CRYSTALS
(CRYptographic SuiTe for Algebraic LatticeS): A New (Classical) Post-Quantum Public-Key Cryptosystem Standard

Advanced Topics in Information Security / Cryptography and Security Protocols

Ph.D. (Doctoral Program) in Information Security

Instituto Superior Técnico, University of Lisbon
(2022/2023 - 2nd Semester)

Rúben Barreiro[1⋆] and Paulo Mateus[1,2⋆⋆]

[1] Instituto Superior Técnico, University of Lisbon, Portugal
{ruben.andre.letra.barreiro,paulo.mateus}
@tecnico.ulisboa.pt
[2] Instituto de Telecomunicações, Portugal
paulo.mateus@lx.it.pt

[⋆] Student Identification: IST1108107

[⋆⋆] Acknowledgments and thanks to Prof. Paulo Mateus, who is the lecturer of the course "Advanced Topics in Information Security / Cryptography and Security Protocols" at Instituto Superior Técnico, University of Lisbon, Portugal.

Abstract. In the last few years, Quantum Computing has been growing exponentially, and several quantum algorithms are threatening most of the cryptography we are using nowadays. In 1994, Peter Shor proposed one of them, known nowadays as Shor's algorithm. This quantum algorithm is capable of factoring large integer numbers into smaller prime numbers and finding discrete logarithms in a polylogarithmic time. These two problems are considered hard to solve on a classical computer, and we apply this computational hardness assumption as the basis of several cryptosystems we used since the 1970s. For instance, the security factor of popular cryptosystems, such as Rivest-Shamir Adleman (RSA), Diffie-Hellman (DH), and Elliptic-Curve Cryptography (ECC), rely on this assumption. However, this hardness assumption does not hold for quantum computers due to the expected impact of Shor's algorithm.

Concerning this imminent threat, the scientific community is putting enormous efforts into selecting new quantum-resistant candidates, arising two new leading families of quantum-resistant cryptography, known as (Classical) Post Quantum Cryptography and Quantum Cryptography. For the former, we consider mathematical approaches and problems believed to be computationally hard to solve using both classical and quantum computers. From this family of candidates, we have a popular set of cryptographic primitives based on mathematical problems related to algebraic lattices, commonly known as Lattice-based Cryptography.

In this paper/report, we discuss Lattice-based Cryptography in detail, its fundamentals, as well as the respective mathematical and computational hardness assumptions involved, presenting a detailed description and analysis of CRYptographic SuiTe for Algebraic LatticeS (CRYSTALS) public-key cryptosystem, namely its variants for asymmetric encryption (CRYSTALS-Kyber) and digital signatures (CRYSTALS-Dilithium), already selected as the new (and first-ever) standards for public-key cryptography to be adopted at the beginning of the post-quantum era.

1 Background

Recently, due to the emergence of Quantum Computing and its expected future impact on Modern Cryptography, from the threats arising from some quantum algorithms such as Grover's [1], Simon's [2], Brassard-Hoyer-Tapp (BHT) [3], and Shor's [4,5] algorithms, new quantum-resistant cryptosystems are being studied.

These new alternatives of Modern Cryptography are mainly focused on Key Exchanges and Public-Key Cryptography since the expected impacts of Grover's, Simon's, and BHT algorithms on Symmetric Cryptography and Cryptographic Hash Functions are minor when compared to the impact Shor's algorithm will have on Asymmetric Cryptography and Public-Key Cryptosystems we use.

We support this claim since we expect symmetric encryption and cryptographic hash algorithms to have their security strength only reduced by half and by a third, respectively. For instance, symmetric encryption algorithms, such as Advanced Encryption Standard (AES) [6], are currently believed to be quantum-resistant when we use them with key sizes doubled regarding what we consider secure nowadays in order to resist to brute-force attacks made by Grover's and Simon's algorithms[3]. Similarly, cryptographic hash algorithms and functions, such as Secure Hash Algorithm (SHA) [7], are currently believed to be quantum-resistant when we use them with output sizes tripled regarding what we consider secure today in order to resist to collision, claw-finding, and birthday attacks made by BHT algorithm, also known as Quantum Birthday Attack [4].

In the opposite direction, we expect the Asymmetric and Public-Key Cryptography we currently use nowadays to be easily "broken" by Shor's algorithm in polylogarithmic time, running in a sufficiently powerful Quantum Computer with a large processing capacity, no matter how large the keys used are nor how we change the security parameters. This claim is valid since Shor's algorithm can solve the factorization of large integer numbers into smaller prime numbers as well as the discrete logarithm problem in a very efficient fashion, while most of the currently used public-key cryptosystems, digital signature schemes, and key exchange protocols in our lives, such as Rivest-Shamir-Adleman (RSA) [8], Diffie-Hellman (DH) [9,10], ElGamal [11], Digital Signature Algorithm (DSA) [12–14], and other primitives based on Elliptic-Curve Cryptography (ECC) [15, 16], have their security relied precisely on the computational hardness from these two mathematical problems when we consider only classical contexts. However, for quantum contexts, all these cryptographic primitives will become obsolete and that is the reason why we need to replace them in the future.

From this incoming need, two new major families of Modern Cryptography arose, one named (Classical) Post-Quantum Cryptography [17,18] and the other named Quantum Cryptography [19–24], as the more suitable candidates to replace the cryptographic primitives we use today, which we will start to refer to as (Classical) Pre-Quantum Cryptography to clarify the presented nomenclature.

The former is very similar to the cryptographic primitives we currently use in (Classical) Pre-Quantum Cryptography and also uses classical information (in the form of bits). However, (Classical) Post-Quantum Cryptography has its basis in mathematical problems and computational assumptions we believe to be hard to be solved by a classical computer and even by a considerably powerful quantum computer. In particular, we have several categories of (Classical) Post-Quantum Cryptography differing on what type of mathematical problems they have their computational hardness assumptions based on, such as Lattice-based Cryptography [25–27], Code-based Cryptography [28–30], Hash-based Cryptography [31–33], Isogeny-based Cryptography [34–36], Multivariate Cryptography [37–39], and Zero-Knowledge Proofs (ZKPs) [40, 41]. Since some cryptographic primitives for symmetric encryption and hash computation, such as AES and SHA, for example, are quantum-resistant when used with sufficiently large sizes for keys and outputs, as mentioned before, we can consider them primitives of (Classical) Post-Quantum Cryptography as well.

The latter follows a very different approach when compared to the previously mentioned (Classical) Pre-Quantum and Post-Quantum Cryptography. Namely,

[3] For AES, 256 bits are considered quantum-resistant.

[4] For SHA, 384 bits or more, are believed to be quantum-resistant.

it uses quantum information (in the form of quantum bits, also known simply as qubits [21,42], or quantum modes, also known simply as qumodes [43–45]) in addition to classical information. Moreover, it does not depend on any mathematical problems and computational hardness assumptions since it relies on postulates and fundamentals from Quantum Mechanics, as well as laws of physics. More specifically, it uses well-studied quantum phenomena such as Quantum Superposition and Entanglement, as well as postulates and properties ruled from them, such as Wave Interference, Heisenberg's Uncertainty Principle, Observer's Effect, Measurement Statistics (e.g., Non-Locality), No-Go Theorems (e.g., No-Cloning and No-Deleting Theorems), Monogamy of Entanglement, Special Relativity, among others. Since these quantum phenomena rule nature and reality themselves, they cannot simply be avoided or modified by an attacker, as well as retire any dependence of cryptographic primitives on future advances in computational power and respective continuous cryptoanalysis, reasons why it offers theoretical unconditional security in most of the cases and can achieve perfect security in the future. In particular, from Quantum Cryptography, we can build several cryptographic primitives and protocols, such as Quantum Key Distribution (QKD) [21, 42–45], Semi-Quantum Key Distribution (SQKD) [46, 47], Quantum Conference Key Agreement (QCKA) [48–50], Quantum Oblivious Transfer (QOT) [51, 52], Quantum Digital Signature (QDS) [53], Quantum Hashing [54, 55], Quantum Bit Commitment (QBC) [56–58], among others. However, due to the current state of technology, the development and implementation of these physical apparatuses and systems still have limitations, and there are gaps between its currently available setups and the ideal ones.

From this current perspective, governments and institutions are mostly developing and adopting Quantum Cryptography for critical infrastructures and services, disposed of fixed nodes that do not require significant mobility requirements but must have higher levels of security, such as military, banking, and healthcare infrastructures, while (Classical) Post-Quantum Cryptography is more targeted to the global community, such as mobile telecommunications, Internet of Things (IoT), and World Wide Web (WWW) we use in quotidian.

One of the most famous and well-accepted initiatives for the new directions to follow globally from the current perspective in Modern Cryptography is the Post-Quantum Cryptography Standardization initiative from the National Institute of Standards and Technology (NIST)[5]. This initiative started in 2016 and is still ongoing, aiming to select the next (quantum-resistant) cryptographic standards for Asymmetric and Public-Key Cryptography. Despite the NIST's Post-Quantum Cryptography Standardization being currently in its 4th round, NIST already selected the (initial) four quantum-resistant asymmetric and public-key cryptosystems: CRYSTALS-Kyber [59,60], CRYSTALS-Dilithium [61, 62], FALCON [63,64], and SPHINCS^{+} [65,66]. For the 4^{th} round, there are other three quantum-resistant asymmetric cryptosystems currently in analysis for a standardization: BIKE [67,68], Classic McEliece [28,69], and HQC [70,71].

[5] The official link for the Post-Quantum Cryptography Standardization from NIST is: `https://csrc.nist.gov/projects/post-quantum-cryptography`

Regarding the current status of the NIST's Post-Quantum Cryptography Standardization initiative with the respective cryptographic algorithms already selected and in analysis for standardization, we have the following table:

Selected for Standardization		
Name	**Purpose**	**Family**
CRYSTALS-Kyber	Key Exchange (Encaps./Decaps.)	Lattice-based
CRYSTALS-Dilithium	Digital Signature Scheme	Lattice-based
FALCON	Digital Signature Scheme	Lattice-based
SPHINCS$^+$	Digital Signature Scheme	Hash-Based
In Analysis for Standardization		
Name	**Purpose**	**Family**
BIKE	Key Exchange (Encaps./Decaps.)	Code-Based
Classic McEliece	Key Exchange (Encaps./Decaps.)	Code-Based
HQC	Key Exchange (Encaps./Decaps.)	Code-Based

Table 1: The current status of the (ongoing) NIST's Post-Quantum Cryptography Standardization initiative.

For a global perspective of the current state of Modern Cryptography regarding both classical and quantum contexts, we have the following summary table:

Type	**Cryptographic Primitive**	**Security Strength (in bits)**		**Is Long-Term Secure?**	**Possible Countermeasures**
		Classical	Quantum		
Symmetric Cryptography	***AES-128***	128	64	No	Use larger symmetric key sizes (with doubled sizes)
	AES-192	192	96	No	
	AES-256	256	128	Yes	
Cryptographic Hash/Digest Functions	***SHA3-224***	112	≈ 74	No	Use larger output digest sizes (with tripled sizes)
	SHA3-256	128	≈ 85	No	
	SHA3-384	192	128	Yes	
	SHA3-512	256	≈ 170	Yes	
	SHAKE128(d)	$\min(\frac{d}{2}, 128)$	$\min(\frac{d}{3}, 128)$	Yes, if $\frac{d}{3} \geq 128$	
	SHAKE256(d)	$\min(\frac{d}{2}, 256)$	$\min(\frac{d}{3}, 256)$	Yes, if $\frac{d}{3} \geq 128$	
Asymmetric Cryptography	***RSA***	$[80, 256]$	≈ 0	No	No direct countermeasures and new solutions are needed, such as (Classical) Post-Quantum Cryptography or Quantum Cryptography
	DH	$[80, 256]$	≈ 0	No	
	DSA	$[80, 256]$	≈ 0	No	
	Elliptic Curve Diffie-Hellman (ECDH)	$[80, 256]$	≈ 0	No	
	Elliptic Curve Digital Signature Algorithm (ECDSA)	$[80, 256]$	≈ 0	No	
	ElGamal	$[80, 256]$	≈ 0	No	

Table 2: The global perspective of the Modern Cryptography for both classical and quantum contexts.

Even if this cryptographic threat does not seem to have immediate repercussions since the current quantum computers are still very limited in terms of processing capacity and noise, the whole scenario changes if we take into consideration the Harvest Now - Decrypt Later (HNDL) paradigm, where an attacker can store information protected by these quantum-vulnerable cryptographic primitives, waiting for a sufficiently powerful quantum computer to appear in the future, to then, be able to "break" such encryption and stole private information, what can lead to critical leakage of confidential information in military and health services, where that information needs to be kept as private for several years.

2 Lattice-based Cryptography

Lattice-based Cryptography is the generic term for cryptographic primitives that use geometric groups, known as lattices, in their construction or security proof. In the current state-of-the-art, we believe these mathematical problems involving lattices are computationally hard to solve efficiently using either classical or quantum computers. For this reason, they are considered prominent candidates for future asymmetric cryptosystems of (Classical) Post-Quantum Cryptography.

Unlike more widely used and known public-key cryptosystems such as the RSA [8], DH [9, 10], El Gamal [11], or even other popular classical public-key schemes of ECC [15, 16] that we currently use daily, which could, theoretically, be defeated using Shor's [4, 5] quantum algorithm for factoring on a sufficiently powerful quantum computer, some lattice-based constructions appear to be quantum-resistant. Namely, we consider many lattice-based cryptosystems to be (asymptotically) and provable secure, under the assumption that it is not known how to solve efficiently some well-studied lattice problems based on their worst-case computational hardness via worst-case-to-average-case security reductions. Furthermore, these asymmetric public-key cryptosystems and cryptographic primitives have robust security proofs based on worst-case hardness, relatively efficient implementations, simplicity, and flexibility.

Namely, we can adapt some proofs, schemes, and techniques of Lattice-based Cryptography for a wide variety of cryptographic functions and primitives, including Asymmetric (Public-Key) Encryption [25, 26], Key Exchange [59, 60, 72–78], Digital Signatures [61–64, 79, 80], Cryptographic (Secure) Hash Functions [81–83], and more complex cryptographic protocols such as the well-desired Homomorphic Encryption [84–86] and ZKPs [87–90]. In particular, Homomorphic Encryption is considered one of the holy grails of cryptography by the scientific community since it allows the direct and correct manipulation of the information of encryption data without decrypting it. On the other hand, while still being optimized, many cryptographic schemes from Lattice-based Cryptography offer practical performance in terms of computation and storage requirements, making them feasible for real-world applications.

As future research directions, Lattice-based Cryptography is also being explored by the scientific community to develop new advanced cryptographic applications, such as Bit Commitment schemes, or Oblivious Transfer (OT) [91, 92] and Secure Multi-Party Computation (SMPC) [93–95] protocols.

2.1 What is a Lattice?

First of all, it is necessary to define what is a lattice geometric group and structure. A lattice is a set of points in m-dimensional space with a periodic "grid-like" geometry. We give an example illustration of a lattice in the Fig. 1:

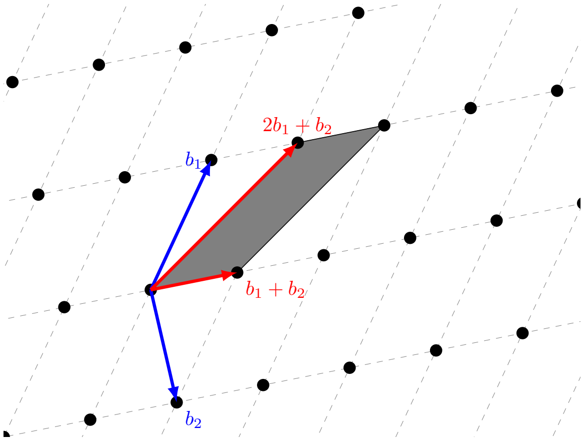

Fig. 1: An illustrative example of a lattice geometric group, with a possible choice of 2 bases and 2 linear combinations of them.

Formally, given n-linearly independent vectors from a basis $\{\boldsymbol{b_1}, \boldsymbol{b_2}, \ldots, \boldsymbol{b_n}\}$ of $\mathbb{R}^n$, a lattice structure $\mathcal{L} \subset \mathbb{R}^n$ generated by them is a linear combination of the basis vectors $\boldsymbol{b}$ and a given set of integer coefficients x, defined as follows:

$$\begin{aligned}\mathcal{L}(\boldsymbol{b_1}, \boldsymbol{b_2}, \ldots, \boldsymbol{b_n}) = & \left\{ \sum_{i=1}^{n} c_i \cdot \boldsymbol{b_i} : x_i \in \mathbb{Z},\ \boldsymbol{b_i} \in \mathbb{R}^n \right\} = \\ & = x_1 \cdot \boldsymbol{b_1} + x_2 \cdot \boldsymbol{b_2} + \cdots + x_n \cdot \boldsymbol{b_n} = \\ & = x_1 \cdot \begin{pmatrix} b_{(1,1)} \\ b_{(1,2)} \\ \vdots \\ b_{(1,n)} \end{pmatrix} + x_2 \cdot \begin{pmatrix} b_{(2,1)} \\ b_{(2,2)} \\ \vdots \\ b_{(2,n)} \end{pmatrix} + \cdots + x_n \cdot \begin{pmatrix} b_{(n,1)} \\ b_{(1,2)} \\ \vdots \\ b_{(n,n)} \end{pmatrix}\end{aligned} \tag{2.1}$$

Indeed, the basis can be represented by the matrix $B = \big(b_1, b_2, \ldots, b_n\big)$, having the basis vectors b_i as their columns, where $1 \leq i \leq n$. Using the matrix notation, a lattice structure generated by a matrix $B \in \mathbb{R}^{(n \times n)}$ can also be defined by $\mathcal{L}(B) = \{\ B \cdot x : B \in \mathbb{R}^{(n \times n)},\ x \in \mathbb{Z}^n\ \}$, using a matrix-multiplication as follows:

$$
\begin{aligned}
\mathcal{L}(B) &= \Big\{\ B \cdot x : B \in \mathbb{R}^{(n \times n)},\ x \in \mathbb{Z}^n\ \Big\} = \\
&= \begin{pmatrix} b_{(1,1)} & b_{(2,1)} & \cdots & b_{(n,1)} \\ b_{(1,2)} & b_{(2,2)} & \cdots & b_{(n,2)} \\ \vdots & \vdots & \ddots & \vdots \\ b_{(1,n)} & b_{(2,n)} & \cdots & b_{(n,n)} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \\
&= \begin{pmatrix} b_{(1,1)} \cdot x_1 + b_{(2,1)} \cdot x_2 + \ldots + b_{(n,1)} \cdot x_n \\ b_{(1,2)} \cdot x_1 + b_{(2,2)} \cdot x_2 + \ldots + b_{(n,2)} \cdot x_n \\ \vdots \quad + \quad \vdots \quad + \ddots + \quad \vdots \\ b_{(1,n)} \cdot x_1 + b_{(2,n)} \cdot x_2 + \ldots + b_{(n,n)} \cdot x_n \end{pmatrix} = \\
&= \begin{pmatrix} b_{(1,1)} \cdot x_1 \\ b_{(1,2)} \cdot x_1 \\ \vdots \\ b_{(1,n)} \cdot x_1 \end{pmatrix} + \begin{pmatrix} b_{(2,1)} \cdot x_2 \\ b_{(2,2)} \cdot x_2 \\ \vdots \\ b_{(2,n)} \cdot x_2 \end{pmatrix} + \cdots + \begin{pmatrix} b_{(n,1)} \cdot x_n \\ b_{(n,2)} \cdot x_n \\ \vdots \\ b_{(n,n)} \cdot x_n \end{pmatrix} = \\
&= \begin{pmatrix} b_{(1,1)} \\ b_{(1,2)} \\ \vdots \\ b_{(1,n)} \end{pmatrix} \cdot x_1 + \begin{pmatrix} b_{(2,1)} \\ b_{(2,2)} \\ \vdots \\ b_{(2,n)} \end{pmatrix} \cdot x_2 + \cdots + \begin{pmatrix} b_{(n,1)} \\ b_{(n,2)} \\ \vdots \\ b_{(n,n)} \end{pmatrix} \cdot x_n = \\
&= x_1 \cdot \begin{pmatrix} b_{(1,1)} \\ b_{(1,2)} \\ \vdots \\ b_{(1,n)} \end{pmatrix} + x_2 \cdot \begin{pmatrix} b_{(2,1)} \\ b_{(2,2)} \\ \vdots \\ b_{(2,n)} \end{pmatrix} + \cdots + x_n \cdot \begin{pmatrix} b_{(n,1)} \\ b_{(1,2)} \\ \vdots \\ b_{(n,n)} \end{pmatrix} = \\
&= \mathcal{L}(\boldsymbol{b_1}, \boldsymbol{b_2}, \ldots, \boldsymbol{b_n}) \text{ (see Equation (2.1))}
\end{aligned}
\tag{2.2}
$$

Given a particular $(n \times n)$ unimodular matrix U, the bases B and $B \cdot U$ generate the same lattice structure. In fact, we have the equality $\mathcal{L}(B) = \mathcal{L}(B')$ if and only if there is a unimodular matrix U such that $B' = BU$. In particular, any lattice structure admits multiple bases, and this mathematical fact represents the core of many lattice-based cryptographic applications and primitives proposed.

The determinant of a lattice structure is given by the absolute value of the determinant of the basis matrix B, that is, $\det(\mathcal{L}(B)) = |\det(B)|$. The value of the determinant is independent of the choice of the basis and geometrically corresponds to the inverse of the density of the points of the lattice structure in $\mathbb{R}^n$. The dual of a lattice structure $\mathcal{L} \in \mathbb{R}^n$ denoted $\mathcal{L}^*$, is the lattice structure given by the set of all column n-vectors $\boldsymbol{y} \in \mathbb{R}^n$ that satisfies $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathbb{Z}$ for all column n-vectors $\boldsymbol{x} \in \mathcal{L}$. Additionally, we can see that for any $B \in \mathbb{R}^n$, $\mathcal{L}(B)^* = \mathcal{L}((B^{-1})^T)$. Therefore, we also obtain the equality $\det(\mathcal{L}^*) = \frac{1}{\det(\mathcal{L})}$.

Other lattice structures that are particularly important in Lattice-based Cryptography are q-ary (modular) lattice groups. These q-ary lattice groups are lattice structures $\mathcal{L}$ satisfying $\mathbb{Z}^n \subseteq \mathcal{L} \subseteq \mathbb{Z}^n$ for some prime integer q. In other words, we determine the membership of a vector $x \in \mathcal{L}$ by $x \mod q$. Then, we define such lattice groups in one-to-one correspondence with linear codes in $\mathbb{Z}_q^n$. Most constructions of Lattice-based Cryptography use q-ary (modular) lattice groups as their hard-on-average computational problem. Note that any lattice structure of integer elements $\mathcal{L} \subseteq \mathbb{Z}^n$ is a q-ary lattice group for some prime integer q. For example, whenever q is an integer multiple of the determinant $\det(\mathcal{L})$. However, we are more concerned with q-ary (modular) lattice groups with a prime integer q much smaller than $\det(\mathcal{L})$ in this specific configuration. Namely, given a matrix $X \in Z_q^{(m \times n)}$ for some integer numbers q, m, and n, we can mathematically define two n-dimensional q-ary lattice structures as follows:

1. $\Lambda_q(X) = \left\{\boldsymbol{y} \in \mathbb{Z}^n : \boldsymbol{y} = X^T \times \boldsymbol{s} \mod q, \text{ for some } \boldsymbol{s} \in \mathbb{Z}^m\right\}$;
2. $\Lambda_q^{\perp}(X) = \{\boldsymbol{y} \in \mathbb{Z}^n : X \times \boldsymbol{y} = \boldsymbol{0} \mod q\}$.

For the first case, we generate a specific q-ary lattice structure from the rows of matrix X. In the second case, a q-ary lattice structure contains all vectors being orthogonal modulo q to the rows of matrix X. In other words, the q-ary lattice structure in the first case corresponds to the linear code generated by the rows of the matrix X. The one from the second case corresponds to the linear code whose parity check matrix is X. It follows since these lattice structures are dual to each other, up to normalization. Namely, we have the following equalities:

1. $\Lambda_q(X) = q \cdot \Lambda_q^{\perp}(X)^*$;
2. $\Lambda_q^{\perp}(X) = q \cdot \Lambda_q(X)^*$.

There are several ways to use lattice structures to build cryptographic primitives for (Classical) Post-Quantum Cryptography that are not always obvious. One first milestone in this line of research is a paper from Miklós Ajtai in 1996 [96,97], which defined the Short Integer Solution (SIS) Problem, and related its average case complexity to the worst-case hardness of finding short vectors in every integer lattice structure, giving novel cryptographic directions, proposing lattice-based One-Way Functions (OWFs) [98] and Trapdoor Functions [63]. Then, in the same year, Miklós Ajtai, jointly with Cynthia Dwork, presented a probabilistic public-key cryptosystem based on the hardness of the Shortest Vector Problem (SVP) [99]. The security proof of this cryptosystem holds unless we can solve the worst-case of a well-known lattice problem in polynomial time.

But before discussing this type of (Classical) Post-Quantum Cryptography in more detail, we first need to describe some mathematical problems that are computationally hard to solve, involving lattice structures at their core.

2.2 Lattice-based Problems

The constructions of cryptographic primitives for Lattice-based Cryptography use the presumed computational hardness of mathematical problems related to lattice structures, on which the SVP instances are the most popular ones and served as the inspiration for other similar lattice-based problems. In this case, we start with a lattice structure represented by an arbitrary basis given as an input for the problem, and the main goal is to output the shortest non-zero vector in it.

Before entering the details of these problems, we need to introduce the notion of the shortest non-zero vector in a lattice structure $\mathcal{L}$ for a metric $\mathcal{M}$, given as:

$$\lambda(\mathcal{L}) = \min_{\hat{\boldsymbol{v}} \, \in \, \mathcal{L} \backslash \{\mathbf{0}\}} ||\hat{\boldsymbol{v}}||_{\mathcal{M}} \tag{2.3}$$

Another important constant associated with a lattice structure $\mathcal{L}$ is the covering radius. We define a covering radius as the smallest radius r for a family of closed spheres (or circles) with a center c_i, where $r > 0$, such that any $x \in R^n$ belong at least to one of those spheres (or circles), covering the entire vector space $\mathcal{V}$. In other words, we can define the covering radius r of a lattice structure $\mathcal{L}$ generated from a basis B as the maximum distance $||(\boldsymbol{x} - \mathcal{L}(B))||_{\mathcal{M}}$, for a given metric $\mathcal{M}$, where the vector $\boldsymbol{x}$ ranges over the linear span of the basis B. Thus, we can define the notion of covering radius r of a lattice structure $\mathcal{L}$ as follows:

$$r = \rho_M(\mathcal{L}(B)) = \max_{\boldsymbol{x} \, \in \, \mathbb{R}^n} ||(\boldsymbol{x} - \mathcal{L}(B))||_{\mathcal{M}} \tag{2.4}$$

Now, we can finally enumerate and briefly describe all the most currently known lattice-based mathematical and computational problems, in their *exact* forms:

1. ***Shortest Vector Problem* (*SVP*)**:

   In the SVP instance, a basis of a vector space $\mathcal{V}$ and a metric $\mathcal{M}$, usually the Euclidean norm L^2, are given for a lattice structure $\mathcal{L}$ of size n as input. Then, the goal is to find the shortest non-zero vector in the space vector $\mathcal{V}$ from the starting point (origin) to another point on the "grid-like" structure using the directions provided by the basis vectors, as measured by $\mathcal{M}$, in the lattice structure $\mathcal{L}$. In other words, the solving algorithm should output a non-zero vector $\boldsymbol{v}$, such that $||\boldsymbol{v}||_{\mathcal{M}} = \lambda(\mathcal{L})$. This mathematical problem in its *exact* form, i.e., without any form of approximation (considering $\gamma(n) = 1$), is only known to be NP-Hard for randomized reductions. By contrast, the corresponding mathematical problem, when considering the use of the uniform norm for the metric $\mathcal{M}$, is also known to be an NP-Hard problem.

2. ***Closest Vector Problem* (*CVP*)**:

   In the CVP instance, a basis of a vector space $\mathcal{V}$ and a metric $\mathcal{M}$, usually the Euclidean norm L^2, are given for a lattice structure $\mathcal{L}$ of size n as input, together with a target vector $\boldsymbol{v}$ in the vector space $\mathcal{V}$ but not necessarily in the lattice structure $\mathcal{L}$. The goal of the solving algorithm is to find the vector $\boldsymbol{v'}$ in the lattice structure $\mathcal{L}$ closest to the given target vector $\boldsymbol{v}$, as measured by the given metric $\mathcal{M}$. It is also known that any computational hardness of an SVP instance implies the same computational hardness for a CVP instance. On the other hand, CVP instances are widely regarded, in theory and practice, as considerably harder problems than SVP instances.

3. ***Shortest Independent Vector Problem*** (***SIVP***):

   In the SIVP instance, a basis of a vector space $\mathcal{V}$ and a metric $\mathcal{M}$, usually the Euclidean norm L^2, are given for a lattice structure $\mathcal{L}$ of size n as input. The goal of the solving algorithm is to find a set (i.e., a basis) of n linearly independent and short non-zero vectors $B' = \{\boldsymbol{b'_1}, \boldsymbol{b'_2}, ..., \boldsymbol{b'_n}\}$, so that $\max_i||\boldsymbol{b'_i}||_{\mathcal{M}} \leq \max_B||\boldsymbol{b_i}||_{\mathcal{M}}$, where $1 \leq i \leq n$ and $B = \{\boldsymbol{b_1}, \boldsymbol{b_2}, ..., \boldsymbol{b_n}\}$. In other words, for the best scenario, we seek to minimize the length of the longest vector in the original basis B and find a new basis B' that yields the same lattice structure $\mathcal{L}$. As far as we know, SIVP instances are also NP-Hard. However, they should generally be harder than SVP instances.

4. ***Shortest Basis Problem*** (***SBP***):

   In the SBP instance, a basis of a vector space $\mathcal{V}$ and a metric $\mathcal{M}$, usually the Euclidean norm L^2, are given for a lattice structure $\mathcal{L}$ of size n as input. The goal is to find an equivalent basis B' that can serve as the new direction vectors for the lattice structure, such that the length of the longest vector in that basis is as short as possible. Essentially, the goal of the solving algorithm is to find the shortest possible set of direction vectors that still allow us to navigate the entire "grid-like" lattice structure. Despite the SBP instances not being explicitly considered NP-Hard problems in general, they are at least considered as hard as approximating the SVP or the CVP instances.

5. ***Covering Radius Problem*** (***CRP***):

   In the CRP instance, a basis of a vector space $\mathcal{V}$, and a metric $\mathcal{M}$, usually the Euclidean norm L^2, are given for a lattice structure $\mathcal{L}$ of size n as input. The goal is to find the smallest radius r', defined as a rational number, such each circle (or sphere for higher-dimensions) of radius r' centered around each point of the lattice structure $\mathcal{L}$ together can cover the entire space of the lattice structure without leaving any gaps. In other words, the goal of the solving algorithm is to find r' such that $r = \rho_{\mathcal{M}}(\mathcal{L}(B)) \leq r$, where $r, r' \in \mathbb{Q}$ and $\rho_{\mathcal{M}}(\mathcal{L}(B))$ is the covering radius ρ using the metric $\mathcal{M}$ that covers the entire space of the lattice structure $\mathcal{L}$ represented by the basis B. Usually, we consider the CRP instances NP-Hard because they involve finding a minimum radius for covering all possible points in a continuous (possibly infinite) space of a lattice structure while ensuring at least one circle (or sphere), centered at a lattice point with that radius, covers them all, which is currently considered to be computationally hard and intensive.

However, practical lattice-based public-key cryptosystems usually involve other approximate versions of the previously mentioned problems, which are outside the regime known to be NP-Hard. In addition, these NP-Hardness results only describe the worst-case asymptotic complexity of the problem, and we do not know how to apply this computational hardness directly to algebraically structured lattices. We can describe these approximate instances as follows:

1. ***γ-Approximate Shortest Vector Problem*** (γ***-Approximate SVP***):

   In the γ-Approximate SVP instance, we consider the setup of the original SVP instance, but with an additional small approximation factor γ that works as a *relaxation* to the original *exact* problem. Namely, we also consider a function defined as $\gamma = \gamma(n) \geq 1$ that depends on the dimension n of the lattice structure $\mathcal{L}$. The goal of the solving algorithm is to find a non-zero lattice vector of a scaled length at most $\gamma \cdot \lambda(L)$, setting a lower bound for its shortest size that is greater than the one from the original *exact* problem.

2. ***γ-Approximate Closest Vector Problem*** (γ***-Approximate CVP***):

   In the γ-Approximate CVP instance, we consider the setup of the original CVP instance, but with an additional small approximation factor γ that works as a *relaxation* to the original *exact* problem. Namely, we also consider a function defined as $\gamma = \gamma(n) \geq 1$ that depends on the dimension n of the lattice structure $\mathcal{L}$. The goal of the solving algorithm is to find the vector $\boldsymbol{v'}$ in the lattice structure $\mathcal{L}$ that is within a factor of γ of the distance to the actual closest vector, as measured by the given metric $\mathcal{M}$, setting a lower bound for the distance to the closest vector that is greater than the one from the *exact* problem. Here, γ is a factor that represents how much further away the approximate solution vector v' can be compared to the actual closest vector on the configuration of the original CVP instance.

3. ***γ-Approximate Shortest Independent Vector Problem***
   (γ***-Approximate SIVP***):

   In the γ-Approximate SIVP instance, we consider the setup of the original SIVP instance, but with an additional small approximation factor γ that works as a *relaxation* to the original *exact* problem. Namely, we also consider a function defined as $\gamma = \gamma(n) \geq 1$ that depends on the dimension n of the lattice structure $\mathcal{L}$. The goal of the solving algorithm is to find a set (i.e., a basis) of n linearly independent and short non-zero vectors $B' = \{\boldsymbol{b'_1}, \boldsymbol{b'_2}, ..., \boldsymbol{b'_n}\}$, so that $\max_i ||\boldsymbol{b'_i}||_{\mathcal{M}} \leq \gamma \cdot \max_B ||\boldsymbol{b_i}||_{\mathcal{M}}$, where $1 \leq i \leq n$ and $B = \{\boldsymbol{b_1}, \boldsymbol{b_2}, ..., \boldsymbol{b_n}\}$. In other words, in the best scenario, we seek to minimize the length of the longest vector in the original basis B and find a new basis B', scaled at most by the given approximation factor γ, that yields the same lattice structure $\mathcal{L}$. Here, γ is a factor representing how much longer the (approximated) linearly independent vectors can be compared to the shortest possible (original) linearly independent vectors.

4. ***γ-Approximate Shortest Basis Problem*** (***γ-Approximate SBP***):

   In the γ-Approximate SBP instance, we consider the setup of the original SBP instance, but with an additional small approximation factor γ that works as a *relaxation* to the original *exact* problem. Namely, we also consider a function defined as $\gamma = \gamma(n) \geq 1$ that depends on the dimension n of the lattice structure $\mathcal{L}$. The goal of the solving algorithm is to find a set (i.e., a basis) of direction vectors, scaled at most by the given approximation factor γ, as measured by the given metric $\mathcal{M}$, that still allow us to navigate the entire "grid-like" lattice structure. Here, γ is a factor representing how much longer the (approximated) solution basis of direction vectors can be compared to the shortest possible (original) basis of direction vectors.

5. ***γ-Approximate Covering Radius Problem (CRP)*** (***γ-Approximate CRP***):

   In the γ-Approximate CRP instance, we consider the setup of the original CRP instance, but with an additional small approximation factor γ that works as a *relaxation* to the original *exact* problem. Namely, we also consider a function defined as $\gamma = \gamma(n) \geq 1$ that depends on the dimension n of the lattice structure $\mathcal{L}$. The goal of the solving algorithm is to find the smallest radius scaled at most by the given approximation factor γ, as measured by the given metric $\mathcal{M}$, such each circle (or sphere for higher-dimensions) with that scaled radius centered around each point of the lattice structure together can cover the entire space of the lattice structure without leaving any gaps. Here, γ is a factor representing how much longer the (approximated) solution radius of the circle or sphere can be compared to the shortest (original) one.

2.3 Algorithms for Lattice-based Problems

Now that we already addressed some of the most popular lattice-based problems used as the basis of Lattice-based Cryptography, we can introduce some of the currently known best algorithms for solving such mathematical problems.

2.3.1 Lattice's Basis Reduction Algorithms

1. ***Lenstra-Lenstra-Lovász*** (***LLL***) ***Algorithm***:

   The Lenstra-Lenstra-Lovasz (LLL) algorithm [100] is a polynomial time algorithm for lattice basis reduction invented jointly by Arjen Lenstra, Hendrik Lenstra, and László Lovász in 1982. Given a basis $B = \{b_1, b_2, ..., b_d\}$ of size d with n-dimensional vectors for a lattice structure $\mathcal{L}$, with $d \leq n$, this reduction algorithm computes a short and nearly orthogonal lattice basis, known as an LLL-reduced lattice basis instance. The algorithm starts by, given the basis B, defining the Gram-Schmidt process orthogonal basis $B^* = \{b_1^*, b_2^*, ..., b_d^*\}$ and the Gram-Schmidt coefficients $\mu_{i,j} = \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle}$[6], for

[6] In this context, $\langle m, n \rangle$ represents the inner product between m and n.

any $1 \leq j < i \leq d$. Then, the basis B is LLL-reduced if there exists a real parameter $\delta \in]0.25, 1]$, such that the following enumerated conditions hold:

(a) For $1 \leq j < i \leq d : |\mu_{i,j}| \leq 0.5$. By definition, this property guarantees the length reduction of the ordered basis. **(Size-reduced condition)**

(b) For $k = 2, 3, ..., d \; : \; \delta \times ||b^*_{(k-1)}||^2 \leq ||b^*_k||^2 + \mu^2_{k,(k-1)} \times ||b^*_{(k-1)}||^2$, where $d \in \mathbb{Z}$, $\delta \in \mathbb{R}$, and $\delta \in]0.25, 1]$. **(Lovász condition)**

Here, estimating the value of the δ parameter, we can conclude how well we are reducing the lattice basis, where greater delta values lead to better reductions of the lattice basis. Initially, Arjen Lenstra, Hendrik Lenstra, and László Lovász demonstrated the LLL-reduction algorithm for $\delta = \frac{3}{4}$. It is important to note that although LLL-reduction is well-defined for $\delta = 1$, the LLL algorithm only guarantees a polynomial-time complexity for $\delta \in]0.25, 1[$.

In this context, the Gram-Schmidt orthogonalized vectors b^*_j are defined as:

$$b^*_j = b_j \; - \sum_{i=1}^{(j-1)} \mu_{j,i} \times b^*_i, \text{ for } j = 2, ..., d, \text{ and where } b^*_1 = b_1$$

There is no known efficient algorithm to compute a lattice basis in which the basis vectors are as short as possible for lattice structures of dimensions greater than 4. However, an LLL-reduced lattice basis is nearly as short as possible, in the sense that there are absolute bounds $c_i > 1$ such that the i^{th} reduced basis vector is no more than c_i times as long as the respective shortest i^{th} basis vector in the lattice structure, with $1 \leq i \leq d$ and $d \in \mathbb{Z}$.
We can describe the general LLL algorithm by defining also two auxiliary sub-routines, the Nearest Plane Algorithm [101] and the Lattice Basis Size Reduction Algorithm [100]. After correctly defining these two sub-routines, it is easier to build the general LLL algorithm since it uses them both.
These two sub-routines jointly select the hyperplane closest to the target vector $c = \left\lfloor \frac{\langle t, b^*_n \rangle}{||b^*_n||^2} \right\rceil$, and recursively search for a lattice point in $c \times b_k + \mathcal{L}(B')$ close to the target vector t, or equivalently, a lattice point in the lower dimensional sub-lattice $\mathcal{L}(B')$ close to $(t - c \times b_k)$. The base case is when the rank of the lattice is reduced to 0 and the only possible output is also 0.
The LLL algorithm for lattice basis reduction, calculates a nearly orthogonal reduced lattice basis (also called LLL-reduced basis) in a complexity time of $\mathcal{O}\left(d^5 \times n \times \log^3(L)\right)$, where L is the largest length of b_i under the Euclidean norm, that is, $L = \max\left(||b_i||_2\right) = \max\left(||b_1||_2, ..., ||b_d||_2\right)$, for $1 \leq i \leq d$.
Additionally, the LLL algorithm has also found numerous other applications in cryptanalysis of public-key encryption schemes and cryptosystems, such as Naccache–Stern Knapsack [102], RSA [8] (with particular settings), or Nth Degree Truncated Polynomial Ring Units - Encrypt (NTRUEncrypt) [26].

The pseudo-code for the sub-routine of the Nearest Plane Algorithm that, given an input lattice basis B and a target vector t, outputs a lattice point $v \in \mathcal{L}(B)$ such that $\frac{\langle t-v, b_i^* \rangle}{||b_i^*||^2} \in \left[-\frac{1}{2}, \frac{1}{2}\right[$ for all $i = 1, ..., d$, is given as follows:

Sub-routine 1 `NearestPlane`: Nearest Plane Algorithm

```
Input: $(B, t)$
Output: $v$
Ensure: $B \in \mathbb{R}^{n \times d},\ t \in \mathbb{R}^n,\ v \in \mathcal{L}(B)$

if $\mathrm{n} = 0$ then
    return 0
else
    $B^* \leftarrow$ GramSchmidt($B$)
    $c \leftarrow \left\lfloor \frac{\langle t, b_n^* \rangle}{||b_n^*||^2} \right\rceil$
    return $c \times b_n +$ NearestPlane$\big([b_1, ..., b_{(d-1)}], t - (c \times b_d)\big)$
end if
```

The pseudo-code for the sub-routine of the Lattice Basis Size Reduction Algorithm that, given an input lattice basis B, outputs the same lattice basis B but with its size reduced, is given in detail as presented below:

Sub-routine 2 `SizeReduction`: Lattice Basis Size Reduction Algorithm

```
Input: $B$
Output: $B$
Ensure: $B \in \mathbb{R}^{n \times d}$

for $i = 2$ to $n$ do
    $x \leftarrow$ NearestPlane($B, b_i - b_i^*$)
    $b_i \leftarrow b_i - B \times x$
end for
return $B$
```

In the context of the LLL algorithm for lattice basis reduction, we need to define $\pi_j(b)$ as the projection of the bases vector b onto the orthogonal complement of the sub-space spanned by the first $(j-1)$ basis vectors. This projection is crucial for the `SizeReduction` step and to maintain the Lovasz condition, which together ensure the generation of a reduced and nearly orthogonal lattice basis. The projection $\pi_j(b)$ is given by removing the components of b that lie in the direction of $b^*_{[1,(j-1)]} = (b_1^*, ..., b^*_{(j-1)})$, and therefore, we can defined it mathematically as presented below:

$$\pi_j(b) = b \ - \sum_{i=1}^{(j-1)} \mu_{j,i} \times b_i^*$$

The pseudo-code for the sub-routine of the LLL Basis Reduction Algorithm that, given an input lattice basis B and a real parameter δ, outputs the same lattice basis B but reduced by the LLL algorithm, is presented below:

Algorithm 1 `LLL`: LLL Lattice Basis Reduction Algorithm

```
Input: $(B, \delta)$
Output: $B$
Ensure: $B \in \mathbb{R}^{n \times d},\ \delta \in \mathbb{R},\ \delta \in \,]0.25, 1]$

$B \leftarrow$ SizeReduction($B$)
for $i = 1$ to $(d - 1)$ do
    if $\delta \times ||\pi_i(b_i)||^2 > ||\pi_i(b_{(i+1)})||^2$ then
        swap($b_i, b_{(i+1)}$)
        return LLL($B, \delta$)
    else
        return $B$
    end if
end for
```

2. ***Block Korkine-Zolotarev*** (***BKZ***) ***Algorithm***:

   Before describing the Block Korkine-Zolotarev (BKZ) algorithm [103] for lattice basis reduction, we need to introduce the Korkine-Zolotarev (KZ) algorithm [104], which is also known as Hermite–Korkine–Zolotarev (HKZ) algorithm. For lattice structures defined in R^n, the KZ algorithm yields a lattice basis with orthogonality defect at most n^n steps, unlike the $2^{\left(\frac{n^2}{2}\right)}$ bound for the steps of the LLL algorithm [100]. The KZ algorithm [104] has an exponential complexity when compared to the polynomial complexity of the LLL algorithm. However, we may prefer the KZ algorithm for solving multiple CVP instances in the same lattice structure, where it can be more efficient. First, Aleksandr Korkine and Yegor Zolotarev defined a KZ-reduced lattice basis in 1877 as a strengthened version of the Hermite-reduction [104]. Later, in 1983, Ravi Kannan proposed the first algorithm to construct a KZ-reduced lattice basis [105]. Then, Masaya Yasuda introduced the BKZ algorithm in 1987 [103], and improved later by Claus-Peter Schnorr and Martin Euchner in 1991 [106], as some refined versions of the algorithm.

   Namely, the BKZ algorithm provides a more scalable and practical approach for lattice basis reduction by generalizing the (original) KZ algorithm to blocks of vectors instead of single ones. The BKZ algorithm applies the KZ algorithm to each block of vectors, thereby achieving stronger lattice basis reductions than the LLL algorithm while being more efficient than a (full) lattice basis reduction from the KZ algorithm. The quality of the lattice basis reduction depends on the block size parameter, usually denoted as β. Namely, β offers a trade-off between accuracy and performance in a sense that larger block sizes yield better reductions but require more computational effort.

There are several variants of the BKZ algorithm for lattice basis reduction, but in this report, we will follow the one given by Claus-Peter Schnorr and Martin Euchner [106]. The pseudo-code of this BKZ algorithm variant is:

Algorithm 2 `BKZ`: BKZ Lattice Basis Reduction Algorithm
(Schnorr-Euchner's variant)

Input: (B, β, γ)
Output: B
Ensure: $B \in \mathbb{R}^{n \times d},\ \beta \in \mathbb{Z},\ \beta \in [2, d],\ \gamma \in \mathbb{R},\ \gamma \in [1, 2]$

```
1:  $z \leftarrow 0$
2:  $j \leftarrow 0$
3:  $B \leftarrow \texttt{LLL}\left(B, \frac{1}{\gamma}\right)$
4:  while $z < (d-1)$ do
5:      $j \leftarrow (j \mod (d-1)) + 1$
6:      $d_j \leftarrow \min((j + \beta - 1), d)$
7:      $h \leftarrow \min((j + \beta), d)$
8:      $\mathrm{A} = (\alpha_j, ..., \alpha_{d_j}) \leftarrow \texttt{Enumerate}\left(\mathcal{L}(B_{[j,d_j]})\right)$
9:      if $||\pi_j(b)|| = \lambda_1\left(L(B_{[j,d_j]})\right)$ then
10:         $b = \sum_{i=j}^{d_j} \alpha_i \times b_i$
11:     end if
12:     if $||b_j^*||^2 > \gamma \times ||\pi_j(b)||^2$ then
13:         $z \leftarrow 0$
14:         $B' \leftarrow B_{[1,(j-1)]} \cup B \cup B_{[j,h]} = (b_1, ..., b_{(j-1)}, b, b_j, ..., b_h)$
15:         $B \leftarrow \texttt{LLL}\left(B', \frac{1}{\gamma}\right)$
16:     else
17:         $z \leftarrow (z + 1)$
18:         $B'' \leftarrow B_{[1,h]} = (b_1, ..., b_h)$
19:         $B \leftarrow \texttt{LLL}(B'', 0.99)$
20:     end if
21: end while
22: return $B$
```

Here, the `Enumerate` step in the BKZ algorithm for lattice basis reduction is a local search procedure within a block of lattice basis vectors, aiming to find the shorter lattice basis vectors by systematically exploring combinations of the former ones. This intermediate enumeration step is computationally demanding but crucial for achieving high-quality lattice basis reduction.

Namely, the `Enumerate` step aims to find a shorter lattice basis vector $\mathbb{A} = (\alpha_j, ..., \alpha_{d_j}) \in \mathbb{Z}^{(d_j - j + 1)}$ for $\mathcal{L}\left(B_{[j,d_j]}\right)$. Additionally, $\lambda_1\left(L(B_{[j,d_j]})\right)$ represents the length of the shortest non-zero lattice basis vector in a lattice structure and a new bases vector $b = \sum_{i=j}^{d_j} \alpha_i \times b_i$ is computed such that $||\pi_j(b)|| = \lambda_1\left(L(B_{[j,d_j]})\right)$. Right after the step 20 of the pseudo-code for the BKZ algorithm, $B_{[jmd_j]}$ may no longer be γ-SVP-reduced due to the calls to the LLL algorithm for lattice basis reduction at the steps 15 and 19. Finally, it is "folklore" in practice to allow a relaxation factor $\gamma = 1$, and it is recommended to run the LLL algorithm with $\frac{1}{\gamma} = 0.99$ at steps 3 and 15.

2.3.2 Shortest Vector Problem (SVP) Algorithms

Regarding the specific algorithms for solving the Shortest Vector Problem (SVP), there are two general types of algorithms: the Exact Algorithms to find exact solutions and the Approximation Algorithms to find approximated solutions.

For the Exact Algorithms, there are three main approaches to the exact solution of SVP instances in d-dimensional lattice structures, enumerated as follows:

1. ***Enumeration Algorithms***:
   The Enumeration Algorithms for lattices [107–111] are methods to list all the elements of a lattice structure or to find certain elements that meet specific criteria defined *a priori*. The common proceeding of an Enumeration Algorithm for lattice structures starts with a pre-processing step, usually through a lattice basis reduction algorithm, such as the LLL algorithm, to make the lattice basis vectors shorter and closer to orthogonal, followed by the setting up of the initial parameters for a (geometric) search space and then starts the enumeration from the origin or a specified starting point, using a search or optimization algorithm to explore the points of the lattice structure, such as Depth-First Search (DFS) and Breadth-First Search (BFS), or even Branch-and-Bound algorithms. At each step of the search or optimization, the algorithm performs pruning using geometric insights based on the search space to eliminate infeasible regions or calculating bounds to determine if further exploration along a branch is necessary. Finally, the algorithm stores or outputs the points of the lattice structure that meet the criteria (e.g., points within a certain distance from the origin/starting point). These Enumeration Algorithms for lattice structures have super-exponential running time $d^{O(d)}$ but have the advantage of being deterministic and using only polynomial space. These algorithms are standard techniques to solve SVP and CVP instances on arbitrary lattice structures by systematically enumerating all lattice points in a bounded region of space (typically a d-dimensional parallelepiped or ellipsoid). Interest in Enumeration techniques for lattice structures is due to their low memory requirements (linear in the lattice structures of dimension d) and excellent practical performance in moderately low dimensions. Within the class of polynomial space algorithms, enumeration methods currently provide the asymptotically fastest algorithms to find exact solutions to SVP and CVP, both in theory and practice, with a worst-case time complexity of $d^{O(d)}$. Sieving Algorithms [112–115] and Voronoi Cell Computation Algorithms [116–119] provide theoretically faster solutions to SVP and CVP instances but at the cost of using exponential space and occasionally randomness. Moreover, several optimizations and heuristics have been developed over the years, making this approach the most attractive in practice for moderately small dimensions.

2. ***Voronoi Cell Computation Algorithms***:

   The Voronoi Cell Computation Algorithms [116–119] for lattice structures involve determining the regions of a vector space closest to each point of the lattice structure. The Voronoi Cell of a lattice point is the set of all points in the vector space closer to that lattice point than any other point of the same lattice structure. Generally, there are two main approaches for this type of algorithm: Direct Geometric Construction (via Voronoi Diagram Generation [120]) and Delaunay Triangulation [121] (using dual lattice structures).

   The Direct Voronoi Cell Computation Algorithms via Voronoi Diagrams usually start with a set of seed points on the lattice structure, and for each pair of seed points, compute the perpendicular bisector (a line equidistant from the two points for two-dimensional spaces and a hyperplane for higher dimensional spaces). For each of the perpendicular bisectors, the algorithm computes the respective intersections to find the vertices of the Voronoi Cells. Finally, those vertices are connected to form the edges and faces of the Voronoi Cells. Each Voronoi Cell corresponds to one seed point and contains all points of the lattice structure closer to it than any other seed point. This approach may require handling the boundaries of the vector space appropriately. For an infinite lattice structure, it might be practical to consider periodic boundary conditions or a large enough bounding box.

   On the other hand, the Indirect Voronoi Cell Computation Algorithms via Delaunay Triangulation usually also start with a set of seed points on the lattice structure, and for the set of seed points, construct the Delaunay Triangulation. Here, the Delaunay Triangulation means forming triangles such that no lattice point is inside the circumcircle of any triangle for a two-dimensional space or forming simplices such that no lattice point is inside the sphere of any simplex for a higher-dimensional space. Next, for each simplex in the Delaunay Triangulation, the algorithm computes its circumcenter (i.e., the center of the circumcircle or sphere). The algorithm uses these circumcenters as vertices of the Voronoi Cells. The edges and faces of the Voronoi Cells correspond to the adjacency of the simplices in the Delaunay Triangulation. Then, the algorithm connects the circumcenters to form the edges and faces of the Voronoi Cells. Each Voronoi Cell corresponds to one seed point in the lattice structure. Similar to the Direct Voronoi Cell Computation Algorithms, handling boundaries might be appropriate.

   This approach improves the running time for solving SVP instances and many other lattice problems to $O(2^{2^d})$, using exponential space $O(2^n)$. These are currently the asymptotically fastest known deterministic algorithms for solving SVP, but they are not competitive in practice with other approaches.

3. ***Sieving Algorithms***:

   The Sieving Algorithms [112–115] for lattice groups are advanced techniques that work by iteratively refining a set of candidate lattice vectors. This refining procedure is done by sieving out longer vectors and keeping shorter ones until finding the optimal vector in the lattice structure. These algorithms usually start by reducing the lattice basis using techniques like

LLL or BKZ algorithms to turn the lattice basis vectors shorter and more orthogonal, simplifying the subsequent sieving process. These algorithms then generate a set of random lattice vectors through random combinations of the lattice basis vectors or by sampling from a Gaussian distribution centered at the origin. Next, for each pair of random lattice vectors, these Sieving Algorithms compute a new difference vector between those vectors, and if this new random lattice vector is shorter than both previous vectors, replace one of those vectors with the new one. These algorithms perform this pairwise reduction process iteratively, and during each iteration, we expect the set of random lattice vectors to contain shorter vectors as the reductions progress. This step pairwise reduction is executed while checking convergence by monitoring the length of the shortest vector in the set. If the length stops decreasing significantly between iterations, the algorithm has likely converged to a solution. We can optimize these algorithms by restricting the set of random lattice vectors to lie within a specific bounding volume, reducing the number of vectors considered at each step, speeding up the convergence, and parallelizing the pairwise reduction step. Additionally, we can employ hashing and compression techniques to quickly find and compare lattice vectors of high dimensions in the pairwise reduction step.
These Sieving Algorithms improve the asymptotic running time of traditional Enumeration Algorithms, reducing the dependency of the running time on the dimension of the lattice structure from $d^{O(d)}$ to single exponential $2^{(c \times d)}$ for some constant $c = O(1)$. These algorithms achieve this improvement at the cost of introducing randomization processes and using exponential space.

The Approximation Algorithms for SVP instances are usually based on lattice basis reduction techniques, such as the LLL and BKZ algorithms, and achieve approximation factors for the solution as small as $2^{O(\log(d)/\log(\log(d)))} = d^{O(1)}$.

2.3.3 Closest Vector Problem (CVP) Algorithms

Regarding the specific algorithms for solving the Closest Vector Problem (CVP), there are two main approaches: the Enumeration Algorithms to find exact solutions and Babai's Nearest Plane Algorithm to find approximated solutions.

1. ***Enumeration Algorithms***:

   The Enumeration Algorithms [107–111] for CVP instances are very similar to the ones for solving SVP instances. These algorithms list all the elements of a lattice structure or find certain elements that meet criteria defined *a priori*. The standard procedure involves a pre-processing step, through a lattice basis reduction, a setup of the initial parameters for a search space, and a search or optimization step from the origin or a starting point. At the end, the algorithm stores/outputs the points of the lattice structure meeting the criteria (e.g., points within a certain distance from the starting point).
   These Enumeration Algorithms for lattice structures have super-exponential running time $d^{O(d)}$ but have the advantage of being deterministic and using

only polynomial space. These algorithms are standard techniques to solve CVP instances on arbitrary lattice structures by systematically enumerating all lattice points in a bounded region of space. Within the class of polynomial space algorithms, enumeration methods currently provide the asymptotically fastest algorithms to find exact solutions to CVP, with a worst-case time complexity of $d^{O(d)}$. Other algorithms provide theoretically faster solutions to CVP instances but at the cost of using exponential space and randomness.

2. ***Babai's Nearest Plane Algorithm***:

   The Babai's Nearest Plane Algorithm [101] (also known as Babai's Nearest Hyperplane Algorithm) is a technique used to find an approximate solution to CVP instances in lattice-based theory, introduced by László Babai in 1986. This algorithm leverages the Gram-Schmidt Orthogonalization process to simplify the problem and provide an approximate solution to the problem. The algorithm receives the lattice basis and a target point in the vector space as inputs. Initially, the algorithm commonly starts by performing some pre-processing through the Gram-Schmidt procedure on the basis vectors to obtain an orthogonal basis. Next, the algorithm projects the target point onto the orthogonal vector of each of the lattice basis vectors, starting from the last one. Then, the algorithm determines the nearest integer coefficients for all lattice bases iteratively, such the product between that coefficient and the respective basis vector is close to the projection computed before, subtracting that multiplication result from the target vector and updating its value. Finally, the Babai's Nearest Plane Algorithm uses all the integer coefficients obtained through the previous iterative projection step to construct the approximate closest vector in the lattice "grid-like" structure. In summary, this algorithm systematically approximates the closest point on a lattice structure to a given target point by orthogonalizing the "grid-like" structure and iteratively adjusting the target point using projections.

2.3.4 Quantum Algorithms

A different approach for solving lattice-based problems is to leverage some of the principles and fundamentals of Quantum Computing to develop Quantum Algorithms that potentially provide significant speedups over classical ones.

1. ***Quantum Lattice Basis Reduction Algorithms***:

   The Quantum Lattice Basis Reduction Algorithms [122–125] are general quantum algorithms that aim to perform the lattice basis reduction using quantum properties, potentially offering improvements upon the (classical) lattice basis reduction algorithms like LLL and BKZ algorithms. These quantum algorithms seek to exploit quantum parallelism and entanglement to enhance the efficiency and effectiveness of the lattice basis reduction.

2. ***Quantum Sieving Algorithms***:

   The Quantum Sieving Algorithms [126–128] aim to solve SVP instances in a lattice structure, adapting the previously mentioned (classical) Sieving techniques but using quantum search, such as Grover's Algorithm [1], to speed up the search steps for short lattice vectors. Thus, this quantum variant solves SVP instances more efficiently than (classical) Sieving Algorithms.

3. ***Quantum (Babai's) Nearest Plane Algorithms***:

   The Quantum (Babai's) Nearest Plane Algorithm [129, 130] is the quantum adaptation of the (classical) Babai's Nearest Plane Algorithm previously mentioned, which we can also use to solve CVP instances in a lattice. Namely, this quantum variant uses some quantum techniques to project the lattice vectors onto planes (or hyperplanes) of the lattice structure more efficiently.

4. ***Quantum Walks Lattice Search Algorithms***:

   The Quantum Walk Lattice Search Algorithms [128, 131] are algorithms that explore the lattice structures using the quantum analogs of (classical) Random Walks [132], known as Quantum Walks [133]. In particular, these Quantum Walks can provide a quadratic speedup in finding the shortest vector in a lattice structure, when compared to (classical) Random Walks.

5. ***Quantum Annealing Lattice Optimization Algorithms***:

   The Quantum Annealing Lattice Optimization Algorithms [134–136] are quantum algorithms that use the principles of Quantum Annealing to find low-energy quantum states of a quantum physical system that correspond to short lattice vectors in a lattice structure. We can use this approach to solve SVP and CVP instances by mapping these lattice-based problems onto an optimization framework suitable for Quantum Annealers. Another similar approach is Quantum Simulated Annealing Lattice Optimization Algorithms that combine the concepts of Simulated Annealing with the fundamentals of Quantum Computing to explore the lattice structures more efficiently.

2.3.5 Hybrid Algorithms

Alternatively, we can combine some of the previously mentioned algorithms and techniques for solving lattice-based problems, obtaining Hybrid Algorithms.

1. ***Progressive BKZ Algorithm***:

   The Progressive BKZ Algorithm [137–140] is an example of a Hybrid Algorithm, which is a combination of lattice-basis reduction and sieving algorithms. Typically, this hybrid algorithm starts with a lattice-basis reduction using the BKZ Algorithm and then applies sieving techniques to solve SVP instances more efficiently. This algorithm typically has an exponential complexity, but we can make it significantly faster due to the combination of both lattice-basis reduction and sieving methods.

2. ***Recursive BKZ Algorithm***:

   The Recursive BKZ Algorithm [138, 141] recursively applies the BKZ Algorithm for lattice-basis reduction to smaller sub-problems within the lattice group. By breaking down the reduction process into smaller and more manageable steps, the Recursive BKZ Algorithm achieves better overall lattice-basis reduction results. In particular, this algorithm is advantageous for very high-dimensional lattices where direct application of the BKZ Algorithm for large blocks of lattice basis vectors would be prohibitive.

3. ***Self-Dual BKZ Algorithm***:

   The Self-Dual BKZ Algorithm [142] is a variant of the BKZ Algorithm that simultaneously reduces both primal and dual lattice bases. This dual reduction method aids in achieving better approximations of the shortest vector and improves the overall quality of a lattice basis reduction. This Hybrid Algorithm can be practical in some cryptographic contexts and scenarios, for which both primal and dual lattice structures are relevant.

4. ***Tuple Lattice Sieving***:

   The Tuple Lattice Sieving Algorithm [143] is another Hybrid Algorithm since it combines the advantages of the Sieving and Enumeration Algorithms previously mentioned. Namely, this algorithm uses Sieving techniques to generate many random lattice basis short vectors and an Enumeration method to combine these same vectors to find even shorter ones. This Hybrid Algorithm allows us to solve SVP and CVP instances more efficiently. It might be helpful for cryptanalysis tasks that require finding very short lattice basis vectors in high-dimensional lattice "grid-like" structures.

5. ***Hybrid Quantum-Classical Algorithms***:

   The Hybrid Quantum-Classical Algorithms [127, 144] for lattice-based problems combine the strengths of both Classical and Quantum Computing to achieve better performance than using either approach alone. One example of such a Hybrid Quantum-Classical Algorithm is the Quantum-Enhanced LLL/BKZ Algorithm, which uses quantum algorithms, like Quantum Sieving [126–128], to accelerate the search for lattice short vectors within the blocks of lattice basis vectors while also using (classical) reduction techniques like LLL [100] or BKZ [103, 106] Algorithms to handle the overall reduction. Another approach could consist of using (classical) lattice-basis reduction, such as the LLL [100] or BKZ [103,106] Algorithms, to pre-process the lattice structure and reduce its dimension, applying then quantum algorithms for search or optimization based, for example, on Grover's Algorithm [1] or Quantum Annealing [134–136] to the reduced lattice-based problem like SVP and CVP instances, leveraging quantum speedups in lower dimensions.

3 Learning With Errors (LWE) Problem

Now that we already presented the concepts of Lattice-based Cryptography and the mathematical Lattice-based Problems related to it, we can introduce other slightly different techniques for building variants of cryptographic primitives and cryptosystems related to lattice structures as well. One of these techniques relies on a mathematical problem known as the Learning With Errors (LWE) Problem [27, 145]. The LWE problem is a lattice-based problem we can use to create secure (classical) post-quantum cryptographic primitives. The main idea of this lattice-based problem is to represent secret information as a set of mathematical equations with errors (or noise). In other words, the LWE Problem represents a way of hiding the value of a secret by introducing noise (on purpose).

Despite not seeming to be a mathematical problem related to lattice structures at first sight, we can also view the LWE Problem as the mathematical problem of finding a point in a high-dimensional lattice structure given some "noisy" information about it. More specifically, we can think of each one of these mathematical equations in the LWE Problem as representing a lattice point in a high-dimensional space, slightly perturbed by an error (noise) vector.

Currently, we conjecture (and believe) the LWE Problem to be a problem computationally hard to solve in both classical and quantum contexts. For this reason, we see this mathematical problem as particularly useful and popular for finding cryptographic candidates for (Classical) Post-Quantum Cryptography.

In simple terms, we can think of the (noisy) mathematical equations involved in the definition of the LWE Problem as having the general form presented below:

$$A \times s + e = t,\ A \in \mathbb{Z}^{(d \times n)},\ s \in \mathbb{Z}^{(n \times 1)},\ e,\ t \in \mathbb{Z}^{(d \times 1)}$$

Here, A is a $d \times n$ algebraic matrix containing the lattice basis vectors and s is a n-column vector, while e, and t are d-column vectors. Furthermore, the matrix A and the column vector t are public, while the column vectors s and e are private. Notice that only by knowing the values of the matrix A and the (target) lattice vector t, we cannot infer the values of the column vector s, which is a secret.

We can build a simple cryptosystem from this mathematical problem, using the matrix A and the target lattice vector t as a public cryptographic key pair.

Then, using the public key pair (A, t), we can define the encryption process on a given plaintext message m and the resulting ciphertext pair (u, v) as follows:

$$u = A \cdot x = A^T \times x,\ A^T \in \mathbb{Z}^{(n \times d)},\ x \in \mathbb{Z}^{(d \times 1)},\ u \in \mathbb{Z}^{(n \times 1)}$$
$$v = t \cdot x + m = t^T \times x + m,\ t^T \in \mathbb{Z}^{(1 \times d)},\ x \in \mathbb{Z}^{(d \times 1)},\ m,\ v \in \mathbb{Z}^{(\ell \times 1)}$$

Here, the ciphertext is composed of two column vectors, u and v. Namely, u is a n-column vector and v is a column vector of the same size ℓ of the plaintext message m. Finally, x is a random d-column vector generated for the encryption.

Regarding the decryption process on a given ciphertext pair (u, v) to recover the plaintext message m using the secret s, we can define it mathematically as:

$$z = u \cdot s = u^T \times s,\ u^T \in \mathbb{Z}^{(1\times n)},\ s \in \mathbb{Z}^{(n\times 1)},\ z \in \mathbb{Z}$$

$$\begin{aligned}
v' &= v - z = t \cdot x + m - u \cdot s = t^T \times x + m - u^T \times s = \\
&= (A \times s + e)^T \times x + m - (A \cdot x)^T \times s = \\
&= \left((A \times s)^T + e^T\right) \times x + m - \left(A^T \times x\right)^T \times s = \\
&= (A \times s)^T \times x + e^T \times x + m - x^T \times \left(A^T\right)^T \times s = \\
&= s^T \times A^T \times x + e^T \times x + m - x^T \times A \times s = \\
&= \left(x^T \times A \times s\right)^{T\,[7]} + e^T \times x + m - x^T \times A \times s = \\
&= x^T \times A \times s + e^T \times x + m - x^T \times A \times s = \\
&= e^T \times x + m,\ x^T \in \mathbb{Z}^{(1\times d)},\ A \in \mathbb{Z}^{(d\times n)},\ s \in \mathbb{Z}^{(n\times 1)}, \\
&\quad e^T \in \mathbb{Z}^{(1\times d)},\ x \in \mathbb{Z}^{(d\times 1)},\ m \in \mathbb{Z}^{(\ell\times 1)}
\end{aligned}$$

$$\begin{gathered}
m = v' - e^T \times x = e^T \times x + m - e^T \times x = m, \\
e^T \in \mathbb{Z}^{(1\times d)},\ x \in \mathbb{Z}^{(d\times 1)},\ m \in \mathbb{Z}^{(\ell\times 1)}
\end{gathered}$$

Notice that, for the receiver party to be able to decrypt the ciphertext pair (u, v) and recover m, it should be able to use the same (or approximate) error (noise) vector e and random vector x that the sender party receiver used to encrypt the original plaintext message m. Furthermore, they should also use the same secret vector s in both encryption and decryption processes that should be securely exchanged between them at some point in the cryptosystem execution.

The error (noise) vector e is typically randomly or pseudo-randomly sampled from a discrete Gaussian or (ideally) Uniform distribution with small values.

Finally, using a modulo q in the vectors and matrices is very common in the setup of most cryptosystems based on the LWE Problem. Using a modulo q in such asymmetric cryptosystems provides a balanced approach to security, efficiency, and practicality. In these cases, each numerical component of the algebraic vectors and matrices used for the mathematical operations of these new public-key cryptosystems are defined in $\mathbb{Z}_q$ rather than simply in $\mathbb{Z}$. On the other hand, using floor functions when we define these cryptographic primitives can also be beneficial to achieve efficient encoding and decoding of information during the encryption and decryption procedures, managing and reducing the impact of errors while ensuring that the values stay within the correct range.

[7] The result of $x^T \times A \times s$ is a single scalar since it is a multiplication between a $(1 \times d)$ row vector, a $(d \times n)$ algebraic matrix, and $(n \times 1)$ column vector. Since the transpose of a scalar is the scalar itself, we have the equality $\left(x^T \times A \times s\right)^T = x^T \times A \times s$.

3.1 Ring Learning With Errors (RLWE) Problem

Another popular variant of the LWE Problem and the related cryptographic primitives we can build from is known as the Ring Learning With Errors (RLWE) Problem [72, 145]. This variant works within the structure of polynomial rings, providing a more algebraic structure that we can exploit more efficiently.

For a general definition of the RLWE Problem, we need to slightly redefine the mathematical formulation of the original LWE Problem as presented below:

– Given an algebraic ring structure $\mathcal{R}$ and polynomial elements $a(x), t(x) \in \mathcal{R}$, find a secret polynomial $s(x) \in \mathcal{R}$, such that the following equality holds:

$$a(x) \times s(x) + e(x) = t(x)$$

Similar to the LWE Problem, $e(x)$ represents a small error (noise) polynomial. Here, if we consider a modulo q and a (modulo) quotient function $f(x)$, we know that $\mathcal{R}$ is a polynomial quotient ring and is defined as $\mathcal{R} = \mathbb{Z}_q[x]/f(x)$. Usually, $f(x)$ is a fixed polynomial, typically chosen to be irreducible. Namely, the modulo q reduces the coefficients of the polynomials in the ring $\mathcal{R}$ to lie within the set of integers modulo q, and therefore, we have $\mathbb{Z}_q = \{0, 1, ..., (q-1)\}$. On the other hand, the quotient function $f(x)$ refers to considering the ring of polynomials with coefficients in $\mathbb{Z}_q$ modulo the ideal subset generated by a specific polynomial $f(x)$, which is the subset of all multiples of $f(x)$ in $\mathbb{Z}_q[x]$. For this reason, the quotient ring $\mathcal{R} = \mathbb{Z}_q[x]/f(x)$ has all the respective polynomial operations performed modulo both q and $f(x)$. In the context of the RLWE Problem, $a(x)$ is a public polynomial, $s(x)$ is a secret (and private) polynomial, $e(x)$ is an error (noise) polynomial, and $t(x)$ is the public (target) polynomial. As obvious, these polynomials are all defined in the algebraic ring $\mathcal{R} = \mathbb{Z}_q[x]/f(x)$.

3.2 Module Learning With Errors (MLWE) Problem

There is another variant of these problems we can use to build lattice-based cryptographic primitives, known as the Module Learning With Errors (MLWE) Problem [146,147], which is related to the previously mentioned LWE and RLWE Problems. The MLWE Problem generalizes both LWE and RLWE Problems by working with modules over rings, providing a flexible framework that balances the efficiency of the RLWE Problem with the generality of the LWE Problem.

For a general definition of the MLWE Problem, we need to redefine and adapt the mathematical formulations of both LWE and RLWE Problems as follows:

– Given an algebraic module M over an algebraic ring structure $\mathcal{R}$, a public algebraic matrix $A \in M^{(d \times n)}$, with entry values $A_{i,j} \in \mathcal{R}$, with $1 \leq i \leq d$, $1 \leq j \leq n$, and $d \geq n$, as well as a public target vector $t \in M^d$, find a (private) secret vector $s \in M^n$, such that the following equality holds:

$$A \times s + e = t$$

In this context, the mathematical module M over the algebraic ring $\mathcal{R}$ is a generalization of a vector space where the scalars come from $\mathcal{R}$ instead of a field.

Similar to the LWE and RLWE Problems, e is a small error (noise) vector in the module M. Usually, we can also consider a modulo q and a (modulo) quotient $f(x)$ to define the algebraic ring $\mathcal{R}$ we are working with. In that case, we know the algebraic ring $\mathcal{R}$ is a quotient ring defined as $\mathcal{R} = \mathbb{Z}_q[x]/f(x)$, similar to the one used in the RLWE Problem, but with a higher degree of freedom in terms of the module structure. Once again, $f(x)$ is a fixed polynomial, typically chosen to be irreducible. Namely, the modulo q reduces the coefficients of the polynomials in the ring $\mathcal{R}$ to lie within the set of integers modulo q, and therefore, we have $\mathbb{Z}_q = \{0, 1, ..., (q-1)\}$. On the other hand, the quotient function $f(x)$ refers to considering the algebraic ring of polynomials with coefficients in $\mathbb{Z}_q$ modulo the ideal subset generated by a specific polynomial $f(x)$, which is the subset of all multiples of $f(x)$ in $\mathbb{Z}_q[x]$. For this reason, the quotient ring $\mathcal{R} = \mathbb{Z}_q[x]/f(x)$ also has all the respective polynomial operations performed modulo both q and $f(x)$, similar to the RLWE Problem. In the context of the MLWE Problem, A is a public polynomial matrix, s is a secret (and private) polynomial, e is an error (noise) polynomial, and t is the public (target) resultant polynomial. As obvious, these polynomials are all defined in the algebraic ring $\mathcal{R} = \mathbb{Z}_q[x]/f(x)$.

4 CRYSTALS (CRYptographic SuiTe for Algebraic LatticeS)

The CRYptographic SuiTe for Algebraic LatticeS (CRYSTALS) is a (classical) post-quantum cryptographic suite encompassing two cryptographic primitives, named Kyber and Dilithium. The former is a secure INDistinguishabile under Chosen Plaintext Attack (IND-CPA) asymmetric encryption algorithm and a secure INDistinguishable under adaptive Chosen Ciphertext Attack (IND-CCA2) Key Encapsulation Mechanism (KEM), which we can also use as a Key Exchange protocol. The latter is a secure digital signature scheme and is strongly Existential UnForgeability under adaptive Chosen-Message Attack (EUF-CMA).

These cryptographic primitives have their security assumptions based on computational hardness from mathematical problems over module lattice groups, which are considered, until the current date, secure against attacks made from both classical and quantum computers. Both Kyber and Dilithium have been submitted to the contest of the Post-Quantum Cryptography Standardization project, being finalists of the 3$^{\text{rd}}$ round of the standardization contest and also selected as some of the first-ever standard cryptographic primitives for the post-quantum era. Each of these cryptographic primitives has three variants offering different levels of security, which we will address in detail in this report.

4.1 CRYSTALS-Kyber

The CRYSTALS-Kyber [59, 60] is a (classical) post-quantum asymmetric cryptosystem designed to be quantum-resistant to future cryptanalytic attacks performed by future powerful quantum computers, ensuring security in the classical contexts as well. This cryptographic primitive uses a variant of the already mentioned MLWE problem on lattice groups as its primary trapdoor function. Namely, this asymmetric cryptosystem won the first spot in the NIST Post-Quantum Cryptography Standardization project and is already selected as a new standard to replace the currently used (classical) pre-quantum asymmetric cryptosystems that are vulnerable to attacks performed by quantum computers.

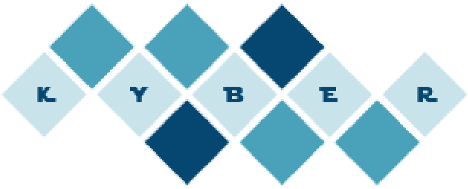

Fig. 2: Logotype of CRYSTALS-Kyber cryptographic primitive.

The design of the primitive CRYSTALS-Kyber has its roots inspired by the LWE-based asymmetric cryptosystem previously proposed by Oded Regev in 2005 [27]. Namely, we can improve the practical efficiency of such cryptosystems by employing the same probability distribution used for the noise to build a secret (i.e., a private key) and using a square matrix rather than a rectangular one as the public key. Another improvement is to use polynomial rings rather than integer numbers, as originally used in the NTRU cryptosystem [26], to define the RLWE and MLWE mathematical problems. From these two main ideas, we build CRYSTALS-Kyber as having its computational hardness assumption based on the MLWE problem. Since from the CRYSTALS-Kyber cryptosystem, we construct the IND-CCA KEM on top of the CPA asymmetric encryption algorithm, the formal definition of the latter will come in the first place.

4.1.1 CRYSTALS-Kyber IND-CPA Asymmetric Encryption Algorithm

Let k, d_u, and d_v be positive integer parameters, and recall that $n = 256$. Additionally, let $\mathcal{M} = \{0,1\}^n = \{0,1\}^{256}$ denote the messages space, where every message $m \in \mathcal{M}$ can be viewed as a polynomial in the algebraic ring $\mathcal{R}$ with binary coefficients (i.e., coefficients in $\{0,1\}$). Now, consider the public-key asymmetric encryption algorithm denoted by the tuple of cryptographic procedures of the form $\texttt{CRYSTALS}_{\texttt{Kyber}}.\texttt{Asym\_Enc} = \big(\texttt{Key\_Gen}, \texttt{Enc}, \texttt{Dec}\big)$. Here, the ciphertexts are composed again by the vectors u and v, having the form:

$$\begin{aligned} ct = (u,v) \in \Big(\{0,1\}^{(n \times k \times d_u)} \times \{0,1\}^{(n \times d_v)}\Big) = \\ = \Big(\{0,1\}^{(256 \times k \times d_u)} \times \{0,1\}^{(256 \times d_v)}\Big) \end{aligned}$$

For the currently recommended standards, we have $k \in \{2,3,4\}$, $d_u = 11$, and $d_v = 3$. However, other configurations are possible as well, depending on the security level desirable for the asymmetric public-key encryption cryptosystem. In order to properly define the $\texttt{CRYSTALS}_{\texttt{Kyber}}$ asymmetric encryption algorithm $\texttt{Asym\_Enc}$, we need to formulate the sub-routines $\texttt{Key\_Gen}$, $\texttt{Enc}$, and $\texttt{Dec}$.

First, we should start by defining the sub-routine `Key_Gen` for the generation of the asymmetric key pair composed of a public and a private (secret) key. This sub-routine receives as inputs, the integer parameters $n \in \mathbb{Z}$, $q \in \mathbb{Z}$, $k \in \mathbb{Z}$, $\eta \in \mathbb{Z}$, and $d_t \in \mathbb{Z}$. Namely, the parameter n determines the dimension of the polynomial ring $\mathcal{R}$, the parameter q defines the modulo for the polynomial coefficients, the parameter k determines the number of polynomials in the module being used, and the parameter η is related to the (random) noise distribution. Both the parameters k and η are related to the security of the asymmetric keys being generated. The former can tune up the complexity of the overall MLWE Problem, while the latter is critical for security by adding randomness to the sub-routine.

The pseudo-code of the sub-routine `Key_Gen` of the $\texttt{CRYSTALS}_{\texttt{Kyber}}$ asymmetric public-key encryption algorithm `Asym_Enc` is defined as we present below:

Sub-routine 3 $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.Asym\_Enc.Key\_Gen}(n, q, k, \eta, d_t)$: Key Generation

```
Input: $(n, q, k, \eta, d_t)$
Output: $(k_{pub}, k_{priv})$
Require: $n = 256$, $q = 7681$[8], $k > 0$, $\eta > 0$, $d_t > 0$
Ensure: $n \in \mathbb{Z},\ q \in \mathbb{Z},\ k \in \mathbb{Z},\ \eta \in \mathbb{Z}$

$\rho, \sigma \leftarrow \{0,1\}^n = \{0,1\}^{256}$
$A \sim \mathcal{R}_q^{(k \times k)} = \mathcal{R}_{7681}^{(k \times k)} := \texttt{Sample}(\rho)$
$(s, e) \sim \left(\beta_\eta^k \times \beta_\eta^k\right) := \texttt{Sample}(\sigma)$
$t := \texttt{Compress}_q(A \times s + e, d_t)$
$k_{pub} := (t, \rho)$
$k_{priv} := s$
return $(k_{pub}, k_{priv})$
```

The resulting output of the sub-routine `Key_Gen` of the $\texttt{CRYSTALS}_{\texttt{Kyber}}$ asymmetric public-key encryption algorithm `Asym_Enc` is a asymmetric key pair composed by the public key k_{pub} and the private key k_{priv}. The public key k_{pub} is a pair composed of the public target vector t and the random (or pseudo-random) seed ρ, while the private key k_{priv} is simply the secret vector s. The random seed ρ is usually a 256-bit string used to send a Pseudo-Random Function (PRF) or cryptographic hash function, from which the coefficients of the polynomials in the matrix A are generated. Furthermore, the public target vector t is compressed by a parameter d_t defined *a priori* that refers to the bit length used for compressed encoded information during the key generation and encryption procedures.

Now we can define the sub-routine `Enc` for the asymmetric encryption of some confidential information. This sub-routine receives as inputs, the integer parameters $n \in \mathbb{Z}$, $q \in \mathbb{Z}$, $k \in \mathbb{Z}$, $\eta \in \mathbb{Z}$, $m \in \mathcal{M}$, as well as the public key k_{pub} generated and received from the other party. Once again, the parameter n determines the dimension of the polynomial ring $\mathcal{R}$, the parameter q defines the modulo for the polynomial coefficients, the parameter k determines the number of polynomials in the module being used, and the parameter η is related to

[8] We can also use $q = 3329$, achieving equal or slightly better performance [148].

the (random) noise distribution. The input m is the confidential message to encrypt, while k_{pub} is the public key generated by the party to which the encrypted information will be sent. Similar to the key generation procedure, both the parameters k and η are related to the security of the random (secret) and error (noise) vectors being randomly sampled from a distribution. The former can tune up the complexity of the overall MLWE Problem, while the latter is critical for security by adding randomness to the overall `Enc` sub-routine.

Namely, the pseudo-code of the sub-routine `Enc` of the $\texttt{CRYSTALS}_{\texttt{Kyber}}$ asymmetric public-key encryption algorithm `Asym_Enc` is defined as we present below:

Sub-routine 4 $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.Asym\_Enc.Enc}(n, q, k, \eta, k_{pub} = (t, \rho),\ m,\ r)$: Asymmetric Encryption

```
Input: $(n, q, k, \eta, k_{pub}, m, r)$      [ $r$ is optional ]
Output: $ct$
Require: $n = 256,\ q = 7681,\ k > 0,\ \eta > 0$
Ensure: $n \in \mathbb{Z},\ q \in \mathbb{Z},\ k \in \mathbb{Z},\ \eta \in \mathbb{Z},\ m \in \mathcal{M}$

$(t, \rho) \leftarrow k_{pub}$
if $r = \varnothing$ then
    $r \leftarrow \{0,1\}^n = \{0,1\}^{256}$
end if
$t := \texttt{Decompress}_q(t, d_t)$
$A \sim \mathcal{R}_q^{(k\times k)} = \mathcal{R}_{7681}^{(k\times k)} := \texttt{Sample}(\rho)$
$(r, e_1, e_2) \sim \left(\beta_\eta^k \times \beta_\eta^k \times \beta_\eta\right) := \texttt{Sample}(r)$
$u := \texttt{Compress}_q(A^T \times r + e_1, d_u)$
$v := \texttt{Compress}_q(t^T \times r + e_2 + \lceil \frac{q}{2} \rfloor \cdot m, d_v)$
$ct := (u, v)$
return $ct$
```

The resulting output of the sub-routine `Enc` of the $\texttt{CRYSTALS}_{\texttt{Kyber}}$ asymmetric public-key encryption algorithm `Asym_Enc` is a ciphertext pair ct composed by the vectors u and v. The construction of the vectors u and v follow the general form of the LWE Problem mentioned before. The public target vector t and the (public) random seed ρ are obtained from the public key k_{pub} of the other party. The target vector t is decompressed using the parameter d_t defined *a priori* and the coefficients of the polynomials of the public matrix A are sampled from the (public) random seed ρ. Furthermore, the vectors u and v composing the ciphertext pair are compressed by the parameters d_u and d_v respectively and defined *a priori*. These parameters d_u and d_v refer to the bit length used for compressed information during the encryption and decryption procedures.

Furthermore, both the `Key_Gen` and `Enc` sub-routines sample and generate the error (noise) vectors e, e_1 and e_2, as well as the random secret vectors t and r from a binomial distribution. The binomial distributions approximate the discrete Gaussian distributions, which are theoretically ideal for security in lattice-based cryptographic primitives. In particular, the Gaussian noise helps to provide strong security guarantees, but it is more complex to generate and handle. In that sense, the Binomial distributions offer a simpler and more efficient alternative while maintaining similar randomness and security properties.

Then, we can define the sub-routine `Dec` for the asymmetric decryption of some confidential information. This sub-routine receives as inputs, the integer parameter $q \in \mathbb{Z}$, as well as the private key k_{priv} generated before and the ciphertext ct. Similar to the encryption procedure, the parameter q defines the modulo for the polynomial coefficients. The input ct is the encrypted message to decrypt and recover the plaintext message $m \in \mathcal{M}$, while the k_{priv} is the private key generated by the party that will decrypt the encrypted information received.

The respective pseudo-code of the sub-routine `Dec` of the $\texttt{CRYSTALS}_{\texttt{Kyber}}$ asymmetric public-key encryption algorithm `Asym_Enc` is defined as follows:

Sub-routine 5 $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.Asym\_Enc}$ $\texttt{.Dec}(q, k_{priv} = s,\ ct = (u, v))$: Asymmetric Decryption

```
Input: $(q, k_{priv}, ct)$
Output: $m$
Require: $q = 7681$
Ensure: $q \in \mathbb{Z}$

$s \leftarrow k_{priv}$
$(u, v) \leftarrow ct$
$u := \texttt{Decompress}_q(u, d_u)$
$v := \texttt{Decompress}_q(v, d_v)$
$m := \texttt{Compress}_q(v - s^T \times u, 1)$
return $m$
```

The resulting output of the sub-routine `Dec` of the $\texttt{CRYSTALS}_{\texttt{Kyber}}$ asymmetric public-key encryption algorithm `Asym_Enc` is a the plaintext message m obtained from the vectors u and v, using the secret vector s from the private key k_{priv} of the party decrypting the encrypted ciphertext ct. The vectors u and v composing the ciphertext ct are decompressed by the parameters d_u and d_v respectively and defined *a priori*. These parameters d_u and d_v refer to the bit length used for compressed information during the encryption and decryption procedures. In the end, the message m is obtained from the compression of the result of the algebraic operation on the ciphertext vectors u, v, jointly with the (private) secret vector s obtained from the private key k_{priv} of the party decrypting ct.

After defining the key generation, encryption, and decryption procedures, we need to show this cryptosystem is correct (or sound) and capable of encrypting and decrypting information correctly (i.e., with a negligible failure probability).

Correctness: In order to show the correctness of the $\texttt{CRYSTALS}_{\texttt{Kyber}}$ asymmetric encryption algorithm `Asym_Enc`, we need to demonstrate that the same has a negligible decryption error δ, concluding that the algorithm is $(1 - \delta)$-correct. Usually, it is considered $\delta < 2^{-128}$ to show the correctness of the cryptosystem.

Theorem 1: Let k be a positive integer parameter. Also, let s, e, r, e_1, e_2 be random variables that have the same probability distribution as in the previously defined sub-routines `Key_Gen` and `Enc` from $\texttt{CRYSTALS}_{\texttt{Kyber}}$ asymmetric encryption algorithm `Asym_Enc`. Then, let $c_t \leftarrow \psi^k_{d_t}$, $c_u \leftarrow \psi^k_{d_u}$, and $c_v \leftarrow \psi_{d_v}$, be distributed according to the probability distribution ψ defined as follows:

- Let ψ^k_d be the following probability distribution over polynomial ring $\mathcal{R}$:
  1. Choose uniformly-random $y \leftarrow \mathcal{R}^k$;
  2. **return** ($y - \texttt{Decompress}_{\text{q}}(\texttt{Compress}_{\text{q}}(y, d), d)$) mod$^{\pm}$ q.[9].
- Denote the decryption error δ as follows:
  - $\delta = \Pr\Big[\; || \; e^T \times r + e_2 + c_v - s^T \times e_1 + c_t^T \times r - s^T \times c_u \; ||_\infty \geq \lceil \frac{q}{4} \rfloor \; \Big]$
- Then, we can conclude that $\texttt{CRYSTALS}_{\texttt{Kyber}}$`.Asym_Enc` is $(1 - \delta)$-correct.

4.1.2 CRYSTALS-Kyber IND-CCA Key Encapsulation Method

Now that we already described the public-key asymmetric encryption algorithm of the Kyber cryptosystem, we can illustrate the related KEM procedures based on the encapsulation and decapsulation of a symmetric session secret key.

In particular, for a correct definition of the $\texttt{CRYSTALS}_{\texttt{Kyber}}$ KEM algorithm `KEM`, we need to formulate the set of sub-routines `Key_Gen`, `Encaps`, and `Decaps`.

Namely, we obtain this asymmetric KEM $\texttt{CRYSTALS}_{\texttt{Kyber}}$`.KEM` algorithm by applying a KEM variant with "implicit rejection" [149] of the Fujisaki-Okamoto Transform [150] to the previously described $\texttt{CRYSTALS}_{\texttt{Kyber}}$`.Asym_Enc` algorithm.

First, let $G : \{0,1\}^* \mapsto \{0,1\}^{(2 \times 256)}$ and $H : \{0,1\}^* \mapsto \{0,1\}^{256}$ be cryptographic hash functions. Then, consider the public-key asymmetric KEM denoted by the procedures set $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.KEM} = \big(\texttt{Key\_Gen}, \texttt{Encaps}, \texttt{Decaps}\big)$.

The pseudo-code of the sub-routine `Key_Gen` of the $\texttt{CRYSTALS}_{\texttt{Kyber}}$ KEM procedure `KEM`, with the previously defined $\texttt{CRYSTALS}_{\texttt{Kyber}}$`.Asym_Enc.Key_Gen` procedure already incorporated, is formulated and defined as follows:

Sub-routine 6 $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.KEM.Key\_Gen}(n,q,k,\eta)$: Key Generation

```
Input: $(n, q, k, \eta)$
Output: $(k_{pub}, k_{priv})$
Require: $n = 256$, $q = 7681$, $k > 0$, $\eta > 0$
Ensure: $n \in \mathbb{Z}$, $q \in \mathbb{Z}$, $k \in \mathbb{Z}$, $\eta \in \mathbb{Z}$

$\chi \leftarrow \{0,1\}^n = \{0,1\}^{256}$
$z \sim \beta^n = \beta^{256} := \texttt{Sample}(\chi)$
$(k_{pub}, k_{priv'}) := \texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.KEM.Key\_Gen}(n, q, k, \eta)$
$k_{priv} := (k_{priv'} || k_{pub} || H(k_{pub}) || z)$
return $(k_{pub}, k_{priv})$
```

[9] Here, the notation mod$^{\pm}$ q denotes a centered modulo q around the origin and defined in the range $[-\frac{(q-1)}{2}, \frac{(q-1)}{2}]$

Here, the sub-routine `Key_Gen` is very similar to the sub-routine `Key_Gen` defined previously in $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.Asym\_Enc}$ with the difference that k_{priv} also contains $k_{pub} = (t, \rho)$ and a secret (binomial) random value z with 256 bits as well.

The pseudo-code of the sub-routine `Encaps` of the $\texttt{CRYSTALS}_{\texttt{Kyber}}$ KEM procedure `KEM`, with the previously defined $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.Asym\_Enc.Enc}$ procedure already incorporated, is formulated and defined as follows:

Sub-routine 7 $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.KEM.Encaps}(n, q, k, \eta, k_{pub} = (t, \rho))$: Key Encapsulation

```
Input: $(n, q, k, \eta, k_{pub})$
Output: $K_{encaps}$
Require: $n = 256$, $q = 7681$, $k > 0$, $\eta > 0$
Ensure: $n \in \mathbb{Z}$, $q \in \mathbb{Z}$, $k \in \mathbb{Z}$, $\eta \in \mathbb{Z}$

$m \leftarrow \{0,1\}^n = \{0,1\}^{256}$
$(\hat{K}, r) := G(H(k_{pub}), m)$
$(u, v) := \texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.Asym\_Enc.Enc}(n, q, k, \eta, k_{pub}, m, r)$
$ct := (u, v)$
$K := H(\hat{K}, H(ct))$
$K_{encaps} := (ct, K)$
return $K_{encaps}$
```

Here, the sub-routine `Encaps` returns the encapsulation K_{encaps} of the symmetric (secret) session key as a pair composed by the ciphertext pair $ct = (u, v)$ and a final hashed secret key K. The ciphertext pair $ct = (u, v)$ results from the asymmetric encryption of a random secret seed m of 256 bits used as a seed for the symmetric (secret) session key. First, the cryptographic hash function H is applied on the asymmetric public key k_{pub}, followed by using the cryptographic hash function G on $H(k_{pub})$ and the random secret seed m. The result of $G(H(k_{pub}), m)$ results in the pair $(\hat{K}, r)$, which can be viewed as two random (coin) strings to be used in the further steps of the encapsulation procedure. Then, the ciphertext pair $ct = (u, v)$ is obtained by applying the asymmetric encryption procedure $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.Asym\_Enc.Enc}$ on the random secret seed m and the random (coin) string r. Next, the final hashed secret key K will be computed, which will work as the symmetric (secret) session key agreed between the parties. The final hashed symmetric (secret) session key K is obtained by applying the cryptographic hash function H on the ciphertext pair $ct = (u, v)$, followed by using the cryptographic hash function H on $\hat{K}$ and $H(ct)$, i.e., $K = H(\hat{K}, H(ct))$. Finally, the encapsulation sub-routine `Encaps` of the $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.KEM}$ finishes by composing the pair $K_{encaps} := (ct, K)$. Despite the encapsulated key being composed by the pair $K_{encaps} := (ct, K)$, only the ciphertext ct, encrypting the random secret seed m and the random (coin) string r, is sent to the other party. The final hashed secret key K will be only used to verify *a posteriori* the data integrity of the (secret) session key being exchanged.

The pseudo-code of the sub-routine `Decaps` of the $\texttt{CRYSTALS}_{\texttt{Kyber}}$ KEM procedure `KEM`, with the previously defined $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.Asym\_Enc.Enc}$ and $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.Asym\_Enc.Dec}$ procedures already incorporated, is defined as:

Sub-routine 8 $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.KEM.Decaps}(n,\ q,\ k,\ \eta,\ k_{priv} = (s, z, t, \rho),\ ct = (u, v))$: Key Decapsulation

```
Input: $(n, q, k, \eta, k_{priv}, ct)$
Output: $K$
Require: $n = 256,\ q = 7681,\ k > 0,\ \eta > 0$
Ensure: $n \in \mathbb{Z},\ q \in \mathbb{Z},\ k \in \mathbb{Z},\ \eta \in \mathbb{Z}$

$(s, z, t, \rho) \leftarrow k_{priv}$
$(u, v) \leftarrow ct$
$k_{pub} := (t, \rho)$
$k_{priv}{}' := s$
$m' := \texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.Asym\_Enc.Dec}(q,\ k_{priv}{}',\ ct)$
$\left(\hat{K}', r'\right) := G(H(k_{pub}), m')$
$(u', v') := \texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.Asym\_Enc.Enc}(n, q, k, \eta, k_{pub},\ m',\ r')$
if $(u', v') = (u, v)$ then
    $K := H(\hat{K}', H(ct))$
else
    $K := H(z, H(ct))$
end if
return $K$
```

Here, the sub-routine `Decaps` returns the (decapsulated) symmetric (secret) session key K shared between the parties at the end of the procedure. First, the components of both public key k_{pub} and private key k_{priv} need to be extracted to use the asymmetric encryption and decryption procedures *a posteriori*. Notice that in the KEM algorithm `KEM`, the k_{priv} also incorporates a random string z, as well as the vectors t and ρ, rather than only incorporating the secret vector s as happens with the k_{priv} of the Asymmetric Encryption algorithm `Asym_Enc`. Additionally, the private key k_{priv} now also incorporates the elements of the public key k_{pub}. Recall also that the private key used in the asymmetric decryption procedure is not the same as the one from the input of the decapsulation procedure. For this reason, we will consider a first private key $k_{priv} = (s, z, t, \rho)$ for the decapsulation procedure `Decaps` of the KEM algorithm `KEM` and a second private key $k_{priv}{}' = s$ for the decryption procedure `Dec` of the Asymmetric Encryption algorithm `Asym_Enc` to distinguish them. Then, a random secret seed m' is obtained from the decryption procedure on the ciphertext pair ct using the respective private key $k_{priv}{}'$. Next, the public key k_{pub} is hashed using the cryptographic hash function H, followed by using the cryptographic hash function G on $H(k_{pub})$ and the random secret seed m'. The result of $G(H(k_{pub}), m')$ results in the pair $(\hat{K}', r')$, which can be viewed as two random (coin) strings to be used in the further steps of the decapsulation procedure. Then, another ciphertext pair (u', v') is obtained by applying the asymmetric encryption procedure $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.Asym\_Enc.Enc}$ on the random secret seed m' and the random (coin) string r'. Next, the final hashed secret key K will be computed, which will work as the symmetric (secret) session key agreed between the parties. If the two ciphertext pairs (u, v) and (u', v') are equal, the final hashed symmetric (secret) session key K will be computed properly. Namely, the final hashed symmetric (secret) session key K is obtained by applying the cryptographic hash function H on the ciphertext pair $ct = (u, v)$, followed by using the cryptographic hash function H on $\hat{K}$ and $H(ct)$, i.e., $K = H(\hat{K}, H(ct))$. The sub-routine $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.KEM.Decaps}$ never returns $\perp$ (i.e., *absurdum*). Instead, in the case of re-encryption failure, a pseudo-random key $K := H(z, ct)$ is computed from a hash function and returned, where z is a random secret seed. This happens when the ciphertext pair (u', v') is not equal to $ct = (u', v')$.

After defining the key generation, encapsulation, and decapsulation methods, we need to show this cryptosystem is correct (or sound) and capable of encapsulating and decapsulating symmetric secret keys correctly (i.e., with a negligible failure probability). Additionally, we need to show that this cryptosystem is secure.

Correctness: If the $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.Asym\_Enc}$ algorithm is considered to be $(1-\delta)$-correct and the cryptographic hash function G represents a random oracle, then the $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.KEM}$ algorithm is also considered $(1-\delta)$-correct [149].

Security: The following concrete security statement proves the security of the $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.Asym\_Enc}$ algorithm when we model the cryptographic hash functions G and H as random oracles. Namely, for the concrete security bounds [149], the KEM variant of the Fujisaki-Okomoto Transform [150] is considered, and a non-zero correctness (or decryption) error δ is also assumed in this case.

Theorem 3: For any classical adversary $\texttt{A}$ that makes at most q_{RO} many queries to the random oracles G and H, and q_{DO} queries to the decryption oracle, there is another classical adversary $\texttt{B}$ such that the following inequality holds:

$$\mathbf{Adv}^{\text{CCA}}_{\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.KEM}}(\texttt{A}) \leq 3 \times \mathbf{Adv}^{\text{CPA}}_{\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.Asym\_Enc}}(\texttt{B}) + q_{RO} \cdot \delta + \frac{(3 \times q_{RO})}{2^{256}}$$

As for security in the Quantum Random Oracle Model (QROM), we can use the fact that $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.Asym\_Enc}$ algorithm is ONE-WAY CPA secure and *sparse pseudo-random* [151] to prove that $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.KEM}$ algorithm is IND-CCA secure. Namely, sparse pseudo-randomness is a slightly stronger security notation than the IND-CPA security and states the following properties:

1. A properly generated ciphertext is *pseudo-random* (i.e., it is computationally indistinguishable from a random high-entropy binary string).
2. A random (high-entropy) binary string is, with high probability, not equal to a properly generated (pseudo-random) ciphertext.

The proof of Theorem 2 shows that the $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.Asym\_Enc}$' algorithm (i.e., the $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.Asym\_Enc}$ algorithm variant without compressing the vector t) is tightly pseudo-random under the Module-LWE computational hardness assumption. Concretely, the pseudo-randomness advantage is bounded by:

$$\mathbf{Adv}^{\mathrm{PR}}_{\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.Asym\_Enc}'}(\texttt{A}) \leq 2 \cdot \mathbf{Adv}^{\mathrm{MLWE}}_{(k+1,k,\eta)}(\texttt{B})$$

Now, we can argue again that the same mathematical bound holds for the $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.Asym\_Enc}$ algorithm. Namely, the $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.Asym\_Enc}$ algorithm has its sparseness property trivially fulfilled since the set of properly generated (pseudo-random) ciphertexts is a sparse subset of the original (pseudo-random) ciphertext space defined as $\left(\{0,1\}^{(256 \cdot k \cdot d_u)} \times \{0,1\}^{d_v}\right)$.

Then, we can also provide the following security statement in the QROM, considering a quantum adversary, for a correctness (decryption) error δ.

Theorem 4: For any quantum adversary $\texttt{A}$ that makes at most q_{RO} many queries to the quantum random oracles G and H, and at most q_{DO} (classical) queries to the decryption oracle, there exists a quantum adversary $\texttt{B}$ such that:

$$\mathbf{Adv}^{\mathrm{CCA}}_{\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.KEM}}(\texttt{A}) \leq 8 \times q_{RO}^2 \cdot \delta + 4 \times q_{RO} \cdot \sqrt{\mathbf{Adv}^{\mathrm{PR}}_{\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.Asym\_Enc}}(\texttt{B})}$$

Unfortunately, the above security bound is non-tight, and we can only use it as an asymptotic indication of the CCA security of the $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.KEM}$ algorithm in QROM [151]. However, we can derive a tight security bound in the QROM from a non-standard security assumption, namely that a deterministic version of $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.Asym\_Enc}$ algorithm is sparse pseudo-random in the QROM. The $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.Asym\_Enc}$ algorithm is already deterministic, but the randomness of the vector r used in the encryption process is derived deterministically from the message m by performing $r := G(m)$. In the Classical Random Oracle Model (CROM), this security assumption is tight and implied by the IND-CPA security of $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.Asym\_Enc}$ algorithm, but the security reduction in the QROM is given by a non-tight bound. For this reason, we need to have the term $q_{RO} \cdot \sqrt{\mathbf{Adv}^{\mathrm{PR}}_{\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.Asym\_Enc}}(\texttt{B})}$ in the inequality given of the Theorem 4.

Hashing k_{pub} into $\hat{K}$: The transformation used in the $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.Asym\_Enc}$ algorithm is essentially the Fujisaki–Okamoto Transform "implicit rejection" [149, 150, 152] with a small adjustment. Most specifically, we need to hash the public key k_{pub} by computing $\hat{K} := H(k_{pub})$. Namely, this adjustment has two effects. First, this adjustment makes the $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.KEM}$ algorithm contributory since the final shared key K does not depend only on the input of one of the two parties involved. Second, this adjustment also offers multi-target protection. In particular, consider an attacker who searches through many values m to find one particular message that produces a decryption failure with a considerable probability. Such decryption failure of a legitimate ciphertext would leak some information about the secret key. In the pre-quantum context, the negligible decryption error δ guarantees that it is not a practical attack. In the post-quantum context, an attacker can use Grover's algorithm to speed up the search for that message m. However, the attacker will have difficulty encoding such message m, which may produce a decryption failure with a considerable probability in Grover's quantum oracle. In particular, this problem is equivalent to identifying noise vectors that likely have a large inner product with the tuple of vectors (s, e). Eventually, the best strategy is to search for a message m that produces noise vectors with a large norm. This attack probably does not offer any better performance than a brute-force attack performed by Grover's algorithm for searching the shared key K with 256 bits. However, applying a hash function and transforming k_{pub} into $\hat{K}$ ensures that an attacker would not be able to use pre-computed values (i.e., pre-images) of messages m against multiple targets.

4.1.3 CRYSTALS-Kyber Key Exchange Protocol

Now that we already described the KEM procedures based on the encapsulation and decapsulation of a symmetric session secret key, we can illustrate how to derive a Key Exchange Protocol using this asymmetric public-key cryptosystem.

Let $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.KEM} = \big(\texttt{Key\_Gen}, \texttt{Encaps}, \texttt{Decaps}\big)$ be the IND-CCA secure KEM algorithm described before. Now, we denote the Key Exchange protocol as $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.KE}$, and we obtain it as a direct cryptographic application of the $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.KEM}$ algorithm. In Key Exchange protocols using a KEM algorithm, a cryptographic hash function is commonly applied to all messages each participant sends and receives in order to compose the final secret key. Namely, we compute a cryptographic hash function on the public key k_{pub} to obtain the pre-shared key $\hat{K}$, and the ciphertext is then cryptographically hashed into the final secret key K. Therefore, the shared secret key K obtained during the Key Exchange Protocol already includes these exchanged messages of each participant. We illustrate the steps of the $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.KE}$ algorithm in Fig. 3:

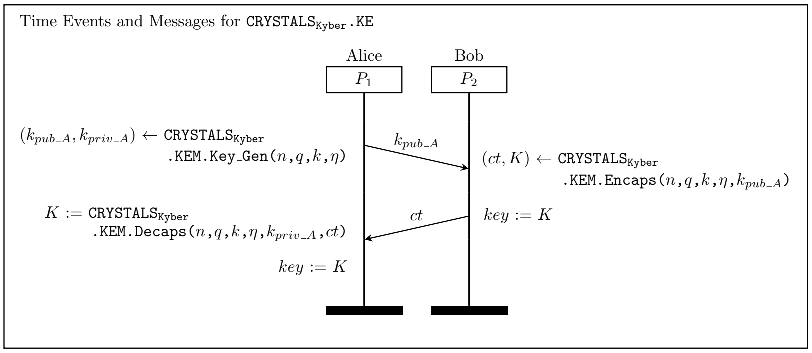
Time Events and Messages for $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.KE}$
Alice
Bob
P_1
P_2
$(k_{pub\_A}, k_{priv\_A}) \leftarrow \texttt{CRYSTALS}_{\texttt{Kyber}}$ $\texttt{.KEM.Key\_Gen}(n,q,k,\eta)$
k_{pub_A}
$(ct, K) \leftarrow \texttt{CRYSTALS}_{\texttt{Kyber}}$ $\texttt{.KEM.Encaps}(n,q,k,\eta,k_{pub\_A})$
$K := \texttt{CRYSTALS}_{\texttt{Kyber}}$ $\texttt{.KEM.Decaps}(n,q,k,\eta,k_{priv\_A},ct)$
ct
$key := K$
$key := K$

Fig. 3: Schematic of the CRYSTALS-Kyber Key Exchange protocol, with the respective Time Events and Messages.

As mentioned before, we can notice that despite the result of the encapsulation `Encaps` of the $\texttt{CRYSTALS}_{\texttt{Kyber}}\texttt{.KEM}$ algorithm being the pair (ct, K), only the ciphertext pair ct is sent to the other party. Then, the other party generates the secret key K by hashing some confidential data decrypted from the ciphertext pair ct. After the encapsulation and decapsulation procedures, both parties should have the same secret key K at the end of the Key Exchanged Protocol.

4.2 CRYSTALS-Dilithium

The CRYSTALS-Dilithium [61,62] is a (classical) post-quantum digital signature scheme developed to be related and complementary to the CRYSTALS-Kyber asymmetric cryptosystem previously presented, being another component of the CRYSTALS cryptosuite. Therefore, this digital signature scheme is also quantum-resistant and intended to be computationally secure against future cryptanalytic attacks performed by future quantum computers with considerable processing power, also ensuring security in the classical contexts. Namely, this cryptographic primitive also uses the computational hardness of the MLWE Problem as its trapdoor function and has most of its security notions based on it. For this reason, this cryptographic digital signature scheme is considered strongly secure under a Chosen-Message Attack (CMA) model. The security notion used on this cryptosystem means that an adversary with access to a signing oracle cannot produce a signature of a message whose signature it has not yet seen nor produce a different signature of a message it already saw signed.

Moreover, CRYSTALS-Dilithium is one of the first selected standards for digital signatures in the NIST Post-Quantum Cryptography Standardization project, together with FAst-fourier Lattice-based Compact signatures Over Ntru (FALCON) [63, 64] and Stateless Practical Hash-based Incredibly Nice Cryptographic Signatures^{+} (SPHINCS^{+}) [65, 66] to replace the current digital signatures that are insecure against to attacks from quantum computers.

Fig. 4: Logotype of CRYSTALS-Dilithium cryptographic primitive.

The core design of the primitive CRYSTALS-Dilithium is based mainly on a cryptographic technique known as “Fiat-Shamir with Aborts”, proposed by Vadim Lyubashevsky in 2009 [153, 154]. This technique uses rejection sampling to make lattice-based Fiat-Shamir digital signatures [155] compact and secure. It is possible to create digital signature schemes applying this approach, and with small signature sizes, based on the hardness assumption of the mathematical problem used in the NTRU lattice-based cryptosystem [26] that crucially uses random samplings from Gaussian probability distributions. However, since it is hard and non-trivial to implement random samplings from the Gaussian probability distribution, the CRYSTALS-Dilithium digital signature scheme uses only the uniform probability distribution to obtain those random samplings.

In particular, this cryptographic primitive improved the previous most efficient digital signature scheme known that only uses the uniform probability distribution for random samplings, proposed by Shi Bai and Steven Galbraith in 2013 [156]. Namely, CRYSTALS-Dilithium digital signature scheme uses a new technique that shrinks the size of the public key by more than half. This digital signature scheme has the smallest public key and signature sizes of any known lattice-based digital signature scheme that only uses a uniform probability distribution for the random samplings. However, it may have larger sizes for the signature, public key, and private key when we compare it to the other digital signature schemes NIST also selected as new standards in the NIST Post-Quantum Cryptography Standardization project. Namely, it has larger public and private keys than SPHINCS$^+$ hash-based digital signature [65, 66] in spite of having shorter signatures. Additionally, the FALCON lattice-based digital signature [63, 64] has smaller sizes for the signature, public key, and private key than the CRYSTALS-Dilithium digital signature scheme. However, the former uses a Gaussian probability distribution for the random samplings.

4.2.1 CRYSTALS-Dilithium Digital Signature

Let n, ℓ, γ_1, ω, and k be positive integer parameters, and recall that $n = 256$. Additionally, let $\mathcal{M} = \{0,1\}^*$ denote the messages space, where every message $m \in \mathcal{M}$ will be cryptographically hashed jointly with a random seed of 256 bits, outputting a value μ of 512 bits to be used then to produce the final signature. Now, consider the digital signature scheme denoted by the tuple of cryptographic procedures of the form $\texttt{CRYSTALS}_{\texttt{Dilithium}}.\texttt{Signature} = \big(\texttt{Key\_Gen}, \texttt{Sign}, \texttt{Verify}\big)$. Here, the signatures are composed of the vectors $\tilde{c}$, z, and h, having the form:

$$\begin{aligned}\sigma = (\tilde{c}, z, h) \in \Big(\{0,1\}^n \times \{0,1\}^{(n\times\ell\times(1+\log_2(\gamma_1))} \times \{0,1\}^{(8\times(\omega+k))}\Big) = \\ = \Big(\{0,1\}^{256} \times \{0,1\}^{(256\times\ell\times(1+\log_2(\gamma_1))} \times \{0,1\}^{(8\times(\omega+k))}\Big)\end{aligned}$$

For the currently recommended standards, we have $\ell \in \{4,5,7\}$, $\gamma_1 \in \{2^{17}, 2^{19}\}$, $\omega \in \{55, 75, 80\}$, and $k \in \{4,6,8\}$. However, other configurations are possible as well, depending on the security level desirable for the digital signature scheme.

In order to properly define the $\texttt{CRYSTALS}_{\texttt{Dilithium}}$ digital signature scheme $\texttt{Signature}$, we need to formulate the sub-routines $\texttt{Key\_Gen}$, $\texttt{Sign}$, and $\texttt{Verify}$.

First, we should start by defining the sub-routine `Key_Gen` for the generation of the asymmetric key pair composed of a public and a private (secret) key. This sub-routine receives as inputs, the integer parameters $n \in \mathbb{Z}$, $k \in \mathbb{Z}$, $\ell \in \mathbb{Z}$, $q \in \mathbb{Z}$, $\eta \in \mathbb{Z}$, and $d_t \in \mathbb{Z}$. Namely, the parameter n determines the dimension of the polynomial ring $\mathcal{R}$, the parameter k determines the number of polynomials in the module being used, ℓ determines the number of secret polynomials, q defines the modulo for the polynomial coefficients, and the parameter η is related to the (random) noise distributions, determining as well the secret key range. The parameters k, ℓ, and η are related to the security of the asymmetric keys generated. The first two can tune up the complexity of the MLWE Problem, while the latter is critical for security by adding randomness to the sub-routine.

The pseudo-code of the sub-routine `Key_Gen` of the $\texttt{CRYSTALS}_{\texttt{Dilithium}}$ asymmetric digital signature scheme algorithm `Signature` is defined as we present below:

Sub-routine 9 $\texttt{CRYSTALS}_{\texttt{Dilithium}}$`.Signature`
`.Key_Gen`(n, k, ℓ, q, η, d_t): Key Generation

```
Input: $(n, k, \ell, q, \eta, d_t)$
Output: $(k_{pub}, k_{priv})$
Require: $n = 256,\ q = 8380417,\ d_t = 13,\ k > 0,\ \ell > 0,\ \eta > 0$
Ensure: $n \in \mathbb{Z},\ k \in \mathbb{Z},\ \ell \in \mathbb{Z},\ q \in \mathbb{Z},\ \eta \in \mathbb{Z},\ d_t \in \mathbb{Z}$

$\zeta \leftarrow \{0,1\}^n = \{0,1\}^{256}$
$(\rho, \rho', K) \sim \left( \{0,1\}^n \times \{0,1\}^{(2 \times n)} \times \{0,1\}^n \right) =$
        $= \left( \{0,1\}^{256} \times \{0,1\}^{512} \times \{0,1\}^{256} \right) := H(\zeta)$
$A \sim \mathcal{R}_q^{(k \times \ell)} = \mathcal{R}_{8380417}^{(k \times \ell)} := \texttt{Sample}(\rho)$
$(s, e) \sim \left( \Upsilon_\eta^\ell \times \Upsilon_\eta^k \right) := \texttt{Sample}(\rho')$
$t := A \times s + e$
$(t_1, t_0) := \texttt{Compress}_q(t, d_t) = \texttt{Power2Round}_q(t, d_t)$
$t_r \sim \{0,1\}^n = \{0,1\}^{256} := H\left(\rho \mid\mid t_1\right)$
$k_{pub} := (\rho, t_1)$
$k_{priv} := (\rho, K, t_r, s, e, t_0)$
return $(k_{pub}, k_{priv})$
```

The resulting output of the sub-routine `Key_Gen` of the $\texttt{CRYSTALS}_{\texttt{Dilithium}}$ digital signature scheme algorithm `Signature` is an asymmetric key pair composed by the public key k_{pub} and the private key k_{priv}. The public key k_{pub} is a pair composed of the random (or pseudo-random) seed ρ and the public target vector t_1, while the private key k_{priv} is composed of the random (or pseudo-random) seed ρ, the (random) secret seed key K, the random (coin) target vector t_r, the secret vector s, the error (noise) vector e, and the low-order bit target vector t_0. The random seed ρ is usually a 256-bit string used to send a Pseudo-Random Function (PRF) or cryptographic hash function, from which the coefficients of the polynomials in the matrix A are generated. The random seed ρ' is usually a 512-bit string used to generate the random secret vector s and the error (noise) vector. Furthermore, the public target vector t is compressed by a parameter d_t

defined *a priori* that refers to the number of bits dropped from that vector on the compression of the encoded information during the key generation procedure.

Now we can define the sub-routine `Sign` for the creation of a digital signature on some given message. Namely, this sub-routine receives as inputs, the integer parameters $n \in \mathbb{Z}$, $k \in \mathbb{Z}$, $\ell \in \mathbb{Z}$, $q \in \mathbb{Z}$, $\gamma_1 \in \mathbb{Z}$, $\gamma_2 \in \mathbb{Z}$, $\tau \in \mathbb{Z}$, $\eta \in \mathbb{Z}$, $m \in \mathcal{M} = \{0,1\}^*$. The input m is the message to be signed, while k_{priv} is the private key generated by the party signing the message m, guaranteeing the authenticity of the same. Once again, the parameter n determines the dimension of the polynomial ring $\mathcal{R}$, the parameter k determines the number of polynomials in the module being used, ℓ determines the number of secret polynomials, and q defines the modulo for the polynomial coefficients. The parameter γ_1 is the coefficient range for a (random) mask vector y, related to the (random) noise distributions used for maskings required to generate randomness and uniqueness for each signature produced. This (random) mask vector y is crucial to create a commitment vector w, which will be used to generate a challenge vector c. The (random) mask vector y and the challenge vector c, jointly with the secret vector s from the private key k_{priv}, will be used to create a response target vector z, following the mathematical structure of the MLWE Problem. The parameter γ_2 sets a low-order rounding range bound on the polynomial coefficients of the commitment vector w to ensure those coefficients are not too large. The parameter τ determines the number of non-zero values (i.e., ±1 values) on the challenge vector c. Similar to the previously described key generation procedure, the input parameters k, ℓ, γ_1, and γ_2 are related to the security of the digital signatures produced. The first two can tune up the complexity of the MLWE Problem, while the remaining ones are critical for efficiency and performance by being closely related to the rate of rejection of the potential signature candidates.

The resulting output of the sub-routine `Sign` of the $\texttt{CRYSTALS}_{\texttt{Dilithium}}$ digital signature scheme algorithm `Signature` is a digital signature σ composed by the (compressed) challenge vector $\tilde{c}$, the response target vector z, and the hint vector h. The (compressed) challenge vector $\tilde{c}$ aims to improve the efficiency of the signature σ in terms of its size and transmission. From the (compressed) challenge vector $\tilde{c}$, the party verifying the signature σ will be able to sample and uncompress it to the (real) challenge vector c. The response target vector z is the response related to the challenge vector c following the mathematical structure of an MLWE Problem. Then, the hint vector h aims to help the other party recover certain polynomial coefficients of the commitment derived from the signature, thus validating the authenticity and integrity of the signed message. This hint vector ensures that the signature σ is valid without needing to store or transmit large amounts of data. The random seed ρ is usually a 256-bit string used to generate the coefficients of the polynomials in the matrix A. The random seed ρ' is usually a 512-bit string used to generate the (random) mask vector y required to create the commitment vector w and the response target vector z.

The pseudo-code of the sub-routine `Sign` of the $\texttt{CRYSTALS}_{\texttt{Dilithium}}$ asymmetric digital signature scheme `Signature` is defined as we present in sub-routine 10.

Sub-routine 10 $\texttt{CRYSTALS}_{\texttt{Dilithium}}$ $\texttt{.Signature.Sign}(n, k, \ell, q, \gamma_1, \gamma_2, \tau, \eta, \omega,$ $k_{priv} = (\rho, K, t_r, s, e, t_0), m)$: Message Signing

```
Input: $(n, k, \ell, q, \gamma_1, \gamma_2, \tau, \eta, \omega, k_{priv}, m)$
Output: $\sigma$
Require: $n = 256,\ q = 8380417,\ k > 0,\ \ell > 0,\ \gamma_1 > 0,\ \gamma_2 > 0,\ \tau > 0,\ \eta > 0,\ \omega > 0$
Ensure: $n \in \mathbb{Z},\ k \in \mathbb{Z},\ \ell \in \mathbb{Z},\ q \in \mathbb{Z},\ \gamma_1 \in \mathbb{Z},\ \gamma_2 \in \mathbb{Z},\ \tau \in \mathbb{Z},$
        $\eta \in \mathbb{Z},\ \omega \in \mathbb{Z},\ m \in \mathcal{M} = \{0,1\}^*$
$(\rho, K, t_r, s, e, t_0) \leftarrow k_{priv}$
$A \sim \mathcal{R}_q^{(k \times \ell)} = \mathcal{R}_{8380417}^{(k \times \ell)} := \texttt{Sample}(\rho)$
$\mu \sim \{0,1\}^{(2 \times n)} = \{0,1\}^{512} := H\,(t_r \;||\; m)$
$\kappa := 0$
$(z, h) := \perp$
$\rho' \sim \{0,1\}^{(2 \times n)} = \{0,1\}^{512} := H\,(K \;||\; \mu)$ (or $\sigma \leftarrow \{0,1\}^{(2 \times n)} = \{0,1\}^{512}$
    for randomized signing)
while $(z, h) = \perp$ do
    $y \sim \tilde{\Upsilon}_{\gamma_1}^{\ell} := \texttt{SampleMask}(\rho', \kappa)$
    $w := A \times y$
    $w_1 := \texttt{HighBits}_q(w, 2 \times \gamma_2)$
    $\tilde{c} \sim \{0,1\}^n = \{0,1\}^{256} := H\,(\mu \;||\; w_1)$
    $c \sim B_\tau := \texttt{SampleInBall}_\tau(\tilde{c})$
    $z := y + c \times s$
    $r_0 := \texttt{LowBits}_q(w - c \times e, 2 \times \gamma_2)$
    $\beta := \tau \times \eta$
    if $\left( \llbracket\ ||z||_\infty \geq \gamma_1 - \beta\ \rrbracket \text{ or } \llbracket\ ||r_0||_\infty \geq \gamma_2 - \beta\ \rrbracket \right)$ then
        $(z, h) := \perp$
    else
        $h := \texttt{MakeHint}_q(-c \times t_0, w - c \times e + c \times t_0, 2 \times \gamma_2)$
        if $\left( \llbracket\ ||c \times t_0||_\infty \geq \gamma_2\ \rrbracket \text{ or } \llbracket\ weight(h) > \omega\ \rrbracket \right)$ then
            $(z, h) := \perp$
        end if
    end if
    $\kappa := \kappa + \ell$
end while
$\sigma = (\tilde{c}, z, h)$
return $\sigma$
```

Then, we can define the sub-routine `Verify` for the verification of a digital signature on some given message. This sub-routine receives as inputs, the integer parameters $n \in \mathbb{Z}$, $k \in \mathbb{Z}$, $\ell \in \mathbb{Z}$, $q \in \mathbb{Z}$, $d_t \in \mathbb{Z}$, $\gamma_1 \in \mathbb{Z}$, $\gamma_2 \in \mathbb{Z}$, $\tau \in \mathbb{Z}$, $\eta \in \mathbb{Z}$, $\omega \in \mathbb{Z}$, $m \in \mathcal{M} = \{0,1\}^*$. The input m is the message supposedly related to the digital signature σ, while k_{pub} is the public key generated by the party verifying the digital signature σ. Once again, the parameter n determines the dimension of the polynomial ring $\mathcal{R}$, the parameter k determines the number of polynomials in the module being used, ℓ determines the number of secret polynomials, and q defines the modulo for the polynomial coefficients. The parameter γ_1 is the coefficient range for the response vector z, setting an upper bound on its polynomial coefficients. The parameter γ_2 sets a low-order rounding range bound on the polynomial coefficients of the commitment vector w to ensure those coefficients are not too large. The parameter τ determines the number of non-zero values (i.e., ±1 values) on the challenge vector c. The parameter ω sets an upper bound to the Hamming weight of the hint vector h. Similar to the previously described key generation procedure, the input parameters k, ℓ, γ_1, and γ_2 are related to the security of the digital signatures verified. The first two can tune up the complexity of the MLWE Problem, while the remaining ones are critical for the efficiency and performance of the sub-routine by being closely related to the rate of success/failure of the verification procedure applied to a signature.

The pseudo-code of the sub-routine `Verify` of the $\texttt{CRYSTALS}_{\texttt{Dilithium}}$ asymmetric digital signature scheme algorithm `Signature` is defined as we present below:

Sub-routine 11 $\texttt{CRYSTALS}_{\texttt{Dilithium}}$ $\texttt{.Signature.Verify}(n, k, \ell, q, d_t, \gamma_1, \gamma_2, \tau, \eta, \omega,$ $k_{pub} = (\rho, t_1), m, \sigma = (\tilde{c}, z, h))$: Signature Verification

```
Input: $(n, k, \ell, q, d_t, \gamma_1, \gamma_2, \tau, \eta, \omega, k_{pub}, m, \sigma)$
Output: $is\_valid$
Require: $n = 256$, $q = 8380417$, $d_t = 13$, $k > 0$, $\ell > 0$, $\gamma_1 > 0$, $\gamma_2 > 0$,
         $\tau > 0$, $\eta > 0$, $\omega > 0$
Ensure: $n \in \mathbb{Z},\ k \in \mathbb{Z},\ \ell \in \mathbb{Z},\ q \in \mathbb{Z},\ d_t \in \mathbb{Z},\ \gamma_1 \in \mathbb{Z},\ \gamma_2 \in \mathbb{Z},$
        $\tau \in \mathbb{Z},\ \eta \in \mathbb{Z},\ \omega \in \mathbb{Z},\ m \in \mathcal{M} = \{0,1\}^*$

$(\rho, t_1) \leftarrow k_{pub}$
$A \sim \mathcal{R}_q^{(k \times \ell)} = \mathcal{R}_{8380417}^{(k \times \ell)} := \texttt{Sample}(\rho)$
$\mu \sim \{0,1\}^{(2 \times n)} = \{0,1\}^{512} := H\left(H(\rho \mid\mid t_1) \mid\mid m\right)$
$c := \texttt{SampleInBall}_\tau(\tilde{c})$
$w_1' := \texttt{UseHint}_q(h, A \times z - c \times t_1 \times 2^{d_t}, 2 \times \gamma_2)$
$\beta := \tau \times \eta$
$is\_valid := \left( [\![\ \left|\left|z\right|\right|_\infty < \gamma_1 - \beta\ ]\!] \text{ and } [\![\ \tilde{c} = H(\mu \mid\mid w_1')\ ]\!] \text{ and } [\![\ weight(h) \leq \omega\ ]\!] \right)$
return $is\_valid$
```

The resulting output of the sub-routine `Verify` of the $\texttt{CRYSTALS}_{\texttt{Dilithium}}$ digital signature scheme algorithm `Signature` is a *boolean* conjunction is_valid of three terms. This sub-routine receives the message m, as well as the (supposedly) related digital signature σ composed by the (compressed) challenge vector $\tilde{c}$, the response target vector z, and the hint vector h. Additionally, the public key k_{pub} is also required for this procedure, since its (extracted) components ρ and t_1 will be crucial in several steps. The random seed ρ is usually a 256-bit string used to generate the coefficients of the polynomials in the matrix A. The random seed ρ is also concatenated with the (compressed) target vector t_1 to produce a digest result from the cryptographic hash function $H(\rho \mid\mid t_1)$. The result of this digest is then concatenated with the message m, being applied another cryptographic hash function on this concatenation, i.e., $H\left(H(\rho \mid\mid t_1) \mid\mid m\right)$, to produce a binary random string μ with 512 bits. From the (compressed) challenge vector $\tilde{c}$, the party verifying the signature σ will be able to sample and uncompress it to the (real) challenge vector c. Then, using the hint vector h and applying algebraic operations involving the lattice matrix A, the response target vector z, the (uncompressed) challenge vector c, and the (compressed) target vector t_1, it is computed a commitment vector w_1'. The algebraic operations related to the computation of the commitment vector w_1' follow the mathematical structure of an MLWE Problem. Finally, the *boolean* conjunction value *is_valid* is computed verifying if all these three following *boolean* conditions hold *simultaneously*:

1. The polynomial coefficients of the response vector z are not greater than the upper bound given by $\gamma_1 - \beta$, where β is computed as $\beta = \tau \times \eta$;
2. The digest result of the cryptographic hash function $H(\mu \mid\mid w_1')$ is equal to the (compressed) challenge vector $\tilde{c}$ extracted from the signature σ;
3. The Hamming weight (i.e., the number of bits equal to 1) of the hint vector h extracted from the signature σ is not greater than the upper bound ω.

If the three *boolean* conditions hold, the verification of the given digital signature σ related to the given message m succeeded and the signature σ is valid.

The CRYSTALS-Dilithium digital signature uses a pseudo-randomness in the signing procedure generated using the well-known cryptographic hash function Secure Hash Algorithm and KECCAK - 256 (SHAKE-256)[10] from the Secure Hash Algorithm - 3 (SHA-3) family as a deterministic function of the message to be signed and a small secret key (see the steps 3 and 6 of the sub-routine 10). Since most of the signing procedure may need to be repeated several times in a computational loop until we build a final signature, a counter κ is also appended during the creation of the (random) mask vector y, which will be the input of the SHAKE-256 cryptographic hash function making its output differs with each signing attempt of the same message m. Because each (possibly long) message m may require several iterations to be successfully signed, we compute an initial hash digest of the (initial) message m using a collision-resistant cryptographic hash function (see the step 3 of the sub-routine 10). Then, we use the respective hash digest as the input of the cryptographic hash function in the computational loop throughout the signing procedure (see steps 3 and 11 of the sub-routine 10).

[10] The cryptographic hash function SHAKE-256(M, d), differently from the well-known cryptographic hash function SHA-256, can have an output of arbitrary size.

Additionally, one of the primary implementation design improvements offered by the CRYSTALS-Dilithium digital signature scheme over the previous ones is the fact that the size of the public key is approximately halved, at the expense of little longer signatures with fewer than a hundred extra bytes. In order to perform this size reduction, the key generation sub-routine outputs $t_1 := \texttt{Power2Round}_q(t, d_t)$ as a part of the public key, dropping at average d_t bits per polynomial coefficient. Then, the public key requires $\lceil \log(q) \rceil - d_t$ bits per coefficient of the ring polynomial. In the configurations proposed by the authors, we assume $q \approx 2^{23} = 8388608$ and $d_t = 13$, which means that instead of 23 bits in each coefficient of the public key, there is instead only 10 bits (i.e., $\lceil \log(q) \rceil - d_t = \lceil \log(2^{23}) \rceil - 13 = 23 - 13 = 10$), what results in a compression of the size of the public key generated (see the step 6 of the sub-routine 9).

The main complication of not having the entire target vector t in the public key k_{pub} is that it is no longer possible to compute exactly the commitment vector w_1' in the signature verification sub-routine. The signature verification will need the high-order bits of the result of the algebraic operation $A \times z - c \times t$ to compute this, but it can only compute $A \times z - c \times t_1 \times 2^{d_t} = A \times z - c \times t + c \times t_0$ (see the step 5 of the sub-routine 11). Even though the high-order bits of $c \times t_0$ are 0, its presence in the sum creates "carry" bits which may affect the higher bits. The signer of the message thus sends these "carries" as a hint vector h to the verifier of the signature. Heuristically, based on the parameter choices proposed by the authors, there should not be more than ω positions in which a "carry" is caused. Therefore, the signer of the message simply sends the positions in which these "carries" occur, which allows the verifier to compute the high order bits of $A \times z - c \times t$ with the support given by the hint vector h (these are the extra bytes in CRYSTALS-Dilithium compared to other previous digital signatures).

In order to keep the size of the public and private keys small, both the `Sign` and `Verify` procedures begin with extracting the matrix A, or more accurately, its Number-Theoretic Transform (NTT) [157] domain representation $\hat{A}$ from the (public) pseudo-random seed ρ. If storage space is not a factor, then the NTT domain representation matrix $\hat{A}$ can be pre-computed and be part of both the public and private keys. The signer of the message m can additionally pre-compute the NTT domain representations of the secret vector s, error (noise) vector e, and target vector t_0 to slightly speed up the signing sub-routine. On the other hand, if the signer of the message m wants to store a private key as small as possible, it only needs to store a secret seed ζ with 256 bits, which is used to generate the pseudo-randomness to create the pseudo-random seed ρ, the secret key K, the secret vector s, and the error (noise) vector e, in the key generation sub-routine. Furthermore, we can keep the memory usage low for some of the intermediate computations required by only keeping the parts of the NTT domain representation we are currently working on within all the procedures. For more details about this implementation with low memory usage, see the original proposal of the CRYSTALS-Dilithium digital signature scheme.

We can use the CRYSTALS-Dilithium cryptographic primitive to produce deterministic or randomized digital signatures. The former always gives the same digital signature output σ for a given message m since we derive the random seed ρ' from the hashed message μ and a secret key K. The latter does not produce the same digital signature output σ for a particular message since we choose the random seed ρ' completely at random (see step 6 of the sub-routine 10).

In situations where some side channels that exploit determinism are possible [158, 159], we may use the randomized variant of this digital signature scheme. Another case where we may want to avoid determinism is when the signer party does not wish to reveal the content of the message m the sender party is signing. When the digital signature scheme is deterministic, despite not occurring timing leakage of the secret key K, there is a timing leakage of the hashed message μ. For this case, and since we derive the randomness of this signature from the value μ, the number of aborts for a particular message will always be the same.

After defining the key generation, signing, and verification procedures, we need to show this digital signature scheme is correct (or sound) and capable of signing messages and verifying digital signatures correctly (i.e., with a negligible failure probability). However, to prove the correctness of this digital signature, we need to define a set of lemmas that highlight the properties of the algorithms $\texttt{MakeHint}_q$, $\texttt{UseHint}_q$, $\texttt{Decompose}_q$, $\texttt{HighBits}_q$, and $\texttt{LowBits}_q$ [61, 62].

Lemma 1: Suppose that q and α are positive integer numbers, where $q > 2 \times \alpha$, $q \equiv 1 \pmod{\alpha}$, and α is an even number. Let r and z be vectors containing elements in $\mathcal{R}_q$ where $||z||_\infty \leq \frac{\alpha}{2}$, and let h, h' be binary hint values. Then, the $\texttt{HighBits}_q$, $\texttt{MakeHint}_q$, and $\texttt{UseHint}_q$ algorithms satisfy the following properties:

1. $\texttt{UseHint}_q(\texttt{MakeHint}_q(z,\, r,\, \alpha),\, r,\, \alpha) = \texttt{HighBits}_q(r + z,\, \alpha)$;
2. Let $v_1 = \texttt{UseHint}_q(h, r, \alpha)$. Then, we also verify that $||r - v_1 \times \alpha||_\infty \leq \alpha + 1$. Furthermore, if the Hamming weight of the vector h is equal to ω, then all except at most w coefficients of the result $r - v_1 \times \alpha$ will have magnitude at most $\frac{\alpha}{2}$ after centered reduction modulo q using modular multiplications;
3. For any values h and h', if we verify $\texttt{UseHint}_q(h, r, \alpha) = \texttt{UseHint}_q(h', r, \alpha)$, then we can conclude those binary hint values h and h' are equal.

Lemma 2: If we verify that $||s||_\infty \leq \beta$ and $||\texttt{LowBits}_q(r, \alpha)||_\infty < \frac{\alpha}{2} - \beta$, where s is the secret vector on the MLWE Problem formulation and $\beta = \tau \times \eta$. Then, we can also verify that the equality $\texttt{HighBits}_q(r, \alpha) = \texttt{HighBits}_q(r + s, \alpha)$ holds.

Lemma 3: Let $(r_1, r_0) = \texttt{Decompose}_q(r, \alpha)$, $(w_1, w_0) = \texttt{Decompose}_q(r + s, \alpha)$, and $||s||_\infty \leq \beta$, where s is the secret vector on the MLWE Problem formulation and $\beta = \tau \times \eta$. Then, $||s + r_0||_\infty < \frac{\alpha}{2} - \beta \Longleftrightarrow w_1 = r_1 \wedge ||w_0||_\infty < \frac{\alpha}{2} - \beta$ holds.

Lemma 4: Let $q \in \mathbb{Z}$, $\alpha \in \mathbb{Z}$, $r \in \mathbb{Z}_q$, and $z \in \mathbb{Z}_q$, with $||z||_\infty \leq \frac{\alpha}{2}$. Then, we verify the equality $\texttt{UseHint}_q(\texttt{MakeHint}_q(z, r, \alpha),\, r, \alpha) = \texttt{HighBits}_q(r + z, \alpha)$.

Lemma 5: Let $(h, r) \in \{0, 1\} \times \mathbb{Z}_q$ and let $v_1 = \texttt{UseHint}_q(h, r, \alpha)$. If $h = 0$ holds, then we verify $||r - v_1 \times \alpha||_\infty \leq \frac{\alpha}{2}$, else we verify $||r - v_1 \times \alpha||_\infty \leq \alpha + 1$.

Lemma 6: Let $q \in \mathbb{Z}$, $\alpha \in \mathbb{Z}$, $r \in \mathbb{Z}_q$, $h \in \{0,1\}$, and $h' \in \{0,1\}$. If the equality $\texttt{UseHint}_q(h, r, \alpha) = \texttt{UseHint}_q(h', r, \alpha)$ holds, then the equality $h = h'$ also holds.

Now, we need to derive the proofs of the first three lemmas, where the last three lemmas serve as auxiliary lemmas for the proof of each one of the three properties of the first lemma. The respective lemma proofs are shown below:

Proof of Lemma 1:

- ***Proof of Property 1 using Lemma 4:*** The output of the algorithm $\texttt{Decompose}_q$ are given by two integer numbers r_0 and r_1, such that $0 \leq r_1 < \frac{(q-1)}{\alpha}$ and $||r_0||_\infty \leq \frac{\alpha}{2}$. Due to $||z||_\infty \leq \frac{\alpha}{2}$, the integer number $v_1 := \texttt{HighBits}_q(r+z, \alpha)$ either stays with the same value as r_1 or becomes equal to $r_1 \pm 1$ modulo $m = \frac{(q-1)}{\alpha}$. More precisely, if $r_0 > 0$, then $-\frac{\alpha}{2} < r_0 + z \leq \alpha$, which implies that $v_1 = r_1$ or $v_1 = r_1 + 1 \bmod m$. If $r_0 \leq 0$, then $-\alpha \leq r_0 + z \leq \frac{\alpha}{2}$, which implies that $v_1 = r_1$ or $v_1 = r_1 - 1 \bmod m$. The algorithm $\texttt{MakeHint}_q$ checks whether $r_1 = v_1$ and outputs the value 0 in that case, and outputs the value 1 when $r_1 \neq v_1$. The algorithm $\texttt{UseHint}_q$ uses the hint vector h to either output the single value r_1 when we verify $y = 0$ or, depending on whether $r_0 > 0$ holds or not, output either the value $r_1 + 1 \bmod^+ m$ or the value $r_1 - 1 \bmod^+ m$.

- ***Proof of Property 2 using Lemma 5:*** The *Lemma 5* shows that the value r is not too far away from the output value of the algorithm $\texttt{UseHint}_q$ algorithm. This lemma is necessary for the security of the digital signature scheme. Namely, let $(r_1, r_0) := \texttt{Decompose}_q(r, \alpha)$. Then, for each of three possible cases of outputs of the algorithm $\texttt{UseHint}_q$, we have the following enumerated set of results:

1. If we verify $h = 0$, we have the equality $v_1 = r_1$ and the equality given by $r - v_1 \times \alpha = r_1 \times \alpha + r_0 - r_1 \times \alpha = r_0$ with an absolute value at most $\frac{\alpha}{2}$;
2. If we verify $h = 1$ and $r_0 > 0$, we have the equality $v_1 = r_1 + 1 - \kappa \times \frac{(q-1)}{\alpha}$ for $\kappa = 0$ or $\kappa = 1$. Therefore, we have the following mathematical equalities:

$$\begin{aligned} r - v_1 \times \alpha &= r_1 \times \alpha + r_0 - \left(r_1 + 1 - \kappa \times \frac{(q-1)}{\alpha}\right) \times \alpha \\ &= -\alpha + r_0 + \kappa \times (q-1) \end{aligned}$$

   After performing a centered reduction modulo q, the latter mathematical expression $-\alpha + r_0 + \kappa \times (q-1)$ has a magnitude value lower or equal to α;
3. If we verify $h = 1$ and $r_0 \leq 0$, we have the equality $v_1 = r_1 - 1 + \kappa \times \frac{(q-1)}{\alpha}$ for $\kappa = 0$ or $\kappa = 1$. Therefore, we have the following mathematical equalities:

$$\begin{aligned} r - v_1 \times \alpha &= r_1 \times \alpha + r_0 - \left(r_1 - 1 + \kappa \times \frac{(q-1)}{\alpha}\right) \times \alpha \\ &= \alpha + r_0 - \kappa \times (q-1) \end{aligned}$$

   After performing a centered reduction modulo q, the latter mathematical expression $\alpha + r_0 - \kappa \times (q-1)$ has a magnitude value lower or equal to $\alpha + 1$.

• ***Proof of Property 3 using Lemma 6:*** The *Lemma 6* plays a significant role in providing the strong existential unforgeability of the digital signature scheme. In particular, the *Lemma 6* states that two different hint values h and h' cannot lead to the equality $\texttt{UseHint}_q(h, r, \alpha) = \texttt{UseHint}_q(h', r, \alpha)$. Namely, note that $\texttt{UseHint}_q(0, r, \alpha) = r_1$ and $\texttt{UseHint}_q(1, r, \alpha) = (r_1 \pm 1) \bmod^+ \frac{(q-1)}{\alpha}$. Since we have the inequality $\frac{(q-1)}{\alpha} \geq 2$, we have that $r_1 \neq (r_1 \pm 1) \bmod^+ \frac{(q-1)}{\alpha}$.

Proof of Lemma 2: We prove this lemma for integer numbers, rather than vectors of polynomials since the algorithm $\texttt{HighBits}_q$ works independently on each polynomial coefficient. If the inequality $||\texttt{LowBits}_q(r, \alpha)||_\infty < \frac{\alpha}{2} - \beta$ holds, then we have $r = r_1 \times \alpha + r_0$, where $-\frac{\alpha}{2} + \beta < r_0 \leq \frac{\alpha}{2} + \beta$ and $\beta = \tau \times \eta$. Furthermore, we also have $r + s = r_1 \times \alpha + (r_0 + s)$ and $-\frac{\alpha}{2} < r_0 + s \leq \frac{\alpha}{2}$. Thus, we have the equality $r + s \bmod^\pm \alpha = r_0 + s$, as well as the following one:

$$(r + s) - \big((r + s) \bmod^\pm \alpha\big) = r_1 \times \alpha = r - (r \bmod^\pm \alpha)$$

From this mathematical equality, the claim on the *Lemma 2* follows.

Proof of Lemma 3: We first prove this lemma in the forward direction. By the same proof as in *Lemma 2*, since $||r_0 + s||_\infty < \frac{\alpha}{2}$, we know that the equality $w_1 = r_1$ holds, and therefore, we can write $(r + s) = r_1 \times \alpha + r_0 + s$. Since $||r_0 + s||_\infty < \alpha$, we know that the following mathematical equality holds:

$$w_0 = \texttt{LowBits}_q(r + s, \alpha) = r_1 \times \alpha + r_0 + s \bmod^\pm \alpha = r_0 + s$$

To prove the lemma in the other direction, we write the following equality:

$$r_1 \times \alpha + r_0 + s = r + s = w_1 \times \alpha + w_0 = r_1 \times \alpha + w_0$$

Therefore, we obtain the final mathematical equality $r_0 + s = w_0$.

Recapping, the $\texttt{Power2Round}_q$ algorithm splits (and compresses) the polynomial coefficients of the target vector into rounded and remainder parts to reduce their magnitude and ensure efficient processing of the digital signature scheme. The $\texttt{MakeHint}_q$ algorithm generates a hint vector to help reconstruct the high-order polynomial coefficient bits in the verification procedure. The $\texttt{UseHint}_q$ algorithm uses the hint vector generated by the $\texttt{MakeHint}_q$ algorithm to adjust the polynomial coefficients, ensuring they match during verification. The $\texttt{Decompose}_q$ algorithm breaks down the polynomial coefficients into high-order and low-order parts for specific operations. The $\texttt{HighBits}_q$ and $\texttt{LowBits}_q$ algorithms extract the high-order and low-order bits of the polynomial coefficient, respectively.

Now that we already defined the set of lemmas that highlight the properties of the algorithms $\texttt{MakeHint}_q$, $\texttt{UseHint}_q$, $\texttt{Decompose}_q$, $\texttt{HighBits}_q$, and $\texttt{LowBits}_q$, supporting the digital signature scheme and demonstrated the respective proofs, we can show the correctness of the CRYSTALS-Dilithium cryptosystem.

Regarding the details of the correctness property of the CRYSTALS-Dilithium digital signature scheme, we have the following enumerated mathematical proof:

- If $\left|\left|c \times t_0\right|\right|_\infty < \gamma_2$, then by *Lemma 1* introduced before, we know that:

  $$\texttt{UseHint}_q(h, w - c \times e + c \times t_0, 2 \times \gamma_2) = \texttt{HighBits}_q(w - c \times e, 2 \times \gamma_2)$$

- Since we verify the results $w = A \times y$ and $t = A \times s + e$, we have that:

  $$w - c \times e = A \times y - c \times e = A \times (z - c \times s) - c \times e = A \times z - c \times t$$

  Additionally, we also verify that the following mathematical equality holds:

  $$w - c \times e + c \times t_0 = A \times z - c \times t_1 \times 2^{d_t}$$

- Then, the party that verifies if the digital signature is valid, computes:

  $$\texttt{UseHint}_q(h, A \times z - c \times t_1 \times 2^{d_t}, 2 \times \gamma_2) = \texttt{HighBits}_q(w - c \times e, 2 \times \gamma_2)$$

- Furthermore, because we set $\beta = \tau \times \eta$, such we verify that $||c \times e||_\infty \leq \beta$ and the signer party checks that $\texttt{LowBits}_q(w - c \times e,\ 2 \times \gamma_2) < \gamma_2 - \beta$ (see the steps 14 and 16 of the sub-routine 10). Then, the *Lemma 2* implies that:

  $$\begin{aligned}\texttt{HighBits}_q(w - c \times e, 2 \times \gamma_2) &= \texttt{HighBits}_q(w - c \times e + c \times e, 2 \times \gamma_2) \\ &= \texttt{HighBits}_q(w, 2 \times \gamma_2) = w_1\end{aligned}$$

- Therefore, the vector w_1' computed by the verifier party to be the input of the hash function is equal to the vector w_1 calculated by the signer party (see the step 5 and 7 of the sub-routine 11, as well as the step 10 and 11 of the sub-routine 10). Thus, the verification procedure will always accept the digital signature produced in this case, verifying the correctness property.

A Demonstration of CRYSTALS Cryptosuite

In the following GitHub link, we can find a code demonstration and analysis of both CRYSTALS-Kyber cryptosystem and CRYSTALS-Dilithium digital signature scheme, built on Script of *Script (SoS) Notebook* (*Java* and *Python*):

- https://github.com/rubenandrebarreiro/crystals-kyber-and-dilithium-study-and-analysis

References

1. Lov Grover. A Fast Quantum Mechanical Algorithm for Database Search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 212–219, New York, NY, USA, 1996. Association for Computing Machinery.
2. Daniel Simon. On the Power of Quantum Computation. *SIAM Journal on Computing*, 26(5):1474–1483, 1997.
3. Gilles Brassard, Peter HØyer, and Alain Tapp. Quantum Cryptanalysis of Hash and Claw-Free Functions. In Cláudio Lucchesi and Arnaldo Moura, editors, *LATIN'98: Theoretical Informatics*, pages 163–169, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.
4. Peter Shor. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.
5. Peter Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.
6. Joan Daemen and Vincent Rijmen. The Block Cipher Rijndael. In Jean-Jacques Quisquater and Bruce Schneier, editors, *Smart Card Research and Applications*, pages 277–284, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.
7. Morris Dworkin. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, 2015.
8. Ron Rivest, Adi Shamir, and Leonard Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.
9. Whitfield Diffie and Martin Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
10. Ralph Merkle. Secure Communications Over Insecure Channels. *Commun. ACM*, 21(4):294–299, 1978.
11. Taher Elgamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.
12. Claus Schnorr. Efficient Identification and Signatures for Smart Cards. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 239–252, New York, NY, 1990. Springer New York.
13. Claus Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991.
14. Lily Chen, Dustin Moody, Andrew Regenscheid, and Angela Robinson. Digital Signature Standard (DSS), 2023.
15. Victor Miller. Use of Elliptic Curves in Cryptography. In Hugh Williams, editor, *Advances in Cryptology — CRYPTO '85 Proceedings*, pages 417–426, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg.
16. Neal Koblitz. Elliptic Curve Cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.
17. Daniel Bernstein, Johannes Buchmann, and Erik Dahmen. *Post-Quantum Cryptography.* Springer Berlin Heidelberg, 2009.
18. Daniel Bernstein and Tanja Lange. Post-Quantum Cryptography. *Nature*, 549(7671):188–194, 2017.
19. Stephen Wiesner. Conjugate Coding. *SIGACT News*, 15(1):78–88, 1983.

20. Gilles Brassard and Claude Crépeau. *Quantum Cryptography*, pages 495–500. Springer US, Boston, MA, 2005.
21. Artur Ekert. Quantum Cryptography Based on Bell's Theorem. *Phys. Rev. Lett.*, 67:661–663, 1991.
22. Stefano Pirandola, Ulrik Andersen, Leonardo Banchi, Mario Berta, Darius Bunandar, Roger Colbeck, Dirk Englund, Tobias Gehring, Cosmo Lupo, Carlo Ottaviani, Jason Pereira, Mohsen Razavi, Jesni Shaari, Marco Tomamichel, Vladyslav Usenko, Giuseppe Vallone, Paolo Villoresi, and Petros Wallden. Advances in Quantum Cryptography. *Advances in Optics and Photonics*, 12(4):1012, 2020.
23. Federico Grasselli. *Quantum Cryptography: From Key Distribution to Conference Key Agreement.* Quantum Science and Technology. Springer International Publishing, 2021.
24. Thomas Vidick and Stephanie Wehner. *Introduction to Quantum Cryptography.* Cambridge University Press, 2023.
25. Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-Key Cryptosystems From Lattice Reduction Problems. In Burton Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, pages 112–131, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.
26. Jeffrey Hoffstein, Jill Pipher, and Joseph Silverman. NTRU: A Ring-Based Public Key Cryptosystem. In Joe Buhler, editor, *Algorithmic Number Theory*, pages 267–288, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.
27. Oded Regev. On Lattices, Learning With Errors, Random Linear Codes, and Cryptography. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '05, page 84–93, New York, NY, USA, 2005. Association for Computing Machinery.
28. Robert McEliece. A Public Key Cryptosystem Based on Algebraic Coding Theory. *Jet Propulsion Laboratory DSN Progress Report*, 4244:114–116, 1978.
29. Harald Niederreiter. Knapsack-Type Cryptosystems and Algebraic Coding Theory. *Prob. Contr. Inform. Theory*, 15(2):157–166, 1986.
30. Nicolas Courtois, Matthieu Finiasz, and Nicolas Sendrier. How to Achieve a McEliece-Based Digital Signature Scheme. In Colin Boyd, editor, *Advances in Cryptology — ASIACRYPT 2001*, pages 157–174, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.
31. Leslie Lamport. Constructing Digital Signatures From a One Way Function. Technical Report CSL-98, Microsoft, 1979. This paper was published by IEEE in the Proceedings of HICSS-43 in January, 2010.
32. Ralph Merkle. *Secrecy, Authentication, and Public Key Systems.* PhD thesis, Stanford University, Stanford, CA, USA, 1979. AAI8001972.
33. Johannes Buchmann, Erik Dahmen, and Andreas Hülsing. XMSS - A Practical Forward Secure Signature Scheme Based on Minimal Security Assumptions. In Bo-Yin Yang, editor, *Post-Quantum Cryptography*, pages 117–129, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
34. Luca De Feo, David Jao, and Jérôme Plût. Towards Quantum-Resistant Cryptosystems From Supersingular Elliptic Curve Isogenies. *Journal of Mathematical Cryptology*, 8(3):209–247, 2014.
35. Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. Csidh: An efficient post-quantum commutative group action. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018*, pages 395–427, Cham, 2018. Springer International Publishing.

36. Luca De Feo, David Kohel, Antonin Leroux, Christophe Petit, and Benjamin Wesolowski. SQISign: Compact Post-Quantum Signatures From Quaternions and Isogenies. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020*, pages 64–93, Cham, 2020. Springer International Publishing.
37. Tsutomu Matsumoto and Hideki Imai. Public Quadratic Polynomial-Tuples for Efficient Signature-Verification and Message-Encryption. In David Barstow, Wilfried Brauer, Per Brinch, David Gries, David Luckham, Cleve Moler, Amir Pnueli, Gerhard Seegmüller, Josef Stoer, Niklaus Wirth, and Christoph Günther, editors, *Advances in Cryptology — EUROCRYPT '88*, pages 419–453, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.
38. Jacques Patarin. Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms. In Ueli Maurer, editor, *Advances in Cryptology — EUROCRYPT '96*, pages 33–48, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.
39. Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced Oil and Vinegar Signature Schemes. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, pages 206–222, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.
40. Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-Quantum Zero-Knowledge and Signatures From Symmetric-Key Primitives. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 1825–1842, New York, NY, USA, 2017. Association for Computing Machinery.
41. Vadim Lyubashevsky, Ngoc Nguyen, and Maxime Plancon. Lattice-Based Zero-Knowledge Proofs and Applications: Shorter, Simpler, and More General. Cryptology ePrint Archive, Paper 2022/284, 2022.
42. Charles Bennett and Gilles Brassard. Quantum Cryptography: Public Key Distribution and Coin Tossing. *Theoretical Computer Science*, 560:7–11, 2014. Theoretical Aspects of Quantum Cryptography – celebrating 30 years of BB84.
43. Mark Hillery. Quantum Cryptography With Squeezed States. *Phys. Rev. A*, 61:022309, 2000.
44. Frédéric Grosshans and Philippe Grangier. Continuous variable quantum cryptography using coherent states. *Phys. Rev. Lett.*, 88:057902, 2002.
45. Frédéric Grosshans, Gilles Van Assche, Jérôme Wenger, Rosa Brouri, Nicolas Cerf, and Philippe Grangier. Quantum Key Distribution Using Gaussian-Modulated Coherent States. *Nature*, 421(6920):238–241, 2003.
46. Michel Boyer, Dan Kenigsberg, and Tal Mor. Quantum Key Distribution with Classical Bob. *Phys. Rev. Lett.*, 99:140501, 2007.
47. Michel Boyer, Ran Gelles, Dan Kenigsberg, and Tal Mor. Semiquantum Key Distribution. *Physical Review A*, 79(3), 2009.
48. Michael Epping, Hermann Kampermann, Dagmar Bruß, and Chiara Macchiavello. Multi-Partite Entanglement Can Speed Up Quantum Key Distribution in Networks. *New Journal of Physics*, 19(9):093012, 2017.
49. Federico Grasselli, Hermann Kampermann, and Dagmar Bruß. Finite-Key Effects in Multipartite Quantum Key Distribution Protocols. *New Journal of Physics*, 20(11):113014, 2018.
50. Gláucia Murta, Federico Grasselli, Hermann Kampermann, and Dagmar Bruß. Quantum Conference Key Agreement: A Review. *Advanced Quantum Technologies*, 3(11), 2020.

51. Charles Bennett, Gilles Brassard, Claude Crépeau, and Marie-Hélèn Skubiszewska. Practical Quantum Oblivious Transfer. In Joan Feigenbaum, editor, *Advances in Cryptology — CRYPTO '91*, pages 351–366, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.
52. Alex Grilo, Huijia Lin, Fang Song, and Vinod Vaikuntanathan. Oblivious Transfer is in MiniQCrypt. Cryptology ePrint Archive, Paper 2020/1500, 2020. `https://eprint.iacr.org/2020/1500`.
53. Daniel Gottesman and Isaac Chuang. Quantum Digital Signatures, 2001.
54. Dan Li, Jie Zhang, Fen-Zhuo Guo, Wei Huang, Qiao-Yan Wen, and Hui Chen. Discrete-Time Interacting Quantum Walks and Quantum Hash Schemes. *Quantum Information Processing*, 12:1501–1513, 2013.
55. Yu-Guang Yang, Peng Xu, Rui Yang, Yi-Hua Zhou, and Wei-Min Shi. Quantum Hash Function and its Application to Privacy Amplification in Quantum Key Distribution, Pseudo-Random Number Generation and Image Encryption. *Scientific Reports*, 6(1):19788, 2016.
56. Gilles Brassard and Claude Crépeau. Quantum Bit Commitment and Coin Tossing Protocols. In Alfred Menezes and Scott Vanstone, editors, *Advances in Cryptology-CRYPTO' 90*, pages 49–61, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.
57. Dominic Mayers. Unconditionally Secure Quantum Bit Commitment is Impossible. *Phys. Rev. Lett.*, 78:3414–3417, 1997.
58. Hoi-Kwong Lo and Hoi Chau. Why Quantum Bit Commitment and Ideal Quantum Coin Tossing Are Impossible. *Physica D: Nonlinear Phenomena*, 120(1):177–187, 1998. Proceedings of the Fourth Workshop on Physics and Consumption.
59. Joppe Bos, Leo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, John Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehle. Crystals - kyber: A cca-secure module-lattice-based kem. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 353–367, 2018.
60. Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Kyber Algorithm Specifications and Supporting Documentation. *NIST PQC Round*, 2(4):1–43, 2019.
61. Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium Algorithm Specifications and Supporting Documentation. *NIST PQC Round*, 1(3), 2017.
62. Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé'. CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(1):238–268, 2018.
63. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for Hard Lattices and New Cryptographic Constructions. Cryptology ePrint Archive, Paper 2007/432, 2007.
64. Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon: Fast-Fourier Lattice-Based Compact Signatures Over NTRU. *NIST PQC Round*, 1(3), 2019.
65. Daniel Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, and Zooko Wilcox-O'Hearn. SPHINCS: Practical Stateless Hash-Based Signatures. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, pages 368–397, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

66. Daniel Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. The SPHINCS+ Signature Framework. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2129–2146, 2019.
67. Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo Barreto. MDPC-McEliece: New McEliece Variants From Moderate Density Parity-Check Codes. In *2013 IEEE International Symposium on Information Theory*, pages 2069–2073, 2013.
68. Nicolas Aragon, Paulo Barreto, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Santosh Ghosh, Shay Gueron, Tim Güneysu, et al. BIKE: Bit Flipping Key Encapsulation. *NIST PQC Round*, 2(4), 2022.
69. Martin Albrecht, Daniel Bernstein, Tung Chou, Carlos Cid, Jan Gilcher, Tanja Lange, Varun Maram, Ingo Von Maurich, Rafael Misoczki, Ruben Niederhagen, et al. Classic McEliece: Conservative Code-Based Cryptography. *NIST PQC Round*, 2(4), 2022.
70. Carlos Melchor, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, and Gilles Zémor. Efficient Encryption From Random Quasi-Cyclic Codes. *IEEE Transactions on Information Theory*, 64(5):3927–3943, 2018.
71. Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loıc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, Gilles Zémor, and IC Bourges. Hamming Quasi-Cyclic (HQC). *NIST PQC Round*, 2(4):13, 2018.
72. Christopher Peikert. Lattice Cryptography for the Internet. Cryptology ePrint Archive, Paper 2014/070, 2014.
73. Daniel Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine Van Vredendaal. NTRU Prime. *IACR Cryptol. ePrint Arch.*, 2016:461, 2016.
74. Erdem Alkim, Joppe Bos, Léo Ducas, Patrick Longa, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Christopher Peikert, Ananth Raghunathan, and Douglas Stebila. FrodoKEM Learning With Errors Key Encapsulation Algorithm. *National Institute of Standards and Technology*, 2017.
75. Bernstein, Daniel and Chuengsatiansup, Chitchanok and Lange, Tanja and van Vredendaal, Christine. NTRU Prime: Reducing Attack Surface at Low Cost. In *Selected Areas in Cryptography–SAC 2017: 24th International Conference, Ottawa, ON, Canada, August 16-18, 2017, Revised Selected Papers 24*, pages 235–260. Springer, 2018.
76. Erdem Alkim, Joppe W Bos, Léo Ducas, Patrick Longa, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Chris Peikert, Ananth Raghunathan, Douglas Stebila, et al. FrodoKEM Learning With Errors Key Encapsulation. *NIST PQC Standardization*, 3:10–20, 2020.
77. Erdem Alkim, Roberto Avanzi, Joppe Bos, Leo Ducas, Antonio de la Piedra, Thomas Pöppelmann, Peter Schwabe, Douglas Stebila, Martin Albrecht, Emmanuela Orsini, et al. NewHope—Submission to the NIST Post-Quantum Project. *Specification Document (Part of the Submission Package)*, 2019.
78. Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-Quantum Key Exchange - A New Hope. Cryptology ePrint Archive, Paper 2015/1092, 2015.
79. Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. Practical Lattice-Based Cryptography: A Signature Scheme for Embedded Systems. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded*

Systems – CHES 2012, pages 530–547, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

80. Erdem Alkim, Paulo Barreto, Nina Bindel, Juliane Kramer, Patrick Longa, and Jefferson Ricardini. The Lattice-Based Digital Signature Scheme qTESLA. Cryptology ePrint Archive, Paper 2019/085, 2019.
81. Kamel Bentahar, Daniel Page, Markku-Juhani Saarinen, Joseph Silverman, and Nigel Smart. Lash. In *NIST: The Second Cryptographic Hash Workshop*, 2006.
82. Vadim Lyubashevsky, Daniele Micciancio, Christopher Peikert, and Alon Rosen. SWIFFT: A Modest Proposal for FFT Hashing. In *Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers*, volume 5086 of *Lecture Notes in Computer Science*, pages 54–72. Springer, 2008.
83. Vadim Lyubashevsky, Daniele Micciancio, Christopher Peikert, and Alon Rosen. SWIFFTX: A Proposal for the SHA-3 Standard, 2008.
84. Craig Gentry. *A Fully Homomorphic Encryption Scheme.* PhD thesis, Stanford University, Stanford, CA, USA, 2009. AAI3382729.
85. Zvika Brakerski and Vinod Vaikuntanathan. Efficient Fully Homomorphic Encryption From (Standard) LWE. Cryptology ePrint Archive, Paper 2011/344, 2011.
86. Zvika Brakerski and Vinod Vaikuntanathan. Lattice-Based FHE as Secure as PKE. Cryptology ePrint Archive, Paper 2013/541, 2013.
87. Daniele Micciancio and Salil Vadhan. Statistical Zero-Knowledge Proofs with Efficient Provers: Lattice Problems and More. In *Advances in Cryptology-CRYPTO 2003: 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003. Proceedings 23*, pages 282–298. Springer, 2003.
88. Fabrice Benhamouda, Jan Camenisch, Stephan Krenn, Vadim Lyubashevsky, and Gregory Neven. Better Zero-Knowledge Proofs for Lattice Encryption and their Application to Group Signatures. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 551–572. Springer, 2014.
89. Jonathan Bootle, Vadim Lyubashevsky, and Gregor Seiler. Algebraic Techniques for Short (er) Exact Lattice-Based Zero-Knowledge Proofs. In *Annual International Cryptology Conference*, pages 176–202. Springer, 2019.
90. Muhammed Esgin, Ron Steinfeld, Joseph Liu, and Dongxi Liu. Lattice-Based Zero-Knowledge Proofs: New Techniques for Shorter and Faster Constructions and Applications. In *Annual International Cryptology Conference*, pages 115–146. Springer, 2019.
91. Christopher Peikert, Vinod Vaikuntanathan, and Brent Waters. A Framework for Efficient and Composable Oblivious Transfer. In *Annual international cryptology conference*, pages 554–571. Springer, 2008.
92. Benoît Libert, San Ling, Fabrice Mouhartem, Khoa Nguyen, and Huaxiong Wang. Adaptive Oblivious Transfer With Access Control From Lattice Assumptions. In *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23*, pages 533–563. Springer, 2017.
93. Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-Homomorphic Encryption and Multiparty Computation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 169–188. Springer, 2011.

94. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional Encryption with Bounded Collusions via Multi-Party Computation. In *Advances in Cryptology–CRYPTO 2012: 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 162–179. Springer, 2012.
95. Pratyay Mukherjee and Daniel Wichs. Two Round Multiparty Computation via Multi-Key FHE. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016*, pages 735–763, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.
96. Miklós Ajtai. Generating Hard Instances of Lattice Problems. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 99–108, New York, NY, USA, 1996. Association for Computing Machinery.
97. Miklós Ajtai. The Shortest Vector Problem in L2 is NP-Hard for Randomized Reductions. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, page 10–19, New York, NY, USA, 1998. Association for Computing Machinery.
98. Daniele Micciancio. Generalized Compact Knapsacks, Cyclic Lattices, and Efficient One-Way Functions. *Computational Complexity*, 16(4):365–411, 2007.
99. Miklós Ajtai and Cynthia Dwork. A Public-Key Cryptosystem With Worst-Case/Average-Case Equivalence. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 284–293, 1997.
100. Arjen Lenstra, Hendrik Lenstra, and László Lovász. Factoring Polynomials With Rational Coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.
101. László Babai. On Lovász' Lattice Reduction and the Nearest Lattice Point Problem. *Combinatorica*, 6(1):1–13, 1986.
102. David Naccache and Jacques Stern. A New Public Key Cryptosystem Based on Higher Residues. In *Proceedings of the 5th ACM Conference on Computer and Communications Security*, CCS '98, page 59–66, New York, NY, USA, 1998. Association for Computing Machinery.
103. Masaya Yasuda. A Survey of Solving SVP Algorithms and Recent Strategies for Solving the SVP Challenge. In Tsuyoshi Takagi, Masato Wakayama, Keisuke Tanaka, Noboru Kunihiro, Kazufumi Kimoto, and Yasuhiko Ikematsu, editors, *International Symposium on Mathematics, Quantum Theory, and Cryptography*, pages 189–207, Singapore, 2021. Springer Singapore.
104. Aleksandr Korkine and Yegor Zolotarev. Sur Les Formes Quadratiques Positives. *Mathematische Annalen*, 11(2):242–292, 1877.
105. Ravi Kannan. Improved Algorithms for Integer Programming and Related Lattice Problems. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, STOC '83, page 193–206, New York, NY, USA, 1983. Association for Computing Machinery.
106. Claus-Peter Schnorr and Martin Euchner. Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems. *Mathematical Programming*, 66(1):181–199, 1994.
107. Ulrich Fincke and Michael Pohst. Improved Methods for Calculating Vectors of Short Length in a Lattice, Including a Complexity Analysis. *Mathematics of Computation*, 44(170):463–471, 1985.
108. Ravi Kannan. Minkowski's Convex Body Theorem and Integer Programming. *Mathematics of Operations Research*, 12(3):415–440, 1987.
109. Guillaume Hanrot and Damien Stehlé. Improved Analysis of Kannan's Shortest Lattice Vector Algorithm. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007*, pages 170–186, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

110. Xavier Pujol and Damien Stehlé. Rigorous and Efficient Short Lattice Vectors Enumeration. In Josef Pieprzyk, editor, *Advances in Cryptology - ASIACRYPT 2008*, pages 390–405, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.
111. Daniele Micciancio and Michael Walter. *Fast Lattice Point Enumeration with Minimal Overhead*, pages 276–294. Association for Computing Machinery, 2015.
112. Miklós Ajtai, Ravi Kumar, and Dandapani Sivakumar. A Sieve Algorithm for the Shortest Lattice Vector Problem. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, STOC '01, page 601–610, New York, NY, USA, 2001. Association for Computing Machinery.
113. Phong Nguyen and Thomas Vidick. Sieve Algorithms for the Shortest Vector Problem are Practical. *Journal of Mathematical Cryptology*, 2(2):181–207, 2008.
114. Daniele Micciancio and Panagiotis Voulgaris. *Faster Exponential Time Algorithms for the Shortest Vector Problem*, pages 1468–1480. Association for Computing Machinery, 2010.
115. Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. Algorithms for the Shortest and Closest Lattice Vector Problems. In Yeow Meng Chee, Zhenbo Guo, San Ling, Fengjing Shao, Yuansheng Tang, Huaxiong Wang, and Chaoping Xing, editors, *Coding and Cryptology*, pages 159–190, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
116. Naftali Sommer, Meir Feder, and Ofir Shalvi. Finding the closest lattice point by iterative slicing. *SIAM Journal on Discrete Mathematics*, 23(2):715–731, 2009.
117. Daniele Micciancio and Panagiotis Voulgaris. A Deterministic Single Exponential Time Algorithm for Most Lattice Problems Based on Voronoi Cell Computations. *SIAM Journal on Computing*, 42(3):1364–1391, 2013.
118. Robby McKilliam, Alex Grant, and Vaughan Clarkson. Finding a Closest Point in a Lattice of Voronoi's First Kind. *SIAM Journal on Discrete Mathematics*, 28(3):1405–1422, 2014.
119. Daniel Dadush and Nicolas Bonifas. *Short Paths on the Voronoi Graph and Closest Vector Problem with Preprocessing*, pages 295–314. Association for Computing Machinery, 2015.
120. Georges Voronoi. Nouvelles Applications des Paramètres Continus à La Théorie des Formes Quadratiques. Premier Mémoire. Sur Quelques Propriétés des Formes Quadratiques Positives Parfaites. *Journal für Die Reine und Angewandte Mathematik (Crelles Journal)*, 1908(133):97–102, 1908.
121. Boris Delaunay. Sur La Sphère Vide. *Bulletin de L'Académie des Sciences de L'URSS. Classe des Sciences Mathématiques et Na*, 1934(6):793–800, 1934.
122. Christoph Ludwig. A Faster Lattice Reduction Method Using Quantum Search. In *Algorithms and Computation: 14th International Symposium, ISAAC 2003, Kyoto, Japan, December 15-17, 2003. Proceedings 14*, pages 199–208. Springer, 2003.
123. Andris Ambainis, Julia Kempe, and Or Sattath. A Quantum Lovász Local Lemma. *Journal of the ACM (JACM)*, 59(5):1–24, 2012.
124. Marcel Tiepelt and Alan Szepieniec. Quantum LLL with an Application to Mersenne Number Cryptosystems. In *Progress in Cryptology–LATINCRYPT 2019: 6th International Conference on Cryptology and Information Security in Latin America, Santiago de Chile, Chile, October 2–4, 2019, Proceedings 6*, pages 3–23. Springer, 2019.
125. András Gilyén. *Quantum Singular Value Transformation & its Algorithmic Applications.* PhD thesis, University of Amsterdam, 2019.

126. Thijs Laarhoven, Michele Mosca, and Joop Van De Pol. Finding Shortest Lattice Vectors Faster Using Quantum Search. *Designs, Codes and Cryptography*, 77:375–400, 2015.
127. Elena Kirshanova, Erik Mårtensson, Eamonn Postlethwaite, and Subhayan Roy Moulik. Quantum Algorithms for the Approximate k-List Problem and their Application to Lattice Sieving. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 521–551. Springer, 2019.
128. André Chailloux and Johanna Loyer. Lattice Sieving via Quantum Random Walks. In *Advances in Cryptology–ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part IV 27*, pages 63–91. Springer, 2021.
129. Lihui Lv, Bao Yan, Hong Wang, Zhi Ma, Yangyang Fei, Xiangdong Meng, and Qianheng Duan. Using Variational Quantum Algorithm to Solve the LWE Problem. *Entropy*, 24(10), 2022.
130. Sergey Grebnev, Maxim Gavreev, Evgeniy Kiktenko, Anton Guglya, Albert Efimov, and Aleksey Fedorov. Pitfalls of the Sublinear QAOA-Based Factorization Algorithm. *IEEE Access*, 11:134760–134768, 2023.
131. Daniel Bernstein, Stacey Jeffery, Tanja Lange, and Alexander Meurer. Quantum Algorithms for the Subset-Sum Problem. In *Post-Quantum Cryptography: 5th International Workshop, PQCrypto 2013, Limoges, France, June 4-7, 2013. Proceedings 5*, pages 16–33. Springer, 2013.
132. Karl Pearson. The Problem of the Random Walk. *Nature*, 72(1865):294–294, 1905.
133. Neil Shenvi, Julia Kempe, and Birgitta Whaley. Quantum Random-Walk Search Algorithm. *Phys. Rev. A*, 67:052307, 2003.
134. David Joseph, Adam Callison, Cong Ling, and Florian Mintert. Two Quantum Ising Algorithms for the Shortest-Vector Problem. *Physical Review A*, 103(3):032433, 2021.
135. Manohar Raavi, Simeon Wuthier, Pranav Chandramouli, Yaroslav Balytskyi, Xiaobo Zhou, and Sang-Yoon Chang. Security Comparisons and Performance Analyses of Post-quantum Signature Algorithms. In Kazue Sako and Nils Ole Tippenhauer, editors, *Applied Cryptography and Network Security*, pages 424–447, Cham, 2021. Springer International Publishing.
136. Junpei Yamaguchi, Toshiya Shimizu, Kazuyoshi Furukawa, Ryuichi Ohori, Takeshi Shimoyama, Avradip Mandal, Hart Montgomery, Arnab Roy, and Takuya Ohwa. Annealing-Based Algorithm for Solving CVP and SVP. *Journal of the Operations Research Society of Japan*, 65(3):121–137, 2022.
137. Nicolas Gama and Phong Nguyen. Predicting Lattice Reduction. In Nigel Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, pages 31–51, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.
138. Yuanmi Chen and Phong Nguyen. BKZ 2.0: Better Lattice Security Estimates. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, pages 1–20, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
139. Claus-Peter Schnorr and Taras Shevchenko. Solving Subset Sum Problems of Density Close to 1 by "Randomized" BKZ-Reduction. Cryptology ePrint Archive, Paper 2012/620, 2012.
140. Mokammel Haque and Mohammad Rahman. Analyzing Progressive-BKZ Lattice Reduction Algorithm. *International Journal of Computer Network and Information Security*, 2019.

141. Mokammel Haque and Josef Pieprzyk. Optimizing Preprocessing Method of Recursive-BKZ Lattice Reduction Algorithm. In *2015 2nd International Conference on Electrical Information and Communication Technologies (EICT)*, pages 19–23, 2015.
142. Daniele Micciancio and Michael Walter. Practical, Predictable Lattice Basis Reduction. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016*, pages 820–849, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.
143. Shi Bai, Thijs Laarhoven, and Damien Stehlé. Tuple Lattice Sieving. *LMS Journal of Computation and Mathematics*, 19(A):146–162, 2016.
144. Martin Albrecht, Miloš Prokop, Yixin Shen, and Petros Wallden. Variational Quantum Solutions to the Shortest Vector Problem. *Quantum*, 7:933, 2023.
145. Vadim Lyubashevsky, Christopher Peikert, and Oded Regev. On Ideal Lattices and Learning With Errors Over Rings. *J. ACM*, 60(6), 2013.
146. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) Fully Homomorphic Encryption Without Bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, page 309–325, New York, NY, USA, 2012. Association for Computing Machinery.
147. Adeline Langlois and Damien Stehlé. Worst-Case to Average-Case Reductions for Module Lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015.
148. Vadim Lyubashevsky and Gregor Seiler. NTTRU: Truly Fast NTRU Using NTT. Cryptology ePrint Archive, Paper 2019/040, 2019.
149. Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A Modular Analysis of the Fujisaki-Okamoto Transformation. Cryptology ePrint Archive, Paper 2017/604, 2017.
150. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. *Journal of Cryptology*, 26(1):80–101, 2013.
151. Tsunekazu Saito, Keita Xagawa, and Takashi Yamakawa. Tightly-Secure Key-Encapsulation Mechanism in the Quantum Random Oracle Model. Cryptology ePrint Archive, Paper 2017/1005, 2017.
152. Ehsan Targhi and Dominique Unruh. Quantum Security of the Fujisaki-Okamoto and OAEP Transforms. Cryptology ePrint Archive, Paper 2015/1210, 2015.
153. Vadim Lyubashevsky. Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures. In *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, volume 5912 of *Lecture Notes in Computer Science*, pages 598–616. Springer, 2009.
154. Vadim Lyubashevsky. Lattice Signatures Without Trapdoors. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, pages 738–755, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
155. Amos Fiat and Adi Shamir. How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In Andrew Odlyzko, editor, *Advances in Cryptology — CRYPTO' 86*, pages 186–194, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg.
156. Shi Bai and Steven Galbraith. An Improved Compression Technique for Signatures Based on Learning With Errors. Cryptology ePrint Archive, Paper 2013/838, 2013.
157. Ramesh Agarwal and Sidney Burrus. Number Theoretic Transforms to Implement Fast Digital Convolution. *Proceedings of the IEEE*, 63(4):550–560, 1975.

158. Niels Samwel, Lejla Batina, Guido Bertoni, Joan Daemen, and Ruggero Susella. Breaking Ed25519 in WolfSSL. In *Topics in Cryptology–CT-RSA 2018: The Cryptographers' Track at the RSA Conference 2018, San Francisco, CA, USA, April 16-20, 2018, Proceedings*, pages 1–20. Springer, 2018.
159. Damian Poddebniak, Juraj Somorovsky, Sebastian Schinzel, Manfred Lochter, and Paul Rösler. Attacking Deterministic Signature Schemes Using Fault Attacks. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 338–352, 2018.