Spring 5-3-2019

# Lattices in Cryptography

Andy Guo
guoandy86@gmail.com

# Lattices in Cryptography

Andy Guo

May 3, 2019

**Abstract**

The computers that exist throughout the world today, rely on two values, 0 and 1 to work and run. Quantum computers, on the other hand, uses qubits which are quantum objects that can be in either a 0 or 1 state, or even a superposition of 0 and 1 state. Quantum computers do not necessarily make traditional computer algorithms run faster, but rather lets us come up with new algorithms specifically for quantum computers that will be faster than traditional computers. Quantum computer algorithms are designed to make full use of superpositional states. These quantum computers will be able to break commonly used public key cryptosystems such as RSA, DSA, and other practical ones as well. Some physicists predict that quantum computers will become powerful enough in the next 10 years to break these existing public key cryptosystems. As a result, a new field of research in post quantum cryptography is on the rise. One of the most well known and researched post-quantum cryptography schemes are ones that revolved around lattices.

## 1 Introduction

The computers that exist throughout the world today, rely on two values, 0 and 1 to work and run. Quantum computers, on the other hand, uses qubits which are quantum objects that can be in either a 0 or 1 state, or even a superposition of 0 and 1 state. Quantum computers do not necessarily make traditional computer algorithms run faster, but rather lets us come up with new algorithms specifically for quantum computers that will be faster than traditional computers. Quantum computer algorithms are designed to make full use of superpositional states. The quantum computations in the algorithm are done in parallel on superpositions of exponentially many inputs [23]. In other words, the quantum computer is able to go down every path simultaneously, and quantum computer algorithms exploit that to their advantage. The superpositional states of each qubit allows it to represent multiple numbers or values simultaneously. The value of a qubit is not determined till the end of the calculation when it has solved the problem. These quantum computers will be able to break commonly used public key cryptosystems such as RSA, DSA, and other practical ones as well [29]. Some physicists predict that quantum computers will become powerful enough in the

next 10 years to break these existing public key cryptosystems [5]. As a result, a new field of research in post quantum cryptography is on the rise [5]. As of right now, there are four research areas in the post-quantum cryptosystems field that can stop quantum computers: code-based, hash-based, lattice-based, and multivariate public key cryptosystems [7]. Out of the four, one of the most well known and researched post-quantum cryptography schemes are ones that revolved around lattices [21]. Lattice cryptography not only is a well known post-quantum cryptography scheme, it also can be used for fully homomorphic encryption.

Fully homomorphic encryption or FHE has been sought after for many years by cryptographers [12], as it has been seen as an elusive goal that can solve the world's problems with security and trust. A homomorphic encryption means that computations can be done directly to the encryption of a message instead of decrypting the ciphertext and then encrypting it again. This would save time and allows confidential data to be computed in unsafe environments and still remain confidential. A system like this would help the flow of data and information remain safe and secure. In unsafe and untrusted places, one would not need to worry about the environment taking their data. The data will not become vulnerable as it will never be decrypted to the original data, as more computation can be done directly to the ciphertext. Since FHE is still an encryption scheme, it would be ideal that it is based off an encryption scheme that is known to be hard to break. As a result, a fully homomorphic encryption scheme has been built from learning from errors assumption or LWE from the works of Gentry [12]. The security of this scheme is not only based the assumed hardness of LWE but also the sparse subset sum problem. The sparse subset sum problem is defined as given a set of integers $A$, and two integers, $t$ and $M$, find a sparse subset of the set that sums up to $t \mod M$. The sparse subset sum problem is hard and only gets harder when the given set is larger [17]. This paper will discuss what makes lattices a good foundation for building quantum computer resistant cryptography schemes.

A lattice is a regular arrangement of points in n-dimensional Euclidean space. The set of points can be described by using a set of n linearly independent vectors, $v_1, ..., v_n \in \mathbf{R}^n$. A lattice is formally defined as the set of all integer combinations of those $n$ linearly independent vectors. These linearly independent vectors are known as the basis of lattice [26]. The lattice $\mathcal{L}$ can be represented as

$$\mathcal{L} = \{\sum_{i=1}^{n} \alpha_i v_i \,|\, \alpha \in \mathbb{Z}\}.$$

This shows that $\mathcal{L}$ is the integer combinations of the n linearly independent vectors of the basis. The lattice is not the vector space spanned by the basis, as the span of the basis is the set of all real linear combinations of the n vectors, whereas the lattice only has the set of all combinations that has an integer coefficient to it. Due to this, a lattice is a discrete set, meaning that the points in the lattice cannot be too close to one another. There is a minimum distance between points in each lattice, where the minimum distance is greater than

0 [20]. This makes it so that lattice problems cannot be solved with simple linear algebra. Another important thing to note about the basis of a lattice is that it is not unique. There can be two bases that could end up generating the same lattice. Two bases are equivalent if multiplying one of the bases by a unimodular matrix will make the two equivalent

$$UB_1 = B_2,$$

where $U$ is the unimodular matrix. An unimodular matrix is any square matrix with integer entries that has a determinant of $\pm 1$.

Now that a lattice has been defined, lets go over some properties it and its basis has. The rank of $\mathcal{L}$ is the number of vectors in B, and the dimension of the lattice is just the dimension of each vector. If the rank is equal to the dimension, then the lattice is called a full ranked lattice. The dimension of a lattice is the number of elements in each column vector. All lattices has a dual to them. Given a lattice, $\mathcal{L}$, the dual of $\mathcal{L}$, is $\mathcal{L}^*$. $\mathcal{L}^*$ is a lattice given by the set of all vectors $y \in \mathbb{R}^n$ and the set of vectors $x \in \mathcal{L}$ such that for all $x \in \mathcal{L}, \langle x, y \rangle \in \mathbb{Z}$ [26]. The dual of a lattice is also equivalent to the inverse of the transpose of the lattice, $\mathcal{L}^* = (\mathcal{L}^T)^{-1}$. From that, it can be shown that the determinant of $\mathcal{L}^*$ is the reciprocal of the determinant of $\mathcal{L}$. The dual of a lattice is important as it can be used to solve some of these hard lattice problems.

A basis that describes $\mathcal{L}$ can also generate a fundamental parallelepiped. The fundamental parallelepiped of the basis is defined as

$$P(B) = \{\sum_{i=1}^{n} x_i v_i \,|\, x_i \in [0, 1)\}.$$

If the choice for the basis that describes $\mathcal{L}$ is clear, then P(B) can be rewritten as P($\mathcal{L}$), which still means the fundamental parallelepiped as generated by the basis that describes $\mathcal{L}$. The volume of the fundamental parallelepiped of the basis is the same as the determinant of $\mathcal{L}$. The volume of the parallelepiped is also the same for two different bases if they describe the same lattice. That is not the only way to calculate $det(\mathcal{L})$. It can also be calculated from finding the absolute value of the determinant of a matrix which is made of the basis vectors that describes $\mathcal{L}$. Geometrically, the determinant of the lattice shows the inverse density of lattice points in space. As mentioned before, a lattice can be described by multiple bases, and each of those bases will have their own respective fundamental parallelepipeds. The space can be "tiled" with the fundamental parallelepiped, where each lattice point is given a copy of the fundamental parallelepiped [21]. The fundamental parallelepiped is also essential in finding the modulo of a point $x \in \mathbb{R}^n$. $x \mod P(\mathcal{L})$ is the unique point $y \in P(\mathcal{L})$ such that $y - x \in \mathcal{L}$ [26].

The Gram-Schmidt orthogonalization of a basis B $= [v_1, ..., v_n]$ can be written as $B^*$. $B^* = [v_1^*, ..., v_n^*]$ is defined as,

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

where $\mu_{ij}$ is,

$$\mu_{ij} = \frac{\langle v_i, v_j^* \rangle}{\langle v_j^*, v_j^* \rangle}, \; for \, 1 \leq j \leq i \leq n.$$

The Gram-Schmidt orthogonalization projects the vector $v_i$ of B onto a space orthogonal to the space spanned by $v_1, ..., v_{i-1}$. However, the vectors produced by the Gram-Schmidt procedure does not necessarily produce a lattice as the vectors may not all be integer combinations of the basis, and as a result cannot be a lattice. The Gram-Schmidt orthogonalization process, however, is useful in help solving lattice problems such as SVP [11]. The procedure is useful in reducing the basis to other forms such as a LLL reduced basis [10] or a BKZ reduced basis [8]. In those examples, the input basis is taken and the Gram-Schmidt orthogonalization is done on the basis first and then other steps are taken to find a reduced basis.

Each lattice has also its shortest distance between any two points in the lattice. The shortest distance of a lattice is represented by the lambda symbol, $\lambda(\mathcal{L})$. The shortest distance is never 0 as that means the points are on top of each other. That would also mean that the shortest (nonzero) vector has a length of 0. However, since the origin vector is in every lattice, it is also the lattice point with the smallest norm. Each lattice will always contain a pair of points that will have the shortest distance of the entire lattice. Finding the shortest distance, or the minimum distance is hard to do, and thus, has become a lattice problem. The problem will be discussed in greater detail in the next section. The more common properties of lattices has been discussed and in the next section, some hard lattice problems will be discussed, specifically SVP, CVP, and LWE, and later on how a public key cryptosystem can be built from those problems, specifically the LWE problem.

## 2 Lattice Problems

The discovery of lattices also brought with it its fair share of problems that are believed to be computationally intractable. Many existing lattice problems have been proven to have average case hardness, and thus making them a good foundation for building a cryptographic schemes [22]. Average case hardness just means given any random instances of the problem, it is computationally hard to solve. Proofs of security for cryptographic schemes is based on the average case hardness of the problem it is built from. Many thought that NP-hard problems would have been the perfect basis for building cryptographic schemes from, but not all NP-hard problems can achieve average case hardness. This means that for some instances of NP-hard problems they can be easy to solve, therefore making it not suitable to build a cryptographic scheme from. On the other hand, most lattice problems have been proven to achieve average case hardness, and some have even been proven to have worst case hardness. Those problems are more likely to withstand an attack from a quantum computer. Worst case hardness means that any instance of the problem is hard to solve.

So what makes these problems hard? The main reason these lattice problems are considered hard is to due to the fact that there are not efficient solutions to these problems. There does exist solutions, but none of them are efficient enough to consider the problems easy to solve. As a result, some of the solutions for these problems rely on finding approximate values instead of finding the exact values, as the solutions for finding approximate values are slightly more efficient [6]. There is a direct correlation between the hardness of a problem and the security of a cryptographic scheme [16]. This is due to the fact that the cryptographic scheme is built directly from the problem. If one can solve the problem, then one can break the cryptographic scheme built from that problem. If the solution is efficient, then the cryptanalysis of the scheme should be efficient as well. However, in the case of these hard lattice problems, since there does not exist an efficient algorithm, the cryptanalysis will be inefficient as well. Due to these reasons, lattice problems has become popular in quantum resistant cryptographic schemes. Some of the problems discussed will be the Shortest Vector Problem (SVP) in section 2.1, Closest Vector Problem (CVP) in section 2.2, and finally, the Learning with Errors (LWE) problem in section 2.3.

## 2.1 Shortest Vector Problem (SVP)

SVP has three variants to it, decision, optimization, and search. Though there are multiple variants to it, the underlying problem remains the same. They all have to do with the shortest vector in the lattice and the shortest vector of $\mathcal{L}$ can be represented as $\lambda(\mathcal{L})$.

- Decision variant - given a basis B that describes $\mathcal{L}$ and a real number $d$, where $d > 0$. Determine whether $\lambda(\mathcal{L})$ is less than or greater than $d$.
- Optimization (or calculation) variant - given basis B that describes $\mathcal{L}$, find $\lambda(\mathcal{L})$.
- Search variant - given basis B that describes $\mathcal{L}$, find a vector $v \in B$ such that the norm of $v$ is equal to $\lambda(\mathcal{L})$.

SVP has been proven to be NP-hard under randomized reductions by Ajtai [3]. Problems that are NP-hard means that if they can be solved in polynomial time than all NP problems can as well. NP problems are problems that are solvable in polynomial time by a nondeterminstic Turing Machine [9]. There are many known algorithms that can solve SVP, show can find the exact shortest vector, while others can find the approximate shortest vector to a certain factor. It has been shown that algorithms for finding the approximate shortest vector takes less time than the ones that require the exact solution [24]. One of the best known algorithms for approximation, can only solve it in polynomial time and achieves an approximation factor of $2^{O(n)}$ where n is the dimension.[1] That

[1] This is interesting as it shows that approximating the shortest vector is not an easy task. In fact, Ajtai showed that if finding an approximation to a factor of $n^c$ is hard in the worst case, than finding the exact shortest vector will be hard in the average case [14].

algorithm is known as the Lenstra-Lenstra-Lovász basis reduction algorithm or the LLL basis reduction algorithm. More algorithms were discovered afterwards with better approximation factors, but most of them still retain their polynomial runtime [19]. Most of the known algorithms for solving the shortest vector problem requires a reduction of some sort on the given basis. Reducing the lattice makes the solution more efficient to find [30]. Though the reduction does makes it the solution slightly easier to find, it still takes up to polynomial time for the entire algorithm to run.

As mentioned before, one of the most well known algorithms to solve SVP is the LLL basis reduction algorithm. The LLL basis reduction algorithm is an algorithm that finds the approximate shortest vector and not the exact shortest vector. The basis reduction algorithm reduces the given basis to a shorter or smaller basis that describes the same lattice $\mathcal{L}$. The shortest vector in integer span of the basis is also the shortest vector in the lattice it describes [10]. Due to that, many of the existing algorithms to solve SVP relies on reducing the given basis to make it more efficient to approximate the shortest vector. (Note that there is some basis that includes the shortest vector, so there is a connection between the norm of basis vectors and difficulty of solving shortest vector.) One of the most basis steps to most lattice reductions starts with a Gram Schmidt Orthogonalization. By producing a more orthogonal basis and still describing the same lattice, it becomes easier to find the shortest vector. A given basis for the lattice could have long and narrow shape to it, and due to that, it will be hard to calculate the shortest vector from that. But if the a second basis could be produced from the first basis, and if the second vector describes the same lattice, but the shape of it is smaller and more box like, then it is easier to determine the shortest vector from that lattice. As a result, that basis is "better," as it is easier to use to solve these lattice problems. The LLL algorithm finds an approximation within an exponential factor in polynomial time. The algorithm was discovered in the early 1980s and still relevant in solving SVP and other problems in number theory, cryptography, and integer programming [10]. The LLL algorithm has two major steps to it.

**LLL − algorithm**

1. Perform Gram Schmidt orthogonalization on the given basis and get a new basis $B^*$.

2. For each $v_i \in B^*$, check for the following condition,

$$\|v_{i+1}^* + \mu_{i,i+1} v_i^*\|^2 < \frac{3}{4}\|v_i^*\|^2.$$

   If it holds, swap $v_i$ and $v_{i+1}$ and repeats the first step, else it continues to the next i.

The $\mu$ are known as the Gram Schmidt coefficients, and which are described earlier in the introduction. The vectors $v$ are from the original basis and the

vectors $v^*$ are from the basis $B^*$. The Gram Schmidt process doesn't necessarily produce an basis that can describe a lattice, so to prevent that round the projection $\mu$ to the nearest integer, that way the basis produced from the Gram Schmidt procedure still describes a lattice. The basis is LLL-reduced if the following two conditions holds,

1. $\forall i \neq k, \mu_{i,k} \leq \frac{1}{2}$
2. $\|v^*_{i+1} + \mu_{i,i+1} v^*_i\|^2 \geq \frac{3}{4}\|v^*_i\|^2$

The first step takes polynomial time and the second time takes linear time, and as a result the entire algorithm runs in polynomial time. It can approximate to at most $2^{\frac{(n-1)}{2}}$ times $\lambda(\mathcal{L})$. From LLL algorithm many other reduction algorithms has risen such as the HKZ reduction and so on [28]. SVP has been studied for very long and so has CVP, a related lattice problem.

## 2.2 Closest Vector Problem (CVP)

CVP is very similar to SVP, and it is believed that SVP is no harder than CVP [1]. CVP has been shown to be NP-hard as well by van Emde Boas in 1981 [14]. In the CVP, a basis $B \in \mathbf{R}^n$ is given, along with a target vector $t \in \mathbf{R}^n$. The goal is to find the closest point in B closest to the point $p$ given. While in SVP the all zero solution is not accepted, in CVP, the all zero solution can an answer [22]. The all zero solution just means that the origin is the closest vector. That cannot be the case for SVP. If the shortest vector could be 0, then SVP would not be a hard problem at all, as the shortest vector will always be the all zero vector. Due to that the zero vector is not an allowed answer for SVP, but could be for CVP. Similarly to SVP, CVP also has three variants to it, search, optimization, and decision.

- Search variant - given a basis B that describes $\mathcal{L}$ and a target vector t, find the closest vector in $\mathcal{L}$ to t.
- Optimization variant - given a basis B that describes $\mathcal{L}$ and a target vector t, compute $d(t, \mathcal{L})$.
- Decision variant - given a basis B that describes $\mathcal{L}$, a target vector t, and a rational number r, determine whether $d(t, B) \leq r$ or $d(t, B) > r$

CVP is consider "harder" than SVP [1]. As a result, a reduction can be shown of CVP to SVP, but no reduction can be shown in the other direction. Instead of looking for the lattice vector that is closest to an arbitrary given vector, in SVP we look for the shortest vector. An oracle that can solve CVP, can easily be alter to output the norms of the vectors in the lattice rather than which vector is closest to the target vector. Due to this reasoning, CVP can be reduced to SVP and if CVP can be solved, then so can SVP.

One of the earliest found algorithms to solve CVP is Kannan's algorithm as well. The time complexity for this algorithm is $n^{n/2+o(n)}$. Since then, more time

efficient has been discovered to solve CVP. For example, the Micciano-Voulgaris Algorithm has been found to be able to solve CVP in $2^{O(n)}$ time. However, one of the more well known algorithm to solve CVP is Babai's Nearest Plane Algorithm [2].

**Babai's Nearest Plane Algorithm**

1. Do a LLL reduction on the input lattice, $\mathcal{L}$.
2. Look for an integer combination of the basis vectors that will make it close to the target vector. This can be done with the simple algorithm:

$b \leftarrow t$
for i = n to 1:
    $b \leftarrow b - c_i b_i$ , where $c_i = \left\lceil \frac{\langle b, b_i^* \rangle}{\langle b_i^*, b_i^* \rangle} \right\rfloor$
output $t - b$

The algorithm works by looking for the a multiple for nth vector in the basis $B$, to bring the nth coordinate in it to within $\pm\frac{1}{2}\|b*_i\|$, where $b_i^*$ is from the LLL reduced basis. After doing that for nth vector, it moves on to the nth - 1 vector and keeps going till it reaches the first vector in the basis. In a sense, it is looking for the closest plane to the target vector, hence the name of the algorithm. The run time of the algorithm is in polynomial time of the input size [2]. The returned vector from Babai's Nearest Plane algorithm is called the Babai point. The Babai point may not be the closest vector to the target vector, but the error of it can be bounded. How close the Babai point is to the true solution is reliant on the given target vector and the basis given as well for the lattice $\mathcal{L}$. Just using Babai's algorithm by itself without doing a LLL reduction on the basis first does not guarantee that the solution vector will be the closest vector to the target vector. However, after doing a LLL reduction first on the given basis, then the solution vector can be found to an approximate exponential factor of the closest distance to the target vector [2]. As one can see, the LLL reduction algorithm is useful for many lattice problems not just SVP. It helps find a better basis that will describe the same lattice to use in the problem. As a result of that, the easier basis will make the algorithms to solve the problems more efficient [10].

## 2.3 Learning with Errors (LWE)

The Learning with Errors problem was formalized by Oded Regev in 2005 and he showed that LWE was as hard to solve as other worst case lattice problems such as SVP and CVP [26]. The LWE problem is defined as such, given a basis B generating $\mathcal{L}$, and the value of Bx + e, the solution to the problem would be successfully determining the value of x. The e is a small error on the Gaussian distribution that is added to Bx to distort Bx so it would be hard to find the value of x. By adding the error value, it makes it significantly harder to solve the problem. There are two variants to this problem, the search and decision versions.

- Search variant - given a basis B that describes $\mathcal{L}$ and the value of Bx + e, find x.
- Decision variant - given a basis B that describes $\mathcal{L}$ and C, determine if C is C = Bx + e or random.

Without adding the error or noise to Bx, the problem would be easy to solve. But adding just a little error to Bx makes it so that Bx is more hidden or secretive. If one were to use the Gaussian elimination method to solve the problem where there is no errors introduced, it would be easy to figure out what the secret x is [27]. Using Gaussian elimination in finding the value of x in $Bx = C$ where $C = Bx$ would be simple. Row operations can be performed on $B|C$, where $B|C$ just represents augmenting C as another column to B, to put it into upper triangular form. Once it is in upper triangular form, the last column of the new matrix is the value of x. A matrix in upper triangular form is when all the values below the diagonal from the top left to bottom right is all zeros. However, if the same method is used on the actual problem, all the errors added would accumulate making it impossible to find out what the actual value of x is. Due to that, a new method for solving had to be found. Similarly to the SVP and CVP, some of the best known algorithms for solving this problem all run in exponential time. The Blum, Kalai, and Wasserman algorithm, or BKW algorithm for short, takes $2^{O(n)}$ time [4]. Since the best known algorithm still runs in exponential time and there is no known algorithm that can solve it faster than that, LWE becomes a hard lattice problem. LWE is also seen as a natural extension of the Learning Parity with Noise problem or LPN for short [25]. LPN itself is widely believed to be hard as well, and if LWE is a natural extension of it, it would be only right if LWE was hard as well. In LPN one is given $x, y$, where $y = \langle x, z \rangle + 1 \mod 2$ with some probability (otherwise given $y = \langle x, z \rangle \mod 2$). The vector $z$ is used repeatedly with different vectors $x$. As one can see, LWE and LPN is very similar and the main difference being that LWE uses lattices and LPN can be done using integers. Either way, when noise is introduced into the problem, it makes them both significantly harder. LWE is also known to be hard based on assumptions regarding the worst case hardness of problems such as GapSVP and SIVP. Due to all these factors, LWE is believed to be hard [27] and hard to approximate as well.

The BKW algorithm takes both $2^{O(n)}$ time and memory. It is the best known algorithm to solve LWE as of right now [15]. The BKW algorithm was originally developed to solve the Learning Parity with Noise problem or LPN for short [18]. LWE is seen as a generalization of LPN to a larger modulo and it has different error values. Since the BKW algorithm was developed for LPN, it was able to be converted over to a similar algorithm that solved LWE [15]. There are three main steps to the BKW algorithm.

1. A sample reduction on the basis.
2. Determine some sub-solutions to recover some components of the secret vector $x$.

3. Substitute what has been figured out so far and in the process getting a smaller LWE instance to solve.

The sample reduction is done through a Gaussian elimination. However, as mentioned before, using the Gaussian elimination on this problem would amplify the noise to the point where the secret vector cannot be recovered. Thus, instead of doing all row reductions, only a few row reductions are done to it so that the noise is low enough to be managable [4]. By only doing a few row reductions at a time, only a few part of the secret vector can recovered at a time. After figuring out parts of the secret vector x, it becomes easier to plug those values back in and recover the rest of the parts of x [15]. However, as previously mentioned, this still takes exponential time, and for a problem to be considered "easy" it has to be able to be solved in a faster time than that. Another reason to considered LWE hard is because it is believed to be hard under on assumptions regarding the worst case hardness of other lattice problems such as GapSVP and SIVP. [27]. Being able to easily solve LWE would mean that those lattice problems can be solved easily as well.

# 3 Cryptography

Lattice based problems such as SVP, CVP, and LWE are used as the foundation for building lattice based cryptography schemes. A simple public key cryptosystem can be built from any of those problems. To show that, a PKC built from LWE will be shown down below.

## 3.1 Public Key Cryptosystem using LWE

The main goal of LWE is to find the secret vector x, and in the PKC, this secret vector x will be used as the private key. Before going into how the system works, there are some basic parameters that needs to be defined first.

- $n$ - the security parameter or dimension of the lattice
- $m$ - number of equations or samples (choose $m = 1.1 * n \log(q)$)
- $q$ - modulus parameter (choose a prime between $n^2$ and $2n^2$)
- $\alpha$ - noise or error parameter (has to be greater than 0 and let $\alpha = 1/(\sqrt{n}\log^2 n)$)

Choosing these parameters with the following guidelines will guarantee both security and correctness. The private key for this system, as previously mentioned, is the secret vector x chosen uniformly from $\mathbb{Z}_q^n$. The public key is the m samples of ($B_i$, $b_i$) from i = 1 to m using the secret vector $x$, $q$, and $\alpha$. $b_i$ in the sample pair is the result of $b_i = B_i x + \alpha_i$.

To encrypt a message, choose a set S uniformly from the $2^{\mathrm{m}}$ subsets of m samples for each bit of the message. If the bit is 0, then encrypt using

$$\left(\sum_{i\in S} B_i, \sum_{i\in S} b_i\right).$$

Otherwise if the bit is 1, then encrypt using

$$\left(\sum_{i\in S} B_i, \lfloor\frac{q}{2}\rfloor + \sum_{i\in S} b_i\right).$$

As one can see, each bit of message is returned with a pair of values for decryption.

To decrypt a ciphertext, given any pair (B, b), if b - $\langle B, x \rangle$ is closer to 0 than $\lfloor q/2 \rfloor$ mod q, then the pair is decrypted as 0. $\langle B, x \rangle$ is the inner product of B and x. If it is closer to $\lfloor q/2 \rfloor$ mod q, then it should be decrypted as 1. $\langle B, x \rangle$ is just the inner product of B and the private key vector x. This decryption works as if there were no $\alpha$ values, then the decryption would strictly equal either 0 or $\lfloor q/2 \rfloor$. However, with the addition of the error, the decryption values become close to and not exactly equal to either 0 or $\lfloor q/2 \rfloor$. It rarely decrypts to wrong bit as the sum of the errors over the set S is almost always under q/4. The probability that it does decrypt to the wrong bit is negligible [27]. If the sum was greater than q/4, then there might be some error in decrypting it as the errors at that point are too large will change the message bits completely. The probability that the sum of the errors is over q/4 is negligible and therefore it can be assumed that it will always decrypt to the correct bits and then message.

The system is pretty straightforward, however, it is not ideal to use this in a large scale setting as it is very inefficient [27]. The public key size is $\mathrm{O}(\mathrm{n}^2)$ and encrypting the message increases the message size by a factor of O(n). It works as a PKC, but using it regularly is not ideal, as it'll end taking up a lot of computing space as the message size increases. Since most of these lattice problems become even harder in higher dimensions [13], including LWE, increasing the dimensions of the lattice for LWE in this PKC will make the public key size even bigger, because the public key size is equal to the square of security parameter or the number of dimensions the lattice has. For LWE to be truly effective, $n$ has to be decent size number, and as a result cryptosystems such as PKC will become inefficient to larger values of $n$. An ideal cryptographic application built of LWE will be one that requires a smaller key size than the key size of the PKC system. The security of the system has been shown to be secured under the LWE hardness assumption [27]. Regev showed that given an adversary that can efficiently guess the encrypted bit won't be able to distinguish between encryptions of 0 and 1 as they are equally distributed. The adversary is given pairs $(a_i, b_i)$, but it is not chosen from the LWE distribution, but rather chosen uniformly from $\mathbb{Z}_q^n \times \mathbb{Z}_q$. From that, LWE samples can be distinguished from the uniform samples for a non-negligible fractions of secret s. This shows that this public key cryptosystem using LWE is secure and correct, but not the

most efficient cryptosystem there could be. As a result, different methods are applied and new cryptosystems using LWE has come to light.

# 4 Conclusion

Even though lattice cryptography is still rather young, it has become an extensively researched field. It has shown that it has the potential to become a foundation for cryptographic schemes in the post-quantum era if it were ever to come. From problems such as SVP to LWE, it has shown to have a variety of problems that are considered hard to solve. Most of the work done in this field can be split up into two major parts [20]. One part is involved in the math part behind lattices, such as connecting the average case complexity of lattice problems to worst case complexity. The other part is involved in building cryptosystems from these existing problems. As of right now only the surface has been scratched in this field, and it has to the potential to grow even bigger than it is right now.

# References


[1] Divesh Aggarwal and Noah Stephens-Davidowitz. Just take the average! an embarrassingly simple $2^n$-time algorithm for svp (and cvp). *arXiv preprint arXiv:1709.01535*, 2017.

[2] Erik Agrell, Thomas Eriksson, Alexander Vardy, and Kenneth Zeger. Closest point search in lattices. *IEEE transactions on information theory*, 48(8):2201–2214, 2002.

[3] 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*, pages 601–610. ACM, 2001.

[4] Martin R Albrecht, Carlos Cid, Jean-Charles Faugere, Robert Fitzpatrick, and Ludovic Perret. On the complexity of the bkw algorithm on lwe. *Designs, Codes and Cryptography*, 74(2):325–354, 2015.

[5] Daniel J Bernstein. Introduction to post-quantum cryptography. In *Post-quantum cryptography*, pages 1–14. Springer, 2009.

[6] Jin-Yi Cai and Ajay Nerurkar. Approximating the svp to within a factor (1-1/dim/sup/spl epsiv//) is np-hard under randomized conditions. In *Proceedings. Thirteenth Annual IEEE Conference on Computational Complexity (Formerly: Structure in Complexity Theory Conference)(Cat. No. 98CB36247)*, pages 46–55. IEEE, 1998.

[7] Lily Chen, Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. *Report on post-quantum*

*cryptography*. US Department of Commerce, National Institute of Standards and Technology, 2016.

[8] Yuanmi Chen and Phong Q Nguyen. Bkz 2.0: Better lattice security estimates. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 1–20. Springer, 2011.

[9] Stephen Cook. The p versus np problem. *The millennium prize problems*, pages 87–104, 2006.

[10] Xinyue Deng. An introduction to lenstra-lenstra-lovasz lattice basis reduction algorithm.

[11] Dan Ding, Guizhen Zhu, and Xiaoyun Wang. A genetic algorithm for searching the shortest lattice vector of svp challenge. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 823–830. ACM, 2015.

[12] Craig Gentry et al. Fully homomorphic encryption using ideal lattices. In *Stoc*, volume 9, pages 169–178, 2009.

[13] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In *Annual International Cryptology Conference*, pages 112–131. Springer, 1997.

[14] Subhash Khot. Hardness of approximating the shortest vector problem in lattices. *Journal of the ACM (JACM)*, 52(5):789–808, 2005.

[15] Paul Kirchner and Pierre-Alain Fouque. An improved bkw algorithm for lwe with applications to cryptography and lattices. In *Annual Cryptology Conference*, pages 43–62. Springer, 2015.

[16] Thijs Laarhoven, Joop van de Pol, and Benne de Weger. Solving hard lattice problems and the security of lattice-based cryptosystems. *IACR Cryptology EPrint Archive*, 2012:533, 2012.

[17] Moon Sung Lee. On the sparse subset sum problem from gentry-halevi's implementation of fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2011:567, 2011.

[18] Éric Levieil and Pierre-Alain Fouque. An improved lpn algorithm. In *International Conference on Security and Cryptography for Networks*, pages 348–359. Springer, 2006.

[19] Daniele Micciancio. The shortest vector in a lattice is hard to approximate to within some constant. *SIAM journal on Computing*, 30(6):2008–2035, 2001.

[20] Daniele Micciancio. The geometry of lattice cryptography. In *International School on Foundations of Security Analysis and Design*, pages 185–210. Springer, 2011.

[21] Daniele Micciancio. Lattice-based cryptography. *Encyclopedia of Cryptography and Security*, pages 713–715, 2011.

[22] Daniele Micciancio and Shafi Goldwasser. *Complexity of lattice problems: a cryptographic perspective*, volume 671. Springer Science & Business Media, 2012.

[23] Michele Mosca. *Quantum computer algorithms*. PhD thesis, University of Oxford. 1999., 1999.

[24] Phong Q Nguyen. Lattice reduction algorithms: Theory and practice. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 2–6. Springer, 2011.

[25] Krzysztof Pietrzak. Cryptography from learning parity with noise. In *International Conference on Current Trends in Theory and Practice of Computer Science*, pages 99–114. Springer, 2012.

[26] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):34, 2009.

[27] Oded Regev. The learning with errors problem. *Invited survey in CCC*, 7, 2010.

[28] Claus Peter Schnorr. Progress on lll and lattice reduction. In *The LLL Algorithm*, pages 145–178. Springer, 2009.

[29] Peter W Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.

[30] Dirk Wubben, Dominik Seethaler, Joakim Jalden, and Gerald Matz. Lattice reduction. *IEEE Signal Processing Magazine*, 28(3):70–91, 2011.