrow \mathbf{A}^T\mathbf{r}+\mathbf{e}_1 \bmod qR$
6. $v \leftarrow \mathbf{t}^T\mathbf{r}+e_2+\mathsf{Decompress}_1(M) \bmod qR$

Output: $(\mathbf{c}_1, c_2) = (\mathsf{Compress}_{d_u}(\mathbf{u}), \mathsf{Compress}_{d_v}(v))$

Algorithm 9.3: Dec (Kyber)

Input: Public parameters, secret key $\mathsf{sk} = \mathbf{s} \in R_q^k$, ciphertext $(\mathbf{c}_1, c_2) \in R_{2^{d_u}}^k \times R_{2^{d_v}}$.

1. $\mathbf{u} \leftarrow \mathsf{Decompress}_{d_u}(\mathbf{c}_1)$
2. $v \leftarrow \mathsf{Decompress}_{d_v}(c_2)$
3. $u \leftarrow v - \mathbf{s}^T\mathbf{u} \bmod qR$
4. $M \leftarrow \mathsf{Compress}_1(u)$

Output: $M \in \mathcal{M}$

Pseudo-randomization of the Encryption

As we will see when presenting the KEM, it is necessary to make the binomial sampling pseudo-random. It means that sampling is performed from a seed so that the same seed gives out the same samples. This is for example what is done for the matrix $\mathbf{A}$ where we sample it from the seed ρ. Concretely, the KeyGen algorithm takes as input a random 256-bit string d. We start by computing 512 pseudorandom bits which we segment into two 256-bit blocks. This is done by $(\rho, \sigma) = \mathsf{SHA3-512}(d\|k)$ (where k is the dimension of the matrix). The string ρ is used to expand $\mathbf{A}$ as above, and the string σ is used to sample the binomial elements $\mathbf{s}$ and $\mathbf{e}$. Concretely, a polynomial p is sampled according to $\mathcal{B}_\eta$ with the function $\mathsf{SamplePolyCBD}_\eta(\sigma, \mathsf{ctr})$ where ctr is a counter which increments after each sampled polynomial. In the end, it means that give the same random string d, we will generate the same key pair.

For the encryption, we generate a pseudorandom bit-string r for the binomial errors involved in the encryption, from the public key and the message itself. In other words, for the same message, and the same public key, the randomness will be the same hence giving the same ciphertext. This seems to contradict the classical IND-CPA security. However, for KEMs, the encrypted message is randomly chosen as it corresponds (or allows for deducing) the exhanged symmetric key. Making the encryption deterministic for a given message is then not a problem and is actually essential for the Fujisaki-Okamoto transform to obtain a KEM that is IND-CCA2-secure.

Parameters

Table 9.1 reports the three parameter sets called Kyber512, Kyber768 and Kyber1024. They correspond to different security levels, where the higher ones are more encline to be used in highly sensitive applications. We also give in Table 9.2 the sizes of the keys and ciphertexts for each security level.

Table 9.1 (Kyber Parameters)

	n	k	q	η_1	η_2	d_u	d_v	δ
Kyber512	256	2	3329	3	2	10	4	2^{-139}
Kyber768	256	3	3329	2	2	10	4	2^{-164}
Kyber1024	256	4	3329	2	2	11	5	2^{-174}

The parameter δ is the decryption failure probability.

Table 9.2 (Key and Ciphertext Sizes)

	$\|\mathsf{pk}\|$ (B)	$\|\mathsf{sk}\|$ (B)	$\|c\|$ (B)
Kyber512	800	1632	768
Kyber768	1184	2400	1088
Kyber1024	1568	3168	1568

9.2 ML-KEM: IND-CCA Key Encapsulation Mechanism

A Key Encapsulation Mechanism (KEM) is essentially the equivalent of a public key encryption scheme allowing for exchanging random fixed-size messages such as symmetric keys. In 1999, Fujisaki and Okamoto [FO99] proposed a generic transformation turning an IND-CPA public key encryption into an IND-CCA KEM, thus resisting to attacks of an adversary having access to a decryption oracle. In the context of KEMs, we talk about encapsulation Encaps and decpasulation

Decaps instead of encryption and decryption. Without entering the details, the idea of the Fujisaki-Okamoto transform is essentially to render the decapsulation oracle useless so that it returns something only when the adversary already knows the message. Hence, the adversary cannot send malformed ciphertexts for which it does not know the underlying message. For that, we ensure the randomness r used in the encryption algorithm depends on the encrypted message. More precisely, we generate a random message M (a 256-bit string), and generate $(K, r) = \mathcal{H}'(M \| \mathcal{H}(\mathsf{pk}))$. The exchanged symmetric key corresponds to K (256 bits) and the bitstring r (256 bits) is used to sample the errors $\mathbf{r}, \mathbf{e}_1, e_2$ in the encryption algorithm. The decapsulation can then decrypt and re-encrypt to ensure the two ciphertexts are identical. If not, the ciphertext was malformed. In that case, instead of sending an abort symbol $\perp$, we return a random value $\overline{K}$ (this method is called *implicit rejection*). Otherwise, the decrypted message is the correct one and the recipient can recompute the key K with $(K, r) = \mathcal{H}'(M \| \mathcal{H}(\mathsf{pk}))$.

The details of this transform go beyond the scope of this course so we only present the algorithms of ML-KEM.

Algorithm 9.4: KeyGen (ML-KEM)

1. $d, z \hookleftarrow U(\{0,1\}^{256})$
2. $(\mathsf{pk}, \mathsf{sk}') \leftarrow \mathsf{Kyber{-}KeyGen}(d)$ ▷ Specifying randomness d
3. $\mathsf{sk} \leftarrow (\mathsf{sk}', \mathsf{pk}, \mathcal{H}(\mathsf{pk}), z)$

Output: pk, sk

Algorithm 9.5: Encaps (ML-KEM)

Input: Public key pk.

1. $M \hookleftarrow U(\mathcal{M})$.
2. $(K, r) \leftarrow \mathsf{SHA3-512}(M \| \mathcal{H}(\mathsf{pk}))$
3. $\mathbf{c} \leftarrow \mathsf{Kyber{-}Enc}(\mathsf{pk}, M, r)$ ▷ Specifying randomness r

Output: Shared key K, ciphertext $\mathbf{c}$.

Algorithm 9.6: Decaps (ML-KEM)

Input: Secret key sk, ciphertext $\mathbf{c}$.

1. $M' \leftarrow \mathsf{Kyber{-}Dec}(\mathsf{sk}', \mathbf{c})$ ▷ sk' secret key for the PKE
2. $(K', r') \leftarrow \mathsf{SHA3-512}(M' \| \mathcal{H}(\mathsf{pk}))$
3. $\overline{K} \leftarrow \mathsf{SHAKE256}(z \| \mathbf{c})$
4. $\mathbf{c}' \leftarrow \mathsf{Kyber{-}Enc}(\mathsf{pk}, M', r')$
5. **If** $\mathbf{c} \neq \mathbf{c}'$ **then** $K' \leftarrow \overline{K}$

Output: Shared key K'

9.3 Comparison with Elliptic Curve Diffie-Hellman

The ML-KEM algorithm was specifically designed for the exchange of symmetric keys between two parties that do not share a common secret. Currently, the usual security protocols like TLS 1.3 use key exchange protocols for that, namely ECDH (or ECDH) for *Elliptic Curve Diffie-Hellman*. This solution relies on elliptic curve cryptography and more precisely on a variant of the discrete logarithm. As a reminder, an elliptic curve (over a prime field $\mathbb{Z}_p$) is a group of points $E = \{(x, y) \in \mathbb{Z}_p^2 : y^2 = x^3 + ax^2 + bx + c \bmod p\mathbb{Z}\} \cup \{P_\infty\}$, we p is large prime, a, b, c are integers and P_∞ is the identity element for the group operation. During an ECDH key exchange, each party sends the x coordinate of a point $P = (x, y)$ on the curve. Hence, this represents one element of $\mathbb{Z}_p$ each, i.e., $\lceil \log_2 p \rceil$ bits. The total communication cost is then $2\lceil \log_2 p \rceil$ bits.

A typical example of a curve used in practice is the X25519 curve where $a = 486662$, $b = 1$, $c = 0$ and $p = 2^{255} - 19$. It means that each element that is sent is 256 bits or 32 bytes, giving a total cost of 64 bytes. The ECDH protocol (and the key exchange protocols a la Diffie-Hellman in general) are not KEMs, and thus operate differently. This is why the two approaches are not entirely comparable. We nevertheless give a comparison between ECDH and ML-KEM-512 (based on Kyber512) in Table 9.3.

Table 9.3 (Comparison of Sizes and Timings)

Algorithme	$\|\mathsf{pk}\|$ (B)	$\|\mathsf{sk}\|$ (B)	$\|c\|$ (B)	KeyGen (ms)	Encaps (ms)	Decaps (ms)
ECDH (X25519)	-	-	**64**	-	0.040	0.040
ML-KEM-512	800	1632	768	0.013	**0.015**	**0.011**

All the sizes are expressed in bytes (B), and the timings in milliseconds (ms). The timings correspond to an implementation in "x86_64 OpenSSL implementation" on an Intel Xeon Platinum 8259CL processor operating at 2.50 GHz [Ope].

RECAP

- The Kyber public key encryption scheme is based on Regev's encryption with the following optimizations
  - Replace the leftover hash lemma with another LWE assumption to reduce parameters
  - Add algebraic structure by replacing $\mathbb{Z}$ with $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$. Allows one to encrypt n buts with a single element of R.
  - Replace the discrete Gaussians with centered binomial distributions with very small parameters (2, 3).
  - Store the matrix $\mathbf{A}$ as a 256-bit seed, and compress ciphertexts. The compression induces a deterministic rounding error which must be taken into account in the security estimation and evaluation of the decryption failure probability.
- The Kyber encryption scheme is IND-CPA under the M-LWE assumption and a hybrid variant of M-LWE due to compression. The encryption can be transformed into a IND-CCA2-secure KEM, called ML-KEM, with the Fujisaki-Okamoto transform.
- ML-KEM has three security levels. The algorithm is standardized by NIST in the FIPS 203 publication, and will serve as the post-quantum key exchange standard.

Bibliography

[BDK+18] J. W. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé. CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM. In *EuroS&P*, 2018.

[FO99] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO*, 1999.

[Lyu24] V. Lyubashevsky. Basic Lattice Cryptography: The concepts behind Kyber (ML-KEM) and Dilithium (ML-DSA). *IACR Cryptol. ePrint Arch.*, page 1287, 2024.

[Ope] OpenQuantumSafe. OpenSSL cryptographic suite benchmark. https://openquantumsafe.org/benchmarking/visualization/openssl_speed.html.

10

ML-DSA: Crystals-Dilithium

In this last chapter, we present the signature algorithm Crystals-Dilithium [DKL+18] which is one of the new digital signature standards. It is officially named ML-DSA and the standard has been published by NIST in the document FIPS 204. Just like we did for Kyber which was a series of optimizations of Regev's encryption, we proceed step by step to construct Dilithium. The scheme indeed follows from a series of optimizations on Lyubashevsky's signature which we covered in Chapter 7. We note here that we do not present the other lattice-based signature standard FN-DSA (following the Hash-and-Sign paradigm) as it requires deeper knowledge in the geometry of numbers and module lattices of rank 2. We however proposed a simplified version in TD 5 . This chapter is heavily inspired by the Dilithium specification and the tutorial of Lyubashevsky [Lyu24].

Contents

10.1 Improvements on Lyubashevsky's Signature

We have seen in Chapter 7 two main lattice-based signature designs. One of them correspond to the Fiat-Shamir paradigm, well known since Schnorr's signature [Sch89] and other subsequent variants such as Okamoto's [Oka92] or Katz and Wang's [KW03]. However, the peculiarities of lattices impose to manipulate short elements to ensure the security of the system (relying on the fact that an adversary should find a short vector in a given lattice without knowledge of a secret that would help them). This is what led to Lyubashevsky to add this rejection sampling step [Lyu09, Lyu12] which is essential to maintain the security of the signature as we have seen in TD 3 , giving its name to the *Fiat-Shamir with Aborts* paradigm. As it was presented initially, Lyubashevsky's signature works over the integers, with discrete Gaussian distributions (which makes the rejection step computationally complex), and without specific optimizations. We thus start from this scheme and present the different optimizations brought to obtain the signature Crystals-Dilithium.

10.1.1 Algebraic Structure

COntrarily to Regev's encryption which allowed for encrypting only one bit at a time, Lyubashevsky's signature hashes the message to be signed and thus allows for signing arbitrarily long messages. However, the scheme suffers from inefficiencies inherent to unstructured lattices. Indeed, the elements to stor are relatively large and the computations relatively inefficient. The public key contains two matrices $\mathbf{A}$ and $\mathbf{B} = \mathbf{AS}$ where $\mathbf{S}$ has k columns. This parameter k is chosen so that the space of possible challenges $\{-1, 0, 1\}^k$ is of size at least $2^{2\lambda}$ where λ is the security parameter.

Typically, when $\lambda = 128$, one would need to choose $k \geq 162$. Only storing the matrix $\mathbf{B}$ would then require $162 \cdot d\lceil \log_2 q \rceil$ bits.

The choice of the challenge space is not random. We indeed need the produced challenges to have a small norm while being in an exponentially large space. Let us now consider the cyclotomic ring $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$ where n is a power of 2. If we consider the set $S_1 = \tau^{-1}(\{-1, 0, 1\}^n)$, this set already contains 3^n elements, that are short with infinity norm bounded by 1, and where the elements are single ring elements instead of vectors. Working on rings thus allows us to choose $k = 1$ while preserving the same security guarantees.

Also, we change a little bit the presentation of the signature by specifying the identity in the matrix $\mathbf{A}$ of Algorithm 7.1. This allows for expressing $\mathbf{B}$ as an LWE instance. Combined with the use of the ring R, we obtain a first intermediate version of Lyubashevsky's signature scheme. We also change the notations to be coherent with those of the Dilithium standard.

KeyGen: Sample $\mathbf{s}_1 \hookleftarrow U(S_1^\ell)$ and $\mathbf{s}_2 \hookleftarrow U(S_1^k)$. Sample $\mathbf{A} \hookleftarrow U(R_q^{k\times\ell})$, and compute $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2 \bmod qR$. Output $\mathsf{pk} = (\mathbf{A}, \mathbf{t})$ and $\mathsf{sk} = \mathbf{s}$.

Sign: Given a message $M \in \{0,1\}^*$, sample $\mathbf{y}_1 \hookleftarrow \mathcal{D}_{R^\ell,s}$ and $\mathbf{y}_2 \hookleftarrow \mathcal{D}_{R^k,s}$. Compute $c = \mathcal{H}(\mathsf{pk}\|\mathbf{A}\mathbf{y}_1 + \mathbf{y}_2 \bmod qR\|M) \in S_1$, and $\mathbf{z}_1 = \mathbf{y}_1 + c\mathbf{s}_1$ and $\mathbf{z}_2 = \mathbf{y}_2 + c\mathbf{s}_2$. Accept $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2)$ with probability $P(\mathbf{z})$. If rejected, start over the procedure, and return $(\mathbf{z}, c)$ otherwise.

Verify: Given a message M and a signature $(\mathbf{z}_1, \mathbf{z}_2, c)$, we compute $c' = \mathcal{H}(\mathsf{pk}\|\mathbf{A}\mathbf{z}_1 + \mathbf{z}_2 - c\mathbf{t} \bmod qR\|M)$. Output 1 if $c' = c$ and if $\left\|[\mathbf{z}_1|\mathbf{z}_2]\right\|_2 \leq s\sqrt{n(k+\ell)}$.

10.1.2 Simpler Rejection: Uniform Distributions

In the previous algorithm, the rejection probability $P(\mathbf{z})$ depends on the chosen source and target distributions. Here, we choose Gaussian masks $\mathbf{y}_1, \mathbf{y}_2$ which means the source distribution is a discrete Gaussian shifter by $c[\mathbf{s}_1|\mathbf{s}_2]$. A fortiori, the simplest in this case is to choose the target distribution to be the same discrete Gaussian but centered at $\mathbf{0}$. This requires, when computing $P(\mathbf{z})$, the computation of a Gaussian mass and thus exponential functions. As the latter are transcendental, their computation requires floating-point arithmetic which is to avoid in cryptographic implementations. Moreover, this step must be protected against side-channel attacks. It is therefore necessary to make this rejection step simpler to protect.

We have briefly seen in TD 3 that it is possible to trade the discrete Gaussian distributions for uniform distributions over hypercubes. The advantage of said uniform distributions is that the rejection step simply consists in computing the infinity norm of $\mathbf{z} = [\mathbf{z}_1|\mathbf{z}_2]$. This change of parameter induces a new security evaluation, especially on the keys (which represent an M-LWE instance). We thus update the sampling of the secret key so as to produce slightly larger errors to increase the hardness of M-LWE. More precisely, instead of choosing $\mathbf{s}_1, \mathbf{s}_2$ uniform in S_1, we choose them uniform in S_η where η ranges between 2 and 4.

In order to guarantee there is no leakage during the rejection step, and thus ensure $\mathbf{z}$ is indeed uniform in its space, we must choose a bound on $\mathbf{z}$ that is smaller than that on $\mathbf{y}$. For that it suffices to choose the bound on $\mathbf{z}$ to be $\gamma_1 - B$ where γ_1 is the bound on $\mathbf{y}$, and B is an upper-bound on the infinity norm of $c[\mathbf{s}_1|\mathbf{s}_2]$. We must then choose the parameters B and γ_1 as small as possible to reduce the signature size.

In the case of rings, we can show that for all c and s in R, we have $\|cs\|_\infty \leq \|c\|_1\|s\|_\infty$. In the scheme above, we impose $\left\|[\mathbf{s}_1|\mathbf{s}_2]\right\|_\infty \leq \eta$. We must then bound $\|c\|_1$. Because c is uniform in S_1, its ℓ_1 norm is bounded by n in the worst-case, which is not entirely satisfying. In particular, if $n = 256$, the constaint on the challenge space $3^n > 2^{256}$ is vastly verified. It is then relevant to reduce the size of this space, by selecting only the challenges that have a small ℓ_1 norm to kill two birds with one stone. This can be achieved by limiting the number of non-zero coefficients of c. More precisely, we choose the challenge space to be

$$\mathcal{C} = \{c \in S_1 : \|c\|_1 = \widetilde{\tau}\}$$

This space is precisely of size $2^{\widetilde{\tau}}\binom{n}{\widetilde{\tau}}$, and we can efficiently sample uniform elements in it using a Fisher-Yates-like algorithm. In particular, to guarantee $|\mathcal{C}| > 2^{256}$, it suffices to choose $\widetilde{\tau} = 60 \ll 256$. This allows one to reduce the size of B. We then define $B = \widetilde{\tau}\eta$. We note that the standard documents use the notation τ instead of $\widetilde{\tau}$. In this course, τ is reserved for the coefficient embedding. We then obtain the following simplified version of the signature scheme.

KeyGen: Sample $\mathbf{s}_1 \hookleftarrow U(S_\eta^\ell)$ and $\mathbf{s}_2 \hookleftarrow U(S_\eta^k)$. Sample $\mathbf{A} \hookleftarrow U(R_q^{k\times\ell})$, and compute $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2 \bmod qR$. Output $\mathsf{pk} = (\mathbf{A}, \mathbf{t})$ and $\mathsf{sk} = \mathbf{s}$.

Sign: Given a message $M \in \{0,1\}^*$, sample $\mathbf{y}_1 \hookleftarrow U(S_{\gamma_1}^{\ell})$ and $\mathbf{y}_2 \hookleftarrow U(S_{\gamma_1}^{k})$. Compute $c = \mathcal{H}(\mathsf{pk} \| \mathbf{A}\mathbf{y}_1 + \mathbf{y}_2 \bmod qR \| M) \in \mathcal{C}$, and $\mathbf{z}_1 = \mathbf{y}_1 + c\mathbf{s}_1$ and $\mathbf{z}_2 = \mathbf{y}_2 + c\mathbf{s}_2$. Accept if $\left\|[\mathbf{z}_1|\mathbf{z}_2]\right\|_\infty \leq \gamma_1 - B$. If rejected, restart the procedure, and output $(\mathbf{z}, c)$ otherwise.

Verify: Given a message M and a signature $(\mathbf{z}_1, \mathbf{z}_2, c)$, we compute $c' = \mathcal{H}(\mathsf{pk} \| \mathbf{A}\mathbf{z}_1 + \mathbf{z}_2 - c\mathbf{t} \bmod qR \| M)$. Output 1 if $c' = c$ and if $\left\|[\mathbf{z}_1|\mathbf{z}_2]\right\|_\infty \leq \gamma_1 - B$.

For security reasons that we omit in this description, γ_1 must be chosen to be approximately $B \cdot n(\ell + k)$.

10.1.3 Signature Compression

Lyubashevsky's signature, and our intermediate variant, can in reality be seen as a zero-knowledge proof of the secret key. In other words, we prove to the verifier that we know the secret key associated to our public key, without leaking information on the secret key. Here, this means that we prove knowledge of[1] $(\mathbf{s}_1, \mathbf{s}_2)$ such that $\mathbf{A}\mathbf{s}_1 + \mathbf{s}_2 = \mathbf{t} \bmod qR$. Because $\mathbf{s}_2$ is very small compared to q, an idea to compress the signature is to only prove knowledge of $\mathbf{s}_1$ such that $\mathbf{A}\mathbf{s}_1 \approx \mathbf{t} \bmod qR$.

For example, if we consider the high order bits of $\mathbf{A}\mathbf{s}_1 + \mathbf{s}_2 \bmod qR$, they match with good probability those of $\mathbf{A}\mathbf{s}_1 \bmod qR$. Said differently, because $\mathbf{s}_2$ is small, it does not introduce carries on the high order bits through addition. We thus define two functions High_q and Low_q which output respectively the high order and low order bits of an element of R_q with a threshold $2\gamma_2$. We will define these functions precisely in Section 10.2, but at high level, $\mathsf{High}_q(x, 2\gamma_2)$ returns the quotient of the Euclidean division of x by $2\gamma_2$, and $\mathsf{Low}_q(x, 2\gamma_2)$ returns the remainder of the Euclidean division of x by $2\gamma_2$ but recentered in the interval $[\![-\gamma_2 + 1, \gamma_2]\!]$.

With these notations and by carefully choosing γ_2, in most cases we would have $\mathsf{High}_q(\mathbf{A}\mathbf{s}_1 + \mathbf{s}_2, 2\gamma_2) = \mathsf{High}_q(\mathbf{A}\mathbf{s}_1, 2\gamma_2)$. Not proving knowledge of the exact value of $\mathbf{s}_2$ means we do not need $\mathbf{z}_2$ and thus $\mathbf{y}_2$. We change the algorithm as follows.

KeyGen: Sample $\mathbf{s}_1 \hookleftarrow U(S_\eta^{\ell})$ and $\mathbf{s}_2 \hookleftarrow U(S_\eta^{k})$. Sample $\mathbf{A} \hookleftarrow U(R_q^{k \times \ell})$, and compute $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2 \bmod qR$. Output $\mathsf{pk} = (\mathbf{A}, \mathbf{t})$ and $\mathsf{sk} = \mathbf{s}$.

Sign: Given a message $M \in \{0,1\}^*$, sample $\mathbf{y} \hookleftarrow U(S_{\gamma_1}^{\ell})$. Compute $\mathbf{w} = \mathbf{A}\mathbf{y} \bmod qR$, and $\mathbf{w}_1 = \mathsf{High}_q(\mathbf{w}, 2\gamma_2)$. Then, compute $c = \mathcal{H}(\mathsf{pk} \| \mathbf{w}_1 \| M) \in \mathcal{C}$, and $\mathbf{z} = \mathbf{y} + c\mathbf{s}_1$. Accept if $\|\mathbf{z}\|_\infty \leq \gamma_1 - B$ and if $\left\|\mathsf{Low}_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2)\right\|_\infty \leq \gamma_2 - B$. If rejected, restart the procedure, and $(\mathbf{z}, c)$ otherwise.

Verify: Given a message M et a signature $(\mathbf{z}, c)$, we compute $c' = \mathcal{H}(\mathsf{pk} \| \mathsf{High}_q(\mathbf{A}\mathbf{z} - c\mathbf{t}, 2\gamma_2) \bmod qR \| M)$. Output 1 if $c' = c$ and if $\|\mathbf{z}\|_\infty \leq \gamma_1 - B$.

We explain here the several changes induced by this compression. Before, the verifier computed $\mathbf{A}\mathbf{z}_1 + \mathbf{z}_2 - c\mathbf{t}$ which was indeed equal to the $\mathbf{A}\mathbf{y}_1 + \mathbf{y}_2$ used during the signing process. Here, the value used in the hash is $\mathbf{w}_1 = \mathsf{High}_q(\mathbf{A}\mathbf{y})$. In our case, we have

$$\mathbf{A}\mathbf{z} - c\mathbf{t} = \mathbf{A}\mathbf{y} + c\mathbf{A}\mathbf{s}_1 - c\mathbf{t} = \mathbf{w} - c\mathbf{s}_2 \bmod qR.$$

The high order bits are thus that of $\mathbf{w} - c\mathbf{s}_2$ and not of $\mathbf{w}$. However, by ensuring the low order bits of $\mathbf{w} - c\mathbf{s}_2$ do not overflow, we ensure that the high order bits of $\mathbf{w} - c\mathbf{s}_2$ match that of $\mathbf{w}$. This is why the verification computes $\mathsf{High}_q(\mathbf{A}\mathbf{z} - c\mathbf{t}, 2\gamma_2)$ and that the signer checks that $\left\|\mathsf{Low}_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2)\right\|_\infty \leq \gamma_2 - B$. Indeed, if this last condition is verified, we have

$$\begin{aligned} & \mathbf{w} - c\mathbf{s}_2 = \mathsf{Low}_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2) + 2\gamma_2 \cdot \mathsf{High}_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2) \\ \iff \quad & \mathbf{w} = (c\mathbf{s}_2 + \mathsf{Low}_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2)) + 2\gamma_2 \cdot \mathsf{High}_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2) \end{aligned}$$

Yet $\left\|(c\mathbf{s}_2 + \mathsf{Low}_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2))\right\|_\infty \leq B + (\gamma_2 - B) = \gamma_2$. By unicity of the decomposition, we obtain that $\mathsf{High}_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2) = \mathsf{High}_q(\mathbf{w}, 2\gamma_2)$.

Beyond the sole correctness of the signature, the rejection performed by the signer with the condition $\left\|\mathsf{Low}_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2)\right\|_\infty \leq \gamma_2 - B$ is in fact paramount for the security of the system. As we have seen in TD 3, if this rejection is not performed, it is possible to mount a statistical attack to recover the secret key. Indeed, the adversary can recover $\mathbf{w} - c\mathbf{s}_2$ where $\mathbf{w}$ would be a centered random variable. By conditioning on c, we could then find the conditional distribution

[1] In reality, we prove knowledge of slightly larger elements $\overline{\mathbf{s}_1}, \overline{\mathbf{s}_2}, \overline{c}$ such that $\mathbf{A}\overline{\mathbf{s}_1} + \overline{\mathbf{s}_2} = \overline{c}\mathbf{t} \bmod qR$. This goes beyond the scope of this course.

whose expectation is linked to $\mathbf{s}_2$. Doing this rejection step, we force the random variable $\mathbf{X} = \mathsf{Low}_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2)$ to be uniform in $[-(\gamma_2 - B), (\gamma_2 - B)]$. Hence

$$\mathbf{w} - c\mathbf{s}_2 = \mathbf{X} + 2\gamma_2 \cdot \mathsf{High}_q(\mathbf{w}, 2\gamma_2)$$

where neither $\mathbf{X}$ nor $\mathsf{High}_q(\mathbf{w}, 2\gamma_2)$ depends on $\mathbf{s}_2$. Hence, the random variable $\mathbf{w} - c\mathbf{s}_2$ is now independent of $\mathbf{s}_2$.

10.1.4 Public Key Compression

First, the same way as we did for Kyber, it is possible to directly reduce the size of the public key by computing $\mathbf{A}$ from a seed ρ. This compacts the public key to only $256 + nk\lceil \log_2 q \rceil$ bits. However, for security reasons, the parameters of Dilithium must be chosen much larger than those of Kyber. In particular, instead of havinv $q \approx 2^{11.7}$ in Kyber, Dilithium must choose $q \approx 2^{23}$ which drastically increases the size of the public key. To limit this excessive size, the authors proposed a method to compress $\mathbf{t}$.

In the same way that it was sufficient to prove knowledge of $\mathbf{s}_1$ such that $\mathbf{A}\mathbf{s}_1 \approx \mathbf{t}$, it is possible to prove knowledge of $\mathbf{s}_1$ such that $\mathbf{A}\mathbf{s}_1 \approx \mathsf{High}_q(\mathbf{t}, 2^d)$ for a carefully chosen d. In other words, we ignore the low order bits of the public key during signing. By decomposing $\mathbf{t}$ into $\mathbf{t}_0 + 2^d\mathbf{t}_1$, and by only sending $\mathbf{t}_1$ as the public key, the verifier would however be incapable of computing $\mathsf{High}_q(\mathbf{A}\mathbf{z} - c\mathbf{t}, 2\gamma_2)$. We would then need to update the verification as we did before. However, if the low order bits of $c\mathbf{t}_0$ do not overflow, the verification would still work. We could thus do just like we did for $\mathbf{w} - c\mathbf{s}_2$ by ensuring $\|c\mathbf{t}_0\|_\infty \leq \gamma_2$ for example. Unfortunately, this simple test would still require a fairly small value of d, leading to only a small compression. Indeed, $c\mathbf{s}_2$ was very small because $\mathbf{s}_2$ had infinity norm $\eta = 2, 4$. Here, we have $\|c\mathbf{t}_0\|_\infty \leq \widetilde{\tau}2^d$, which would impose choosing a small d to use the same approach as before in an efficient way.

To improve the latter, the authors proposed to transmit an extra bit of information per coefficients, which would in reality correspond to the carry induced by $c\mathbf{t}_0$. This small modification of the signature allows one to choose a much larger d and compress the public key more efficiently. To compute this extra bit of information, we make use of the function $\mathsf{MakeHint}$, producing a vector $\mathbf{h} \in \tau^{-1}(\{0,1\}^n)^k$. The verifier will then use this carry vector $\mathbf{h}$ with the help of a function $\mathsf{UseHint}$ to recompute the proper information.

10.2 Description of ML-DSA: Crystals-Dilithium

We first define additional notations, as well as the functions necessary for compressing the signatures and the public key. We define $\widetilde{S}_\kappa$ as the subset of S_κ but where the lower bound is strict, i.e., $-\kappa < \tau_i(r) \leq \kappa$ for all i or simply $\widetilde{S}_\kappa = \tau^{-1}(\llbracket -\kappa + 1, \kappa \rrbracket^n)$. For an even (resp. odd) positive integer α, we define $r' = r \bmod^{\pm} \alpha$ as the unique element r' such that $-\alpha/2 < r' \leq \alpha/2$ (resp. $-(\alpha-1)/2 \leq r' \leq (\alpha-1)/2$) such that α divides $r' - r$. For all positive integer α, we also define $r' = r \bmod^{+} \alpha$ as the unique integer $r' \in \llbracket 0, \alpha - 1 \rrbracket$ such that α divides $r' - r$. We define the following functions. These functions extend to polynomials coefficient-wise, and then to vectors entry-wise.

We now give the complete algorithms of ML-DSA or Crystals-Dilithium in Algorithms 10.1, 10.2 and 10.3. We only simplify the presentation of certain parts like hashes, encodings, etc. In particular, just like Kyber, it is possible to specify a random string r as input to make the rest of the scheme deterministic. Said differently, given the same seed r, the scheme would produce the same keys and most importantly the same signatures.

Algorithm 10.1: KeyGen (Dilithium)

Input: Ring R of degree n, integers $k, \ell, q, \eta, \widetilde{\tau}, \gamma_1, \gamma_2, d, \omega$.

1. $\rho \hookleftarrow U(\{0,1\}^{256})$
2. $\mathbf{A} \leftarrow XOF(\rho)$ ▷ $\mathbf{A} \in R_q^{k \times \ell}$
3. $\mathbf{s}_1 \hookleftarrow U(S_\eta^\ell)$
4. $\mathbf{s}_2 \hookleftarrow U(S_\eta^k)$
5. $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2 \bmod qR$
6. $(\mathbf{t}_1, \mathbf{t}_0) \leftarrow \mathsf{Power2Round}_q(\mathbf{t}, d)$

Output: $\mathsf{pk} = (\rho, \mathbf{t}_1)$, $\mathsf{sk} = (\mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0)$

$\underline{\mathsf{Power2Round}_q(r, d)}$
00 $r^+ \leftarrow r \bmod^+ q$
01 $r_0 \leftarrow r^+ \bmod^\pm 2^d$
02 **return** $(r_1, r_0) = ((r^+ - r_0)/2^d, r_0)$

$\underline{\mathsf{MakeHint}_q(z, r, 2\gamma_2)}$
03 $r_1 \leftarrow \mathsf{HighBits}_q(r, 2\gamma_2)$
04 $v_1 \leftarrow \mathsf{HighBits}_q(r + z, 2\gamma_2)$
05 **if** $r_1 \neq v_1$ **then** $h \leftarrow 1$ **else** $h \leftarrow 0$
06 **return** h

$\underline{\mathsf{UseHint}_q(h, r, 2\gamma_2)}$
07 $\alpha \leftarrow (q-1)/(2\gamma_2)$
08 $(r_1, r_0) \leftarrow \mathsf{Decompose}_q(r, 2\gamma_2)$
09 **if** $h = 1$ et $r_0 > 0$ **return** $(r_1 + 1) \bmod^+ \alpha$
10 **elif** $h = 1$ et $r_0 \leq 0$ **return** $(r_1 - 1) \bmod^+ \alpha$
11 **else return** r_1

$\underline{\mathsf{Decompose}_q(r, 2\gamma)}$
12 $r^+ \leftarrow r \bmod^+ q$
13 $r_0 \leftarrow r^+ \bmod^\pm 2\gamma_2$
14 **if** $r^+ - r_0 = q - 1$
15 **then** $(r_1, r_0) \leftarrow (0, r_0 - 1)$
16 **else** $r_1 \leftarrow (r^+ - r_0)/(2\gamma_2)$
17 **return** (r_1, r_0)

$\underline{\mathsf{HighBits}_q(r, 2\gamma_2)}$
18 $(r_1, r_0) \leftarrow \mathsf{Decompose}_q(r, 2\gamma_2)$
19 **return** r_1

$\underline{\mathsf{LowBits}_q(r, 2\gamma_2)}$
20 $(r_1, r_0) \leftarrow \mathsf{Decompose}_q(r, 2\gamma_2)$
21 **return** r_0

Figure 10.1: Necessary functions for Dilithium compression.

Algorithm 10.2: Sign (Dilithium)

Input: Public parameters, public key $\mathsf{pk} = (\rho, \mathbf{t}_1) \in \{0,1\}^{256} \times R_q^k$, secret key $\mathsf{sk} = (\mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0) \in R^{\ell+k+k}$, message $M \in \{0,1\}^*$.

1. $\mathbf{y} \hookleftarrow U(\widetilde{S}_{\gamma_1}^{\ell})$.
2. $\mathbf{w} \leftarrow \mathbf{A}\mathbf{y} \bmod qR$
3. $\mathbf{w}_1 \leftarrow \mathsf{HighBits}_q(\mathbf{w}, 2\gamma_2)$
4. $c \leftarrow \mathcal{H}(\mathsf{pk} \| \mathbf{w}_1 \| M)$. ▷ $c \in \mathcal{C}$
5. $\mathbf{z} \leftarrow \mathbf{y} + c\mathbf{s}_1$
6. $\mathbf{r}_0 \leftarrow \mathsf{LowBits}_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2)$
7. **if** $\|\mathbf{z}\|_\infty \geq \gamma_1 - \eta\widetilde{\tau}$ **or** $\|\mathbf{r}_0\|_\infty \geq \gamma_2 - \eta\widetilde{\tau}$ **goto 1.**
8. $\mathbf{h} \leftarrow \mathsf{MakeHint}_q(-c\mathbf{t}_0, \mathbf{w} - c\mathbf{s}_2 + c\mathbf{t}_0, 2\gamma_2)$
9. **if** $\|c\mathbf{t}_0\|_\infty \geq \gamma_2$ **or** $\|\mathbf{h}\|_1 > \omega$ **goto 1.**

Output: $\mathsf{sig} = (\mathbf{z}, \mathbf{h}, c)$

Algorithm 10.3: Verify (Dilithium)

Input: Public parameters, public key $\mathsf{pk} = (\rho, \mathbf{t}_1) \in \{0,1\}^{256} \times R_q^k$, signature $\mathsf{sig} = (\mathbf{z}, \mathbf{h}, c) \in R^{\ell+k+1}$, message $M \in \{0,1\}^*$.

1. $\mathbf{w}_1' \leftarrow \mathsf{UseHint}_q(\mathbf{h}, \mathbf{A}\mathbf{z} - 2^d \cdot c\mathbf{t}_1 \bmod qR, 2\gamma_2)$
2. $b \leftarrow (c = \mathcal{H}(\mathsf{pk} \| \mathbf{w}_1' \| M)) \wedge (\|\mathbf{z}\|_\infty \leq \gamma_1 - \eta\widetilde{\tau}) \wedge (\|\mathbf{h}\|_1 \leq \omega)$

Output: $b \in \{0,1\}$

Parameters

We give in Table 10.1 the three parameter sets called Dilithium-2, Dilithium-3, and Dilithium-5, corresponding to the different security levels. We also give in Table 10.2 the key and signature sizes for each security level.

Table 10.1 (Parameters of Dilithium)

	n	q	k	ℓ	η	$\widetilde{\tau}$	γ_1	γ_2	d	ω
Dil-2	256	8380417	4	4	2	39	2^{17}	$\frac{q-1}{88}$	13	80
Dil-3	256	8380417	6	5	4	49	2^{19}	$\frac{q-1}{32}$	13	55
Dil-5	256	8380417	8	7	2	60	2^{19}	$\frac{q-1}{32}$	13	75

Table 10.2 (Sizes of keys and signatures of Dilithium)

	|pk| (B)	|sk| (B)	|sig| (B)
Dilithium-2	1312	2528	2420
Dilithium-3	1952	4000	3293
Dilithium-5	2592	4864	4595

10.3 Comparison with Elliptic Curve Digital Signature Algorithm

Digital signature algorithms like Falcon or Dilithium aim at authenticating messages, data, or keys during more complex security protocols. Today, the signature algorithm prioritized in all these protocols, such as TLS 1.3, is the ECDSA algorithm for *Elliptic Curve Digital Signature Algorithm*. An ECDSA signature simply consists in the x coordinate of a point $P = (x, y)$ of an elliptic curve $E = \{(x, y) \in \mathbb{Z}_p^2 : y^2 = x^3 + ax^2 + bx + c \bmod p\mathbb{Z}\} \cup \{P_\infty\}$, as well as a scalar of $\mathbb{Z}_{|E|}$. Hence, each signature has bit size $\lceil \log_2 p \rceil + \lceil \log_2 |E| \rceil$. One of the most commonly used curves is known as NISTP256 where $p = 2^{256} - 2^{32} - 977$ and $|E| \approx 2^{256}$. Hence, each signature is 512 bits or 64 bytes. Table 10.3 compares the sizes and timings of ECDSA with those of Falcon and Dilithium. We observe that Falcon and Dilithium are less compact than ECDSA in terms of size, but are very efficient and competitive with ECDSA in terms of timing. Moreover, Falcon is more compact than Dilithium but is slightly slower. Thence, the use of Falcon or Dilithium will depend on the application, some of which might want to minimize bandwidth or memory resources thus prefering Falcon, and others which might want to optimize computational resources thus prefering Dilithium. Each algorithm will most likely find relevant applications.

Table 10.3 (Comparison in sizes and timings)

Algorithme	|pk| (B)	|sk| (B)	|sig| (B)	KeyGen (ms)	Sign (ms)	Verify (ms)
ECDSA (NISTP256)	**32**	**32**	**64**	-	**0.024**	0.074
Falcon512	897	1998	666	8.64	0.352	0.057
Dilithium-II	1312	2528	2420	**0.035**	0.093	**0.035**

All sizes are expressed in bytes (B), and timings in milliseconds (ms). The timings correspond to an implementation in "x86_64 OpenSSL implementation" on an Intel Xeon Platinum 8259CL processor operating at 2.50 GHz [Ope].

RECAP

- The Dilithium signature scheme is based on Lyubashevsky's signature with a series of optimizations
  - Add algebraic structure by replacing $\mathbb{Z}$ by $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$. Allows for reducing the dimension of the challenge c to be only a single ring element, and thus reducing the dimension of the public key.
  - Replace discrete Gaussian distributions by uniform distributions over hypercubes. Allows for simplifying the rejection step which now simply consists in testing the infinity norm of a vector.
  - Compression of the signature by only keeping the high order bits in the commitment step. Requires another rejection step on the low order bits of $\mathbf{Ay} - c\mathbf{s}_2$ to guarantee correctness and security.
  - Storage of the matrix $\mathbf{A}$ as a 256-bit seed and compressing the public key $\mathbf{t}$ using a *hint* or *carry* vector. Requires another rejection step on $c\mathbf{t}_0$ and the computation of the carry vector, but allows for cutting the public key size in half.
- The Dilithium signature scheme is EUF-CMA under the M-LWE assumption as well as a variant of M-SIS due to compression. The signature can be made deterministic offering sEUF-CMA security.
- Dilithium has three security levels. The algorithm is standardized by NIST in the publication FIPS 204 under the name ML-DSA, and will serve as the post-quantum signature standard.

Bibliography

[DKL+18] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé. CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme. *TCHES*, 2018.

[KW03] J. Katz and N. Wang. Efficiency Improvements for Signature Schemes with Tight Security Reductions. In *CCS*, 2003.

[Lyu09] V. Lyubashevsky. Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures. In *ASIACRYPT*, 2009.

[Lyu12] V. Lyubashevsky. Lattice Signatures without Trapdoors. In *EUROCRYPT*, 2012.

[Lyu24] V. Lyubashevsky. Basic Lattice Cryptography: The concepts behind Kyber (ML-KEM) and Dilithium (ML-DSA). *IACR Cryptol. ePrint Arch.*, page 1287, 2024.

[Oka92] T. Okamoto. Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In *CRYPTO*, 1992.

[Ope] OpenQuantumSafe. OpenSSL cryptographic suite benchmark. https://openquantumsafe.org/benchmarking/visualization/openssl_speed.html.

[Sch89] C.-P. Schnorr. Efficient Identification and Signatures for Smart Cards. In *CRYPTO*, 1989.

Université de Rennes