Master's Thesis
Theoretical and Computational Methods
Theoretical Physics

NIST's Module-Lattice-Based Key-Encapsulation Mechanism and its security

Anna Jokiniemi

February 26, 2025

Supervisor(s): Esko Keski-Vakkuri and Valtteri Niemi

Examiner(s): Esko Keski-Vakkuri and Valtteri Niemi

University of Helsinki

Faculty of Science

PL 64 (Gustaf Hällströmin katu 2a)
00014 Helsingin yliopisto

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Degree programme
Faculty of Science	Theoretical and Computational Methods Theoretical Physics

Tekijä — Författare — Author

Anna Jokiniemi

Työn nimi — Arbetets titel — Title

NIST's Module-Lattice-Based Key-Encapsulation Mechanism and its security

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidantal — Number of pages
Master's Thesis	February 26, 2025	71

Tiivistelmä — Referat — Abstract

This thesis is an in-depth introduction to NIST's post-quantum Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM). We begin by introducing the problem at the core of ML-KEM, known as the Module Learning With Errors (MLWE) problem. In this problem, an adversary attempts to distinguish between samples selected from a specific distribution and samples from uniformly random samples. We also discuss the hardness of the MLWE problem, under the conjecture that certain lattice problems over modules are hard.

ML-KEM utilizes a variant of the Fujisaki-Okamoto transformation, which employs a weakly secure Public-Key Encryption (PKE) scheme to construct a strongly secure key-encapsulation mechanism in the Random Oracle Model (ROM). We explain the working principle of the MLWE-based public-key encryption scheme and demonstrate how ML-KEM is constructed from it.

Finally, we analyze the security of ML-KEM. We provide a tight security bound for IND-CCA security of ML-KEM in ROM, under the assumption that the underlying PKE is IND-CPA secure. Additionally, we present two security theorems in the Quantum Random Oracle Model (QROM). The first theorem establishes non-tightly IND-CCA security of ML-KEM in QROM if the MLWE problem is hard to solve. The second theorem assumes that the deterministic PKE is ciphertext-indistinguishable. Under this assumption, we derive a tighter bound for the IND-CCA security of ML-KEM in QROM.

Avainsanat — Nyckelord — Keywords

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

1. Introduction

It has been predicted that sufficiently powerful quantum computers will be built within the next 20 years [33]. Such computers could break a significant portion of modern cryptographic methods using Shor's algorithm.

Twenty years may sound like a long time, but even though such quantum computers do not exist yet, we should start preparing for them. An adversary could store data encrypted using non-quantum-safe methods today and wait until quantum computers become available. Once quantum computers exist, the adversary could break the encryption using this new technology. If data encrypted today remain sensitive long into the future, it should be protected in a quantum-safe way. Fortunately, experts of cryptography are already considering how to update today's security protocols to make them quantum-safe.

One of the largest international standardization bodies, the National Institute of Standards and Technology (NIST), develops cybersecurity standards that are widely adopted. NIST is the de facto global standards organization when it comes to information security, and nearly all other standardization organizations follow its recommendations. NIST recognizes the quantum threat, and in August 2024 NIST released three quantum-safe cryptography standards [34][42][43]. This thesis provides an in-depth introduction to one of these standards, the Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM)[34] standard and its security.

In the second chapter, we discuss why cryptography is important and examine the cryptographic and mathematical preliminaries required. In the third chapter, we introduce the Module Learning With Errors problem, which is the hard problem at the core of ML-KEM. Chapter 4 discusses the working principle of ML-KEM and Chapter 5 includes the security proofs. The classical security proof follows the works of Dennis Hofheinz, Kathrin Hovelmanns and Eike Kiltz in [24] and [23]. The quantum proof is based on the works of Tsunekazu Saito, Keita Xagawa and Takashi Yamakawa in [37]. Finally, Chapter 6 brings all of the above together.

2. Preliminaries

In this chapter, we begin with a discussion on the importance of cryptography and the threat posed by quantum computers. Next, we provide the cryptographic and mathematical prerequisites necessary for understanding security.

2.1 Role of cryptography in security

We will now discuss the importance of cryptography from Nokia's perspective. A radio access network (RAN) is part of a mobile telecommunication system, which connects user equipment, such as a phone or a computer, to a core network using radio access technology. The core network provides core services we expect from mobile phones, like messages, voice calls and internet access. RAN contains all the components required to connect a phone to the core network, including antennas, radio network controllers and base transceiver stations.

As the information a user accesses on their personal mobile phone is sensitive, numerous safety mechanisms are in place to ensure privacy and protection. These mechanisms are defined by standards created by the 3rd Generation Partnership Project (3GPP), which specifies the architecture of the systems, including algorithms used and their tasks. The secure environment is defined by 3GPP TS 33.401 for LTE and by TS 33.501 for 5G [1][2]. These standards utilize protocols like TLS (Transport Layer Security) and IPsec (Internet Protocol Security), standardized by Internet Engineering Task Force (IETF). These protocols rely on cryptographic schemes like Diffie-Hellman.

The security of cryptographic schemes is derived from the presumed hardness of a certain mathematical problem. These assumptions hold true when only classical computers are considered. However, the situation changes drastically with quantum computers. In 1994, Peter Shor discovered a quantum algorithm capable of solving efficiently factoring and discrete logarithm problems [39]. As a result, all systems whose security relies on these problems can become vulnerable to attacks with quantum computers. The vulnerable schemes include encryption schemes like RSA public key encryption, Diffie-Hellman key exchange, and elliptic curve encryption schemes. If sufficiently large quantum computers become available, all data secured using these methods will be at risk, posing a

significant threat to mobile phone security.

Even though sufficiently efficient quantum computers do not exist yet, we should already start preparing for their appearance. Transitioning to new security systems takes a significant amount of time. Moreover, there is a possibility that an adversary could store data encrypted using non-quantum-safe methods and wait until quantum computers become available. Once such computers exist, the adversary could use them to break the encryption and access the data. Therefore, if data encrypted today remains sensitive far in the future, where quantum computers may exist, it should be protected in a quantum-safe manner. [33]

Fortunately, there are still mathematical problems where quantum algorithms have not been successfully applied. For instance, quantum computers have little impact on the security of hash functions and symmetric encryption. A few years after Shor's algorithm was published, another quantum algorithm was introduced by Grover [21]. Grover's algorithm can be applied to hash algorithms, but it is not extraordinarily fast. By increasing the size of the key or hash, symmetric-key and hash function systems can be considered secure.[9]

The academic field that studies cryptographic methods resistant to both classical and quantum computers is known as post-quantum cryptography. This field differs from quantum cryptography, which studies ways to use quantum mechanics for design of cryptographic methods. In post-quantum cryptography, all designed algorithms remain classical, i.e., using them does not require a quantum computer. However, they are secure against attacks by both classical and quantum computers. Currently, four main families of post-quantum algorithms are considered for the design of quantum-safe methods. The first is hash-based signatures, which utilizes hash functions. The second family is multivariate polynomial schemes, which are based on the difficulty of solving systems of multivariate polynomials over finite fields. The third is code-based cryptography, which relies on difficult problems in coding theory. The fourth family is lattice-based cryptography, which uses lattice problems. The last one is the category we will focus on in this thesis. In addition, there are some proposed schemes that do not fall into these families, such as those based on the conjugacy search problem in braid groups. [14]

The international community has begun considering how today's security protocols should be updated to post-quantum era. One of the largest international standardization bodies is the National Institute of Standards and Technology (NIST), an agency of the United States Department of Commerce. NIST develops widely used standards through formal processes and rigorous studying. Another key organizations is the Internet Engineering Task Force (IETF), which provides standards related to the internet. Nearly all other standardization organizations, including 3GPP, follow the recommendations of NIST and IETF in the area of post-quantum cryptography.

In 2016, NIST initiated a process to standardize post-quantum algorithms. Initially, they received 82 candidates, but through evaluation and analysis, the list was narrowed down to three. The standard specifications of these three algorithms were released in August 2024. In this thesis, we focus on one of the three new standards: the Module-Lattice-based Key-Encapsulation Mechanism (ML-KEM). As the name suggests, the algorithm specified by this standard is based on lattice problems.

Many security standards in 3GPP rely on cryptographic methods that quantum computers could eventually break. This poses a significant threat for mobile phone security, making it highly likely that 3GPP standards will be updated to incorporate the new post-quantum algorithms recommended by NIST. Therefore, NIST's post-quantum standards will play a crucial role in our efforts to move towards a quantum-safe future and are of great interest to organizations like Nokia.

2.2 Basics of cryptography

Cryptography is the science of encrypting and decrypting data. It allows us to send sensitive information privately so that only the intended recipient can read it. [35] The original readable data that we want to send is called *plaintext*. The encryption algorithm is a mathematical function that *encrypts* this plaintext into an unreadable form called *ciphertext*. The ciphertext can be *decrypted* back into plaintext using the decryption algorithm. The algorithm requires a key, which specifies how the algorithm encrypts and decrypts the data. Encrypting the same plaintext with different keys will result in different ciphertexts. In practice, keys are long random bitstrings that are stored in encrypted form in computers and similar devices. If a recipient wants a ciphertext to be decrypted, they need to know the key.

Symmetric-key encryption uses the same key for both encryption and decryption. Let us consider Alice and Bob, who are trying to send each other messages securely. Alice chooses a key and uses it to encrypt a plaintext message into a ciphertext. She securely transfers the key to Bob. After sending the key, she can send him ciphertexts, knowing that only him, with the key, can decrypt them. This method works very quickly and is effective if there is a secure way for Alice to transfer the key to Bob. For example, Alice and Bob could meet in person to exchange the key, which they can use to decrypt and encrypt messages between each other.

However, if one needs to send a message to someone without having a secure way to transfer a key, this method does not work. In such cases, *asymmetric*, or *public key*, cryptography, which uses a pair of keys, is needed. One key, known as the *public key*, is used for encryption, and the other, called the *private key*, is used for decryption. The public key, as the name suggests, is public and anyone can see it and use it to encrypt

data. The holder of the private key is the only one capable of decrypting the message that has been encrypted with the public key. This way, there is no need to transfer the secret key.

Alice chooses two keys that are mathematically linked to each other. She makes one key public so that anyone can see it while keeping the other key secret. Bob, who wants to send a message to Alice, sees this public key and uses it to encrypt his message to Alice. Only Alice's secret key can decrypt the resulting ciphertext, and Bob can send the ciphertext to Alice, while knowing that only she can read it.

Unfortunately, asymmetric schemes are inefficient and require significant computational power to send large messages. If we want to send a large message or multiple messages, we use a hybrid cryptosystem known as Key-Encapsulation Mechanism (KEM), which combines both asymmetric and symmetric encryption schemes. In a KEM, asymmetric encryption is used to send a symmetric key, which can then be used to efficiently encrypt messages using symmetric encryption. We will discuss these different methods in more detail in the section 2.3.

A person or algorithm attempting to break an encryption scheme is called an adversary. The initial instinct might be to keep the entire encryption scheme secret from the adversary. However, in practice, a cryptographic scheme may have hundreds or even thousands users, making it impossible to guarantee that the scheme remains secret. Therefore, relying on keeping the scheme secret is not a feasible solution. An encryption scheme should be designed to remain secure even if the adversary knows its working principle and manages to obtain the ciphertext. Our goal is to keep only the key secret, minimizing the possibilities for breaks because of information leakage.

We can go even further and require that the working principle of the scheme is completely public. This allows multiple researchers to test and study the scheme extensively, ideally finding its weaknesses before a real-life adversary can find them. Schemes that are deemed secure after this rigorous study are standardized by various organizations. Then users do not need to know all the details of the scheme and can simply select a standard and follow its recommendations. Standardization also provides a way for multiple systems to work together and ensures interoperability. [35][27]

2.3 Encryption schemes and their properties

Let us formalize the discussion of the previous section. The plaintext message space $\mathcal{M}$ is a set of all possible messages that can be encrypted by the encryption scheme. A plaintext in this space is denoted as $m \in \mathcal{M}$. We also have a key space $\mathcal{K}$ that defines all possible keys $k \in \mathcal{K}$ that the scheme can use, and a ciphertext space $\mathcal{C}$ that includes all possible ciphertexts $c \in \mathcal{C}$ that the scheme can produce.

An encryption scheme Π is defined by three algorithms: key generation (Gen), encryption (Enc), and decryption (Dec) algorithms. We denote the scheme as: $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$.

A symmetric encryption scheme is presented in Algorithm 1. An asymmetric, or Public Key Encryption (PKE), scheme is displayed in Algorithm 2.

Algorithm 1 Symmetric encryption scheme Π

Gen(1^n)**:** The key generation algorithm Gen generates a key k for a scheme according to some distribution. We denote this as $k \leftarrow \text{Gen}(1^n)$. Here, n is the security parameter of the system, presented in unary form, which we will introduce later.

Enc(k, m)**:** The encryption algorithm Enc takes as input the key k and plaintext m, and produces a ciphertext c. We denote this as $c \leftarrow \text{Enc}(k, m)$.

Dec(k, c)**:** The decryption algorithm Dec takes as input a key k and ciphertext c, and returns the original plaintext m, denoted as $m \leftarrow \text{Dec}(k, c)$. [27]

Algorithm 2 Asymmetric or public key encryption scheme Π

Gen(1^n)**:** The key generation algorithm Gen generates two keys (pk, sk). The private key sk is kept secret, while the public key pk is made public.

Enc(pk, m)**:** The encryption algorithm Enc takes as input the public key pk and plaintext m. It produces a ciphertext c and we denote this as $c \leftarrow \text{Enc}(pk, m)$.

Dec(sk, c)**:** The decryption algorithm Dec takes as input the secret key sk and ciphertext c, and returns the original plaintext m. We denote this as $m \leftarrow \text{Dec}(sk, c)$. [27]

As previously mentioned, asymmetric schemes are inefficient, so instead, we use a Key-Encapsulation Mechanism (KEM) where a public key encryption scheme is used to send a symmetric key. This key can then be used to symmetrically encrypt messages. The working principle is similar to that of asymmetric schemes in Algorithm 2, but KEM does not allow the user to choose a message to be encrypted. In KEM, the encryption algorithm generates a symmetric key and returns it to the user. It also returns a ciphertext that contains the key in encrypted form and which is sent to the recipient. The recipient then decrypts the ciphertext using the decryption algorithm to obtain their copy of the symmetric key. The generic KEM is shown in the Algorithm 3.

For the symmetric encryption scheme to be *perfectly correct*, we require that for every key $k \in \mathcal{K}$ and a plaintext $m \in \mathcal{M}$ it holds that $\text{Dec}(k, \text{Enc}(k, m)) = m$. Similarly, for the asymmetric scheme, perfect correctness is defined with the equation $\text{Dec}(sk, \text{Enc}(pk, m)) = m$. Some encryption schemes are not perfectly correct, rather, their internal design will cause some infrequent errors.

Algorithm 3 Key-encapsulation mechanism Π

Gen(1^n)**:** The key generation algorithm Gen generates two keys $(pk, sk) \in \mathcal{K}$. The private key sk is kept secret, while the public key pk is made public.

Enc(pk)**:** The encryption algorithm Enc takes as input the public key pk and generates a shared symmetric key K according to some distribution. It produces a ciphertext c from the shared key K and returns both the key K and the ciphertext c to the user.

Dec(sk, c)**:** The decryption algorithm Dec takes as input the secret key sk and ciphertext c, and returns the shared symmetric key K to the user. This is denoted as $K \leftarrow \mathrm{Dec}(sk, c)$. [27]

We define a looser correctness definition for public-key encryption, where we allow the scheme to make mistakes occasionally, but not too often, because otherwise the quality of the scheme would suffer. For fixed keys (pk, sk), the most probable decryption error occurs with a probability $\max_{m \in \mathcal{M}} \Pr[\mathrm{Dec}(sk, c) \neq m | c \leftarrow \mathrm{Enc}(pk, m)]$. We take an average of this probability over the key space. If this average probability is smaller than δ, such that

$$E[\max_{m \in \mathcal{M}} \Pr[\mathrm{Dec}(sk, c) \neq m | c \leftarrow \mathrm{Enc}(pk, m)]] \leq \delta$$

we say PKE is δ-correct. When PKE is perfectly correct it makes no decryption errors and δ is zero.

We can also view δ-correctness as a game where an adversary A plays a correctness game G^A_{COR} displayed in Algorithm 4. In this game, there is an imaginary game master, called the challenger, who *challenges* the adversary A to play. The challenger runs the key generation algorithm (line 1), generating a pair of a secret and a public key. It gives both keys to the adversary A, who then outputs a message m and gives it to the challenger, denoted as $m \leftarrow A(pk, sk)$ (line 2). The adversary A wins the game if the message it outputs induces a decryption error. The challenger evaluates whether the message causes an error in lines 3 and 4.

If the PKE is δ-correct, there is an upper bound on A's success. The probability of A winning the game and triggering a decryption failure is given by

$$\Pr\left[G^A_{\mathrm{COR}} = 1\right] \leq \delta$$

[23][24]

Secure public-key encryption algorithms are always probabilistic; they output a different ciphertext when run multiple times, even with the same key. They utilize a random *coin*. We flip a coin, resulting in a random bit. We then flip a coin multiple times and form a random sequence of bits, denoted as r, which serves as a source of randomness in the public-key encryption scheme. A new r is chosen randomly from a *coin space*,

Algorithm 4 Correctness game G^A_{COR} [24]. We use a notation $[\![\cdot]\!]$ to denote a boolean value of a statement.

1: $(pk, sk) \leftarrow \text{Gen}(1^n)$
2: $m \leftarrow A(sk, pk)$
3: $c \leftarrow \text{Enc}(pk, m)$
4: **return** $[\![\text{Dec(sk,c)} \neq m]\!]$

denoted as $\mathcal{R}$, every time the encryption algorithm is executed. This randomness is crucial for the security of the scheme because repeated patterns could be exploited by an adversary. For example, when people send messages, they often use common short words or phrases that can be guessed, such as "hello" at the beginning of an email. Additionally, communication protocols include many repeating message patterns as part of their standardized interaction procedures. If these phrases are always encrypted to the same ciphertext, an adversary could detect the repeated patterns and infer their meaning. Most of the time, this randomness inside encryption is not explicitly denoted, as it is assumed to hold implicitly. However, if we want to specifically indicate this, we use the notation $\text{Enc}(pk, m, r)$ instead of $\text{Enc}(pk, m)$.

We want to evaluate how well our scheme produces different ciphertexts over a coin space. For a fixed plaintext and a public key pk, we encrypt the message using all possible values in the coin space and measure the probability of the most probable ciphertext. The *min-entropy* for an encryption algorithm is defined as the negative logarithm of this probability as:

$$\gamma(pk, m) := -\log_2 \max_{c \in \mathcal{C}} \Pr_{r \leftarrow \mathcal{R}}[c = \text{Enc}(pk, m, r)]$$

If there exists a lower bound such that $\gamma(pk, m) \geq \gamma$ for all values of pk and m, we say that the PKE is γ-spread.

If a PKE is γ -spread, it follows that the probability of obtaining specific ciphertext is small. Consequently, for every possible ciphertext $c \in \mathcal{C}$, it holds that

$$\Pr_{r \leftarrow \mathcal{R}}[c = \text{Enc}(pk, m; r)] \leq 2^{-\gamma}$$

This ensures that no ciphertext becomes overly probable when encrypting any message with any public key. [23]

2.4 Security definitions

An adversary can attempt many different types of an attack against the scheme. If we want to prove the security of our scheme, we need to model these real-world threats mathematically. We create mathematical models to capture how much information an

adversary can obtain in these different attack types. Similar to the correctness game, we construct an attack game played between the adversary and a *challenger*. The challenger is again an entity that runs these games and validates if the adversary has succeeded. These games are modeling the attack, specifying what it entails and what it means for the adversary to break the scheme. We aim to provide the adversary with at least as much knowledge and power in these attack games as an attacker would have in an equivalent real-life scenario. On the other hand, the games are defined so that if an attacker would succeed in a real-life scenario, then the corresponding adversary in the idealized game would also succeed.

There is a risk that the adversary can obtain a ciphertext, so in some attack games, the adversary is given access to the ciphertext and attempts to gain information from it. Additionally, there is a possibility that the adversary could impersonate someone else and deceive the user of the cryptographic scheme into sending the adversary the ciphertext of a plaintext the adversary has selected. To simulate this, some attack games allow the adversary to query the encryption of plaintexts it chooses.

Let $\mathcal{C}(m)$ denote the distribution of ciphertexts that the encryption algorithm produces when encrypting the message m over all possible keys and random coins. We consider the scheme to be secure if the produced ciphertext reveals little to nothing about the plaintext from which it was generated. This means that the plaintext should be independent of the ciphertext, and that the ciphertext distribution should not change when two different messages are encrypted. For every $m_0, m_1 \in \mathcal{M}$, it should hold that $\mathcal{C}(m_0)$ and $\mathcal{C}(m_1)$ are identical or at least indistinguishable. In a corresponding attack game, the adversary may choose two plaintexts, and one of them is encrypted to a ciphertext. Given the ciphertext, the adversary should not be able to determine which of the two plaintexts was encrypted. This property, known as *indistinguishability*, forms the basis for most attack games.

If the scheme is *perfectly secure*, the adversary would not obtain any information about the plaintext even with unlimited computational power. It could only guess which plaintext was encrypted and would succeed with a probability of 1/2. But it has been proven that in a perfectly secure scheme the key must be as long as the sent message itself. Most of the time, this condition is unattainable, making perfectly secure schemes rarely used. Most modern cryptography relies on *computationally secure* schemes, which are unbreakable when the adversary uses realistic computational power.

In the real world, no adversary has unlimited computational power, so we do not worry about such a case. Instead, we focus on adversaries that are efficient. An efficient adversary is defined as one with polynomial running time t. This means that there exists a polynomial $\mathrm{p}(n) = a \cdot n^c$, for some constants a and c, such that the adversary's running time t satisfies $t \leq \mathrm{p}(n)$. Here, n represents the *security parameter* of the scheme, which

is chosen based on the desired level of security. The larger the value of n, the more secure the system, in the sense that it can resist more powerful attacks. Greater security usually comes at the cost of efficiency. Often, n represents the length of the key.

We allow an adversary to succeed in breaking the scheme with some very small probability, which in practical usage, we are not concerned about. We allow the success probability to be negligible, i.e., it grows slower than any inverse polynomial n^{-c} for all large enough value of n. A function $\epsilon(n)$ is negligible, if for all polynomials $\mathrm{p}(n)$, there exists an integer N such that for all integers $n > N$, it holds that $\epsilon(n) < \mathrm{p}(n)^{-1}$. Then, for an adversary A, the probability of successfully breaking the scheme should be negligible such that for some negligible function $\epsilon(n)$ it holds that $\Pr[A \text{ Succeeds}] \leq \epsilon(n)$.

Negligible functions satisfy the following calculation rules. Let ε_1 and ε_2 be negligible functions, and let $p(n)$ denote an arbitrary polynomial. Then, functions $\varepsilon_3 = \varepsilon_1(n) + \varepsilon_2(n)$ and $\varepsilon_4 = p(n) \cdot \varepsilon_1(n)$ are also negligible. [27]

Based on these discussions, we define a scheme to be secure if every probabilistic polynomial-time adversary A, carrying out a specific attack succeeds with only a negligible probability.

2.4.1 Indistinguishability Attacks

In this subsection, we introduce two common attack scenarios; Indistinguishability under Chosen-Plaintext Attack (IND-CPA) and Indistinguishability under Chosen-Ciphertext Attack (IND-CCA). The IND-CPA is displayed in Algorithm 5 and IND-CCA in Algorithm 6 [27]. However, we would like to emphasize that there is a wide variety of attack scenarios and even a single security proof may use multiple of them. In Chapter 5, we encounter many different attack games. Understanding the logic behind these common cases makes it easier to work with other attack scenarios.

The adversary is given access to the public key and, depending on the type of attack, access to some *oracle*. An oracle is a black-box system, meaning the adversary can provide inputs and receive outputs but has no knowledge of the internal working of the oracle. The adversary can only gain knowledge of oracle's behavior by observing the input-output interaction. When an adversary has access to an oracle, this is denoted using the superscript indicating the oracle. In the IND-CPA scenario, the adversary A is given oracle access to the encryption algorithm, allowing it to encrypt any message it chooses. This is denoted as $A^{\mathrm{Enc}(pk,\cdot)}$. In the IND-CCA scenario, the adversary A is given additional oracle access to the decryption algorithm, denoted as $A^{\mathrm{Enc}(pk,\cdot),\mathrm{Dec}(sk,\cdot)}$.

Let $\Pi = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ be a public key encryption scheme and A a probabilistic polynomial-time adversary. The adversary has access to either encryption or both encryption and decryption oracle depending on which game we are playing. Challenger runs

the key-generation algorithm and generates keys (line 1). The public key is given to the adversary, who then generates two plaintexts $m_0, m_1 \in \mathcal{M}$ for the challenger (line 2). It is crucial that messages m_0 and m_1 are of the same length. Typically, schemes produce larger ciphertext for larger plaintext. If the adversary gives two different length messages to the challenger, it would receive two different length ciphertexts and could easily deduce which text is which, thus always winning the game. [27] We denote the adversary choosing the messages as $m_0, m_1 \leftarrow A^{\mathrm{Enc}(pk,\cdot)}(pk)$, where $|m_0| = |m_1|$ in the IND-CPA scenario. It is important to note that a real-world attacker may deduce some information from the length of the ciphertext. These games are mathematical models, and they do not account for all types of attacks.

The challenger randomly selects a *challenge bit* b, denoted as $b \xleftarrow{\$} \{0,1\}$, and uses it to determine which of the plaintexts provided by the adversary will be encrypted (line 3). It encrypts the chosen plaintext m_b into a *challenge ciphertext* $c^* \leftarrow \mathrm{Enc}(pk, m_b)$. This challenge ciphertext is then given to the adversary, who tries to distinguish which of the two plaintexts was encrypted and deduce the bit b using oracles accesses it has (line 4).

In the IND-CCA scenario, the adversary is given oracle access to the decryption algorithm, allowing it to request the decryption of any ciphertext it chooses. However, it is prohibited from querying the decryption of the challenge ciphertext, as doing so would allow it to always succeed in determining which plaintext was encrypted.

The adversary can make as many queries to the oracle as it wishes and, after some time, outputs its *guess bit* b' (line 5). The output of the game is defined to be 1 if A correctly guessed the challenge bit such that $b' = b$, and 0 otherwise. We use the notation $[\![\cdot]\!]$ to indicate the truth value of the statement inside. If the attack game returns 1, we say that A has won the game.

Algorithm 5 IND-CPA game $G_{\text{IND-CPA}}$

1: $(pk, sk) \leftarrow \mathrm{Gen}(1^n)$
2: $m_0, m_1 \leftarrow A^{\mathrm{Enc}(pk,\cdot)}(pk)$, where $|m_0| = |m_1|$.
3: $b \xleftarrow{\$} \{0,1\}$
4: $c^* \leftarrow \mathrm{Enc}(pk, m_b)$
5: $b' \leftarrow A^{\mathrm{Enc}(pk,\cdot)}(pk, c^*)$
6: **return** $[\![b = b']\!]$

Algorithm 6 IND-CCA game $G_{\text{IND-CCA}}$

1: $(pk, sk) \leftarrow \text{Gen}(1^n)$
2: $m_0, m_1 \leftarrow A^{\text{Enc}(pk,\cdot),\text{Dec}(sk,\cdot)}(pk)$, where $|m_0| = |m_1|$.
3: $b \xleftarrow{\$} \{0, 1\}$
4: $c^* \leftarrow \text{Enc}(pk, m_b)$
5: $b' \leftarrow A^{\text{Enc}(pk,\cdot),\text{Dec}(sk,\cdot)}(pk, c^*)$
6: **return** $[\![b = b']\!]$

If the adversary is simply tossing a coin and making a random guess, it succeeds in these attack games with a probability of 1/2. If it is able to determine which text was encrypted with a probability higher than 1/2, we need to be concerned that it is gaining some information about the plaintext from the ciphertext. To quantify this, we define a measure called *advantage*, which tells us how well the adversary is performing compared to the ideal that they gain no information. An IND-CPA advantage of adversary A against the scheme Π is defined as

$$\text{Adv}_{\Pi}^{\text{IND-CPA}}(A) = \Pr[G_{\text{IND-CPA}} = 1] - \frac{1}{2}. \tag{2.1}$$

If the adversary is just guessing at random, the advantage is zero. We say that the scheme is IND-CPA secure if any probabilistic polynomial-time adversary has a negligible advantage in the IND-CPA game, i.e., it succeeds with probability that differs from that of random guessing just with a negligible amount. [8]

We can define a similar advantage for the IND-CCA game and obtain the IND-CCA advantage as follows

$$\text{Adv}_{\Pi}^{\text{IND-CCA}}(A) = \Pr[G_{\text{IND-CCA}} = 1] - \frac{1}{2}. \tag{2.2}$$

We say that the scheme is IND-CCA secure if any probabilistic polynomial-time adversary has only a negligible advantage. [8]

2.4.2 Advantage of distinguishing two games

We extend the definition of advantage to distinguishing between two games, G_0 in Algorithm 7 and G_1 in Algorithm 8. The adversary A alternates between the games and tries to determine whether it is interacting with G_0 or G_1. A returns a bit b' indicating which game it believes it is interacting with.

Algorithm 7 The game G_0

1: Rules for the game G_0
2: $b' \leftarrow A$
3: **return** b'

Algorithm 8 The game G_1

1: Rules for the game G_1
2: $b' \leftarrow A$
3: **return** b'

If the adversary A is effective at distinguishing between the two games, its probability of outputting 1 when interacting with G_1 is higher than when interacting with G_0. Conversely, if A cannot distinguishing between the games, it will return 1 with the same probability, regardless of which game it is interacting with. The advantage of adversary A in distinguishing between games G_0 and G_1 is defined as:

$$\text{Adv}(A) = |\Pr[G_1 = 1] - \Pr[G_0 = 1]| \tag{2.3}$$

If A cannot distinguish games, the advantage is zero.[8]

2.5 Reduction

Unfortunately, today's security schemes are too complex for us to prove them secure in the information-theoretic sense. We would need mathematical tools that do not yet exist, and providing full proof of security would require a breakthrough in complexity theory. Instead, we rely on empirical evidence. If a problem has been studied for long time – say, a decade – by many researchers and organizations, and they have been unable to find an efficient algorithm to solve the problem, we can assume that the problem is hard to solve.

There is a limited number of problems that can be trusted to be sufficiently analyzed. Additionally, many of these well-studied problems would result in inefficient and poorly functioning security schemes, so we need more problems that we can trust to be hard. To achieve this, we use a method called *reduction*. If we believe that a problem P is hard to solve, we can use this conjecture to prove the hardness of another problem, denoted R. This way we can find new problems that we can trust to be difficult.

We start with the knowledge that the problem P seems to be hard to solve. The problem P can then be reduced to a problem R by showing that if an adversary could solve R, it could also solve P. But since we assume that the problem P cannot be solved, we can conclude that R cannot be solved either. [8]

When we find a hard problem that can be used to construct an efficient security scheme, we again use a reduction to prove the security of the scheme. We conjecture that problem P is hard, meaning that no polynomial-time algorithm can solve it with non-negligible probability. Now, we want to prove the security of a cryptographic scheme Π. Real-life breaking attempts are modeled by an adversary A playing an attack game against the scheme. We construct another adversary A', which attempts to solve problem

P by using adversary A as a subroutine. This construction is designed so that if A succeeds in the attack game and thus succeeds to break Π, then A' will succeed in solving P. [27]

We say that this reduction is *tight*, if the running times and success probabilities for adversaries A and A' are approximately the same. This ensures that if the problem is hard for A' – meaning the running time of A' is large and success probability is low – then the same holds for A. When the reduction is *non-tight*, A may be significantly more efficient compared to A'. It may succeed with a significantly higher probability or it may be significantly faster. In this case, the reduction does not directly guarantee that the scheme is secure. If problem P is hard for A', reduction can only provide an asymptotic analysis of the security. [37]

We use the *sequence-of-games* approach, that is also called the *game hopping* approach, in reductions. We construct a sequence of games $G_0, G_1, \ldots, G_n$, where the game G_0 is the attack game against Π for the adversary A and the last game G_n is solving the problem P for the adversary A'. We aim to transition from game G_0 to game G_n by introducing intermediate games, where each game's success probability differs from that of the previous game in a small, calculable, way.

Let us consider intermediate games G_i and G_{i+1}. Sometimes, these games are essentially the same, but certain quantities are reformulated while preserving the behavior of the game. These changes do not affect the outcome and success probability of the game, i.e.,

$$\Pr[G_{i+1} = 1] = \Pr[G_i = 1] \tag{2.4}$$

but they make the proof more readable. Other times, the two games proceed identically unless some event F occurs. Then the success probabilities between the games differ only by the probability of F occurring, and we can write an upper bound

$$|\Pr[G_{i+1} = 1] - \Pr[G_i = 1]| \leq \Pr[F] \tag{2.5}$$

We then show that the probability of F occurring is negligible, and it follows that the success probabilities of these games differ only negligibly.

The third way the two games can differ is based on indistinguishability. Games G_i and G_{i+1} differ in such a way that distinguishing between them would mean that the adversary could differentiate between two indistinguishable distributions [40]. Let P_0 and P_1 be distributions that are assumed to be indistinguishable. There exists a distinguishing algorithm D that alternates between games G_i and G_{i+1}. When D receives as input a value sampled from the distribution P_0, it outputs 1 with probability $\Pr[G_i = 1]$. When D receives an input sampled from P_1, it outputs 1 with probability $\Pr[G_{i+1} = 1]$. Since the distributions P_0 and P_1 are assumed to be indistinguishable, the notion

$$|\Pr[G_{i+1} = 1] - \Pr[G_i = 1]| \tag{2.6}$$

is negligible [40].

We analyze the sequence of games $G_0, G_1, \ldots, G_n$ examining each pair G_i, G_{i+1} and applying one of the three Equations (2.4),(2.5) or (2.6) to each pair. Then we have shown that the success probability of each game differ from the previous one only negligibly or not at all. Consequently, the success probability of the first game, G_0, differs from that of the last game, G_n, at most negligibly. [40]

Since the first game in the sequence, G_0, is the attack game against Π for adversary A, and the last game, G_n, is the attack game where A' attempts to solve problem P, then the success probabilities of these two attack games differ at most negligibly. Since we assume that the problem P is hard to solve, A' can win the attack game of solving the problem P with only negligible probability. It then follows that A can also win the attack game against Π only with negligible probability, thereby proving the security of the scheme Π under the assumption that the problem P is hard to solve. [40]

However, in some cases, analyzing the sequence of games reveals that the success probabilities of A winning the attack game against Π differ significantly more than a negligible amount from A' solving the problem P. In such cases, the assumption that the success probability of solving problem P is negligible does not guarantee that the success probability of the attack game against Π is also negligible.

2.6 Random oracle model

Designing a game-hopping sequence and providing proofs for each step is often challenging or even impossible without certain assumptions. We now introduce the Random Oracle Model (ROM), where we assume the existence of idealized black-box oracles. This assumption enables the use of mathematical tools that make reductions significantly easier and allows us to prove a much wider range of cryptographic schemes to be secure.

A *random oracle* takes an input and returns a random output. However, the random oracle is consistent, meaning that for the same input, it will return the same output. The oracle generates values "on-the-fly" as needed.

Initially, the random oracle, denoted as G, is defined with an empty list $\mathcal{L}_G$ of input-output pairs (m_i, r_i). When a query m_i is made, the oracle checks from the list if this query has already been made. If the query has been previously made, it returns the corresponding output value r_i from the list. If not, the oracle uniformly generates a new value r_i and adds the input-output pair to the list. The behavior of random oracle is illustrated in Algorithm 9.

Choosing values on-the-fly is a valuable feature of the random oracle model, known as *programmability*, and it is often utilized in ROM reductions. In such a reduction, there is an algorithm A' that uses another algorithm A as a subroutine. When A makes queries

to the random oracle, A' answers these queries and can randomly generate values for A like in Algorithm 9. All queried values are stored in the list, allowing A' to track which values the adversary A queries. [27]. This property is referred to as *preimage awareness*. [24]

Algorithm 9 Random oracle $G(m)$ [23]:

1: **if**: $\exists r \; s.th. \; (m, r) \in \mathcal{L}_G$
2: **return** r
3: **else** :
4: $r \xleftarrow{\$} \mathcal{R}$
5: $\mathcal{L}_G := \mathcal{L}_G \cup \{(m, r)\}$
6: **return** r

The random oracle behavior seems like a behavior of *hash functions*. A hash function maps an arbitrarily sized input into a fixed-sized output, called a *hash*. [35] Originally, hash functions were used to save memory in data storage. Quite soon their properties were noticed to be suitable for cryptographic purposes.

The hash function, denoted by H, is initialized by allocating N empty entries. The *hash* of a data string x is denoted as $y = H(x)$. Different data strings produce distinct hashes. If two different data strings were to produce the same hash, it is called a *collision*. In cryptography, we only use hash functions that produce collisions with a negligible probability; these are known as *collision resistant* hash functions. Collision resistance also implies *preimage resistance*, which means, that given a hash y, it should be difficult to find an input value x that produces it. Collision and preimage resistance make hash functions suitable real-world approximations for random oracles. From the point of view of a user, their behavior appears to be similar to a random oracle. [8]

When designing a cryptographic system, we use random oracles in theoretical models. In practical implementations, we replace them with collision resistant hash functions and assume that they behave similarly to their idealized versions. Even if the chosen hash function turns out not to be collision resistant and does not behave like a random oracle, we can restore security by replacing the hash function with a more secure one. [27]

2.7 Quantum random oracle

In post-quantum security we want to design classical cryptographic systems that are secure even when an adversary has a quantum computer. In such a case, the adversary can make quantum queries to the random oracle.

We view a *qubit* as a linear combination of the form $|b\rangle = \alpha |0\rangle + \beta |1\rangle$, where $\{|0\rangle, |1\rangle\}$ represents the orthonormal *basis*, and $\alpha, \beta \in \mathbb{C}$ are the *probability amplitudes*

satisfying the relation $|\alpha|^2 + |\beta|^2 = 1$. A *quantum register* consists of multiple qubits and can be expressed as $|\psi\rangle = \sum_{(b_1b_2\cdots b_n)\in\{0,1\}^n} \alpha_{(b_1b_2\cdots b_n)} |b_1b_2\cdots b_n\rangle$, where $\alpha_{(b_1b_2\cdots b_n)} \in \mathbb{C}$, and it holds $\sum_{(b_1b_2\cdots b_n)\in\{0,1\}^n} |\alpha_{(b_1b_2\cdots b_n)}|^2 = 1$. The orthonormal basis of the register is $\{|b_1b_2\cdots b_n\rangle\}_{(b_1b_2\cdots b_n)\in\{0,1\}^n}$. When we measure the register with respect to this orthonormal basis, the outcome will be the state $|b_1b_2\cdots b_n\rangle$ with a probability of $|\alpha_{(b_1b_2\cdots b_n)}|^2$. We can also express the probability of obtaining a specific basis state of the register, denoted as $|x\rangle$, using a projection operator Q_x. This operator acts on the register as $Q_x|\psi\rangle = \alpha_x|x\rangle$. Since $|x\rangle$ is an orthonormal basis vector, it has a norm of 1. Then the probability of observing the state $|x\rangle$ is given by the squared norm $||Q_x|\psi\rangle||^2$.

We have a register state $|x\rangle$, and a target state $|y\rangle$, where $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$. Honest parties have only classical access to a random oracle H, but now an adversary has quantum access to it. The *quantum random oracle* is a unitary transformation O_H such that

$$O_H|x\rangle|y\rangle \to |x\rangle|y \oplus H(x)\rangle$$

Let $|\phi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x|x,y\rangle$ be a quantum state of the adversary. The adversary can query $|\phi\rangle$ from the random oracle and receive $\sum_{x\in\{0,1\}^n} \alpha_x|x, y\oplus H(x)\rangle$, meaning that the adversary can query all the values of x at once! In this Quantum Random Oracle Model (QROM), where we assume the existence of quantum random oracles, many classical proof techniques, such as programmability and preimage awareness are no longer suitable. Therefore, we need new methods to prove security. [10]

2.7.1 One-Way to hiding in QROM

To illustrate the difficulty of working in the Quantum Random Oracle Model (QROM), we study the *one-way to hiding* from [44] by Dominique Unruh. We derive the *one-way to hiding lemma*, which has a significant impact on post-quantum reductions. We omit many details of the proof, but we aim to showcase how quantum mechanics complicates proof techniques. The full proof can be found in [44].

Algorithm 10 The game G_0^A

1: $x \xleftarrow{\$} \{0,1\}^n$
2: $b' \leftarrow A^{H(\cdot)}(x, H(x))$
3: **return** b'

Let $H : \{0,1\}^n \to \{0,1\}^m$ be a random oracle. Our goal is to show that $H(x)$ is indistinguishable from a uniformly random y when H is viewed as a random oracle. In the game formulation, we define two games, G_0^A in Algorithm 10 and G_1^A in Algorithm 11. In game G_0^A, the adversary A receives as input a random value x and its hash $H(x)$. In

Algorithm 11 The game G_1^A

1: $(x, y) \xleftarrow{\$} \{0,1\}^{n+m}$
2: $b' \leftarrow A^{H(\cdot)}(x, y)$
3: **return** b'

game G_1^A, it receives two random values x and y. The adversary's goal is to distinguish whether it is interacting with game G_0^A or G_1^A, and it outputs a bit to indicate its guess. The adversary has oracle access to the function H in both games. To demonstrate that the hash is indistinguishable from a uniformly random y, we need to show that

$$|\Pr\left[G_0^A = 1\right] - \Pr\left[G_1^A = 1\right]| \tag{2.7}$$

is negligible [44].

In a classical setting, we could construct a specific adversary B, presented in Algorithm 13. The adversary B is given two random inputs x and y, and it outputs a bitstring x'. It simply receives an input and outputs value specified by Algorithm 13. If this output is x, i.e., $x' = x$, we say that B wins a game G^B displayed in Algorithm 12.

Algorithm 12 The game G^B

1: $(x, y) \xleftarrow{\$} \{0,1\}^{n+m}$
2: $x' \leftarrow B^H(x, y)$
3: **return** $[\![x = x']\!]$

Since B is a probabilistic algorithm specified by Algorithm 13, it does not simply return x and automatically win the game G^B. The adversary B uses A as a subroutine, providing it with x and either $H(x)$ or a random y as input. While B runs the adversary A as a subroutine, it knows the values A queries from the random oracle. We have defined B in such a way that it outputs a value x, and wins the game G^B, if A queries x from the oracle H during its execution. Then the adversary B succeeds with a probability is denoted as $\Pr\left[G^B = 1\right]$ [44].

The distinguisher algorithm A is given two inputs: x and either y or $H(x)$. Its goal is to distinguish whether the second input is uniformly random y or a hash $H(x)$. The adversary is allowed to query values from the oracle H and receive the corresponding outputs. If H did not resemble a uniform distribution, A could analyze the distribution formed from the oracle queries and deduce whether the second input is y or $H(x)$. However, since we are modeling H as a random oracle, its outputs are uniformly distributed. Then the only way for A to determine whether the second input is uniformly random or equal to $H(x)$ is by querying the oracle H with the value x and comparing the result with the second input. When the adversary A queries x from H, the adversary B will win the

game G^B. Since A is also a probabilistic algorithm, it does not simply query x from the random oracle H to determine which game it interacts with [44]. Instead, it queries the values randomly. Let $\mathcal{X}$ be the input space that contains all possible values of x. The size of the input space is denoted by $|\mathcal{X}|$. Then, A queries x from the random oracle with a probability of $1/|\mathcal{X}|$ [44].

Thus, we have reduced the indistinguishability of $H(x)$ from a random y, i.e., the hiding of $H(x)$, to the probability of randomly guessing x, i.e., the one-wayness. If the input space is large, the probability of querying x is very low. This holds even when B gets x as an input [44].

Algorithm 13 Classical algorithm $B(x)$

$y_0 \xleftarrow{\$} \mathcal{Y}$
$y_1 \leftarrow H(x)$
$b \xleftarrow{\$} \{0,1\}$
$b' \leftarrow A^H(x, y_b)$
if: $\exists(x, y_1) \in \mathcal{L}_H$
return x
else:
$x' \xleftarrow{\$} \mathcal{X}$
return x'

But in the quantum setting things are not as straightforward. Let $|\Phi(x,y)\rangle$ denote the initial state of the adversary A in the game G_0^A. This state depends on the input values x and y, and can also be expressed as $|x,y\rangle$. The quantum random oracle operates on the state as $O_H|x,y\rangle = |x, y \oplus H(x)\rangle$. The adversary A makes a sequence of such queries, and the state after i queries is $(O_H)^i|\Phi(x,y)\rangle$. After q queries, we obtain the final state of A, denoted $(O_H)^q|\Phi(x,y)\rangle$. Similarly, in the game G_1^A, let the initial state of A be $|\Psi(x,y)\rangle$. Then the state after i queries is $(O_H)^i|\Psi(x,y)\rangle$, and the final state after q queries is $(O_H)^q|\Psi(x,y)\rangle$.

In a classical setting, the output bit of A indicates its guess on whether it is interacting with game G_0^A or G_1^A. Here, we measure A's final state and the result similarly indicates A's guess. The proof in [44] does not specify the quantum measurement used for this purpose, as Unruh intends the proof to hold for any measurement. We will denote the generic quantum measurement as $Measure(\cdot)$.

The random oracle H is randomly selected from the set of functions $\{0,1\}^n \to \{0,1\}^m$, while the value x is chosen from the input space $\mathcal{X}$ and y from the output space $\mathcal{Y}$. Let α be the probability for the selected values H, x, and y. Given these values, we denote the probability of measuring 1 from the final state in the game G_0^A as

$\alpha \Pr[Measure((O_H)^q|\Phi(x,y)\rangle) = 1]$. A similar probability can be calculated for the game G_1^A. We sum over all possible values of H, x and y, and obtain the total probabilities as:

$$\Pr\Big[G_0^A = 1\Big] = \sum_{H,x,y} \alpha \Pr[Measure((O_H)^q|\Phi(x,y)\rangle) = 1]$$

$$\Pr\Big[G_1^A = 1\Big] = \sum_{H,x,y} \alpha \Pr[Measure((O_H)^q|\Psi(x,y)\rangle) = 1]$$

In the quantum mechanical setting, it is not straightforward to define what it means for A to query x from the oracle H. The adversary A can query a superposition $\sum_{x\in\mathcal{X}} |x,y\rangle$ from H and receive a quantum state $\sum_{x\in\mathcal{X}} |x, y \oplus H(x)\rangle$. However, A cannot gain information about the output without performing a measurement. So, can we say that A has queried x from the oracle in this scenario? Furthermore, each quantum random oracle query alters the state, preventing us from gaining information about the past queries. We would need to either copy or measure the state during the execution. But any measurement would alter the state and disturb the ongoing execution of A.

To address these challenges, we define a quantum version of the adversary B in Algorithm 14. This version runs A but stops its execution at a random query to perform a measurement on the state. The measurement yields an output x for B with the same probability that A would obtain it if A attempted to gain information about x. If the measurement returns x, then B succeeds in the game G^B. After each round, A's execution is restarted to ensure that the measurement does not interfere with A's subsequent execution.

The adversary B picks the random value $i \xleftarrow{\$} \{0,\ldots,q\}$ with a probability $1/q$. It stops the execution of A just before the ith query and obtains A's state, which is either $(O_H)^{i-1}|\Phi(x,y)\rangle$ or $(O_H)^{i-1}|\Psi(x,y)\rangle$ depending on which game A is interacting with. The adversary B then measures the state and yields the value of x with a probability $||Q_x(O_H)^{i-1}|\Phi(x,y)\rangle||^2$ or $||Q_x(O_H)^{i-1}|\Psi(x,y)\rangle||^2$, where Q_x is the projection operator for $|x\rangle$. We sum over all possible values of H, x, y and i to obtain the total probability of B's success denoted as:

$$\Pr\Big[G^B = 1\Big] = \sum_{H,x,y,i} \frac{\alpha}{q} ||Q_x(O_H)^{i-1}|\Psi(x,y)\rangle||^2$$

or

$$\Pr\Big[G^B = 1\Big] = \sum_{H,x,y,i} \frac{\alpha}{q} ||Q_x(O_H)^{i-1}|\Phi(x,y)\rangle||^2$$

depending on which game A is interacting with.

If A makes q queries to the random oracle H, it has been proved by Unruh in [44] that it then holds that

$$\Big|\Pr\Big[G_0^A = 1\Big] - \Pr\Big[G_1^A = 1\Big]\Big| \leq 2q\sqrt{\Pr[G^B = 1]} \tag{2.8}$$

Algorithm 14 Quantum algorithm $B(x, y)$

1: $i \xleftarrow{\$} \{1, 2, \ldots, q\}$
2: Run $A(x, y)$ just before the ith query $|x'\rangle$ to H
3: $x' \leftarrow Measure(|x'\rangle)$
4: **return** x'

which we refer to as the *one-way to hiding lemma*.

Indistinguishability is a crucial feature employed in many reductions. Consequently, the one-way to hiding lemma is widely used in the QROM, and all QROM reductions we encounter in this thesis, such as those in [37][26][15][23], rely on it. Unfortunately, the square root in the lemma introduces non-tightness in the theorems that employ it. This lemma has made many security theorems non-tight and has made it difficult to provide a tight proof in the QROM.

Suppose we have proven the security of a scheme Π based on the hardness of a problem P using the one-way to hiding lemma. This means that the advantage of solving the problem P serves as an upper bound for the advantage of breaking Π. Since the one-way to hiding lemma is used, the advantage of breaking Π is bounded by square root of the advantage of solving the problem P. As P is assumed to be hard, this advantage of solving it is negligible. For a small value, taking a square root increases it. This causes the advantage of breaking the scheme Π to be much greater than the advantage of solving the problem P. The hardness of the problem P does not directly provide security for the scheme Π.

2.7.2 Simulating quantum random oracles

We can either accept a non-tight proof or avoid relying on the one-way to hiding lemma. In practice, often the consequence of the latter is that we do not derive security from IND-CPA or IND-CCA definitions. Instead, we derive security from non-standard security notions, which we will encounter in Chapter 5. However, when using non-standard security notions and avoiding the one-way to hiding lemma, we will face another challenge when working with QROM.

If the scheme includes hash functions, we model them as random oracles in the ROM. In the reduction, we have an adversary A' that uses another adversary A as a subroutine. In QROM, adversaries can exploit quantum mechanics and make quantum queries to the random oracles they have access to, meaning they can query superpositions of inputs. If A' does not have direct access to any quantum random oracle, it must simulate itself responses to the quantum queries made by A. However, if A' has access to a random oracle, it can respond to A's queries by querying the random oracle. There are three

methods for simulating quantum random oracles against an adversary, as outlined in [37]. We omit the details and only briefly mention the three strategies. The important result is that this simulation increases the running time of A' [37].

Let q represent the number of queries made to the quantum random oracles. The first method, proposed in [46], simulates a quantum random oracle using $2q$-wise independent hash functions. This approach increases the running time of A' by $\mathcal{O}(q^2)$.

The second method, introduced in [10], employs a quantum-secure pseudorandom function (PRF) for simulation. In this case, the running time increases by $\mathcal{O}(q \cdot t_{prf})$, where t_{prf} denotes the time required to evaluate PRF.

The third method involves using a real hash function and treating it as a random oracle, similarly to how real hash functions are idealized in ROM. This simulation increases the running time by $\mathcal{O}(q \cdot t_{hash})$, where t_{hash} is the time needed to evaluate the hash function. We do not take a specific stance on which of these three simulation methods should be chosen. Instead, we generalize the running time for simulation as $\mathcal{O}(q) \cdot t_{RO}$, where t_{RO} represents either $\mathcal{O}(q)$, t_{prf} or t_{hash}, depending on the chosen simulation method.

Because of the one-way hiding lemma and the time cost of simulating quantum oracles, it has proven to be challenging to provide tight security in the QROM.

3. Module Learning With Errors

In this chapter, we introduce a lattice problem known as Module Learning With Errors (MLWE), which serves as the foundation for the NIST's Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM). We begin by introducing lattices and discussing the hardness of lattice problems. Next, we examine the Learning With Errors (LWE) problem, which was introduced by Odev Regev. We then extend this problem by adding structure to define the Module version of the LWE problem, i.e., MLWE. We conclude with a discussion on the hardness of the MLWE problem.

3.1 Lattice-based cryptography

Lattice-based cryptography is a very promising area in post-quantum cryptography. These encryption schemes often enjoy strong security proofs and are usually relatively efficient and simple, compared to other post-quantum algorithm families. The use of lattices in cryptography was introduced by Ajtai in 1996 [3], and since then, the field has developed into a broad and active research area. [9]

A *lattice* is a set of points in an n-dimensional space with a regular and repeating structure. Let (column) vectors $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n \in \mathbb{R}^n$ be linearly independent and form the basis of the lattice. The lattice is then defined as the set of vectors generated by this basis using all possible linear combinations with integer coeffcients:

$$\mathcal{L}(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n) = \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i : x_i \in \mathbb{Z} \right\}. \tag{3.1}$$

We can also represent the lattice using a matrix of the basis vectors such that $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n] \in \mathbb{R}^{n \times n}$. The lattice generated by the matrix $\mathbf{B}$ is given by

$$\mathcal{L}(\mathbf{B}) = \{\mathbf{B}x : x \in \mathbb{Z}^n\} \tag{3.2}$$

[9].

There are several presumably hard problems related to lattices. For example, the Shortest Vector Problem (SVP), where we aim to find the shortest nonzero vector in the lattice. Another example is the Shortest Independent Vectors Problem (SIVP), where

we aim to find n linearly independent vectors $\mathbf{s}_1, \mathbf{s}_2 \ldots, \mathbf{s}_n$ that minimize the quantity $\max_i ||\mathbf{s}_i||$.

Typically, one tries to find only an approximation of the solution for these problems, as finding the exact solution would be too time-consuming. We define an approximation factor γ. With this relaxation in the SIVP, we aim to find the value of $\max_i ||\mathbf{s}_i||$ within a factor of γ from the optimal minimized value. This variant is denoted as SIVP_γ. [28]

The best-know algorithm for solving lattice problems is the LLL algorithm, introduced by Lenstra, Lenstra and Lovasz in 1982. This polynomial-time algorithm solves the SVP with an approximation factor of $2^{\mathcal{O}(n)}$, where n is the security parameter. Unfortunately, the achieved approximation factor is large, which limits the applicability of this algorithm. Following its introduction, the LLL algorithm was improved by Schnorr in 1987. The improved algorithm achieves better approximation factor, but at the cost of running time. Since the 1980s, many additional attempts have been made to develop more efficient algorithms for lattice problems. Numerous elegant methods have been proposed, but no significant breakthroughs have been made. All known algorithms still exhibit exponential time complexity for approximation factors $\gamma \leq \text{p}(n)$. This highlights the relative inefficiency of these algorithms compared to algorithms designed to solve other problems. For example, factoring integers can be done with subexponential-time algorithms such as the number field sieve, yet cryptographic schemes based on factoring are still considered secure. Given this context, we can safely conjecture that there is no classical polynomial-time algorithm for solving lattice problems with polynomial approximation factors, and that solving lattice problems using classical computers is computationally hard.

Ever since Shor introduced his algorithm in 1994, researchers have attempted to solve lattice problems using quantum algorithms. However, no one has succeeded thus far. It appears that Shor's algorithm and other known quantum algorithms are not applicable to lattice problems. All existing quantum algorithms for solving lattice problems exhibit similar time complexities to their classical counterparts. This leads us to conjecture that there is no polynomial-time quantum algorithm for solving lattice problems, and that lattice problems remain hard to solve, even with quantum computers.

3.2 Learning With Errors

The Learning With Errors (LWE) problem was introduced by Oded Regev in 2005 [36] as an extension of the learning parity with error problem to higher moduli. In the LWE problem, we are trying to solve linear equations with errors.

Let $\mathbb{Z}_q$ denote a set of integers $\{0, 1, 2, \ldots, q-1\}$, and let $\mathbb{Z}_q^n$ represent the space of vectors of length n whose entries belong to the set $\mathbb{Z}_q$. We choose a vector $\mathbf{s}$. We also choose vectors $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n$ independently and uniformly from $\mathbb{Z}_q^n$, for some integer

$q = q(n) \leq \mathrm{p}(n)$. We form linear equations of the form:

$$\begin{aligned} \mathbf{s} \cdot \mathbf{a}_1 &= c_1 (\mathrm{mod}\ q) \\ \mathbf{s} \cdot \mathbf{a}_2 &= c_2 (\mathrm{mod}\ q) \\ &\vdots \\ \mathbf{s} \cdot \mathbf{a}_m &= c_m (\mathrm{mod}\ q) \end{aligned}$$

Given the vectors $\mathbf{a}_1, \mathbf{a}_2 \ldots, \mathbf{a}_m$ and the results of these linear equations $c_1, c_2, \ldots, c_m$, the vector $\mathbf{s}$ can be solved in polynomial time using Gaussian elimination. However, when we add noise terms $e_1, e_2, \ldots, e_m \in \mathbb{Z}_q$, specified by a probability distribution $\chi : \mathbb{Z}_q \to \mathbb{R}^+$, the equations become computationally hard to solve. The noise distribution χ is typically set to be a discrete Gaussian distribution centered at zero with a width parameter αq. [36] The equations with errors take the form:

$$\begin{aligned} \mathbf{s} \cdot \mathbf{a}_1 + e_1 &= c_1 (\mathrm{mod}\ q) \\ \mathbf{s} \cdot \mathbf{a}_2 + e_2 &= c_2 (\mathrm{mod}\ q) \\ &\vdots \\ \mathbf{s} \cdot \mathbf{a}_m + e_m &= c_m (\mathrm{mod}\ q) \end{aligned}$$

Attempting to solve the vector $\mathbf{s}$ from these noisy equations is computationally hard. In Gaussian elimination, each step involves taking linear combinations of these equations, which causes the noise terms to accumulate, thereby amplifying the error. By the end of the process, the accumulated error becomes so large that solving $\mathbf{s}$ becomes infeasible.[4]

We can write the above linear equations with errors in the matrix form as

$$A\mathbf{s} + \mathbf{e} = \mathbf{c} \tag{3.3}$$

where $\mathbf{e} = (e_1, e_2, \ldots, e_m)^T$ is the noise vector and the vector $\mathbf{c} = (c_1, c_2, \ldots, c_m)^T$ contains the results of the linear equations. The problem where we aim to solve the vector $\mathbf{s}$ from these noisy equations is known today as the *search version of the LWE problem.*

If all the noise terms are chosen according to a probability distribution χ on $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ (which is isomorphic to [0,1)) and number q is a prime, we can establish an equivalence between the search version and the *distinguishing version of the LWE problem.* In the distinguishing version of the LWE, we have m independent samples of the form

$$(\mathbf{a}_i, \frac{1}{q}(\mathbf{a}_i \cdot \mathbf{s}) + e_i)$$

where e_i is chosen from $\mathbb{Z}_q$ according to χ on $\mathbb{T}$. We denote the distribution of these independent samples $(\mathbf{a}_i, \frac{1}{q}(\mathbf{a}_i \cdot \mathbf{s}) + e_i)$ as $A_{\mathbf{s},\chi}$. The goal is to distinguish these samples following the distribution $A_{\mathbf{s},\chi}$ from samples chosen uniformly from $\mathbb{Z}_q$. So, we are trying

to distinguish between two distributions. As noted in Chapter Two, aiming to separate two distributions is a common practice in cryptography, and the distinguishing version is the one typically used in reductions. The distinguishing version of the LWE problem is commonly referred as the LWE problem.

The distinguishing version can also be interpreted using linear algebra. If we have m samples $\mathbf{a}_i$, we can construct a matrix $\mathbf{A}$, where the rows are the samples $\mathbf{a}_i$. Then, the matrix version of distinguishing LWE can be formulated by choosing a matrix $\mathbf{A}$ uniformly from ${\mathbb{Z}_q}^{m \times n}$. Let $\mathbf{s} \in {\mathbb{Z}_q}^n$ be a secret vector. We construct equations of the form $\mathbf{c} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$ and the same number of elements are chosen uniformly from $\mathbb{Z}_q$. We try to distinguish these elements from each other. [28]

The reduction from the search version to the distinguishing version of LWE can be seen rather easily. If we solve the search version, we obtain the secret vector $\mathbf{s}$ and can use it to compute errors $e_i = c_i - \mathbf{a}_i \cdot \mathbf{s}$. Then, we can distinguish samples with small errors from uniform samples without error, thereby solving the distinguishing version of the LWE. However, the reduction from the distinguishing LWE to the search version is not as trivial. The equivalence was proven by Regev in [36].

Let q be a prime. We choose an integer l uniformly from $\mathbb{Z}_q$. We begin by solving the first element of the secret vector, denoted as s_0. We consider a sample $(\mathbf{a}_i, c_i)$ and transform it into the form $(\mathbf{a}_i + (l, 0, \ldots, 0), c_i + lk)$ for some $k \in \mathbb{Z}_q$. If the sample had been chosen uniformly, the transformed sample would be uniform as well. If the sample is of the form $(\mathbf{a}_i, \mathbf{a}_i \cdot \mathbf{s} + e_i)$, the transformation returns $(\mathbf{a}_i + (l, 0, \ldots, 0), \mathbf{a}_i \cdot \mathbf{s} + lk + e_i)$. If $k \neq s_0$ and as q is a prime, then this distribution is uniform as l masks the error. If $k = s_0$, the transformed sample is of the form $(\mathbf{a}_i + (l, 0, \ldots, 0), (\mathbf{a}_i + (l, 0, \ldots, 0)) \cdot \mathbf{s} + e_i)$ and follows the distribution $A_{\mathbf{s}+l,\chi}$. Since we can solve the distinguishing LWE problem, we can separate this distribution from the uniform one, and we can determine whether k is equal to s_0 or not. There are $q \leq \text{poly}(n)$ possible values for the integer k, and we can test them all to find the correct value for k. We repeat this procedure to find the remaining elements of $\mathbf{s}$. [4] [36] Now we can solve the search version based on the fact that we can solve the distinguishing version of LWE.

The LWE problem has been studied extensively since its introduction. There are three main strategies for solving LWE. One strategy is to solve the search version using an exhaustive search. The goal is to directly find a suitable $\mathbf{s}$ such that the quantity $||\mathbf{A} \cdot \mathbf{s} - \mathbf{c}||$ is minimized. The second strategy is to solve the distinguishing version by implementing algorithms designed to solve another lattice problem: the Short Integer Solution problem [28][4]. We use one such algorithm to find a short vector $\mathbf{v}$ such that $\mathbf{v} \cdot \mathbf{A} = \mathbf{0}$. If the studied m samples are of the form $\mathbf{c} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$, then $\mathbf{v} \cdot \mathbf{c} = \mathbf{v} \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}) = \mathbf{v} \cdot \mathbf{e}$ which follows Gaussian noise distribution χ^m. If the samples $\mathbf{c}$ are uniformly distributed, then the quantity $\mathbf{v} \cdot \mathbf{c}$ is uniformly distributed as well. Depending on the noise distribution,

we may be able to distinguish it from a uniform distribution, allowing us to solve the distinguishing version of the LWE problem. In the third strategy, we view the search LWE problem as a Bounded Distance Decoding (BDD)[4] problem in a lattice. If the noise distribution is Gaussian, most of the noise is within three times the standard deviation from the center. We interpret the point $\mathbf{As}$ as the lattice point and the sample $\mathbf{c}$ as a point close to the lattice. The distance of $\mathbf{c}$ from the lattice point $\mathbf{As}$ is then bounded by three standard deviations. We give the sample point $\mathbf{c}$ to the BDD algorithm, which attempts to find the closest lattice point within this bound. The result is the point $\mathbf{As}$, from which we can obtain the secret $\mathbf{s}$. [4]

There are several algorithms that exploit these three strategies, but all of them have exponential running times. This leads us to conjecture that the LWE problem is hard for classical computers. However, we would like to note that the LWE problem is relatively new, and new algorithms may be discovered with more time and research.

The LWE problem is also assumed to be hard for quantum computers. In the original article [36], Regev provided a quantum reduction from the SIVP_γ problem to LWE. It is conjectured that for an approximation factor $\gamma \leq \mathcal{O}(1)$, SIVP_γ is computationally hard to solve. If this conjecture holds, then LWE can also be considered hard.

3.3 MLWE

Unfortunately, cryptographic encryption schemes based on the LWE problem are inefficient due to the size of the public key [28]. The efficiency can be improved by adding more structure to linear equations. We now introduce the module version of LWE, where we sample from a different algebraic structure than $\mathbb{Z}_q$.

3.3.1 Modules

In this subsection, we will introduce the concept of modules generally, and then focus on one specific type of a module: module over the ring of integers of a cyclotomic number field. We will provide all necessary background, ensuring that readers with no previous familiarity with these concepts can follow the discussion.

We denote the set of all single-variable polynomials with rational number coefficients as $\mathbb{Q}[X]$. The set of single-variable polynomials with integer coefficients is denoted as $\mathbb{Z}[X]$. Let $g(X)$ be a polynomial in $\mathbb{Q}[X]$. A root of this polynomial is called an *algebraic number*, and we denote one such number as ξ.

A *monic polynomial* is a polynomial in $\mathbb{Q}[X]$ whose leading coefficient is 1, i.e., a polynomial of the form $x^n + c_{n-1}x^{n-1} + \cdots + c_1 x + c_0$ with $n > 1$ and $c_i \in \mathbb{Q}$. If the polynomial $g(X)$ that defines an algebraic number ξ is monic, irreducible in $\mathbb{Q}[X]$, and

there is no other monic polynomial of lower degree with the same root ξ, we say that $g(X)$ is the *minimal polynomial* of ξ. If the coefficients of the minimal polynomial of ξ are all integers, then $g(X) \in \mathbb{Z}[X]$, and ξ is classified as an *algebraic integer*. [28] [32]

Let us examine whether the complex number i is an algebraic integer. This complex number i is a root of the monic polynomial $X^2 + 1$. Furthermore, there is no monic polynomial of lower degree that has the root i. Thus, X^2+1 is the minimal polynomial of i. Polynomial $X^2 + 1$ belongs to the set $\mathbb{Z}[X]$, and it follows that i is indeed an algebraic integer.

Let F and E be fields such that $F \subset E$. We say that E is an extension of F. The degree of this field extension is defined to be the dimension of E as a vector space over F [18]. We now denote the smallest extension field of $\mathbb{Q}$ that contains all elements of $\mathbb{Q}$ with addition of an algebraic number ξ as $\mathbb{Q}(\xi)$. This finite extension of the rational number field is called the *number field*, and from here on we denote $\mathbb{Q}(\xi)$ simply as K.

The algebraic number ξ has a minimal polynomial of degree n over $\mathbb{Q}$. Thus, there cannot exist any polynomial in $\mathbb{Q}[X]$ of smaller degree than n for which ξ is a root, which implies that $a_{n-1}\xi^{n-1} + a_{n-2}\xi^{n-2} + \cdots + a_1\xi + a_0 = 0$ holds only when all coefficients a_i are zero. Consequently, the roots of unity $\{1, \xi, \xi^2, \ldots, \xi^{n-1}\}$ are linearly independent and form the basis for K as a vector space over $\mathbb{Q}$. Therefore, the degree of K is n. [28]

We continue with our example and examine the complex number i. We aim to generate an extension field of $\mathbb{Q}$, i.e., the number field of i, denoted as $K = \mathbb{Q}(i)$. For a set to be a field, it must be closed under addition, multiplication and division (excluding the division by zero). Simply adding the number i to $\mathbb{Q}$ would not result in a field, as these operations would not be closed within this set. However, the set $\{a + bi : a, b \in \mathbb{Q}\}$ is a field as it is closed under addition, multiplication and division. It also contains the element i. Therefore, this set is the number field $\mathbb{Q}(i)$, the smallest field containing both $\mathbb{Q}$ and i. The field $\mathbb{Q}(i)$ forms a two-dimensional vector space over the rational field $\mathbb{Q}$, with 1 and i as the basis elements. Thus, the degree of this number field is two.

A *ring* is an algebraic structure consisting of a set equipped with two operations: addition and multiplication. These operations satisfy the ring axioms, which state that the ring forms an abelian group under addition, a monoid under multiplication, and that multiplication is distributive with respect to addition. The set of all algebraic integers belonging to the number field K satisfy the ring axioms, and is called the *ring of integers of* K. We denote this ring as R. [32] Now that we understand what rings of integers are, we will focus on a specific instance. However, to proceed, we first need to explore a few additional number-theoretic concepts.

The *νth root of unity* is a root of $x^\nu - 1$. The *νth primitive root of unity* ω_ν is a root of the polynomial $x^\nu - 1$, but not a root of any other polynomial $x^d - 1$ with $d < \nu$. As the roots of unity satisfy the equation $x^\nu - 1$, it holds that $(\omega_\nu)^\nu = 1$. These definitions

can be applied over any field or ring. When applied to the field of complex numbers, the roots of unity take the form $\omega_\nu = e^{2\pi i/\nu} \in \mathbb{C}$. The 2νth root of unity is periodic, satisfying the relation $\omega_{2\nu}^{\alpha+2\nu} = \omega_{2\nu}^{\alpha}$, and symmetric, i.e., $\omega_{2\nu}^{\alpha+\nu} = -\omega_{2\nu}^{\alpha}$ for a non-negative integer α [38].

Let the set $\mathbb{Z}_\nu^*$ denote all coprimes of ν that are smaller than ν itself. The *νth cyclotomic polynomial* is a polynomial of the form

$$\Phi_\nu(X) = \prod_{j \in \mathbb{Z}_\nu^*} \left(X - \omega_\nu^j\right) \in \mathbb{Z}[X]$$

All numbers of the form ω_ν^k, where $k \in \mathbb{Z}_\nu^*$, are roots of this polynomial. Thus, the degree of this cyclotomic polynomial is given by the number of elements in the set $\mathbb{Z}_\nu^*$, i.e., $n = |\mathbb{Z}_\nu^*|$. [32]

With these concepts, we can define the *ring of integers of the cyclotomic number field*. If an algebraic integer ξ is a νth primitive root of unity, then the number field $K = \mathbb{Q}(\xi)$ is called a *cyclotomic number field*. The ring that contains algebraic integers belonging to this cyclotomic number field is called the ring of integers of the cyclotomic number field. It can be shown that for the cyclotomic number field, the associated ring of integers is $R = \mathbb{Z}[\xi]$. [30] [32] The ring $\mathbb{Z}[\xi]$ contains all numbers of the form $a_{n-1}\xi^{n-1} + a_{n-2}\xi^{n-2} + \cdots + a_1\xi + a_0$, where $a_i \in \mathbb{Z}$, and the degree n corresponds to that of the cyclotomic polynomial. [25]

The elements in the ring of integers of the cyclotomic field $R = \mathbb{Z}[\xi]$ are numbers that can be expressed as "polynomials" of ξ. By mapping $\xi \to X$, we can represent these elements in the ring of polynomials of the form $a_{n-1}X^{n-1}+a_{n-2}X^{n-2}+\cdots+a_1X+a_0$, where $a_i \in \mathbb{Z}$. [28] For n as a power of two, R is isomorphic to ring consisting of polynomials of degree at most $n-1$, with addition and multiplication modulo $X^n + 1$, i.e,

$$R \cong \mathbb{Z}[X]/(X^n + 1) \tag{3.4}$$

[28]. The isomorphism to $\mathbb{Z}[X]/(X^n + 1)$ is particularly important, because most encryption schemes based on cyclotomic number fields use the ring $\mathbb{Z}[X]/(X^n + 1)$ for sampling. We will return to this ring later.

For example, the number i is a root of $X^4 - 1$, and it is not a root of any other polynomial of the form $X^d - 1$ with $d < 4$. Therefore, i is a 4th primitive root of unity. The coprimes of number 4 are $\mathbb{Z}_w^* = \{1, 3\}$, and the 4th cyclotomic polynomial is

$$\Phi_4(X) = (X - i^1)(X - i^3) = (X - i)(X + i) = X^2 + 1$$

which has a degree of 2. Since i is both an algebraic integer and a 4th primitive root of unity, the number field $\mathbb{Q}(i)$ is a cyclotomic number field. The associated ring of integers is the set of $R = \mathbb{Z}[i] = \{a+bi : a, b \in \mathbb{Z}\}$. Then R is isomorphic to the ring of polynomials $\mathbb{Z}[X]/(X^2 + 1)$., and the isomorphism is by mapping $a + bi$ to $a + bX$.

A *module* M is an algebraic structure that generalizes the notion of a vector space. Recall that a vector space V is an abelian group consisting of *vectors* v. These vectors can be scaled by *scalars* f, which are elements of a field F. The multiplication between scalars is defined by the calculation rules of fields, and a different set of rules produces a multiplication between a scalar and a vector, producing a *scaled vector* of the form $f \cdot v$. We use the same symbol, $\cdot$, to represent both the multiplication between a scalar and a vector and the multiplication between two vectors. The operations in the vector space satisfy the distributive properties $f \cdot (v_1 + v_2) = f \cdot v_1 + f \cdot v_2$ and $(f_1 + f_2) \cdot v = f_1 \cdot v + f_2 \cdot v$, as well as the associative property defined as $(f_1 \cdot f_2) \cdot v = f_1 \cdot (f_2 \cdot v)$.

Similarly to a vector space, an *R-module*, denoted M, is an abelian group consisting of elements m, which can be scaled by scalars r belonging to a ring R. Then, the scaled element is of the form $r \cdot m$. The distributive and associative properties are defined similarly to those of a vector space, but in this context, the scalars are elements of a ring and the vectors are elements of the module. [32]

We use this general definition of a module to define a *module over the ring of integers of the cyclotomic number field*. From now on, K denotes the cyclotomic number field and R denotes the ring of integers of the cyclotomic field. Let K^d denote the set of elements that are tuples of the form $(x_0, x_1, \ldots, x_d)$ with $x_i \in K$. If a subset $M \subseteq K^d$ is closed under addition and multiplication by elements of R, then this subset M is a R-module, called a module over the ring of integers of the cyclotomic number field. We will use this specific module for sampling in LWE, when we introduce a module version of LWE.

We define *canonical embeddings*

$$\sigma_j(\xi) = \xi^j \text{ for any } j \in \mathbb{Z}_\nu^* \tag{3.5}$$

which maps the root of unity to its higher powers. For general element $x \in K$, which is represented in the basis $(1, \xi, \xi^2, \ldots, \xi^{n-1})$ as $x = x_0 + x_1\xi + x_2\xi^2 + \cdots + x_{n-1}\xi^{n-1}$, the canonical embedding is defined by

$$\sigma_j(x) = x_0 + x_1\xi^j + x_2\xi^{2j} + \cdots + x_{n-1}\xi^{j(n-1)}$$

Since j can take values of ν's coprimes, there exist $n = |\mathbb{Z}_\nu^*|$ such embeddings. We use the canonical embeddings to define a trace for an element in the number field as

$$\mathrm{Tr}(x) = \sum_j \sigma_j(x). \tag{3.6}$$

The root of unity defining the 4th cyclotomic number field $K = \{a + bi : a, b \in \mathbb{Q}\}$ is i. The canonical embeddings are: $\sigma_1(i) \to i$ and $\sigma_3(i) \to i^3 = -i$. For a generic element in the field $a + bi$, where $a, b \in \mathbb{Q}$, the embeddings are:

$$\sigma_1(a + bi) = a + b \cdot \sigma_1(i) = a + bi$$

and

$$\sigma_3(a + bi) = a + b \cdot \sigma_3(i) = a - bi$$

We observe that these embeddings are complex conjugates to each other. This is an important fact, and its implications will be discussed in the next subsection. The trace for an element in K is given by

$$\mathrm{Tr}(a + bi) = \sigma_1(a + bi) + \sigma_3(a + bi) = a + bi + a - bi = 2a$$

which is a rational number.

An *integral ideal* $I \subseteq R$ of the ring R is a subgroup that is closed under multiplication [28]. This means that for all $r \in R$ and all $x \in I$ it holds that the product $rx \in I$. A *fractional ideal*, also denoted as I, is a subset of K such that $dI \subseteq R$, for some $d \in R$, where dI is an integral ideal. The *inverse* of fractional ideal is defined as a set:

$$I^{-1} = \{x \in K : xI \subseteq R\}$$

and the *dual of fractional ideal* is given by:

$$I^{\vee} = \{x \in K : \mathrm{Tr}(xI) \subseteq \mathbb{Z}\}$$

The dual ideal of the ring, $R^{\vee}$, is often referred to as the *codifferent*.

For example, let us show that the set $I = \{2A + 2Bi : A, B \in \mathbb{Z}\}$ is an integral ideal of the ring of integers of the 4th cyclotomic number field $K = \{a + bi : a, b \in \mathbb{Q}\}$. For all elements in the ring, $r = a + bi \in R$ and for all elements $x = 2A + 2Bi \in I$, where $a, b, A, B \in \mathbb{Z}$, the product

$$rx = (a + bi)(2A + 2Bi) = 2(Aa - Bb) + 2(Ab + Ba)i$$

always belongs to I. Thus, the set is an integral ideal. Furthermore, I is also a fractional ideal. It is a subset of K, and for all integers $d \in \mathbb{Z}$, the set dI is an integral ideal. This can be proven by similar calculation as we did above. Let us also calculate the inverse and dual of I. Let $x = x_1 + x_2 i \in K$, where $x_1, x_2 \in \mathbb{Q}$, denote a general element in K. The inverse of I contains elements of K of the form

$$(x_1 + x_2 i)(2A + 2Bi) = 2(Ax_1 - Bx_2) + 2(Ax_2 + Bx_1)i$$

that belongs to the ring R. This holds for all values of A and B when x_1 and x_2 are multiples of $1/2$. Hence, the inverse of I is the set $I^{-1} = \{x_1 + x_2 i : x_1 = \frac{x_1'}{2}, x_2 = \frac{x_2'}{2}$,where $x_1', x_2' \in \mathbb{Z}\}$. The trace of xI is

$$\begin{aligned}\mathrm{Tr}(xI) &= \{\mathrm{Tr}(2(Ax_1 - Bx_2) + 2(Ax_2 + Bx_1)i) : A, B \in \mathbb{Z}, x_1, x_2 \in \mathbb{Q}\} \\ &= \{4Ax_1 - 4Bx_2 : A, B \in \mathbb{Z}, x_1, x_2 \in \mathbb{Q}\} \quad (3.7)\end{aligned}$$

The dual of I contains all elements from this group that are integers. The term $4Ax_1$ is an integer for all values of A when x_1 are multiples of $1/4$. Applying a similar deduction for x_2, we find that the dual of I is the set $I^\vee = \{x_1+x_2i : x_1 = \frac{x_1'}{4}, x_2 = \frac{x_2'}{4}$, where $x_1', x_2' \in \mathbb{Z}\}$. As a final note, let us calculate the dual of the ring R, i.e., the codifferent. For $A+Bi \in R$, the trace of xR is

$$\begin{aligned}\mathrm{Tr}(xR) &= \{\mathrm{Tr}((Ax_1 - Bx_2) + (Ax_2 + Bx_1)i) : A, B \in \mathbb{Z}, x_1, x_2 \in \mathbb{Q}\} \\ &= \{2Ax_1 - 2Bx_2 : A, B \in \mathbb{Z}, x_1, x_2 \in \mathbb{Q}\} \quad (3.8)\end{aligned}$$

Which is a subset of R when x_1 and x_2 are multiples of $1/2$. Therefore, the codifferent is $R^\vee = \{x_1 + x_2i : x_1 = \frac{x_1'}{2}, x_2 = \frac{x_2'}{2}$, where $x_1', x_2' \in \mathbb{Z}\}$.

From the above example, we observe an interesting fact: the elements in the codifferent are multiples of $1/2$ of the elements in the ring R, such that $R^\vee = \frac{1}{2}R$. In other words, the codifferent is a scaled version of R. This observation can be generalized to hold for cyclotomic number fields generated by adjoining a power-of-2 root of unity to the rational number field in the following way. Let $m = k^2$ for some $k \in \mathbb{Z}$, be the degree a power-of-2 cyclotomic field. The elements of the ring of integers of the mth cyclotomic number field, $r \in R$, are related to the elements of the codifferent $r' \in R^\vee$ by relation $r = m/2 \cdot r'$. This means that all the elements in the codifferent can be easily transformed to elements in the ring of integers. [31]

Some R-modules, but not all, have a basis. However, in our case, where K is a cyclotomic field, there always exists a *pseudo-basis*. There always exist nonzero ideals I_k of R and linearly independent pseudo-basis vectors $(\mathbf{b}_k)_k$ in K^d that generate the module M. The module can be expressed as $\sum_{k=i}^{d} I_k \cdot \mathbf{b}_k$. The cardinality of this pseudo-basis is called the *rank* of the module, denoted by d.

3.3.2 Module lattice

Now that we have introduced modules over the ring of integers of a cyclotomic number field, we turn our attention to mapping these modules to lattices. In this subsection, we describe ring homomorphisms for the cyclotomic number field, which enables us to embed modules into lattices and define the concept of module lattices.

We continue with our example of 4th cyclotomic number field $K = \{a+bi : a, b \in \mathbb{Q}\}$, where the basis of K over $\mathbb{Q}$ is $(1, i)$. We aim to represent elements of the number field as two-dimensional vectors to map these elements to lattices. Let $a + bi \in K$ for some $a, b \in \mathbb{Q}$. A natural first approach would be to take $(1, i)$ as a basis and form a vector $(a, b)^T$. However, this definition causes problems with geometric quantities, such as the norm, which lead to issues in security reductions. [30] While we will omit the details of these problems, it has been established that the approach we describe next provides a

geometrically better alternative.

Instead, we will use canonical embeddings in Equation (3.5). We construct a vector from all these embeddings called the *canonical embedding vector*, $\sigma_C : K \to \mathbb{C}^n$, defined as $\sigma_C(y) = (\sigma_j(y))_{j \in \mathbb{Z}_\nu^*}$.

For example, the canonical embedding vector of the element $a + bi$ in $K = \mathbb{Q}(i)$ is

$$\sigma_C(a+bi) = \begin{bmatrix} \sigma_1(a+bi) \\ \sigma_3(a+bi) \end{bmatrix} = \begin{bmatrix} a+bi \\ a-bi \end{bmatrix} \in \mathbb{C}^2$$

We observe that the embeddings σ_1 and σ_3 are complex conjugates of each other, as $\sigma_3 = \overline{\sigma_1}$. Then, the embedding σ_3 can be represented using the embedding σ_1.

This fact can be generalized. We can represent all embeddings using only the first half. To formalize this, we define the set $[\nu/2] = \{0, 1, \ldots, \nu/2 - 1\}$ for even ν. Using this, we can separate the first half of ν's coprimes $\mathbb{Z}_\nu^*$ into a set $\mathbb{J} = [\nu/2] \cap \mathbb{Z}_\nu^*$. Then, the relation

$$\sigma_{\nu-j}(\xi) = \overline{\sigma_j(\xi)} \tag{3.9}$$

holds for all $j \in \mathbb{J}$.

The canonical embedding vector is in the space $\mathbb{C}^n$. The basis vectors of this space are typically denoted as $\mathbf{e}_1 = (1, 0, \ldots, 0)^T, \mathbf{e}_2 = (0, 1, 0, \ldots, 0)^T, \ldots, \mathbf{e}_n = (0, \ldots, 0, 1)^T$. We will reindex these basis vectors to match the indices of the embeddings, denoting them as $\mathbf{e}_j$ and $\mathbf{e}_{\nu-j}$ for all $j \in \mathbb{J}$. This notation is often counterintuitive. For example, in the case of $K = \mathbb{Q}(i)$, the canonical embedding vector is in a two-dimensional space where the basis is typically denoted as $\mathbf{e}_1 = (1, 0)^T, \mathbf{e}_2 = (0, 1)^T$. However, here we will reindex the basis to match coprimes 1 and 3, so the basis vectors are reindexed to $\mathbf{e}_1 = (1, 0)^T, \mathbf{e}_3 = (0, 1)^T$.

With these reindexed basis vectors combined with the relation in Equation (3.9), the canonical embedding vector for $x \in K$ can be represented as:

$$\sigma_C(x) = \sum_j \sigma_j(x) \cdot \mathbf{e}_j + \sigma_{\nu-j}(x) \cdot \mathbf{e}_{\nu-j} \tag{3.10}$$

for all $j \in \mathbb{J}$.

Continuing with the example of the 4th cyclotomic number field, the coprimes of the number 4 are $\mathbb{Z}_4 = \{1, 3\}$, and the dividing set is $[4/2] = \{0, 1\}$. Therefore, the set of first half of the coprimes is $\mathbb{J} = \{1\}$. Then the canonical embedding vector can be represented in the form:

$$\sigma_C(a+bi) = \sum_{j=1} \sigma_j(a+bi)\mathbf{e}_j + \sigma_{4-j}(a+bi)\mathbf{e}_{4-j} = (a+bi)\mathbf{e}_1 + (a-bi)\mathbf{e}_3$$

We notice that canonical embedding vectors belong to a subspace $H \subseteq \mathbb{C}^n$, defined as:

$$H = \{(x_j)_{j \in \mathbb{Z}_\nu^*} \in \mathbb{C}^n : \text{such that } \forall j x_{\nu-j} = \overline{x_j}\}$$

We will now change the basis of this space to represent the canonical embedding vectors as real-valued vectors. The new basis vectors are defined as $\mathbf{h}_j = \frac{1}{\sqrt{2}}(\mathbf{e}_j + \mathbf{e}_{\nu-j})$ and $\mathbf{h}_{\nu-j} = \frac{i}{\sqrt{2}}(\mathbf{e}_j - \mathbf{e}_{\nu-j})$ for all $j \in \mathbb{J}$. Then, the basis of the canonical embedding vector in Equation (3.10) can changed by applying the transformation:

$$\begin{bmatrix} x_j \\ x_{\nu-j} \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -i & i \end{bmatrix} \begin{bmatrix} \sigma_j(x) \\ \sigma_{\nu-j}(x) \end{bmatrix} \tag{3.11}$$

separately for each $j \in \mathbb{J}$. Using this transformation, we define an embedding $\sigma_H : K \to \mathbb{R}^n$, which maps an element $x \in K$ to a real-valued vector such that

$$\sigma_H(x) = \sum_j (x_j \mathbf{h}_j + x_{\nu-i}\mathbf{h}_{\nu-j}) \in \mathbb{R}^n \tag{3.12}$$

Let us continue with our example of the 4th cyclotomic number field. Previously, we represented the elements in this field as the canonical embedding vector of the form $(a+bi)\mathbf{e}_1 + (a-bi)\mathbf{e}_3$. For a generic element in the field $x = a + bi$, where $a, b \in \mathbb{Q}$, we can change the basis using Equation (3.11) and obtain

$$\begin{bmatrix} x_1 \\ x_3 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -i & i \end{bmatrix} \begin{bmatrix} \sigma_1(a+bi) \\ \sigma_3(a+bi) \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -i & i \end{bmatrix} \begin{bmatrix} a+bi \\ a-bi \end{bmatrix} = \sqrt{2} \begin{bmatrix} a \\ b \end{bmatrix}$$

Then, $\sigma_{\mathbf{H}}(a+bi) = \sqrt{2}a \cdot \mathbf{h}_1 + \sqrt{2}b \cdot \mathbf{h}_3$, where the basis vectors are $\mathbf{h}_1 = \frac{1}{\sqrt{2}}(\mathbf{e}_1 + \mathbf{e}_3)$ and $\mathbf{h}_3 = \frac{i}{\sqrt{2}}(\mathbf{e}_1 - \mathbf{e}_3)$. The resulting vector belongs to the real-valued vector space $\mathbb{R}^2$.

As we have successfully represented an element in the number field as a real-valued vector, we can map modules to lattices. A *module lattice* is a special type of a lattice, where the structure of a module is imposed on the lattice. As the lattice is in the real vector space, we can use the map σ_H for this purpose. Let $M \subseteq K^d$ be an R-module. The elements in the module are tuples of the form $(x_0, x_1, \ldots, x_d)$. Each x_i is mapped separately into the real vector space. We do this with an embedding $\sigma_{\mathbf{H}} = (\sigma_H, \ldots, \sigma_H) : K^d \to \mathbb{R}^N$, with $N = nd$, where n is the degree of the number field K. The resulting lattice is in the N-dimensional vector space. When the entire module M is mapped, we call the set $\sigma_{\mathbf{H}}(M)$ a module lattice. [31] [28]

3.3.3 Module version of the LWE

Using the above concepts and discussions, we can formulate a *ring version of the LWE* problem introduced a year after the LWE problem by Lyubashevsky et al. in [31]. As we noted earlier, the cryptographic schemes based on the LWE problem are inefficient. The authors of [31] aimed to improve the efficiency by adding more structure to the problem. Instead of sampling elements from $\mathbb{Z}_q^n$, we now sample elements from the ring of integers of a cyclotomic number field R.

Let $R_q = R/qR$ denote the quotient ring for some integer q, which is the set of equivalence classes $r + qR$ for $r \in R$. Similarly, for the codifferent, let $R_q^\vee = R^\vee/qR^\vee$. For the example of the 4th cyclotomic number field, the ring is of the form $R = \{a + bi : a, b \in \mathbb{Z}\}$. Let us choose $q = 2$, which forms the ring $2R = \{2a + 2bi : a, b \in \mathbb{Z}\}$. Then, there are four equivalence classes: $0 + 2R, 1 + 2R, i + 2R$ and $1 + i + 2R$, and the quotient ring is $R/2R = \{a + bi : a, b \in \mathbb{Z}_2\}$. The codifferent is the set $R^\vee = \{a + bi : a = \frac{a'}{2}, b = \frac{b'}{2}, \text{ where } a', b' \in \mathbb{Z}\}$. Then, the set $2R^\vee$ is the set $\{a + bi : a, b \in \mathbb{Z}\}$. The equivalence classes are now: $0 + 2R, \frac{1}{2} + 2R, \frac{1}{2}i + 2R, \frac{1}{2} + \frac{1}{2}i + 2R$, and the quotient ring is $R_2^\vee = \{a + bi : a = \frac{a'}{2}, b = \frac{b'}{2}, \text{ where } a', b' \in \mathbb{Z}_2\}$.

When this example is generalized, we find that, for a generic cyclotomic field, the quotient ring is given by $R/qR = \{x_0 + x_1\xi + \cdots + x_{n-1}\xi^{n-1} : x_0, \ldots, x_{n-1} \in \mathbb{Z}_q\}$. By mapping $\xi \to X$, we obtain a homomorphism from this set to

$$R_q \cong \{x_0 + x_1X + \cdots + x_{n-1}X^{n-1} : x_0, \ldots, x_{n-1} \in \mathbb{Z}_q\} \bmod (X^n + 1) \tag{3.13}$$

which contains $n - 1$ degree polynomials with coefficients from $\mathbb{Z}_q$. The multiplication and addition in this ring are performed modulo $X^n + 1$.

In the ring LWE setting, the public value $\mathbf{a}_i$ is sampled from the ring R_q, while the secret $\mathbf{s}$ is chosen from the codifferent $R_q^\vee$. The decision to sample from these rings is not self-evident and could have been set differently. For example, both $\mathbf{a}$ and $\mathbf{s}$ could have been sampled from R_q. However, this choice to sample $\mathbf{a}_i$ from R_q and $\mathbf{s}$ from $R_q^\vee$ emerges naturally in the security reductions in [31] and provides an analogy between the LWE and the short integer solution problem. It also enhances security in the ring LWE setting by allowing the largest amount of error while still being able to recover the secret from the noisy products.

The cryptographic schemes based on the ring LWE are indeed more efficient than those based on LWE. But the ring LWE can be shown to be only as hard as lattice problems on certain special classes of ideal lattices, which limits their security. On the other hand, the LWE problem has been proven to be as hard as certain extremely hard lattice problems, but cryptographic schemes based on it are inefficient. The Module-LWE (MLWE) problem, introduced in [12], serves as a generalization of the LWE and the ring LWE. MLWE provides a middle-ground approach between these two extremes, allowing the construction of sufficiently efficient cryptographic schemes, while also being provably as hard as standard lattice problems for module lattices.

In MLWE, we will use an R-module $M \subseteq R^d$. Let the set $(R_q^\vee)^d$ contain tuples of the form $(x_1, x_2, \ldots, x_d)$, where each entry belongs to $R_q^\vee$. We sample $\mathbf{a}_i$ and $\mathbf{s}$ from $(R_q^\vee)^d$.

We also define the circle group $\mathbb{T}_{R^\vee} = K/R^\vee$, which consists of "small" elements that we will use as errors. For example, for the 4th cyclotomic number field $K = \mathbb{Q}(i) =$

$\{a+bi : a,b \in \mathbb{Q}\}$ and its ring of integers $R^\vee = \{a+bi : a = \frac{a'}{2}, b = \frac{b'}{2}, \text{ where } a', b' \in \mathbb{Z}\}$, the circle group is $\mathbb{T}_{R^\vee} = \{a+bi : a,b \in [0, \frac{1}{2})\}$.

To construct MLWE, we also need distributions for sampling. Let D_{r_j} denote a normalized continuous Gaussian probability distribution with width $r_j \in \mathbb{R}^+$. We have a vector of widths $\mathbf{r} = (r_1, r_2, \ldots, r_n)^T$. A vector sampled using the Gaussian distribution is denoted as $D_{\mathbf{r}} = (D_{r_1}, D_{r_2}, \ldots, D_{r_n})$. Elements in the cyclotomic number field can be represented as real-valued vectors, as described in Equation (3.12). These vectors exhibit internal structure, and we require that components x_j and $x_{\nu-j}$ are sampled with the same width, such that $r_j = r_{\nu-j}$. We use a gamma distribution $\Gamma(2,1)$ with shape parameter 2 and scale parameter 1 for this purpose. When each x_j and $x_{\nu-j}$ for $j \in \mathbb{J}$ are sampled independently from $\Gamma(2,1)$, the widths are $r_j = r_{\nu-j} = \alpha\sqrt{1+\sqrt{nx_j}}$ for some $\alpha > 0$. We denote distribution $\Gamma(2,1)$ over the cyclotomic number field K as ψ and call it an *elliptic Gaussian distribution.* [28]

Implementing an efficient Gaussian sampler while protecting against timing attacks has proven to be challenging. In a timing attack, an adversary attempts to gain information about the scheme by analyzing the running times of operations. To address this issue Alkim et al. [5] proposed replacing the elliptical Gaussian distribution used in sampling with a *centered binomial distribution* in practical applications. The centered binomial distribution resembles the elliptical Gaussian distribution while being computationally efficient to sample. It has been shown that this change does not compromise the security of the scheme. The centered binomial distribution for a positive integer η is defined by sampling η pairs (a_i, b_i) uniformly from $(\{0,1\}^2)^\eta$ and computing $\sum_{i=1}^{\eta} a_i - b_i$. Next, we will present the MLWE problem as it is described in [28], but later the distribution will be replaced with the centered binomial distribution.

Let $q \geq 2$ be a prime, n be the dimension of R and d the rank of the module M. The dimension of the module lattice is then $N = nd$. Let ψ be an elliptical Gaussian probability distribution over the cyclotomic number field K. We choose a vector $\mathbf{s} \in (R_q^\vee)^d$, and a vector $\mathbf{a}$ is sampled uniformly at random from the module $(R_q^\vee)^d$. A perturbation e is sampled according to ψ from K, and we form a sample $(\mathbf{a}_i, \frac{1}{q}(\mathbf{a}_i \cdot \mathbf{s}) + e)$. Given arbitrarily many independent samples of this form, it is difficult to distinguish them from an equal number of samples chosen uniformly from $\mathbb{T}_{R^\vee}$. Trying to distinguish samples of the form $(\mathbf{a}, \frac{1}{q}(\mathbf{a} \cdot \mathbf{s}) + e)$ from samples chosen uniformly from $\mathbb{T}_{R^\vee}$ is know as the *distinguishing version of module-LWE* or simply the MLWE problem.

In the *search version of MLWE*, the perturbation e is sampled according to a Gaussian probability distribution over $\mathbb{T}_{R^\vee}$. Given samples of the form $(\mathbf{a}_i, \frac{1}{q}(\mathbf{a}_i \cdot \mathbf{s}) + e)$, it is computationally hard to recover the vector $\mathbf{s}$ from these samples.

We can interpret the search version of MLWE using linear algebra, though it is not as straightforward as the algebraic interpretation of the standard search LWE. We will

demonstrate this only for rings whose dimensions are power of 2. Then, we can sample from the ring R_q instead of the codifferent $R_q^\vee$. Using the isomorphism in Equation (3.13), we obtain the quotient ring $R_q = \mathbb{Z}_q[X]/(X^n+1)$. Elements in this quotient ring are $n-1$ degree polynomials, where coefficients are elements of $\mathbb{Z}_q$.

Now we sample the elements of the MLWE problem from $R_q^d = (\mathbb{Z}_q[X]/(X^n+1))^d$. We sample a vector $\mathbf{a}_i = [a_{i,1}, a_{i,2}, \dots a_{i,d}]^T$ from R_q^d, where each entry is a polynomial of degree $n-1$. We want to represent these polynomials as matrices. In the ring R_q^d, the multiplication between elements is calculated in modulo (X^n+1). We need to preserve this structure in our matrix interpretation, which can be achieved by representing each polynomial $a_{i,j}$ as a negacyclic matrix of the form $\mathrm{Rot}(a_{i,j})$. For the coefficients of the polynomial of the form $b = \sum_{i=0}^{n-1} b_i x^i$, we define

$$\mathrm{Rot}(b) := \begin{bmatrix} b_0 & -b_{n-1} & \cdots & -b_1 \\ b_1 & b_0 & \cdots & -b_2 \\ \vdots & \vdots & \ddots & \vdots \\ b_{n-1} & b_{n-2} & \cdots & b_0 \end{bmatrix}$$

Then, the sampled vector can be rewritten as $\mathbf{a}_i = [\mathrm{Rot}(a_{i,1}), \mathrm{Rot}(a_{i,2}), \dots \mathrm{Rot}(a_{i,d})]^T$. Given m samples $\mathbf{a}_i$, we can construct a matrix $\mathbf{A}$ as follows

$$\mathbf{A} = \begin{pmatrix} \mathrm{Rot}(a_{1,1}) & \cdots & \mathrm{Rot}(a_{1,d}) \\ \vdots & \ddots & \vdots \\ \mathrm{Rot}(a_{m,1}) & \cdots & \mathrm{Rot}(a_{m,d}) \end{pmatrix} \tag{3.14}$$

Now, the MLWE problem can be viewed as a special case of the LWE problem, where the matrix $\mathbf{A}$ has the above structured form. This structure makes the problem easier to solve compared to LWE, which lacks such structure. Nevertheless, MLWE is still considered hard, as we will discuss in the next section.

Continuing with the example of the 4th cyclotomic number field, this ring has a dimension that is a power of 2. We will sample from the set $(R_q)^d = (\mathbb{Z}_q[X]/(X^2+1))^d$. Let $q = 2$ and $d = 2$. We choose a vector $\mathbf{s} = (1+X, X)^T$, and sample a vector $\mathbf{a}_1 = (1, 1+X) = (a_{1,1}, a_{1,2})$ uniformly from $(R_q)^d$. We sample an error vector $e_1 = 0.05+0.225X$ according to Gaussian distribution from $\mathbb{T}_{R^\vee}$. We then form a sample $(\mathbf{a}_1, \frac{1}{2}(\mathbf{a}_1 \cdot \mathbf{s}) + e_1)$, where

$$(\mathbf{a}_1 \cdot \mathbf{s}) = \begin{bmatrix} 1, 1+X \end{bmatrix} \cdot \begin{bmatrix} 1+X \\ X \end{bmatrix} (\mathrm{mod}\ X^2+1) = 1 + X + X + X^2 (\mathrm{mod}\ X^2+1) = 2X = 0$$

where $X^2 + 1 = 0$ holds because of the modulus X^2+1. Additionally, the coefficient of $2X$ satisfies $2 \mod 2 = 0$. Then,

$$\frac{1}{2}(\mathbf{a}_1 \cdot \mathbf{s}) + e_1 = 0 + 0.05 + 0.225X = 0.05 + 0.225X$$

and the first sample becomes

$$(\mathbf{a}_1, \frac{1}{2}(\mathbf{a}_1 \cdot \mathbf{s}) + e_1) = \Big(\Big[1, 1 + X\Big], 0.05 + 0.225X \Big)$$

We sample another vector $\mathbf{a}_2 = (0, 1) = (a_{2,1}, a_{2,2})$ and $e_2 = 0.0335 + 0.22X$. The number of samples is now $m = 2$. We form a matrix

$$\mathbf{A} = \begin{bmatrix} \text{Rot}(a_{1,1}) & \text{Rot}(a_{1,2}) \\ \text{Rot}(a_{2,1}) & \text{Rot}(a_{2,2}) \end{bmatrix}$$

For the element $a_{1,1} = 1 + 0 \cdot X$, we calculate $\text{Rot}(a_{1,1}) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Calculating similar results for the other elements, we obtain the matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 & -1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Denoting the secret vector as a real-valued vector whose entries are the coefficients, we have $\mathbf{s} = (1, 1, 0, 1)^T$, and similarly for the error vector, $\mathbf{e} = (0.05, 0.225, 0.0335, 0.22)^T$. The MLWE problem can then be interpreted as a LWE problem of the form $(\mathbf{A}, \frac{1}{2}(\mathbf{A} \cdot \mathbf{s}) + \mathbf{e})$ where the rows of $\mathbf{A}$ are the samples of the LWE.

3.4 Hardness of MLWE problem

The Shortest Independent Vector Problem $(\text{SIVP})_\gamma$ is a well studied lattice problem and serves as a primitive for our reduction. In $(\text{SIVP})_\gamma$, given a lattice basis $\mathbf{B} \in \mathbb{Z}^{n \times n}$, we try to find n linearly independent vectors $\mathbf{s}_1, \mathbf{s}_2 \ldots \mathbf{s}_n$, where $\mathbf{s}_i \in \mathcal{L}(\mathbf{B})$, that minimize the quantity $||S|| = \gamma \cdot \max_i ||\mathbf{s}_i||$. [9] Here, the number γ is the approximation factor. Finding the exact minimum of the quantity would be too difficult, so this factor makes the task a little bit easier. The problem is still hard, but the cryptographic scheme based on it is more efficient.

The fastest known classical algorithm for solving SIVP exhibits exponential time complexity, and thus we can conjecture that this problem is computationally hard for classical computers. Since the discovery of Shor's algorithm, there have been attempts to solve lattice problems using quantum computers with no success so far [9]. It is conjectured that no quantum algorithm can solve this problem in polynomial time for an approximation factor $\gamma \leq \text{poly}(n)$, which provides assurance that $(\text{SIVP})_\gamma$ is computationally hard for quantum computers. The number n is the security parameter, which can

be scaled depending on the desired security level. The larger the value of n, the harder the problem.

The security of the MLWE problem relies on a variant of $(\text{SIVP})_\gamma$ over modules, denoted as $(\text{Mod-SIVP})_\gamma$. This variant is less studied than the ordinary SIVP. It may be possible that an adversary could exploit the internal structure of modules to solve $(\text{Mod-SIVP})_\gamma$, but no such algorithm is currently known for module ranks greater than 1. The quantum reduction from $(\text{Mod-SIVP})_\gamma$ to MLWE was established by Langlois and Stehle in [28]. This reduction involves several intermediate steps, each imposing specific restrictions on the parameters of the MLWE problem.

We begin by generalizing $(\text{SIVP})_\gamma$. In the Generalized Independent Vector Problem $(\text{GIVP})_\gamma^\phi$ the goal is to find $n = dim(\mathcal{L}(\mathbf{B}))$ linearly independent vectors $\mathbf{s}_1, \mathbf{s}_2, \dots \mathbf{s}_n \in \mathcal{L}(\mathbf{B})$ such that $\max_i ||\mathbf{s}_i|| \leq \gamma \cdot \phi(\mathcal{L}(\mathbf{B}))$. Here ϕ is some real-valued function of a lattice. By setting the function ϕ to so called *smoothing parameter* for which it holds that $\eta_\epsilon \leq \sqrt{\frac{\ln\left(2n\left(1+\frac{1}{\epsilon}\right)\right)}{\epsilon}}$, and setting $\gamma' = \gamma/\eta_\epsilon$ we obtain a reduction from $(\text{SIVP})_\gamma$ to $(\text{GIVP})_{\gamma'}^{\eta_\epsilon}$ [28]. We further restrict $(\text{GIVP})_{\gamma'}^{\eta_\epsilon}$ to module lattices, denoting this restricted version as $(\text{Mod-GIVP})_{\gamma'}^{\eta_\epsilon}$.

Let n denote a security parameter, $\epsilon(n)$ be a negligible function, and $2 \leq q \leq \text{poly}(n)$. Then, there exists a quantum reduction from solving $(\text{Mod-GIVP})_{\gamma'}^{\eta_\epsilon}$ in polynomial time to solving the MLWE. As long as the problem $(\text{Mod-GIVP})_{\gamma'}^{\eta_\epsilon}$ is considered hard, the MLWE problem can also be considered hard.

We define the advantage for the MLWE problem by considering two games, illustrated in Algorithms 15 and 16. In game G_0, we have samples of the form $\mathbf{A} \cdot \mathbf{s} + \mathbf{e}$. We use the notation $\xleftarrow{\eta} R_q$ to denote the sampling of a polynomial from the ring R_q, whose coefficients are sampled according to the binomial distribution. In game G_1, the samples $\mathbf{b}$ are chosen uniformly at random. The goal of an adversary B is to distinguish between these two games, and this corresponds to the distinguishing version of MLWE. We can use Equation (2.3) and define the advantage of an algorithm B solving the distinguishing MLWE as

$$\text{Adv}_{m,k,\eta}^{\text{MLWE}}(B) = |Pr[G_1 = 1] - Pr[G_0 = 1]| \tag{3.15}$$

[11]. The parameters m and k define the sizes of the matrices and vectors, and η deines the centered binomial distribution used for sampling. We have conjectured that the MLWE problem is computationally hard for both classical and quantum computers; therefore this advantage is negligible.

Algorithm 15 Game G_0

1: $\mathbf{A} \xleftarrow{\$} R_q^{m \times k}$
2: $(\mathbf{s}, \mathbf{e}) \xleftarrow{\eta} R_q^k \times R_q^m$
3: $\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$
4: $b' \leftarrow B(\mathbf{A}, \mathbf{b})$
5: **return** b'

Algorithm 16 Game G_1

1: $\mathbf{A} \xleftarrow{\$} R_q^{m \times k}$
2: $\mathbf{b} \xleftarrow{\$} R_q^m$
3: $b' \leftarrow B(\mathbf{A}, \mathbf{b})$
4: **return** b'

4. Module-Lattice-Based Key-Encapsulation Mechanism

In 2016, the National Institute of Standards and Technology (NIST) initiated a process to standardize quantum-resistant cryptographic signature and key-encapsulation schemes. NIST received numerous proposals and after a rigorous evaluation process, selected four algorithms – three signature algorithms and one key-encapsulation algorithm – to be developed into official standards. The selected key-encapsulation algorithm is known as Kyber, and its security relies on the difficulty of the Module Learning With Errors problem. NIST has standardized a slightly modified version of the Kyber under the name of the *Module-Lattice-based Key-Encapsulation Mechanism standard* (ML-KEM). In this chapter, we introduce ML-KEM algorithm, as specified in [34]. The standard does not contain extensive background information. Therefore, most of our discussion will be drawn from the documentation of Kyber, [11] and [6], as these two articles provide reasoning behind the design decisions.

In the ML-KEM, there are two global parameters, $n = 256$ and $q = 3329$, which specify the cyclotomic ring being used. Additionally, there are five global parameters k, η_1, η_2, d_u and d_v. Parameter k specifies sizes of the keys by defining the dimensions of matrices and vectors used in the protocol. The parameters η_1 and η_2 define the distributions used for sampling. Parameters d_u and d_v determine the size of the encoded and compressed ciphertext. The parameters can be chosen from three different sets, which are presented in Table 4.1. Each set has it own advantages and disadvantages with respect to the security-efficiency trade-off. The ML-KEM-1024 is the most secure, but it is also the least efficient.

Scheme	**k**	η_1	η_2	d_u	d_v
ML-KEM-512	2	3	2	10	4
ML-KEM-768	3	2	2	10	4
ML-KEM-1024	4	2	2	11	5

Table 4.1: Parameter specifications for different ML-KEM schemes

The MLWE problem discussed in the previous chapter applies to the ring of integers of any cyclotomic number field. In the ML-KEM, the dimension of the ring is chosen to be $n = 256$, which is a power of 2. Then the codifferent $R^\vee$ is a scaling of the R and it can be mapped to R. Instead of sampling from the codifferent $R_q^\vee$, we can sample public and secret values from R_q.

Using the isomorphism between R and $\mathbb{Z}[X]/(X^n+1)$, we choose to sample from the ring $R \cong \mathbb{Z}[X]/(X^n+1)$. The corresponding quotient ring is $R_q \cong \mathbb{Z}_q[X]/(X^n+1)$, where the global parameters $n = 256$ and $q = 3329$ are used. The ring $\mathbb{Z}_q[X]/(X^n+1)$ consists of 255-degree polynomials of the form $f = f_0 + f_1X + \cdots + f_{255}X^{255}$, where $f_i \in \mathbb{Z}_q$, with addition and multiplication performed modulo X^n+1. This ring is particularly convenient to work with as arithmetic operations can be performed efficiently using the Number Theoretic Transformation, an optimization technique for polynomial multiplication, which we will introduce later. Then the MLWE problem $(\mathbf{A}, \frac{1}{q}(\mathbf{A}\cdot\mathbf{s})+\mathbf{e})$ is constructed by choosing a vector $\mathbf{s}$ from $(R_q)^d$, sampling a matrix $\mathbf{A}$ uniformly from $(R_q)^{m\times d}$, and sampling the error vector $\mathbf{e}$ from $(R_q)^d$ according to the binomial distribution.

Instead of representing elements of $(R_q)^d$ in the negacyclic form, as in Equation (3.14), we will use pseudocode in algorithms where the coefficients of polynomials of the above form are represented as vectors $f = (f_0, f_1, \ldots, f_{255}) \in \mathbb{Z}_q^{256}$. This representation allows us to sample elements of R_q from $\mathbb{Z}_q^{256}$.

Then, an element $\mathbf{v}$ in R_q^d is a vector with d components. Each component $\mathbf{v}[i]$ is a vector representing a polynomial in R_q. Thus, we can sample vectors from $(\mathbb{Z}_q^{256})^d$. Similarly, each element in the matrix $\mathbf{A}$ from $R_q^{m\times d}$ is a polynomial represented as a vector. Then matrices can be sampled from $(\mathbb{Z}_q^{256})^{m\times d}$. When calculating $\mathbf{A}\cdot\mathbf{v}$, each element-wise operation involves the multiplication between two polynomials modulo (X^n+1). For large polynomials, this operation becomes computationally inefficient. To address this, we introduce the Number Theoretic Transformation (NTT), an optimization method for efficiently performing polynomial multiplication. This technique reduces the computational cost of these operations, enabling efficient calculations. After presenting the NTT, we describe a public-key encryption scheme based on the MLWE problem. Finally, we explain the working principles of the key encapsulation mechanism itself.

4.1 Number Theoretic Transformation

The elements in the ring $R = \mathbb{Z}_q[X]/(X^n+1)$ are polynomials of degree 255. Addition between two polynomials can be performed efficiently element-wise with modulo q. However, multiplying two high-degree polynomials is computationally inefficient. Suppose $G(X)$ and $H(X)$ are polynomials of degree $n-1$ in the ring $\mathbb{Z}_q[X]/(X^n+1)$. The

polynomial multiplication between them is given by

$$Y(X) = G(X) \cdot H(X) \mod (X^n + 1) = \sum_{k=0}^{2(n-1)} y_k X^k \mod (X^n + 1)$$

where $y_k = \sum_{i=0}^{k} g_i h_{k-i} \mod q$. For the ring $\mathbb{Z}_q[X]/(X^n + 1)$, this operation has a time complexity of $\mathcal{O}(n^2)$ [38]

One way to improve efficiency is by using the map σ_H to transform the ring elements into a real space H, where multiplication can be performed relatively efficiently. However, an even faster method for polynomial multiplication is the *Number Theoretic Transformation* (NTT) [30]. Instead of calculating a discrete Fourier transformation over the complex numbers, the NTT performs a Fourier transformation over a finite field [29]. We can use techniques analogous to the fast Fourier transformation, which allow us to compute multiplications with a time complexity of $\mathcal{O}(n \log n)$. [34]

Let N be an integer formed by the product $N = n_1 \cdot n_2 \cdot \cdots \cdot n_k$ of numbers $n_i \geq 1$ that are pairwise coprimes. The *Chinese Remainder theorem* states that for an integer $0 \leq x < N$, the map

$$f : x \mod N \to (x \mod n_1, x \mod n_2, \ldots, x \mod n_k)$$

defines an isomorphism $\mathbb{Z}/N \to \Pi_{j=0}^{k} \mathbb{Z}/(n_j)$. [45] Isomorphisms preserver multiplication, which allows us to compute $a \cdot b \mod N$ using the Chinese remainder theorem. Instead of directly calculating the result of the multiplication, we first compute separately:

$$(a \mod n_1, a \mod n_2, \ldots, a \mod n_k)$$

and

$$(b \mod n_1, b \mod n_2, \ldots, b \mod n_k)$$

Then, we multiply component-wise in the Chinese remainder domain to obtain:

$$(a \cdot b \mod n_1, a \cdot b \mod n_2, \ldots, a \cdot b \mod n_k)$$

Finally, we reconstruct $a \cdot b \mod N$ using the inverse function, and obtain the result. [29]

To apply the above to the ring $\mathbb{Z}_q[X]/(X^{256} + 1)$ we need to factor $X^{256} + 1$ into its coprimes. We can easily perform the first factorization as follows:

$$(X^{256} - (-1)) = (X^{128} - i)(X^{128} + i)$$

However, we cannot continue factoring $(X^{128} + i)$ further. To overcome this limitation, we utilize the roots of unity. We denote the 8th root of unity as ξ_8, which satisfies $\xi_8^8 = 1$. Due to the property of symmetry, $\xi_8^k = -\xi_8^{k+4 \mod 8}$, it follows that $\xi_8^4 = -1$ and $\xi_8^2 = -\xi_8^6$. By incorporating the roots of unity, we can rewrite the factorization as:

$$(X^{256} + 1) = (X^{256} - \xi_8^4) = (X^{128} - \xi_8^2)(X^{128} + \xi_8^2) = (X^{128} - \xi_8^2)(X^{128} - \xi_8^6)$$

Now we can continue factorizing further:

$$(X^{256}+1) = (X^{64}-\xi_8)(X^{64}+\xi_8)(X^{64}-\xi_8^3)(X^{64}+\xi_8^3)$$

Using relations $\xi_8 = -\xi_8^5$ and $\xi_8^3 = -\xi_8^7$, we obtain the final result of

$$(X^{256}+1) = (X^{64}-\xi_8)(X^{64}-\xi_8^3)(X^{64}-\xi_8^5)(X^{64}-\xi_8^7)$$

[41].

Each step of the factorization process reduces the polynomial degree by half while doubling the number of terms. We could continue this process until we obtain 256 factors, each being a polynomial of the degree 1. In order to do so, we should have selected the 512th root of unity ξ_{512} to represent number -1 in the first step, but the ring R_q contains polynomials whose coefficients come from $\mathbb{Z}_q$. Unfortunately, with $q = 3329$ there is no 512th root of unity in the set $\mathbb{Z}_q$. We only have 256th root of unity $\xi_{256} = 17$. This holds because $17^{256} \mod q = 1$ while $17^{128} \mod q = -1$. Thus, we can only continue the factorization up to 128 factors of polynomials of degree 2. The result of this factorization is

$$X^{256}+1 = \prod_{i=0}^{127}\left(X^2 - \zeta_{256}^{2i+1}\right)$$

Using the Chinese remainder theorem, a polynomial $f \in R_q$ can be mapped to the ring

$$\hat{f} = (f \mod (X^2-\zeta_{256}^{2\cdot 0+1}), f \mod (X^2-\zeta_{256}^{2\cdot 1+1}), \ldots, f \mod (X^2-\zeta_{256}^{2\cdot 127+1})) \in T_q \tag{4.1}$$

where $T_q = \bigoplus_{i=0}^{127} \mathbb{Z}_q[X]/(X^2-\xi_{256}^{2i+1})$. The result $\hat{f}$ is known as the Number Theoretic Transformation (NTT) of f. [6]

We still need to consider how we can efficiently calculate the elements in $\hat{f}$. While we could calculate each $f \mod (X^2+\xi_{256}^{2i+1})$ separately, there is a much more efficient approach. Each time we half the power of the modulus, it takes on the form

$$(X^d - \xi^2) = (X^{d/2}-\xi)(X^{d/2}+\xi) \tag{4.2}$$

with

$$f \to (f \mod (X^{d/2}-\xi), f \mod (X^{d/2}+\xi)) \tag{4.3}$$

for some root of unity ξ. The elements in the initial domain are polynomials of degree $d-1$, expressed as $f = \sum_{i=0}^{d-1} a_i X^i$. In the factorized Chinese remainder domain, the polynomials are of degree $d/2-1$ and can be represented as:

$$f \mod (X^{d/2-1}-\xi) = \sum_{i=0}^{d/2-1} b_i X^i \tag{4.4}$$

and

$$f \mod (X^{d/2-1} + \xi) = \sum_{i=0}^{d/2-1} c_i X^i \tag{4.5}$$

Using basic algebra, we can determine that the coefficients of these polynomials are:

$$b_i = a_i + \xi a_{i+d/2} \text{ and } c_i = a_i - \xi a_{i+d/2}$$

[29] Each time we factorize the modulo polynomial $X^d + 1$, the elements in this Chinese remainder domain are polynomials of degree $d/2-1$ with coefficients given by the expression above for b_i and c_i. We repeat this process recursively, further reducing the degree of the modulo, until the desired form is achieved.

Let us consider an example with the modulo $X^4 + 1$. Elements in this domain are polynomials of degree 3, expressed as $f = a_0 + a_1X + a_2X^2 + a_3X^3$. We will use the 8th root of unity to denote the number -1, such that $\xi_8^4 = -1$. The first factorization gives $X^4 + 1 = (x^2 - \xi_8^2)(X^2 + \xi_8^2)$, which maps the polynomial f as follows:

$$f \to (f \mod X^2 - \xi_8^2, f \mod X^2 + \xi_8^2)$$

Here, the elements of $f \mod X^2 - \xi_8^2$ are polynomials of the form $g = b_0 + b_1X$, where $b_0 = a_0 + \xi_8^2 a_2$ and $b_1 = a_1 + \xi_8^2 a_3$. Similarly, the elements of $f \mod X^2 + \xi_8^2$ are polynomials of the form $h = c_0 + c_1X$, where $c_0 = a_0 - \xi_8^2 a_2$ and $c_1 = a_1 - \xi_8^2 a_3$. As a list representation, the two polynomials are $g = [a_0 + \xi_8^2 a_2, a_1 + \xi_8^2 a_3]$ and $h = [a_0 - \xi_8^2 a_2, a_1 - \xi_8^2 a_3]$.

We substitute $\xi_8^2 = -\xi_8^6$ and continue factoring, which gives:

$$X^4 + 1 = (X - \xi_8)(X + \xi_8)(X - \xi_8^3)(X + \xi_8^3)$$

This maps the polynomials g and h to the following domains:

$$g \to (g \mod X - \xi_8, g \mod X + \xi_8)$$

and

$$h \to (h \mod X - \xi_8^3, h \mod X + \xi_8^3)$$

Elements of $g \mod X - \xi_8$ are "degree 0 polynomials", i.e., numbers of the form $d = b_0 + \xi_8 b_1 = (a_0 + \xi_8^2 a_2) + \xi_8(a_1 + \xi_8^2 a_3)$. Elements of $g \mod X + \xi_8$ are numbers of the form $e = b_0 - \xi_8 b_1 = (a_0 + \xi_8^2 a_2) - \xi_8(a_1 + \xi_8^2 a_3)$. We can compute the results for h similarly, yielding numbers: $(a_0 - \xi_8^2 a_2) + \xi_8^3(a_1 - \xi_8^2 a_3)$ and $(a_0 - \xi_8^2 a_2) - \xi_8^3(a_1 - \xi_8^2 a_3)$.

At the end of the factorization, we obtain four numbers:

$$(a_0 + \xi_8^2 a_2) + \xi_8(a_1 + \xi_8^2 a_3)$$

$$(a_0 + \xi_8^2 a_2) - \xi_8(a_1 + \xi_8^2 a_3)$$

$$(a_0 - \xi_8^2 a_2) + \xi_8^3(a_1 - \xi_8^2 a_3)$$

$$(a_0 - \xi_8^2 a_2) - \xi_8^3(a_1 - \xi_8^2 a_3)$$

which correspond to moduli $X-\xi_8, X+\xi_8, X-\xi_8^3$ and $X+\xi_8^3$. We again use the calculation rules for roots of unities, and obtain that corresponding moduli are $X-\xi_8, X-\xi_8^5, X-\xi_8^3$ and $X-\xi_8^7$. We change the order so that the power of the moduli are in an increasing order, and label these numbers as:

$$\hat{a_0} = (a_0 + \xi_8^2 a_2) + \xi_8(a_1 + \xi_8^2 a_3)$$

$$\hat{a_1} = (a_0 - \xi_8^2 a_2) + \xi_8^3(a_1 - \xi_8^2 a_3)$$

$$\hat{a_2} = (a_0 + \xi_8^2 a_2) - \xi_8(a_1 + \xi_8^2 a_3)$$

$$\hat{a_3} = (a_0 - \xi_8^2 a_2) - \xi_8^3(a_1 - \xi_8^2 a_3)$$

These four numbers are the NTT representation of the polynomial f, and we can denote this as $\hat{f} = [\hat{a_0}, \hat{a_1}, \hat{a_2}, \hat{a_3}]$. The recursive calculation of this example is illustrated in Figure 4.1, where we represent polynomials as lists of coefficients.

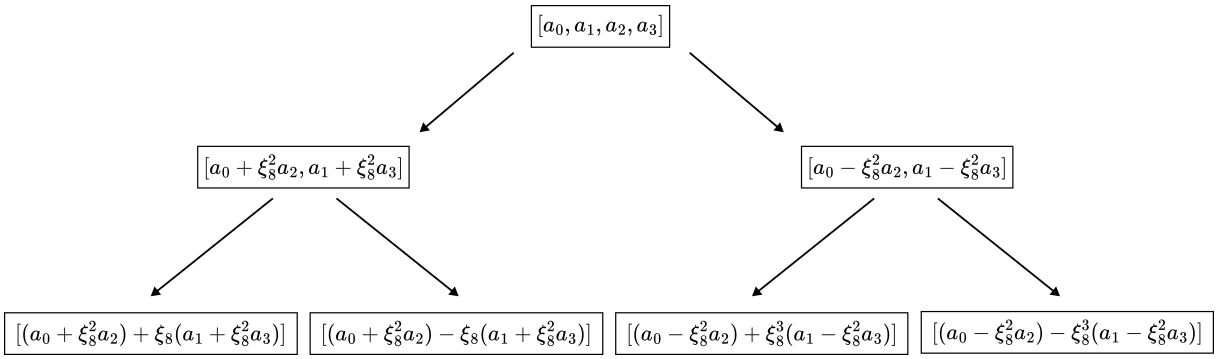

Figure 4.1: Graph illustrating the recursive factorization for $n = 4$

We can observe that the NTT representation exhibits a structured pattern. The terms $(a_0 \pm \xi_8^2 a_2)$ and $(a_1 \pm \xi_8^2 a_3)$ appear multiple times and are summed in a structured manner. Instead of calculating these numbers multiple times for each leaf, we compute them once, and distribute accordingly. This optimization method was independently proposed by Cooley-Tukey and Gentleman-Sande. [38]

We continue with the example of the modulo $X^4 + 1$. At the beginning we have a polynomial $f = a_0 + a_1X + a_2X^2 + a_3X^3$, and we represent it as a list $[a_0, a_1, a_2, a_3]$. We begin by computing the pairs a_0, a_2 and a_1, a_3 together, using a method called butterfly. These operations result in terms $(a_0 \pm \xi_8^2 a_2)$ and $(a_1 \pm \xi_8^2 a_3)$. The butterfly operation is visualized for the pair a_0, a_2 in Figure 4.2 [38]. This circuit representation resembles the shape of a butterfly, hence the name. Once we have calculated the butterflies for these

two pairs, we store the results in a list. Now our list contains $[a_0 + \xi_8^2 a_2, a_1 + \xi_8^2 a_3, a_0 - \xi_8^2 a_2, a_1 - \xi_8^2 a_3]$.

We then take the first two elements from this list and butterfly them together, resulting in $\hat{a_0} = (a_0 + \xi_8^3 a_2) + \xi_8(a_1 + \xi_8^2 a_3)$ and $\hat{a_2} = (a_0 + \xi_8^2 a_2) - \xi_8^3(a_1 + \xi_8^2 a_3)$. We replace the first two elements in the list with these new values. Next, we take the second half of the elements and butterfly them together, which results in $\hat{a_1} = (a_0 - \xi_8^2 a_2) + \xi_8(a_1 - \xi_8^2 a_3)$ and $\hat{a_4} = (a_0 - \xi_8^2 a_2) - \xi_8^3(a_1 - \xi_8^2 a_3)$, and store the result. At the end of this process, we have the list $[\hat{a_0}, \hat{a_2}, \hat{a_1}, \hat{a_4}]$ that represents the NTT of the original polynomial. This process is visualized in Figure 4.3.

However, the elements are now stored in a different order than they were initially: they are in the bit-reversal order. For example, the bit reversal of the number $126 = 111110$ is $011111 = 63$. We will use the function $\mathbf{br}(k)$ to denote the bit reversal of the number k. [16] [38] In the example of $X^4 + 1$, where $4 = 2^2$, we are working with 2-bit integers. The order changes as follows $1 = 01 \to 10 = 2$ and $2 = 10 \to 01 = 1$. Numbers $0 = 00$ and $3 = 11$ do not change the order.

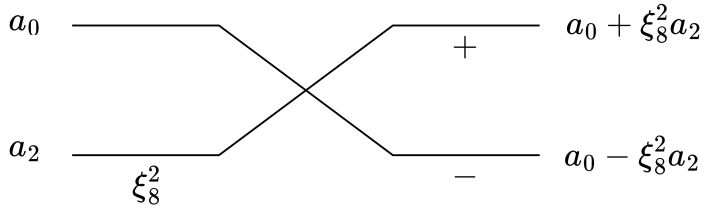

Figure 4.2: Cooley-Tukey butterfly

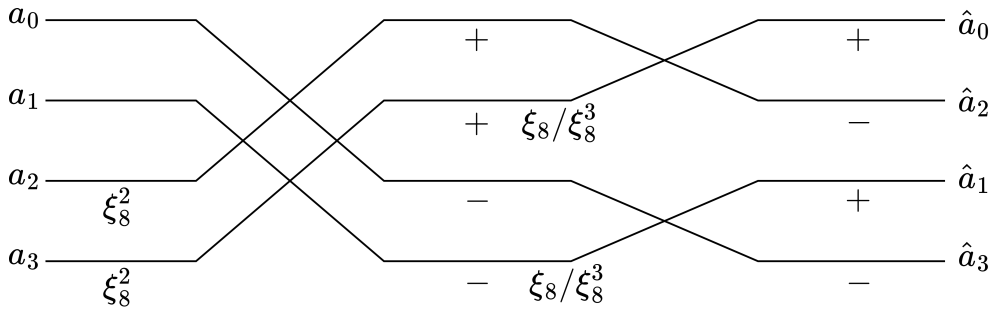

Figure 4.3: Butterflies for $n = 4$

Similarly to the example of a polynomial of the degree 3, we can recursively factorize a ring element polynomial of degree 255 with modulo $X^{256}+1$ until we obtain polynomials of degree 2 as the modulus. The elements in this factorized domain are polynomials of the degree 1 of the form

$$\mathrm{NTT}(f) = \hat{f} = (\hat{f}_0 + \hat{f}_1 X, \hat{f}_3 + \hat{f}_4 X, \ldots, \hat{f}_{244} + \hat{f}_{255} X)$$

where $\hat{f}_{2j} = \sum_{j=0}^{127} f_{2j}\xi^{(2\mathbf{br}(i)+1)j}$ and $\hat{f}_{2i+1} = \sum_{j=0}^{127} f_{2j+1}\xi^{(2\mathbf{br}(i)+1)j}$ [6]. Multiplication in this basis $\hat{f} \cdot \hat{g} = \hat{h}$ can be performed element-wise and involves 128 operations of the form

$$\hat{h}_{2i} + \hat{h}_{2i+1}X = (\hat{f}_{2i} + \hat{f}_{2i+1}X)(\hat{g}_{2i} + \hat{g}_{2i+1}X) \mod (X^2 - \zeta^{2^{\mathrm{br}7(i)}+1})$$

This approach is much faster than directly multiplying two polynomials of degree 255.

The inverse NTT, denoted as NTT^{-1}, is performed using the Gentelman-Sande butterfly, which reverses the operations of the Cooley-Tukey butterfly. For example, applying the Gentelman-Sande butterfly to elements $\hat{a}_0$ and $\hat{a_2}$ yields $(\hat{a_0} + \hat{a_2}) = 2(a_0 + \xi_8^2 a_2)$ and $(\hat{a_0} - \hat{a_2})\xi_8^{-1} = 2(a_1 + \xi_8^2 a_3)$. Similarly, applying the Gentelman-Sande butterfly to $\hat{a_1}$ and $\hat{a_3}$ produces: $(\hat{a_1} + \hat{a_3}) = 2(a_0 - \xi_8^2 a_2)$ and $(\hat{a_1} - \hat{a_3})\xi_8^{-3} = 2(a_1 - \xi_8^2 a_3)$. We have four elements and we store them into the list:

$$[2(a_0 + \xi_8^2 a_2), 2(a_1 + \xi_8^2 a_3), 2(a_0 - \xi_8^2 a_2), 2(a_1 - \xi_8^2 a_3)]$$

Next, we take the first and third elements from this list and apply another Gentelman-Sande butterfly. This yields:

$$(2(a_0 + \xi_8^2 a_2) + 2(a_0 - \xi_8^2 a_2)) = 4a_0$$

and

$$(2(a_0 + \xi_8^2 a_2) - 2(a_0 - \xi_8^2 a_2))\xi_8^{-2} = 4a_2$$

Similarly, applying the Gentleman-Sande butterfly to the second and fourth elements produces $4a_1$ and $4a_3$. Finally, to recover the coefficients of the original polynomial, we divide these results by 4. This process is visualized in Figure (4.4).

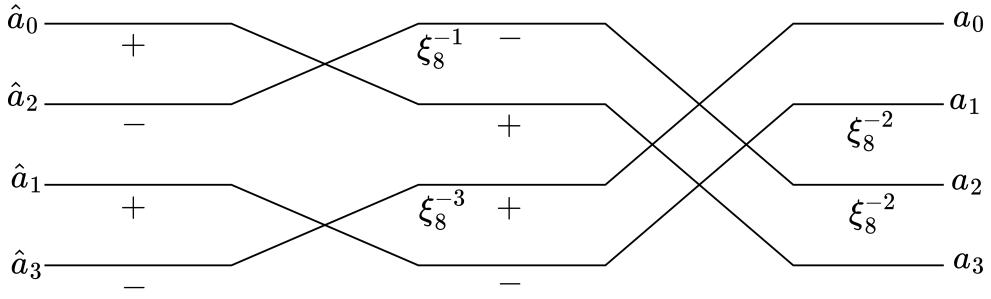

Figure 4.4: Gentelman-Sande butterflies for $n = 4$

In pseudocode, every time we multiply two elements $f, g \in R_q$, we transform them to the NTT domain, multiply them in the NTT basis, and then transform the results back to the original ring $f \cdot g = \mathrm{NTT}^{-1}(\mathrm{NTT}(f) \cdot \mathrm{NTT}(g)) = h$. When we run the NTT for a vector element $\mathbf{v}$ in a module, we need to apply the NTT separately to each element in the vector. We do not explicitly denote this in pseudocode, but the reader should keep it in mind. [34][38][6]

4.2 MLWE based Public-Key Encryption scheme

In this subsection, we will present a simplified version of the MLWE-based PKE scheme used in the ML-KEM. We will omit details about the seeds used as sources of randomness, the algorithms used to sample elements, and any algorithms for converting between data types such as bits to bytes or bits to integers. All these details can be found in the official NIST document [34]. We will only include details necessary to understand the working principle of this scheme.

Let us construct a Public-Key Encryption (PKE) scheme whose security is based on the assumed hardness of the MLWE problem. First, we define the auxiliary function and algorithms used in the PKE.

We define auxiliary algorithms $\text{Compress}_d(x)$ and $\text{Decompress}_d(x)$. For an element $x \in \mathbb{Z}_q$, the algorithms are defined as:

$$\text{Compress}_d(x) = \lceil (2^d/q) \cdot x \rfloor \mod 2^d$$

and

$$\text{Decompress}_d(x) = \lceil (q/2^d) \cdot x \rfloor$$

where the global parameter $q = 3329$ is used. We also use the notation $\lceil \cdot \rfloor$ to denote the rounding to the nearest integer.

All lattice-based schemes involve some degree of error [9]. We will now define an auxiliary function f that is used in error correction. The function rounds the result in such a way that the generated error diminishes. The function $f : \mathbb{Z}_q \to \mathbb{Z}_2$ is defined as $f(x) = \text{Decompress}_1(x)$. We denote the inverse function as $f^{-1}(x) = \text{Compress}_1(x)$. The correctness of this function is set to satisfy

$$f^{-1}(f(x)) - x \mod q \leq q/4 \tag{4.6}$$

for all $x \in \mathbb{Z}_q$.

We also define auxiliary algorithms $\text{Encode}_d(x)$ and $\text{Decode}_d(x)$. While we omit the details of these algorithms, they handle conversions between integers and byte arrays. The algorithm $\text{Encode}_d(x)$ converts a d-bit integer x into a byte array of length $\mathbb{B}^{32d}$. The algorithm $\text{Decode}_d(x)$ performs the reverse conversion. These algorithms determine the size of the ciphertext.

We use a centered binomial distribution $\mathcal{D}$ to sample polynomials with small coefficients from R_q, which are errors. This distribution, denoted by $\mathcal{D}_\eta$, is parameterized by $\eta \in \{2, 3\}$. Let B be a vector of four or six elements (for $\eta = 2$ or $\eta = 3$, respectively) such that

$$B = (a_1, \ldots, a_\eta, b_1, \ldots, b_\eta)$$

The centered binomial distribution of B is defined as:

$$\mathcal{D}_\eta(B) := \sum_{i=1}^{\eta}(a_i - b_i)(\text{mod } q)$$

This distribution is employed in the auxiliary algorithm CBD_η, which takes as input a uniformly random vector. The algorithm processes the elements of the vector in sets of 4 or 6 elements (depending on whether $\eta = 2$ or $\eta = 3$). For each set, it computes a centered binomial distribution value according to $\mathcal{D}_\eta$ and stores this centered value. The output of CBD_η is a list of values that follow the centered binomial distribution. [6] We sample a polynomial uniformly at random from the ring R_q using a seed r. This polynomial is represented as a vector and is provided as an input to the algorithm CBD_η. The output is a vector representation of the polynomial whose coefficients are "small" and follow a binomial distribution. This polynomial can be interpreted as an error.

In pseudocode, when an element a is sampled from the ring R_q according to CBD_η, we denote this as: $a \xleftarrow{\text{CBD}_\eta} R_q$. If we want to sample a vector $\mathbf{a}$, all the elements of the vector are sampled independently. For simplicity, sometimes we omit the seed used for sampling the uniformly random polynomial. However, if we explicitly want to indicate the seed, we use the notation $a \xleftarrow{\text{CBD}_\eta(r)} R_q$.

Now that the auxiliary function and algorithms have been introduced, we present a public-key encryption scheme based on the MLWE problem in Algorithm 17. The PKE.KeyGen algorithm generates two keys: a public key pk and a secret key sk. The secret key is kept private, while the public key is published. A plaintext m and a random seed r are chosen. This plaintext will later serve as a symmetric key in KEM. The PKE.Enc algorithm takes the public key, the plaintext and the seed as inputs and outputs a ciphertext (c_1, c_2). The ciphertext can then be decrypted using the PKE.Dec algorithm with the secret key sk. This decryption algorithm returns the plaintext m the user.

In the last step of the decryption, the function rounds the result such that $f^{-1}(\mathbf{e}^T\mathbf{y} + e_2 + \mathbf{s}^T\mathbf{e_1} + f(m)) \approx f^{-1}(f(m)) \approx m$. This approximation is crucial; if it rounds incorrectly, the decryption will fail. The function f has been designed to satisfy the correctness condition given by Equation (4.6). For the approximation to be accurate, we then require $|\mathbf{e}^T\mathbf{y} + e_2 + \mathbf{s}^T\mathbf{e_1}| < q/4$.

Let δ denote the probability of a decryption error, for which it holds $\delta = \Pr\Big[|e^T r + e_2 + s^T e_1| > q/4\Big]$. Then, the PKE is $(1-\delta)$-correct. The parameters in Table 4.1 have been selected such that the error probability δ is negligible: $\delta \leq 2^{-128}$. [6][11]

4.3 MLWE Key-Encapsulation Mechanism

Fujisaki and Okamoto [19][20] introduced a method to transform a weakly secure PKE scheme into a strongly secure KEM in the random oracle model. Unfortunately, the

Algorithm 17 NIST MLWE-based (PKE)

Integers $q = 3329$ and $n = 256$. Parameters k, η_1, η_2, d_u and d_v. Random number r specified by a user.

Key generation algorithm PKE.KeyGen(1^n):

Secret key: $sk = \mathbf{s} \xleftarrow{\text{CBD}_{\eta_1}} (\mathbb{Z}_q^{256})^k$

Public key: $pk = (\mathbf{A}, \mathbf{t} = \mathbf{As} + \mathbf{e})$, where $\mathbf{A} \xleftarrow{\$} (\mathbb{Z}_q^{256})^{k \times k}$ and $\mathbf{e} \xleftarrow{\text{CBD}_{\eta_1}} (\mathbb{Z}_q^{256})^k$

return (pk, sk)

Encryption PKE.Enc(pk, m, r):

$m \xleftarrow{\$} \mathbb{B}^{32}$

$pk = (\mathbf{A}, \mathbf{t})$

$\mathbf{y} \xleftarrow{\text{CBD}_{\eta_1}(r)} (\mathbb{Z}_q^{256})^k$

$\mathbf{e_1} \xleftarrow{\text{CBD}_{\eta_2}(r)} (\mathbb{Z}_q^{256})^k$ and $e_2 \xleftarrow{\text{CBD}_{\eta_2}(r)} (\mathbb{Z}_q^{256})$

$\mathbf{u} = \mathbf{A}^T\mathbf{y} + \mathbf{e_1}$ and $\nu = \mathbf{t}^T\mathbf{y} + e_2 + f(m))$

$c_1 = \text{Encode}_{d_u}(\text{Compress}_{d_u}(\mathbf{u})) \in \mathbb{B}^{32d_uk}$

$c_2 = \text{Encode}_{d_\nu}(\text{Compress}_{d_\nu}(\nu)) \in \mathbb{B}^{32d_\nu}$

return (c_1, c_2)

Decryption PKE.Dec($(c_1, c_2), sk$):

$sk = \mathbf{s}$

$\mathbf{u} = \text{Decode}_{d_u}(\text{Decompress}_{d_u}(c_1))$

$\nu = \text{Decode}_{d_\nu}(\text{Decompress}_{d_\nu}(c_2))$

$f^{-1}(\nu - \mathbf{s}^T\mathbf{u}) = f^{-1}(\mathbf{e}^T\mathbf{y} + e_2 + \mathbf{s}^T\mathbf{e_1} + f(m)) \approx f^{-1}(f(m)) \approx m$

return m

security reductions for these KEMs are non-tight in ROM. The ML-KEM is based on a variant of the Fujisaki-Okamoto transformation, which converts an IND-CPA secure PKE tightly into an IND-CCA secure KEM.

We introduce a simplified version of the ML-KEM. In the NIST's version, the ML-KEM.KeyGen algorithm is accompanied by a deterministic algorithm, ML-KEM.KeyGen_internal. All random sampling is performed within the KeyGen algorithm, which then uses the sampled random values to execute KeyGen_internal. A similar two-level structure exists for ML-KEM.Enc and ML-KEM.Dec. These deterministic algorithms can be used for testing during implementation. While this two-level structure allows for a clear separation between randomness and deterministic operations, which may be useful in testing and implementation, it introduces unnecessary complexity for our discussion. Therefore, we opt for a simpler, single-level design. In this version, all sampling is performed directly within the algorithms themselves, and we do not separate the sampling into its own algorithm.

We need two hash functions that take a random byte input $s \in \mathbb{B}^*$ and produce a 32-byte output $z \in \mathbb{B}^{32}$. We use two widely adopted hash functions SHA3-256 and SHAKE256. We define the following functions using these hash functions

$$\mathrm{H}(s) := \text{SHA3-256}(s) = z$$

$$\mathrm{J}(s) := \text{SHAKE256}(s) = z$$

In addition, we require a hash function that takes two random length byte strings $a, b \in \mathbb{B}^*$ as inputs and produces two hashes $K, r \in \mathbb{B}^{32}$ as outputs. We define this function as

$$\mathrm{G}(a||b) := \text{SHA3-512}(a||b) = (K, r)$$

The MLWE based KEM is presented in Algorithm 18. Bob and Alice want communicate securely. To achieve this, they want to share a symmetric key, which can be used to encrypt messages efficiently using symmetric encryption. Alice uses the key generation algorithm ML-KEM.KeyGen, which generates two keys: an encapsulation key ek and a decapsulation key dk. Alice makes the encapsulation key public, while keeping the decapsulation key private. Bob uses Alice's public encapsulation key to run the encapsulation algorithm ML-KEM.Enc. This algorithm produces a symmetric key K and returns it to Bob. The encapsulation algorithm also produces a ciphertext c from the key K, which Bob sends to Alice. Alice then uses this ciphertext and her private decapsulation key to run the decapsulation algorithm ML-KEM.Dec. If the decapsulation is successful, the algorithm returns her the copy of the symmetric key K. Now, both Bob and Alice have a copy of the shared symmetric key that they can use to encrypt messages using any symmetric-key encryption method.

However, if the decapsulation fails, the scheme does not explicitly notify whether the decapsulation was successful or not. In the case of a decapsulation failure, the scheme returns Alice an incorrect key of the same data type as K. This design choice prevents attackers from gaining information about the scheme. If both parties are honest, the probability of a decapsulation failure is very small. [34] If an encryption error still occurs, it will be noticed when the two parties attempt to use the symmetric key for communication.

Algorithm 18 NIST's Module-Lattice-based Key-Encapsulation Mechanism (ML-KEM)

Key generation algorithm ML-KEM.KeyGen(1^n):
$(\mathbf{A}, \mathbf{t}) \leftarrow$ PKE.KeyGen(1^n).
Encapsulation key: $ek = (\mathbf{A}, \mathbf{t}) \in \mathbb{B}^{384k+32}$.
Decapsulation key $dk = (\mathbf{s}, (\mathbf{A}, \mathbf{t}), H(\mathbf{A}||\mathbf{t}), z) \in \mathbb{B}^{768k+96}$, where $z \xleftarrow{\$} \mathbb{B}^{32}$.
return (ek, dk)
Encapsulation algorithm ML-KEM.Enc(ek):
$m \xleftarrow{\$} \mathbb{B}^{32}$
$ek = (\mathbf{A}, \mathbf{t})$
$(K, r) = G(m||H(\mathbf{A}||\mathbf{t}))$.
$c \leftarrow$ PKE.Enc(ek, m, r)
return (K, c)
Decapsulation algorithm ML-KEM.Dec(c, dk):
$dk = (\mathbf{s}, (\mathbf{A}, \mathbf{t}), H(\mathbf{A}||\mathbf{t}), z)$
$ek = (\mathbf{A}, \mathbf{t})$
$m' \leftarrow$ PKE.Dec(dk, c).
$(K', r') = G(m'||H(\mathbf{A}||\mathbf{t}))$.
$c' \leftarrow$ PKE.Enc(ek, m', r')
if: $c = c'$: encapsulation has been successful.
return $K = K'$
else:
return $K = J(z||c)$

5. Security of ML-KEM

Security models cannot account for all possible real-world attacks. There may be flaws in implementations, or adversaries might extract information from the system's physical behavior, such as time and power consumption. Instead of directly attacking the algorithm, they may exploit various other methods to gain information, and mathematical security models do not take all the mentioned threats into account. Additionally, these models rely on heavy assumptions, which may not reflect reality. Particularly the random oracle model assumptions have received critique [13].

Perhaps for similar reasons, NIST decided not to require a security proof for post-quantum algorithms they selected. They opted to rely more on empirical evidence of security. However, the design team of Kyber provides security theorems for Kyber in its supporting documentation [6]. The authors apply the findings of Hofheinz et al. [23] and Saito et al. [37], when deriving three security theorems.

The theorem in classical setting states that Kyber is IND-CCA secure against adversaries making at most q_{RO} queries to the random oracle, under the assumption that MLWE problem is hard. The first security theorem in quantum setting ties Kyber's IND-CCA security to the hardness of the MLWE problem in the QROM. However, this security theorem provides a non-tight reduction, which is undesirable. To address this, the authors present a second quantum security theorem, which provides a significantly tighter reduction. This theorem is based on an assumption that the deterministic PKE of Kyber satisfies a non-standard security notion called *ciphertext-indistinguishability*, which we will introduce later in this chapter. If this condition is met, Kyber is IND-CCA secure in QROM under the assumption that the MLWE problem is hard.

However, it turned out that the findings of Saito et al. do not directly apply to Kyber as it uses a different variant of the Fujisaki-Okamoto transformation. The FO transformation, first introduced by Fujisaki and Okamoto in [19][20], employs a weakly secure public-key encryption scheme and constructs a strongly secure key encapsulation mechanism in the random oracle model. The proof by Saito et al. applies to one specific variant of the FO transformation. In the literature, this FO transformation is denoted as $FO_m^{\not\perp}$. It uses a PKE scheme and follows a structure detailed in Algorithm 19. There are other variants of the FO transformation, such as a version where the symmetric key

is of the form $K = H(m, c)$. This version is denoted as $FO^{\not\perp}$. Much of the confusion surrounding the security of Kyber arises from differences in interpretations as to which FO transformation applies to it. Furthermore, the discussion becomes even more complex when we consider the differences between Kyber and the FO transformations. At its core, the structure of Kyber is a FO transformation. But Kyber's structure includes additional operations, and some of these modifications interfere with reductions.

The first version of Kyber, as described in [7], employs the FO transformation $FO^{\not\perp}$. Thus, the findings of Saito are inapplicable to it, meaning that the security theorems provided in the supporting documentation are not valid for this version. However, the FO transformation $FO^{\not\perp}$ was proven to be non-tightly IND-CCA secure in the QROM in 2017 by Jiang et al. [26].

The third version of Kyber, presented in [6], introduces additional hashes to the symmetric key, resulting in a key in the form of a double-nested-hash: $K = H(G(m), H'(c))$. In the ROM, these additional hashes do not affect the reduction. But in the QROM they complicate poofs, as noted by Grubbs at al. [22]. Consequently, the findings of Saito or Jiang do not apply to this version. Moreover, these additional hashes increase the running time of the encapsulation algorithm, as observed in [17]. This double-nested-hash version of Kyber was eventually proven non-tightly secure in QROM by Chen et al. in [15].

However, before Chen et al. published their security proofs, the design team of Kyber decided to remove the additional hashes for ML-KEM. The ML-KEM now employs the FO transformation $FO_m^{\not\perp}$, making the findings in [23] and [37] applicable to it. The security theorems in the supporting documentation of Kyber [7][6] are not applicable to Kyber as such, but should be applicable to ML-KEM.

Algorithm 19 Variant of Fujisaki-Okamoto transformation denoted as $FO_m^{\not\perp}$

Enc(ek):
$ek = pk$
$c \leftarrow \text{PKE.Enc}(pk, m, G(m))$
$K = H(m)$
Dec(dk, c):
$dk = (sk, s)$
$m = \text{PKE.Dec}(sk, c)$
if: $m \neq \perp$ No decryption error
return $K = H(m)$
else:
return $K = H(c, s)$

We now encounter the second challenge in ML-KEM security. The authors of the

supporting documentation [6] apply the findings of Hofheinz et al. and Saito et al. to their needs but omit all details of their derivation. Since NIST did not require security proofs, omitting these details aligns with the NIST's goal of emphasizing empirical evidence. However, this makes the rigorous examination of ML-KEM security more challenging. To address this, we conduct our own reasoning to understand how the findings of Saito are applied to ML-KEM.

In the section 5.1, we provide a theorem establishing the IND-CPA security of the PKE of the ML-KEM, assuming the hardness of the MLWE problem. This theorem and its proof is directly from [11]. In the section 5.2, we discuss the classical security of ML-KEM based on the findings of Hofheinz et al. in [23] and [24]. Finally, we present theorems addressing the security of ML-KEM in quantum setting in the section 5.3. In this section, we also conduct some inference and derive theorems similar to the ones in [6].

We would like to conclude this preface with a note of caution. As demonstrated with the previous versions of Kyber, even a small detail can have a significant impact on the security of the scheme. While ML-KEM includes the FO transformation $FO_m^{\not\perp}$, it involves additional operations. It is possible that these additional operations hinder the reductions, similarly to how the inclusion of the additional hashes affected the security of the third version of Kyber. The discussion surrounding ML-KEM security is ongoing, and details remain under review. To the best of our knowledge, the results presented here reflect the state-of-the-art understanding of ML-KEM security in January 2025. However, these findings may become outdated over time.

5.1 Security of PKE

We follow the security reduction from [11], which applies to a simplified version of PKE of ML-KEM that omits Compress and Encode algorithms. We will prove that this simplified PKE is IND-CPA secure, assuming that the MLWE problem is hard.

Let us define three games, G_0, G_1 and G_2. A simplified version of the PKE.KeyGen and changes made to it in the games are shown in Algorithm 20. The IND-CPA game against the PKE for the adversary B is shown in Algorithm 21. In the pseudocode, the lines that apply only to game G_i are marked with $//G_i$ at the end of the line.

The game G_0 is the IND-CPA game against the MLWE-based PKE scheme for the adversary A. We define an advantage of the form $\mathrm{Adv}_{\mathrm{PKE}}^{\text{IND-CPA}} = |\Pr[G_0 = 1] - 1/2|$, similar to the general formula in Equation (2.1). From here, we can determine that $\Pr[G_0 = 1] = \mathrm{Adv}_{\mathrm{PKE}}^{\text{IND-CPA}} + 1/2$

The game G_1 is identical to G_0, except that we modify the PKE.KeyGen algorithm. Instead of using samples of the form $\mathbf{t} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$ as public values, we replace them with

uniformly random samples. The difference between games G_0 and G_1 can be viewed as an instance of the $\text{MLWE}_{k,k,\eta}$ problem in Equation (3.15) played by the adversary B. Then, the advantage of solving the $\text{MLWE}_{k,k,\eta}$ problem serves as an upper bound for the games, expressed as:

$$|\Pr[G_1 = 1] - \Pr[G_0 = 1]| \leq \text{Adv}^{\text{MLWE}}_{k,k,\eta}(B) \leq \text{Adv}^{\text{MLWE}}_{k+1,k,\eta}(B) \tag{5.1}$$

Here, the article [11] uses the inequality $\text{Adv}^{\text{MLWE}}_{k,k,\eta}(B) \leq \text{Adv}^{\text{MLWE}}_{k+1,k,\eta}(B)$. The article does not explain why this inequality holds. We hypothesize that the reasoning is as follows: if the size of the vector $\mathbf{s}$ remains the same, but the number of equations is increased by adding rows to a matrix $\mathbf{A}$, the problem becomes easier to solve. This would then increase the advantage.

The final game, G_2, is similar to G_1, but with a change in the rules of the IND-CPA game. In game G_1 we randomly choose a bit b and encrypt the corresponding plaintext m_b into a challenge ciphertext c^*. In game G_2, on the other hand, the challenge ciphertext is chosen uniformly at random and does not depend on the challenge bit b. The adversary cannot obtain any information about the bit b from the challenge ciphertext c^*. Thus, it can only guess the bit b with a probability of $1/2$. Then, the success probability of the game G_2 is $\Pr[G_2 = 1] = 1/2$.

The difference between the games G_1 and G_2 can also be seen as an instance of the MLWE problem where the adversary aims to distinguish the challenge ciphertext from a uniformly chosen one. The size of the challenge ciphertext is $(\mathbb{Z}_q^{256})^k \times (\mathbb{Z}_q^{256})$, and thus the difference between games is the $\text{MLWE}_{k+1,k,\eta}$ problem. It then holds that

$$|\Pr[G_2 = 1] - \Pr[G_1 = 1]| \leq \text{Adv}^{\text{MLWE}}_{k+1,k,\eta}(B) \tag{5.2}$$

Algorithm 20 PKE.KeyGen(1^n)

1: $\mathbf{s} \xleftarrow{\text{CBD}_{\eta_1}} (\mathbb{Z}_q^{256})^k$
2: $\mathbf{t} = \mathbf{As} + \mathbf{e}$, where $\mathbf{A} \xleftarrow{\$} (\mathbb{Z}_q^{256})^{k \times k}$ and $\mathbf{e} \xleftarrow{\text{CBD}_{\eta_1}} (\mathbb{Z}_q^{256})^k$ $\quad //G_0$
3: $\mathbf{t} \xleftarrow{\$} (\mathbb{Z}_q^{256})^k$ $\quad //G_1, G_2$
4: **return** $(pk = (\mathbf{A}, \mathbf{t}), sk = \mathbf{s})$

Using the triangle inequality, we obtain the relation:

$$\begin{aligned} |\Pr[G_2 = 1] - \Pr[G_0 = 1]| &= |\Pr[G_2 = 1] - \Pr[G_1 = 1] + \Pr[G_1 = 1] - \Pr[G_0 = 1]| \\ &\leq |\Pr[G_2 = 1] - \Pr[G_1 = 1]| + |\Pr[G_1 = 1] - \Pr[G_0 = 1]| \end{aligned} \tag{5.3}$$

Substituting Equations (5.1) and (5.2) into this, we obtain:

$$|\Pr[G_2 = 1] - \Pr[G_0 = 1]| \leq 2\text{Adv}^{\text{MLWE}}_{k+1,k,\eta}(B)$$

Algorithm 21 IND-CPA Game against the MLWE-based PKE

1: $(pk, sk) \leftarrow \text{PKE.KeyGen}(1^n)$
2: $m_0, m_1 \leftarrow B^{\text{PKE.Enc}(\cdot)}(pk)$
3: $b \xleftarrow{\$} \{0,1\}$
4: $c^* = (\mathbf{u}, \nu) \leftarrow \text{PKE.Enc}(pk, m_b)$ $//G_0, G_1$
5: $c^* = (\mathbf{u}, \nu) \xleftarrow{\$} (\mathbb{Z}_q^{256})^k \times (\mathbb{Z}_q^{256})$ $//G_2$
6: $b' \leftarrow B^{\text{PKE.Enc}(\cdot)}(pk, c^*)$
7: **return** $[\![b = b']\!]$

As the success probabilities of the games G_0 and G_2 are $\Pr[G_0 = 1] = \text{Adv}_{\text{PKE}}^{\text{IND-CPA}}(A) + 1/2$ and $\Pr[G_2 = 1] = 1/2$, respectively, it now holds that, for any probabilistic polynomial-time adversary A, there exist an another adversary B such that

$$\text{Adv}_{\text{PKE}}^{\text{IND-CPA}}(A) \leq 2\text{Adv}_{k+1,k,\eta}^{\text{MLWE}}(B) \tag{5.4}$$

[11]

We have conjectured that based on the hardness of the module lattice problems, the MLWE problem is hard to solve. The advantage of solving $\text{MLWE}_{k+1,k,\eta}$ is thus negligible. It follows that the IND-CPA advantage against the MLWE-based PKE is also negligible, and Equation (5.4) establishes the IND-CPA security of the MLWE-based PKE.

5.2 IND-CCA security of ML-KEM in ROM

Having proven that the underlying PKE is IND-CPA secure, given this, it can be proven that the ML-KEM is IND-CCA secure when the hashes G and J are modeled as random oracles. This result was originally proven in [23], and the updated version is available in [24].

The documentation of Kyber contains two IND-CCA theorems in the ROM: one in [11] and one in [6]. However, these theorems are significantly different in form. Theorem 3 in [11] is directly derived from the work of Hofheinz et al. Although Kyber's supporting documentation in [6] claims to rely on the results of Hofheinz et al., their theorem does not resemble any result from their articles in [23] or [24]. We will directly utilize the results from Hofheinz et al. in [23] regarding the FO transformation $FO_m^{\not\perp}$ and apply them to ML-KEM. Therefore, our theorem resembles the one in [11] but not the one in [6].

It holds that, for an IND-CPA adversary A against the underlying δ-correct PKE issuing at most q_{RO} queries to the random oracles, there exists an IND-CCA adversary B against the ML-KEM such that

$$\text{Adv}_{\text{ML-KEM}}^{\text{IND-CCA}}(B) \leq (q_{\text{RO}} + 1)\delta + \frac{q_{\text{RO}}}{|\mathcal{M}|} + 3\text{Adv}_{\text{PKE}}^{\text{IND-CPA}}(A) \tag{5.5}$$

and the running time of B is approximately the same as that of A. As we have proven that the MLWE-based PKE is IND-CPA secure, Equation (5.5) states that ML-KEM is IND-CCA secure in ROM.

5.3 IND-CCA security of ML-KEM in QROM

Now, we will show two theorems regarding the quantum security of ML-KEM. Both theorems rely on non-standard security notions and assumptions from [37], which we will introduce in the next subsection. We will demonstrate that ML-KEM satisfies these conditions, and reduce the hardness of the MLWE problem to these non-standard notions.

5.3.1 Non-standard security notions

We define a non-standard security notion of *OneWayness under Chosen-Plaintext Attacks* (OW-CPA). OW-CPA states that, given a public key and a challenge ciphertext, it should be hard for an adversary to deduce the plaintext that produces the challenge ciphertext. The OW-CPA game for an adversary A is described in Algorithm 22. We say that the scheme is OW-CPA secure if the success probability, $\mathrm{Adv}^{\text{OW-CPA}}_{\text{PKE}}(A) = \Pr\left[G^{A}_{\text{OW-CPA}} = 1\right]$, is negligible for any probabilistic polynomial-time adversary A. [37]

Algorithm 22 OW-CPA game $G^{A}_{\text{OW-CPA}}$

1: $(pk, sk) \leftarrow \text{PKE.Gen}(1^n)$
2: $r \xleftarrow{\$} \mathcal{R}$
3: $m^* \xleftarrow{\$} \mathcal{M}$
4: $c^* \leftarrow \text{PKE.Enc}(pk, m^*, r)$
5: $m' \leftarrow A(pk, c^*)$
6: **return** $[\![m' = m^*]\!]$

We construct a *deterministic PKE* from the MLWE-based PKE, denoted as PKE_1. This deterministic PKE is similar to the probabilistic PKE, except that its encryption algorithm is deterministic. The deterministic encryption algorithm, denoted as PKE.Enc_1, is derived from the probabilistic encryption algorithm PKE.Enc, but instead of a random seed, it uses a hash of the plaintext as the seed. Then, the deterministic encryption algorithm is defined as $\text{Enc}_1(pk, m) = \text{Enc}(pk, m, G(m))$.

The deterministic PKE can satisfy the notion of *sparse pseudorandomness*, which combines the properties of *pseudorandomness* and *sparseness*. Informally, pseudorandomness ensures that it is difficult to distinguish between a ciphertext chosen uniformly at random from the ciphertext space and a ciphertext produced by encrypting a randomly selected plaintext. Sparseness, on the other hand, measures how sparse valid ciphertexts

are within the ciphertext space. This property ensures that a randomly generated bit string is highly unlikely to be a valid ciphertext.

Formally, for a deterministic PKE, denoted as PKE_1, satisfying the property of pseudorandomness, it holds that

$$\begin{aligned}&\mathrm{Adv}^{\mathrm{pr}}_{\mathrm{PKE}_1}(B) := \\ &\Big|\Pr\Big[1 \leftarrow B(pk, c^*) : (pk, sk) \xleftarrow{\$} \mathrm{PKE.Gen}(1^n); m^* \xleftarrow{\$} \mathcal{M}; c^* := \mathrm{PKE.Enc}_1(pk, m^*)\Big] - \\ &\qquad\qquad \Pr\Big[1 \leftarrow B(pk, c^*) : (pk, sk) \xleftarrow{\$} \mathrm{PKE.Gen}(1^n); c^* \xleftarrow{\$} \mathcal{C}\Big]\Big| \end{aligned} \tag{5.6}$$

is negligible for all probabilistic polynomial time algorithm B, and for a sparse PKE_1 it holds that

$$\mathrm{Sparse}_{\mathrm{PKE}_1} := \max_{(pk,sk)\in \mathrm{PKE.Gen}(1^n,\mathcal{R})} \frac{\Big|\mathrm{PKE.Enc}_1(pk, \mathcal{M})\Big|}{|\mathcal{C}|} \tag{5.7}$$

is negligible. [37]

We also introduce another non-standard security notion, *disjoint simulatability*, for a deterministic PKE. This notion involves a simulator $\mathcal{S}$, which is a probabilistic polynomial-time algorithm. Given a public key, $\mathcal{S}$ attempts to generate a "fake ciphertext" that is indistinguishable from a ciphertext produced by encrypting a randomly chosen plaintext [37]. We require that this fake ciphertext belongs to the valid ciphertext space only with a negligible probability.

Formally, we require that the disjoint simulatable deterministic PKE satisfies *statistical disjointness*, i.e., there exists a probabilistic polynomial time adversary $\mathcal{S}$ such that

$$\mathrm{Disj}_{\mathrm{PKE}_1,\mathcal{S}} := \max_{(pk,sk)\in \mathrm{PKE.Gen}(1^n;\mathcal{R})} \Pr[c \in \mathrm{PKE.Enc}_1(pk, \mathcal{M}) : c \leftarrow \mathcal{S}(pk)] \tag{5.8}$$

is negligible. Additionally, the deterministic PKE satisfies the notion of *ciphertext-indistinguishability*: such that

$$\begin{aligned}&\mathrm{Adv}^{\text{ds-ind}}_{\mathrm{PKE}_1,\mathcal{S}}(B) := \\ &\Big|\Pr\Big[1 \leftarrow B(pk, c^*) : (pk, sk) \xleftarrow{\$} \mathrm{PKE.Gen}(1^n);\ m^* \xleftarrow{\$} \mathcal{M};\ c^* = \mathrm{PKE.Enc}_1(pk, m^*)\Big] \\ &\qquad - \Pr\Big[1 \leftarrow B(pk, c^*) : (pk, sk) \xleftarrow{\$} \mathrm{PKE.Gen}(1^n);\ c^* \xleftarrow{\$} \mathcal{S}(pk)\Big]\Big| \end{aligned} \tag{5.9}$$

is negligible for all probabilistic polynomial time algorithms B. If we choose $\mathcal{S}$ to generate fake ciphertext randomly, we observe that sparse pseudorandomness implies disjoint simulatability.

The total ciphertext space of ML-KEM is $\mathbb{B}^{256(d_u k+d_v)}$. For the deterministic PKE to exhibit sparseness, the properly generated ciphertext space must be significantly smaller

than this. Then, the ratio of the size of the valid ciphertext space and that of the total ciphertext space becomes very small, ensuring that the quantity in Equation (5.7) is negligible. The Kyber's documentation [11] confirms that this condition holds for MLWE-based PKE. Therefore, the deterministic MLWE-base PKE is sparse.

In the MLWE problem the adversary B is trying trying to distinguish uniformly random samples from the samples of the form $(\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e})$. As ciphertexts produced by the MLWE-based PKE are of this form, the pseudorandomness of the MLWE-based PKE for adversary A is equivalent to that of the MLWE problem for adversary B, and it holds that

$$\mathrm{Adv}^{\mathrm{pr}}_{\mathrm{PKE}_1}(A) \leq \mathrm{Adv}^{\mathrm{MLWE}}_{k+1,k,\eta}(B) \tag{5.10}$$

[11]. As the MLWE problem is considered hard, the advantage of solving it is negligible. Then, the notion $\mathrm{Adv}^{\mathrm{pr}}_{\mathrm{PKE}_1}(A)$ is negligible, meaning that the PKE_1 satisfies the property of pseudorandomness. As the deterministic version of the PKE satisfies both sparseness and pseudorandomness, it satisfies sparse pseudorandomness. This further implies that the deterministic MLWE-based PKE is also disjoint simulatable.

Now, we reduce pseudorandomness to OW-CPA security. It seems likely that the authors of the supporting documentation [6] have done similar reduction. Let the adversary A play the OW-CPA game in Algorithm 22 against the MLWE-based PKE. A is given a ciphertext as an input, and its goal is to determine the plaintext that produced this ciphertext. The adversary B aims to distinguish between a ciphertext chosen uniformly at random from the ciphertext space and a ciphertext produced by encrypting a randomly chosen plaintext as defined in the pseudorandomness in Equation (5.6). It outputs a bit that indicates whether it believes the input it received was randomly chosen or encrypted. If B distinguishes whether the input it receives is encrypted or randomly chosen, it wins. The adversary B receives as input that is a randomly chosen ciphertext or a ciphertext produced by encrypting a randomly chosen plaintext. B uses the adversary A as a subroutine, providing it as an input the challenge ciphertext that it itself received. If A wins the OW-CPA game, it successfully determined the plaintext that produced the challenge ciphertext. It then returns this plaintext to B, which allows B to determine whether the ciphertext it received as input was encrypted or randomly chosen. When B distinguishes the input, it wins. Thus, it holds that

$$\mathrm{Adv}^{\text{OW-CPA}}_{\mathrm{PKE}}(A) \leq \mathrm{Adv}^{\mathrm{pr}}_{\mathrm{PKE}_1}(B) \tag{5.11}$$

We have previously proven that pseudorandomness is upper bounded by the hardness of the MLWE problem, as shown in Equation (5.10). We can further reduce the MLWE problem to OW-CPA, and obtain that, for an OW-CPA adversary A against PKE, there exist a MLWE adversary C such that

$$\mathrm{Adv}^{\text{OW-CPA}}_{\mathrm{PKE}}(A) \leq \mathrm{Adv}^{\mathrm{MLWE}}_{k+1,k,\eta}(C) \tag{5.12}$$

and the running time of C is about that of A.

5.3.2 Quantum security theorems for ML-KEM

We use Theorems 3.5 and 4.2 from [37] and apply them to ML-KEM, modeling the hashes G and J as quantum random oracles. For any quantum adversary A against the ML-KEM, issuing at most q_G and q_J queries to the random oracles G and J, and q_{Enc_1} encryption and q_{Dec} decryption queries, there exist an adversary B against OW-CPA of the PKE such that

$$\text{Adv}^{\text{IND-CCA}}_{\text{ML-KEM}'}(A) \leq 4q_G\sqrt{\text{Adv}^{\text{OW-CPA}}_{\text{PKE}}(B)} + \text{Disj}_{\text{PKE}_1,\mathcal{S}} + 4q_J \cdot 2^{-128} \tag{5.13}$$

and the running times are $\text{Time}(B) \approx \text{Time}(A) + q_G \cdot \text{Time}(\text{Enc}_1) + (q_G + q_J + q_{\text{Dec}}) \cdot t_{RO} + q_J \cdot t_{RO}$. Here, t_{RO} denotes the running time of simulating quantum random oracles.

Now, we can insert Equation (5.12) to Equation (5.13) and obtain:

$$\text{Adv}^{\text{IND-CCA}}_{\text{ML-KEM}'}(A) \leq 4q_G\sqrt{\text{Adv}^{\text{MLWE}}_{k+1,k,\eta}(B)} + \text{Disj}_{\text{PKE}_1,\mathcal{S}} + 4q_J \cdot 2^{-128} \tag{5.14}$$

However, this theorem assumes the underlying PKE to be perfectly correct, which is not the case for the PKE of ML-KEM. The authors of Kyber's supporting documentation incorporate an additional term from [23] to account for the δ-correctness of the PKE. Hofheinz et al prove in [23] that we can construct an $8(q_G+1)^2 \cdot \delta$-correct deterministic PKE from a δ-correct PKE in QROM, provided that any adversary issues at most q_G quantum queries to the random oracle G that forms the deterministic encryption algorithm of the form $\text{PKE.Enc}(pk, m, G(m))$. The term $8(q_G+1)^2 \cdot \delta$ is added to the security theorem in the supporting documentation, but the documentation does not explain how this is derived. They have probably reduced ML-KEM with a perfectly correct PKE to ML-KEM with a δ-correct PKE similar to what we do next.

As the ML-KEM is a variant of the FO transformation in Algorithm 19, it utilizes the δ-correct PKE deterministically. This means that in the encapsulation algorithm of ML-KEM, the ciphertext is produced using the deterministic version of the δ-correct PKE as $c \leftarrow \text{PKE.Enc}(pk, m, G(m))$. This deterministic PKE is then $8(q_G + 1)^2 \cdot \delta$-correct in QROM, and it will cause a decryption error with a probability of $8(q_G + 1)^2 \cdot \delta$.

We have an adversary A playing the IND-CCA game against ML-KEM with a δ-correct PKE, denoted $\text{Adv}^{\text{IND-CCA}}_{\text{ML-KEM}}(A)$. We also have another adversary B playing IND-CCA game against ML-KEM with a perfectly correct PKE, denoted as $\text{Adv}^{\text{IND-CCA}}_{\text{ML-KEM}'}(B)$. Both adversaries can make queries to the decapsulation oracle. These two games differ only when the deterministic PKE makes a decryption error such that $\text{PKE.Dec}(sk, \text{PKE.Enc}(pk, m, G(m))) \neq m$, when A queries the decapsulation oracle.

When such error occurs, the decapsulation oracle returns A an incorrect key $K = H(m)$ instead of the key $K = H(c, s)$. If the adversary A can distinguish when it receives

an incorrect key, it can detect when a decryption error has occurred. The knowledge of the decryption error may allow A to deduce additional information about its query. If certain ciphertexts are more likely to cause decryption errors, A may exploit this fact to its advantage. As result, the IND-CCA game may become easier for A compared to B who has no possibility for obtaining such additional information. Thus, the decryption errors increase A's IND-CCA advantage compared to the IND-CCA advantage of B. A decryption error occurs with a probability of $8(q_G+1)^2 \cdot \delta$, and it holds that:

$$\mathrm{Adv}^{\text{IND-CCA}}_{\text{ML-KEM}}(A) \leq \mathrm{Adv}^{\text{IND-CCA}}_{\text{ML-KEM'}}(B) + 8 \cdot (q_G+1)^2 \cdot \delta$$

We substitute Equation (5.14) into this, and obtain the final theorem

$$\mathrm{Adv}^{\text{IND-CCA}}_{\text{ML-KEM}}(A) \leq 4q_G\sqrt{\mathrm{Adv}^{\text{MLWE}}_{k+1,k,\eta}(B)} + \mathrm{Disj}_{\mathrm{PKE}_1,\mathcal{S}} + 4q_J \cdot 2^{-128} + 8(q_G+1)^2 \cdot \delta \quad (5.15)$$

which states that ML-KEM is secure in the quantum random oracle model, if the MLWE problem is hard.

In the above security theorem, the IND-CPA advantage appears under a square root, which is a consequence of using the one-way to hiding lemma in Equation (2.8) in the reduction. The advantage of solving the MLWE problem is negligible, and taking a square root of this very small value increases it, thereby increasing the upper bound of the IND-CCA advantage. Then, the IND-CCA advantage may be significantly larger than the advantage of solving MLWE, making this theorem non-tight. Thus, this theorem can only provide an asymptotic indication of security.

One way to avoid this non-tightness is to assume that the deterministic version of the PKE of ML-KEM is *ciphertext-indistinguishable*, which implies pseudorandomness. Using Equation (5.10), we can further reduce the hardness of the MLWE problem to pseudorandomness. Taking this approach, we can achieve a tighter bound. For any quantum adversary A against the ML-KEM issuing at most q_G and q_J queries to random oracles G and J, and q_{Enc_1} encryption and q_{Dec} decryption queries, there exists an adversary B against the MLWE problem such that

$$\mathrm{Adv}^{\text{IND-CCA}}_{\text{ML-KEM'}}(A) \leq 2\mathrm{Adv}^{\text{MLWE}}_{k+1,k,\eta}(B) + \mathrm{Disj}_{\mathrm{PKE}_1,\mathcal{S}} + 4q_J \cdot 2^{-128} \quad (5.16)$$

and the running times are $\mathrm{Time}(B) \approx \mathrm{Time}(A) + q_J \cdot \mathrm{Time}(\mathrm{Enc}_1) + (q_G + q_J + q_{\mathrm{Dec}}) \cdot t_{RO}$.

This theorem again assumes the PKE of ML-KEM to be perfectly correct. We can perform a similar reduction as above, reducing ML-KEM with a perfectly correct PKE to ML-KEM with a δ-correct PKE. We then obtain the second theorem for the quantum security of ML-KEM

$$\mathrm{Adv}^{\text{IND-CCA}}_{\text{ML-KEM}}(A) \leq 2\mathrm{Adv}^{\text{MLWE}}_{k+1,k,\eta}(B) + \mathrm{Disj}_{\mathrm{PKE}_1,\mathcal{S}} + 4q_J \cdot 2^{-128} + 8 \cdot (q_G+1)^2 \cdot \delta \quad (5.17)$$

and the running times are $\mathrm{Time}(B) \approx \mathrm{Time}(A) + q_J \cdot \mathrm{Time}(\mathrm{Enc}_1) + (q_G + q_J + q_{\mathrm{Dec}}) \cdot t_{RO}$.

The advantages are now roughly the same, but the runtime of algorithm A is faster than that of B. It depends on the interpreter whether these runtimes differ significantly enough that this theorem cannot be classified as tight. However, strictly speaking this theorem is non-tight. It is believed that the impact on the runtime is not as significant as the increase in the advantage observed in the previous theorem, and this theorem is at least tighter compared to the previous one [37].

6. Conclusions

In this thesis, we have provided an in-depth introduction to the NIST's Module-Lattice-based Key-Encapsulation Mechanism (ML-KEM), with an emphasis on the mathematical security proofs. We introduced the Module Learning With Errors problem that serves as the basis for ML-KEM, and discussed the hardness of this problem. We then explained the working principle of ML-KEM and presented three theorems for its security.

We showed that the MLWE-based PKE is IND-CPA secure in the random oracle model, if the MLWE problem is hard. Based on this fact, we presented a theorem for the IND-CCA security of the ML-KEM, when the hashes G and J are modeled as random oracles.

We also derived a non-tight security theorem for the IND-CCA security of the ML-KEM in the quantum random oracle model under the assumption that the MLWE problem is hard. Furthermore, by assuming that the deterministic PKE satisfies ciphertext-indistinguishability, we derived a tighter bound for the IND-CCA security of ML-KEM in quantum random oracle model.

As both of these quantum security theorems are, strictly speaking, non-tight, it remains an open problem to prove rigorously that ML-KEM is tightly IND-CCA secure in QROM. However, finding a tight bound for ML-KEM quantum security may be a very challenging task, as finding a tight bound for Kyber has proven difficult. All existing quantum security theorems for the previous versions of Kyber are non-tight. Achieving a tight theorem for the quantum security of the ML-KEM may require further advancement in reduction techniques in QROM.

Bibliography

[1] 3rd Generation Partnership Project (3GPP). *3GPP TS 33.401: 3GPP System Architecture Evolution (SAE); Security Architecture*. Tech. rep. 33.401. Available at: https://www.3gpp.org/. 3GPP, 2024.

[2] 3rd Generation Partnership Project (3GPP). *3GPP TS 33.501: Security architecture and procedures for 5G System*. Tech. rep. 33.501. Available at: https://www.3gpp.org/. 3GPP, 2025.

[3] Miklós Ajtai. "Generating hard instances of lattice problems". In: *Electron. Colloquium Comput. Complex.* TR96 (1996). URL: https://api.semanticscholar.org/CorpusID:6864824.

[4] Martin R. Albrecht, Rachel Player, and Sam Scott. "On the concrete hardness of Learning with Errors". In: *Journal of Mathematical Cryptology* 9.3 (2015), pp. 169–203. DOI: 10.1515/jmc-2015-0016.

[5] Erdem Alkim et al. *Post-quantum key exchange - a new hope*. Cryptology ePrint Archive, Paper 2015/1092. 2015. URL: https://eprint.iacr.org/2015/1092.

[6] Roberto Avanzi et al. *CRYSTALS-Kyber Algorithm Specifications And Supporting Documentation (version 3.01)*. Version 3.01. Jan. 2021. URL: https://pq-crystals.org/kyber/data/kyber-specification-2021-01.pdf.

[7] Roberto Avanzi et al. *CRYSTALS-Kyber Algorithm Specifications And Supporting Documentation (version 3.01)*. Version 1. Jan. 2021. URL: https://pq-crystals.org/kyber/data/kyber-specification-2021-01.pdf.

[8] Mihir Bellare and Phillip Rogaway. *Introduction to modern cryptography*. 2005. URL: https://web.cs.ucdavis.edu/~rogaway/classes/227/spring05/book/main.pdf.

[9] Daniel J Bernstein, Johannes Buchmann, and Erik Dahmen, eds. *Post-Qufantum Cryptography*. 1st ed. Hardcover ISBN: 978-3-540-88701-0, Softcover ISBN: 978-3-642-10019-2, eBook ISBN: 978-3-540-88702-7. Springer Berlin, Heidelberg, 2009, pp. X, 246. DOI: 10.1007/978-3-540-88702-7.

[10] Dan Boneh et al. "Random Oracles in a Quantum World". In: *Advances in Cryptology – ASIACRYPT 2011*. Ed. by Dong Hoon Lee and Xiaoyun Wang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 41–69. ISBN: 978-3-642-25385-0.

[11] Joppe Bos et al. "CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM". In: *2018 IEEE European Symposium on Security and Privacy (EuroSP)*. 2018, pp. 353–367. DOI: 10.1109/EuroSP.2018.00032.

[12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. "Fully Homomorphic Encryption without Bootstrapping". In: *Electron. Colloquium Comput. Complex.* TR11 (2011). URL: https://api.semanticscholar.org/CorpusID:2182541.

[13] Ran Canetti, Oded Goldreich, and Shai Halevi. *The Random Oracle Methodology, Revisited*. 2000. arXiv: cs/0010019 [cs.CR]. URL: https://arxiv.org/abs/cs/0010019.

[14] Lily Chen et al. *Report on Post-Quantum Cryptography*. NIST Interagency/Internal Report (NISTIR). Gaithersburg, MD: National Institute of Standards and Technology, 2016. URL: https://doi.org/10.6028/NIST.IR.8105 (visited on 05/28/2024).

[15] Zhao Chen et al. "IND-CCA Security of Kyber in the Quantum Random Oracle Model, Revisited". In: *Information Security and Cryptology*. Ed. by Yi Deng and Moti Yung. Cham: Springer Nature Switzerland, 2023, pp. 148–166. ISBN: 978-3-031-26553-2.

[16] Thomas H Cormen et al. *Introduction to Algorithms*. 3rd. Cambridge, MA: MIT Press, 2009. ISBN: 978-0262033848.

[17] CRYSTALS-Kyber submission team. *Kyber decisions, part 2: FO transform*. https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/C0D3W1KoINY/m/99kIvydoAwAJ. PQC-Forum Post. 2023.

[18] David S. Dummit and Richard M. Foote. *Abstract Algebra*. 3rd. Hoboken, NJ: John Wiley & Sons, 2004. ISBN: 978-0471433347.

[19] Eiichiro Fujisaki and Tatsuaki Okamoto. "Secure Integration of Asymmetric and Symmetric Encryption Schemes". In: *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*. Vol. 1666. Lecture Notes in Computer Science. Springer, 1999, pp. 537–554. ISBN: 978-3-540-66456-7. DOI: 10.1007/3-540-48405-1_34. URL: https://doi.org/10.1007/3-540-48405-1_34.

[20] Eiichiro Fujisaki and Tatsuaki Okamoto. "Secure Integration of Asymmetric and Symmetric Encryption Schemes". In: *Journal of Cryptology* 26 (2013), pp. 80–101. DOI: 10.1007/s00145-012-9125-5.

[21] Lov K. Grover. "A Fast Quantum Mechanical Algorithm for Database Search". In: *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC)*. ACM. 1996, pp. 212–219.

[22] Paul Grubbs, Varun Maram, and Kenneth G. Paterson. *Anonymous, Robust Post-Quantum Public Key Encryption*. Cryptology ePrint Archive, Paper 2021/708. 2021. URL: https://eprint.iacr.org/2021/708.

[23] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. "A Modular Analysis of the Fujisaki-Okamoto Transformation". In: *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I*. Ed. by Yael Tauman Kalai and Leonid Reyzin. Vol. 10677. Lecture Notes in Computer Science. Springer, Cham, 2017, pp. 341–371. ISBN: 978-3-319-70499-9. DOI: 10.1007/978-3-319-70500-2_12. URL: https://doi.org/10.1007/978-3-319-70500-2_12.

[24] Kathrin Hövelmanns. "Generic constructions of quantum-resistant cryptosystems". doctoralthesis. Ruhr-Universität Bochum, Universitätsbibliothek, 2021. DOI: 10.13154/294-7758.

[25] Kenneth Ireland and Michael Rosen. *A Classical Introduction to Modern Number Theory*. 2nd ed. Vol. 84. Graduate Texts in Mathematics. Springer New York, NY, 1990, pp. XIV, 394. ISBN: 978-0-387-97329-6. DOI: 10.1007/978-1-4757-2103-4.

[26] Haodong Jiang et al. *IND-CCA-secure Key Encapsulation Mechanism in the Quantum Random Oracle Model, Revisited*. Cryptology ePrint Archive, Paper 2017/1096. 2017. DOI: 10.1007/978-3-319-96878-0_4. URL: https://eprint.iacr.org/2017/1096.

[27] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. 3rd ed. CRC Press, 2020. ISBN: 9780367331580.

[28] Adeline Langlois and Damien Stehle. *Worst-Case to Average-Case Reductions for Module Lattices*. Cryptology ePrint Archive, Paper 2012/090. https://eprint.iacr.org/2012/090. 2012. URL: https://eprint.iacr.org/2012/090.

[29] Vadim Lyubashevsky. *Basic Lattice Cryptography: The concepts behind Kyber (ML-KEM) and Dilithium (ML-DSA)*. Cryptology ePrint Archive, Paper 2024/1287. 2024. URL: https://eprint.iacr.org/2024/1287.

[30] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. "A Toolkit for Ring-LWE Cryptography". In: *Advances in Cryptology – EUROCRYPT 2013*. Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2013, pp. 35–54. DOI: 10.1007/978-3-642-38348-9_3. URL: https://doi.org/10.1007/978-3-642-38348-9_3.

[31] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. "On Ideal Lattices and Learning with Errors over Rings". In: *Advances in Cryptology – EUROCRYPT 2010*. Ed. by Henri Gilbert. Vol. 6110. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2010, pp. 1–23. DOI: 10.1007/978-3-642-13190-5_1. URL: https://doi.org/10.1007/978-3-642-13190-5%5C_1.

[32] Richard A. Mollin. *Algebraic Number Theory, Second Edition*. 2nd. Chapman & Hall/CRC, 2011. ISBN: 9781439845981.

[33] Michele Mosca and Marco Piani. *Quantum Threat Timeline Report 2023*. Co-Founder & CEO, evolutionQ Inc. and Senior Research Analyst, evolutionQ Inc. Waterloo, Ontario, Canada: evolutionQ Inc., Dec. 2023.

[34] National Institute of Standards and Technology. *FIPS 203: Federal Information Processing Standards Publication Module-Lattice-based Key-Encapsulation Mechanism Standard*. Available online at https://doi.org/10.6028/NIST.FIPS.203.ipd. Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899-8900. Aug. 2024.

[35] Network Associates, Inc. *An Introduction to Cryptography*. 3965 Freedom Circle, Santa Clara, CA 95054: Network Associates, Inc., 1999. URL: http://www.nai.com.

[36] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *J. ACM* 56.6 (Sept. 2009). ISSN: 0004-5411. DOI: 10.1145/1568318.1568324. URL: https://doi.org/10.1145/1568318.1568324.

[37] Tsunekazu Saito, Keita Xagawa, and Takashi Yamakawa. "Tightly-Secure Key-Encapsulation Mechanism in the Quantum Random Oracle Model". In: (Mar. 2018), pp. 520–551. DOI: 10.1007/978-3-319-78372-7_17.

[38] Ardianto Satriawan, Rella Mareta, and Hanho Lee. *A Complete Beginner Guide to the Number Theoretic Transform (NTT)*. Cryptology ePrint Archive, Paper 2024/585. https://eprint.iacr.org/2024/585. 2024. DOI: 10.1109/ACCESS.2023.3294446. URL: https://eprint.iacr.org/2024/585.

[39] P.W. Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In: *Proceedings 35th Annual Symposium on Foundations of Computer Science*. 1994, pp. 124–134. DOI: 10.1109/SFCS.1994.365700.

[40] Victor Shoup. *Sequences of games: a tool for taming complexity in security proofs*. Cryptology ePrint Archive, Paper 2004/332. https://eprint.iacr.org/2004/332. 2004. URL: https://eprint.iacr.org/2004/332.

[41] Amber Sprenkels. *The Number Theoretic Transform in Kyber and Dilithium*. Accessed: 2025-01-21. 2020. URL: https://electricdusk.com/ntt.html.

[42] National Institute of Standards and Technology. *Module-Lattice-Based Digital Signature Standard (MLDSS)*. Tech. rep. FIPS 204. National Institute of Standards and Technology, 2024. URL: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.pdf.

[43] National Institute of Standards and Technology. *Stateless Hash-Based Digital Signature Standard (SLH-DSA)*. Tech. rep. FIPS 205. https://doi.org/10.6028/NIST.FIPS.205. National Institute of Standards and Technology, 2024. URL: https://csrc.nist.gov/pubs/fips/205/final.

[44] Dominique Unruh. "Revocable Quantum Timed-Release Encryption". In: *J. ACM* 62.6 (Dec. 2015). ISSN: 0004-5411. DOI: 10.1145/2817206. URL: https://doi.org/10.1145/2817206.

[45] Guangwu Xu. *A Remark on Fourier Transform*. 2023. arXiv: 1807.05829 [math.HO]. URL: https://arxiv.org/abs/1807.05829.

[46] Mark Zhandry. "Secure Identity-Based Encryption in the Quantum Random Oracle Model". In: *Advances in Cryptology – CRYPTO 2012*. Ed. by Reihaneh Safavi-Naini and Ran Canetti. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 758–775.