Lattice-Based Post-Quantum Cryptography

Enrique Pérez Haro and Pablo Gutiérrez Félix

1 Overview

As quantum computing hardware advances, its security implications are of paramount interest to decision-makers in the field of cybersecurity, particularly in cryptography. Shor's algorithm has been shown to significantly reduce the computational hardness of widely used public-key cryptosystems that rely on mathematical problems such as prime factorization, the discrete logarithm problem, and elliptic curve cryptography. Consequently, Post-Quantum Cryptography (PQC) aims to develop new cryptographic schemes based on mathematical problems that remain computationally infeasible to solve, even with quantum computers. Depending on the mathematical problem at hand, different paradigms within PQC have emerged: hash-based, multivariate, code-based, isogeny-based, or lattice-based cryptography. This report focuses on the latter, aiming to provide insights into its historical development, mathematical underpinnings, key algorithms, and the challenges it faces.

As will be described later, lattice-based cryptography has emerged as the main candidate to lead the PQC migration from the various PQC paradigms already mentioned. In fact, the National Institute of Standards and Technology (NIST) recently released the standards for post-quantum algorithms, which are now recommended for cryptographic use. Two out of the three standardized algorithms rely on lattice-based cryptographic schemes: CRYSTALS-Kyber (renamed ML-KEM) as a Key Encapsulation Mechanism (KEM), and CRYSTALS-Dilithium (renamed ML-DSA) for digital signatures. As a result, this exploration aims to provide key insights into the mathematical functioning of such algorithms, as well as an overview of the current investigative landscape within the field of lattice-based cryptography.

The use of lattices in modern cryptosystems dates back to 1996 when Aijtai demonstrated that solving a random instance of the Small Integer Solution (SIS) problem is as challenging as solving certain problems that are believed to be difficult for all lattices [1]. This "reduction" reassures that the computational hardness found within the SIS problem (and, therefore, the other believed-to-be complex lattice problems) is not a coincidence linked to a specific instance, but rather linked to deep, fundamental properties of lattices, making them suitable for cryptographic schemes. Aijtai and Dwork would later propose a public-key encryption scheme based on these security guarantees but proved to be inefficient and complex in its implementation, providing more of a theoretical approach than a tangible result [2]. During that time, in 1998, Hoffstein, Pipher, and Silverman developed NTRU, a public-key encryption scheme based on a lattice problem over polynomial rings, a new potentially hard lattice-based problem [6]. However, the security of NTRU is based on its best-known attacks, rather than on a theoretical foundation.

In 2009, Regev suggested a new type of lattice-based problem, the Learning with Errors

(LWE) problem, a flexible problem that perfectly suits cryptographic systems [9]. Similar to what Aijtai suggested slightly over a decade before, Regev proved that solving a particular instance of the LWE problem was as complex as any known-to-be-difficult lattice-based problem. With this, cryptographers built upon these previously mentioned works, designing new cryptosystems that were theoretically robust yet secure and efficient in their implementation. This is how lattice-based algorithms became the best-positioned proposal to redesign public key cryptography for the quantum era.

2 Lattices

A lattice is an abstract mathematical structure that encompasses a set of points in an n-dimensional space with a periodic structure. The points in the lattice ($\mathcal{L}$) are generated by a basis matrix $B \in \mathbb{R}^{n \times m}$ which contains the vectors $b_1, \ldots, b_m \in \mathbb{R}^n$ in its columns, where m is the number of columns in B. The resultant lattice can be defined through the following set

$$\mathcal{L}(b_1, \ldots, b_m) = \left\{ \sum_{i=1}^{m} x_i b_i : x_i \in \mathbb{Z} \right\}$$

The same expression can be described using its equivalent matrix definition

$$\mathcal{L}(b_1, \ldots, b_m) = \{B\vec{x} : \vec{x} \in \mathbb{Z}^m\}$$

In any case, x_i in Equation 1 and the elements within vector x in Equation 2 are integer constants, which makes the lattice discrete. Therefore, these expressions define the lattice as the set of all vectors (points in the lattice) that are integer linear combinations of B's columns [3].

A key characteristic about lattices is that the same lattice can be generated from two different matrix bases, as depicted in Figure 1. In this case, the red basis (formed by vector v and u) and the green basis (formed by w and a generate the same lattice. This is better understood by introducing the concept of unimodular matrices: matrices with integer coefficients whose determinant is 1 or -1. Two bases B and C will generate the same lattice if and only if there exists a unimodular matrix U such that $B = C \cdot U$. Recall that the determinant of a matrix describes the area (or volume in higher dimensional scenarios) of the fundamental region (parallelepiped) of a lattice, portrayed in Figure 1 as the red and green areas of both bases. Therefore, $B = C \cdot U$ ensures that both bases have equal determinant up to sign, which preserves the lattice structure. Equivalently, B and C will generate the same lattice if C can be obtained by applying the following operations on B: Column swapping, multiplying a column by -1, and adding a column's multiple to the other column. Any of these operations are equivalent to multiplying C by a unimodular matrix.

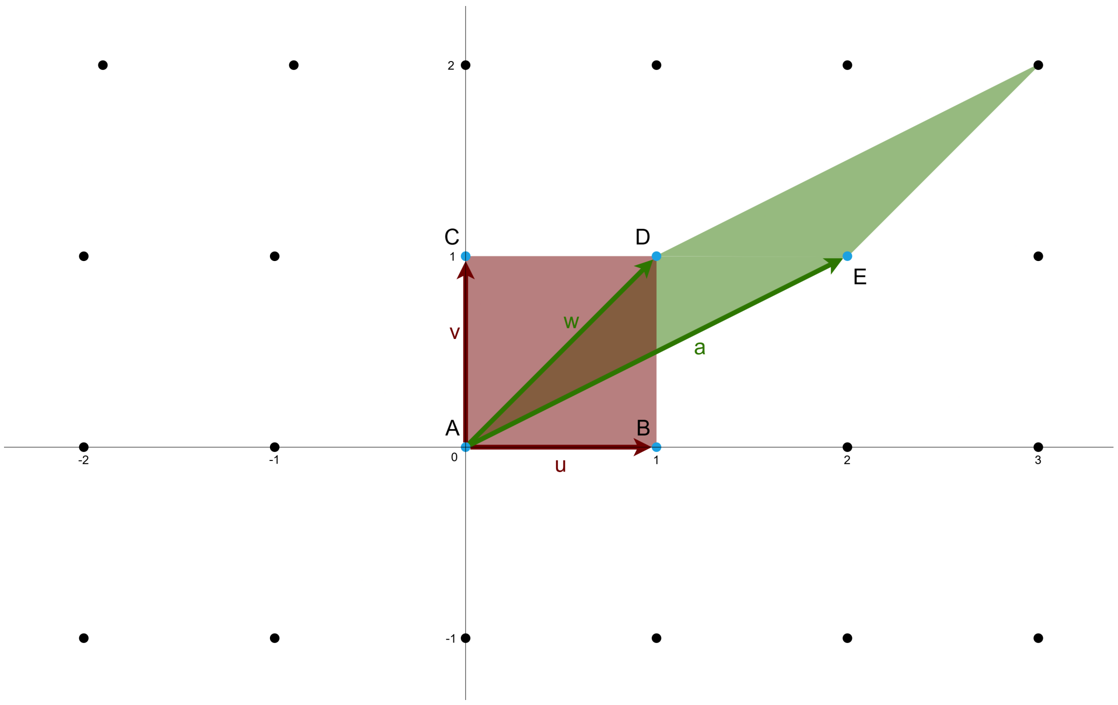

Figure 1: Two different bases generating the same lattice.

3 Lattice-Based Problems

3.1 SVP and CVP

There are certain problems related to lattices that are thought to be very hard. These problems are the foundation for some well-known cryptographic schemes, like the Kyber or Dilithium, two algorithms recently standardized by NIST. The first problem is known as the Shortest Vector problem (SVP). It aims to find an integer linear combination of the basis so that the resultant vector is the closest point to the origin (without counting the origin) that belongs to the lattice. Essentially, the output of this problem is the nonzero vector with the shortest norm (x) where

$$\|x\| = min\,\{\|x\| : x \in \mathcal{L} - \{0\}\}$$

The closest vector problem (CVP) is very similar to the SVP, but instead of interacting with the origin, a reference random point in the space is provided together with the lattice basis. The aim is to find the closest lattice point to the reference point. For a given vector $w \in \mathbb{R}^m$ and a lattice basis B, find the vector $v^* \in \mathcal{L}(B)$ so that

$$\|w - v^*\| = min\,\{\|w - v\| : v \in \mathcal{L}(B)\}$$

Finding the linear combination that outputs the correct vector for the SVP and the CVP is thought to be very hard, particularly in certain instances. At least as complex as solving an SIS problem, as Aijtai proved in 1996 [1].

3.1.1 Babai's algorithm

An initial approach to solving the CVP is the well-known Babai's algorithm. But before outlining its functioning, we must introduce the concept of orthogonality of a lattice basis.

Given a basis $v_1, \dots, v_m \in \mathbb{R}^n$ for a lattice, the fundamental domain corresponding to this basis is defined as:

$$\mathcal{F}(v_1, \dots, v_m) = \{\sum_{i=1}^{m} t_i v_i : 0 \leq t_i < 1\}$$

As it can be inferred, the n-dimensional volume of two fundamental domains corresponding to bases of the same lattice is also the same. Then, this volume is an invariant of the lattice called the determinant of the lattice, denoted as $det(\mathcal{L})$.

If the matrix of a lattice basis is a square matrix ($m = n$), we can also compute the determinant of the lattice as the determinant of that matrix. This is a particular case of the general result

$$det(\mathcal{L}(B)) = \sqrt{det(B^T B)}$$

The determinant of a lattice allows us to calculate how "orthogonal" a lattice basis is. This is due to the following inequality

$$det(\mathcal{L}) \leq ||v_1|| \cdots ||v_m||$$

Moreover, the equality occurs only when the vectors v_i are orthogonal. Therefore, we define the Hadamard ratio of a lattice basis as

$$\mathcal{H}(B) = \left(\frac{det(\mathcal{L})}{||v_1|| \dots ||v_m||} \right)^{\frac{1}{m}}$$

which gives a number in (0,1]. The closer this number is to 1, the more orthogonal the basis is considered to be.

We are interested in nearly orthogonal basis, since this makes SVP and CVP easier. This is clear if we have an orthogonal basis, i.e., $\mathcal{H}(B) = 1$. Since every vector of the lattice is of the form $a_1 v_1 + \cdots + a_m v_m$, with $a_i \in \mathbb{Z}$, using the orthogonality of v_i,

$$||a_1 v_1 + \cdots + a_m v_m||^2 = a_1^2 ||v_1||^2 + \cdots + a_m^2 ||v_m||^2$$

Then it is obvious that the solution of SVP is in the set $\{\pm v_1, \dots, \pm v_m\}$.

We can solve CVP similarly to the process outlined for SVP. If $w \in \mathbb{R}^n$ is a vector not necessarily in the lattice, we can solve a linear system to find t_i such that $w = t_1 v_1 + \cdots + t_m v_m$. If $v = a_1 v_1 + \cdots + a_m v_m$ is any lattice vector,

$$||v - w||^2 = (a_1 - t_1)^2 ||v_1||^2 + \cdots + (a_m - t_m)^2 ||v_m||^2$$

The closest lattice vector to w can be obtained taking each a_i as the nearest integer to t_i.

The process we made for solving CVP with an orthogonal basis is known as Babai's algorithm, and can be extended to any basis. Given a lattice basis $v_1, \dots, v_m \in \mathbb{R}^n$ and a vector $w \in \mathbb{R}^n$, execute the following steps:

- Find t_i that verify $w = t_1 v_1 + \cdots + t_m v_m$.
- Calculate a_i as the nearest integer to t_i.
- We return the lattice vector $v = a_1 v_1 + \cdots + a_m v_m$.

We then know that Babai's algorithm solves CVP correctly when the basis of the lattice is orthogonal. In general, this algorithm gives an approximated solution to CVP, and the more orthogonal the basis is, the better the approximation obtained may be. Let us illustrate the functioning of this algorithm with an example:

First, choose a lattice with basis $B = (v_1, v_2)$, $v_1 = (137, 312)$, $v_2 = (215, -187)$. We apply the algorithm to $w = (53172, 81743)$ and obtain:

- $t_1 \approx 296.85$, $t_2 \approx 58.15$.
- $a_1 = 297$, $a_2 = 58$.
- We return $v = 297 v_1 + 58 v_2 = (53159, 81818)$.

We see that $||v - w|| \approx 76.12$ and that $\mathcal{H}(B) \approx 0.977$.

Then, choose another basis $B' = (v_1', v_2')$ of the same lattice, with $v_1' = 5v_1 + 6v_2 = (1975, 438)$, $v_2' = 19 v_1 + 23 v_2 = (7548, 1627)$. Applying Babai's algorithm to w with this basis, we obtain:

- $t_1 \approx 5722.66$, $t_2 \approx -1490.34$.
- $a_1 = 5723$, $a_2 = -1490$.
- We return $v' = 5723 v_1 - 1490 v_2 = (56405, 82444)$.

In this case, $||v' - w|| \approx 3308.12$ and $\mathcal{H}(B') \approx 0.077$.

This example naturally leads to the question of obtaining a nearly orthogonal basis when starting with an arbitrary basis of a lattice. This is the purpose of another important algorithm, known as LLL, which will be examined later in this section.

3.1.2 Gaussian reduction

When working with a 2-dimensional lattice, there is an algorithm which always solves SVP. This algorithm is known as Gaussian reduction, and is an integer variant of the Gram-Schmidt orthogonalization method.

Recall that when we have a vectorial space basis $v_1, \ldots, v_n \in \mathbb{R}^n$, the Gram-Schmidt orthogonalization process gives us an orthogonal basis $v_1^*, \ldots, v_n^*$, obtained as

$$v_i^* = v_i - \sum_{j=1}^{i-1} \mu_{i,j} v_j^*, \quad \mu_{i,j} = \frac{v_i \cdot v_j^*}{v_j^* \cdot v_j^*}$$

The method can be also applied only to m linearly independent vectors, giving an orthogonal basis of the subspace generated by those vectors.

If we try to apply this method to a lattice basis, we find the problem that $\mu_{i,j}$ are not necessarily integers, so the resultant vectors may not be in the lattice, and we do not obtain a new lattice basis. However, if the $\mu_{i,j}$ were integers, the set obtained is an orthogonal basis of the same lattice (it is easy to see that both bases are related by an upper triangular unimodular matrix with 1's in the main diagonal).

In Gaussian reduction algorithm, the integer version of Gram-Schmidt method is applied to a lattice basis of dimension 2. The only difference with the method previously described is that we use the nearest integer to each $\mu_{2,1}$ when calculating v_2^*.

The Gaussian algorithm can be applied to a lattice basis v_1, v_2, through the following steps:

1. If $||v_2|| < ||v_1||$, swap v_1 and v_2.
2. Apply integer Gram-Schmidt to the basis $\{v_1, v_2\}$.
3. If the basis changed in step 2 (the nearest integer to $\mu_{2,1}$ is not zero, we go back to step 1.

One important result about this algorithm is that the first vector of the final basis is always a solution to SVP. Besides, as it can be inferred, the angle between the vectors of the final basis is in $[\frac{\pi}{3}, \frac{2\pi}{3}]$.

3.1.3 LLL

In 1982, Lenstra, A.K., Lenstra, H.W. and Lovász, L. invented an algorithm for finding a short, nearly orthogonal basis of a lattice in polynomial time [7]. In this algorithm, the concept of LLL-reduced basis is used.

Given a lattice basis $B = (v_1, \ldots, v_n)$, $B' = (v_1^*, \ldots, v_n^*)$ the set obtained after applying Gram-Schmidt (not the integer version described previously, and then B' not necessarily being a basis of the lattice), and $\mu_{i,j}$ the coefficients of Gram-Schmidt. We say that the lattice basis B is LLL-reduced if there exists $\delta \in (0.25, 1)$ such that:

1. $|\mu_{i,j}| \leq \frac{1}{2}$, for $1 \leq j < i \leq n$ (Size condition).
2. $\delta ||v_{i-1}^*||^2 \leq ||v_i^*||^2 + \mu_{i,i-1}^2 ||v_{i-1}^*||^2$, for $1 < i \leq n$ (Lovász condition).

The interest in LLL-reduced bases comes from the fact that, if $\{v_1, \ldots, v_n\}$ is an LLL reduced basis for a lattice $\mathcal{L}$, then $||v_1|| \leq 2^{(n-1)/2}\lambda$, with $\lambda = min\,\{\|x\| : x \in \mathcal{L} - \{0\}\}$.

In addition,

$$\prod_{i=1}^{n} ||v_i|| \leq 2^{n(n-1)/4} det(\mathcal{L})$$

The first inequality means that an LLL-reduced basis gives us an approximated solution to SVP, and the second one gives a lower bound for Hadamard ratio, in both cases with an exponential factor.

The LLL algorithm takes a lattice basis, and returns an LLL-reduced basis of the same lattice in polynomial time. A description of this algorithm can be found in [7].

Hence, if we are given a lattice basis, a way of attempting to solve CVP is to obtain an LLL-reduced basis of the lattice (which should be more orthogonal) and then apply Babai's algorithm.

3.2 LWE

The Learning with Errors (LWE) problem has a different nature, but cryptographers reframed it into a lattice problem to prove that LWE is as complex as hard lattice-based problems like the SVP and CVP [9]. The key functionality of the LWE problem is its versatility, making it perfectly suitable for cryptographic schemes. The fundamental concept of the LWE problem involves a system of linear equations containing intentionally added noise to the independent terms. It is important to note that these systems of equations are defined modulo a large prime number, reducing the problem to a finite structure. The LWE problem demands finding the variables within the approximate equations, for example, s_1, s_2, s_3, s_4 in the following system of equations, as described by Regev [10].

$$\begin{cases} 14s_1 + 15s_2 + 5s_3 + 2s_4 = 8 \pmod{17} \\ 13s_1 + 14s_2 + 14s_3 + 6s_4 = 16 \pmod{17} \\ 6s_1 + 10s_2 + 13s_3 + 1s_4 = 3 \pmod{17} \\ 10s_1 + 4s_2 + 12s_3 + 16s_4 = 12 \pmod{17} \end{cases}$$

Note that this system of equations includes noise added to the independent terms. By highlighting the added error in red, the correct independent terms (without the error) can be identified for secrets $s_1 = 0, s_2 = 13, s_3 = 9, s_4 = 11$

$$\begin{cases} 14s_1 + 15s_2 + 5s_3 + 2s_4 = 7 + 1 \pmod{17} \\ 13s_1 + 14s_2 + 14s_3 + 6s_4 = 0 + 16 \pmod{17} \\ 6s_1 + 10s_2 + 13s_3 + 1s_4 = 3 + 0 \pmod{17} \\ 10s_1 + 4s_2 + 12s_3 + 16s_4 = 13 - 1 \pmod{17} \end{cases}$$

The inclusion of noise within each equation of this system complicates the process of finding the secrets substantially. Without the errors, Gaussian elimination can solve this problem in polynomial time. With the errors, because Gaussian elimination takes linear combinations of the equations within the system, the errors are spread throughout the equations, making it significantly complicated to find the original secrets [10].

4 Lattice-Based Cryptographic Algorithms

4.1 GGH

The already mentioned lattice-based problems serve as the foundation for several known cryptosystems. For example, Goldreich, Goldwasser, and Halevi suggested a scheme based on the CVP [4]. In it, Alice generates two matrices, where both can be the basis of the same lattice. On one basis, the vectors will be almost perpendicular (the "good" basis), and the other will have both vectors close to parallel (the "bad" basis). Note that solving the CVP is a significantly harder task when solving it with a bad basis. This is because, to generate the same lattice, finding linear combinations of the vectors requires an extensive amount of computations. Therefore, Alice keeps the good basis as a private key and shares the bad basis as her public key. Bob, therefore, uses the bad basis to embed a message as a lattice point. Then, he selects a non-lattice point close to the message point and sends the coordinates to Alice. Alice can find the closest lattice point (Bob's message) through the use of the good basis. Nonetheless, an eavesdropper will have to do it on a bad basis, a significantly harder task.

The following example describes the inner functioning of the GGH public-key cryptographic scheme [5]. First Alice chooses a ser of linearly independent vectors which are reasonably orthogonal. This vector basis will serve as her private key or her "good basis", for example

$$v_1 = \begin{bmatrix} -97 \\ 19 \\ 19 \end{bmatrix}, v_2 = \begin{bmatrix} -36 \\ 30 \\ 86 \end{bmatrix}, v_3 = \begin{bmatrix} -184 \\ -64 \\ 78 \end{bmatrix}$$

Alice can verify whether it is a good choice of vectors if computing the Hadamard ratio of her basis (the set of vectors chosen) outputs a number relatively close to 1

$$\mathcal{H}(v_1, v_2, v_3) = \left(\frac{859516}{\|v1\|\|v2\|\|v3\|}\right)^{\frac{1}{3}} \approx 0.74620$$

Next, Alice chooses a unimodular matrix U with determinant $det(U) = \pm 1$, in this case

$$U = \begin{bmatrix} 4327 & -15447 & 23454 \\ 3297 & -11770 & 17871 \\ 5464 & -19506 & 29617 \end{bmatrix}$$

With this, she multiplies her original basis (her private key) with U, obtaining her pubic key or "bad basis"

$$w_1 = \begin{bmatrix} -4179163, \\ -1882253 \\ 583183 \end{bmatrix}, w_2 = \begin{bmatrix} -3184353 \\ -1434201 \\ 444361 \end{bmatrix}, w_3 = \begin{bmatrix} -5277320 \\ -2376852 \\ 736426 \end{bmatrix}$$

By calculating their Hadamard ratio of the public basis, it is possible to verify the non-orthogonality of this basis

$$\mathcal{H}(v_1, v_2, v_3) = \left(\frac{det(U)}{\|w1\|\|w2\|\|w3\|}\right)^{\frac{1}{3}} \approx 0.0000208$$

Assume the message that Bob wants to send to Alice is the lattice point $w_1 = \begin{bmatrix} 86, \\ -35 \\ -32 \end{bmatrix}$. By using the random element $r = \begin{bmatrix} -4, \\ -3 \\ 2 \end{bmatrix}$, Bob can generate the non-lattice point "close" to his message. Thus, the corresponding ciphertext will be

$$e = \begin{bmatrix} -4179163 & -1882253 & 583183 \\ -3184353 & -1434201 & 444361 \\ -5277320 & -2376852 & 736426 \end{bmatrix} \cdot \begin{bmatrix} 86, \\ -35 \\ -32 \end{bmatrix} + \begin{bmatrix} -4, \\ -3 \\ 2 \end{bmatrix} = \begin{bmatrix} -79081427, \\ -35617462 \\ 11035473 \end{bmatrix}$$

Using Babai's algorithm described in section 3.1.1, Alice expresses e as a linear combination of her private key with real coefficients

$$e \approx 81878.97v_1 - 292300.00v_2 + 443815.04v_3$$

Then, Alice rounds to the nearest integer and operates usign the values of her private key, obtaining a lattice vector that is close to e

$$v = \begin{bmatrix} -79081423, \\ -35617459 \\ 11035471 \end{bmatrix}$$

Alice now expresses v as a linear combination of her public key and m is found in the coefficients

$$v = 86w_1 - 35w_2 - 32w_3$$

Nonetheless, in the case of an eavesdropper Eve trying to decrypt m, after applying Babai's algorithm using Alice's public key, the output is

$$e \approx 75.76w_1 - 34.52w_2 - 24.18w_3$$

Which, after rounding and computing the result

$$v' = \begin{bmatrix} -79508353, \\ -35809745 \\ 11095049 \end{bmatrix}$$

Expressing v' as a linear combination of Alice's public key will output the incorrect $m' = \begin{bmatrix} 76, \\ -35 \\ -24 \end{bmatrix}$ instead of $m = \begin{bmatrix} 86, \\ -35 \\ -32 \end{bmatrix}$. In fact, computing Babai's algorithm on the public basis produces a remarkably higher difference in norm when compared to computing it with the private basis, where $\|e - v\| \approx 472004.09$ and $\|e - v'\| \approx 5.39$

4.2 Cryptographic schemes using LWE

It is worth noting that two of the NIST standardized algorithms (Kyber, which became ML-KEM after the standardization process, and Dilithium, renamed ML-DSA) rely on the LWE problem. In short, Kyber's security is based on the Decisional Module Learning With Errors (D-MLWE) problem. This problem is a variation of the classic LWE problem, and it involves working with matrices and polynomials over a ring. In simple terms, the D-MLWE problem asks whether it is possible to find the private key in the form of a polynomial from a set of noisy equations derived from a different polynomial. Similarly, Dilithium uses this approach together with the "Fiat-Shamir with Aborts" technique to generate a quantum-resistant lattice-based Schnorr-like signature with several optimizations [8].

We will now show an implementation of the LWE problem in a cryptographic scheme, an approach that serves as the foundation of several important algorithms like the standardized ML-KEM algorithm mentioned earlier. For simplicity, we will use an example.

First of all, a prime modulo is selected (3329 in ML-KEM). We also have to select the size of the system of equations (the number of equations). We will use the prime 47 and

4x4 systems in this example. Then, a random matrix modulo 47 (entries between 0 and 46) and a random small secret vector (small in the sense of close to 0 modulo 47) are generated.

For example:

$$A = \begin{pmatrix} 44 & 20 & 2 & 41 \\ 36 & 35 & 11 & 27 \\ 11 & 12 & 3 & 34 \\ 16 & 45 & 33 & 29 \end{pmatrix}, \quad s = \begin{pmatrix} 0 \\ 1 \\ 46 \\ 46 \end{pmatrix}$$

The following step is to compute As and to add a small random vector e:

$$As + e = \begin{pmatrix} 44 & 20 & 2 & 41 \\ 36 & 35 & 11 & 27 \\ 11 & 12 & 3 & 34 \\ 16 & 45 & 33 & 29 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 46 \\ 46 \end{pmatrix} + \begin{pmatrix} 1 \\ 2 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1999 \\ 1785 \\ 1714 \\ 2897 \end{pmatrix} \equiv \begin{pmatrix} 25 \\ 46 \\ 22 \\ 30 \end{pmatrix} \pmod{47}$$

The resultant vector will be denoted as

$$t = \begin{pmatrix} 25 \\ 46 \\ 22 \\ 30 \end{pmatrix}$$

The public key will be the system $Ax = t$, and s the private key. We recall that due to the error e added to the linear system, the secret s is difficult to obtain from the system. Until this point, we have just generated the keys to be used in encryption and decryption. Let's see how we can encrypt a bit.

For encryption, we first make a random linear combination, by small coefficients, of the rows of the system. If we sum the first two equations and substract the fourth one:

$$\begin{pmatrix} 1 & 1 & 0 & -1 \end{pmatrix} \begin{pmatrix} 44 & 20 & 2 & 41 \\ 36 & 35 & 11 & 27 \\ 11 & 12 & 3 & 34 \\ 16 & 45 & 33 & 29 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \equiv \begin{pmatrix} 1 & 1 & 0 & -1 \end{pmatrix} \begin{pmatrix} 25 \\ 46 \\ 22 \\ 30 \end{pmatrix} \pmod{47}$$

$$\begin{pmatrix} 17 & 10 & 27 & 39 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \equiv 41 \pmod{47}$$

We obtain the new equation

$$17x_1 + 10x_2 + 27x_3 + 39x_4 \equiv 41 \pmod{47}$$

After obtaining the new equation, we add small random small errors in each coefficient:

$$16x_1 + 11x_2 + 28x_3 + 41x_4 \equiv 40 \pmod{47}$$

Finally, if a 0 is encrypted, the previous equation is published, and if a 1 is encrypted instead, a big error (normally half of the modulo) is added to the independent term, and then the equation is published:

$$16x_1 + 11x_2 + 28x_3 + 41x_4 \equiv 40 + 23 \pmod{47}$$
$$16x_1 + 11x_2 + 28x_3 + 41x_4 \equiv 16 \pmod{47}$$

For decryption, we will just evaluate the secret key in the equation published by the sender of the message:

$$16 \cdot 0 + 11 \cdot 1 + 28 \cdot 46 + 41 \cdot 46 = 3185 \equiv 36 \pmod{47}$$

Finally, we compare the result of the previous operation with the independent term of the published equation. If a 0 was encrypted, we see that both numbers are close (40 and 36). If a 1 was encrypted, there is a big difference between them (16 and 36). In practice, we decrypt a 0 if the difference between both numbers is closer to 0 modulo 47 than to 23, and we decrypt 1 otherwise.

The key part of decryption is that we know an approximated solution of the system, that also has small coefficients. We could try to use the exact solution of the system, but this solution will not have small coefficients in general. If we try to decrypt with the exact solution, the small errors added after making a linear combination of the rows of the matrix during encryption, will distort the result when evaluating the equation during decryption, being unable to distinguish whether the result differs from the independent term because the sender encrypted a 1 or because of the other errors added during encryption.

The generation, encryption and decryption in ML-KEM are very similar to the process previously described, although the main difference is that instead of integer coefficients, polynomials over a ring are used. For example, in one of the versions of ML-KEM, a 4x4 matrix is generated, and each element of the matrix is an element of the polynomial ring $\mathbb{Z}_p[X]/(X^{256}+1)$, with $p = 3329$ as mentioned before. Then, each of this polynomials can be described by 256 coefficients in $\mathbb{Z}_p$.

References

[1] M. Ajtai. Generating hard instances of lattice problems (extended abstract). 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.

[2] Miklos 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*, STOC '97, page 284–293, New York, NY, USA, 1997. Association for Computing Machinery.

[3] D.J. Bernstein, J. Buchmann, and E. Dahmen, editors. *Post-quantum cryptography.* PQCrypto : international workshop on post-quantum cryptography. Springer, Germany, 2009.

[4] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In Burton S. Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, pages 112–131, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[5] J. Hoffstein, J. Pipher, and J.H. Silverman. *An Introduction to Mathematical Cryptography.* Undergraduate Texts in Mathematics. Springer, New York, NJ, USA, 2008.

[6] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. Ntru: A ring-based public key cryptosystem. In Joe P. Buhler, editor, *Algorithmic Number Theory*, pages 267–288, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[7] Arjen K. Lenstra, Hendrik W. Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.

[8] Vadim Lyubashevsky. Basic lattice cryptography: The concepts behind kyber (ML-KEM) and dilithium (ML-DSA). Cryptology ePrint Archive, Paper 2024/1287, 2024.

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

[10] Oded Regev. The learning with errors problem (invited survey). In *2010 IEEE 25th Annual Conference on Computational Complexity*, pages 191–204, 2010.