QUANTUM RESISTANT LWE BASED ENCRYPTION FOR SECURE END-TO-END COMMUNICATION

A Project Report Submitted

by

JADHAV NEHA KISHOR

SAMEER KANT

(2001EE21

2001EE60)

In Partial Fulfilment

of the Requirements for the award of the degree

BACHELOR OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY PATNA

MAY 2024

THESIS CERTIFICATE

This is to certify that the thesis titled **Quantum resistant LWE based encryption for secure end-to-end communication**, submitted by **Jadhav Neha Kishor and Sameer Kant**, to the Indian Institute of Technology, Patna, for the award of the degree of **Bachelor of Technology**, is a bonafide record of the research work done by them under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Jawar Singh
Supervisor
Professor
Dept. of Electrical Engineering
IIT-Patna, 800 013

Place: Patna

Date: 9th May 2024

ACKNOWLEDGEMENTS

I would like to thank my project supervisor, **Prof. Jawar Singh**, for his unwavering assistance and always being available to clear doubts. . A special mention and thanks to my colleague and my co-worker **Jadhav Neha Kishor** for assisting me during the whole project without her it wouldn't be possible.

I am immensely thankful to **Indian Institute of Technology, Patna** for providing me with the resources and the environment to conduct my research and work related to the project.

I am grateful to my parents, and my friends for their encouragement and personal support. Without their support, I wouldn't have been able to reach this point.

ABSTRACT

KEYWORDS: Post-Quantum Cryptography, Lattice-based, Public key encryption, Short vector Problem, and Shor's Algorithm.

Quantum computers are not hypothetical and they posses severe challenges to the existing IT infrastructure as they could break any existing encryption in a fraction of a minute. The National Institute of Standards and Technology (NIST) of the U.S. Department of Commerce has approved four algorithms that are said to be quantum resistant. This paper presents a novel and efficient approach for achieving quantum-resistant encryption using lattice-based cryptography. Specifically, we address the challenge of encrypting extremely small units of data, such as a single letter or a single-bit message, by constructing a multidimensional lattice. The proposed technique leverages the Short Vector Problem (SVP) in lattice-based cryptography and incorporates the Learning with Errors (LWE) methodology to encrypt and decrypt the data. We demonstrate the feasibility and robustness of this approach in protecting sensitive information, even in the presence of quantum threats.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

LWE	Learning with error
PQC	Post Quantum cryptography
SVP	Short vector Problem
u-SVP	unique - Short vector Problem
Gap-SVP	Gap - Short vector Problem
SIVP	Short integer vector Problem
PKE	Public key encryption
KEM	Key exchange mechanism
QKD	Quantum key distribution
SHA	Secure Hash Algorithm
M-LWE	Modulue-Learning with error
NTT	Number theory transform
CCA	Chosen ciphertext attack
CPA	Chosen plaintext attack
IETF	Internet Engineering Task Force
MLS	Message layer security
XOF	Extendable Output Function
LLL	Lenstra, Lenstra, and Lovász
BKZ	Block Korkin-Zolotarev algorithm

NOTATION

$\mathbb{Z}$	Set of integers
q	Modulo number
$\mathbb{Z}_{\|\|}$	The finite field of cardinality q and are represented by interval $[-[\frac{q}{2}], \frac{q}{2}]$.
$B(n)$	Encoded array of n integers like $\{a_1, a_2, a_3...., a_n\}$
$\mathcal{D}$	Gaussian distribution
$\mathcal{D}_\sigma$	Gaussian distribution standard deviation σ
$x \leftarrow \chi$	et of x belongs from the distributed error terms generated which is denoted by χ.
$\mathbf{Pr}(X)$	Probability of X
$\mathcal{B}_n^k$	an array of 32 bits representing the ASCII value for a character

CHAPTER 1

INTRODUCTION

In today's rapidly evolving digital landscape, the security of sensitive information faces an unprecedented threat, particularly in the impending quantum era. Traditional cryptographic algorithms, such as the widely used RSA (Rivest-Shamir-Adleman) public key encryption, confront a daunting challenge. The emergence of quantum computing, with its unparalleled computational power, poses a substantial risk to the reliability of these conventional encryption methods. Notably, quantum computers can effortlessly break the existing algorithms and thereby undermining data security.

In response to this pressing need, scientific communities were initially very interested in lattice cryptography because designs for cryptographic constructions came with security proofs derived from the worst-case situations involving lattice problems. Ajtai and Dwork [8] proposed the first lattice-based encryption scheme, however, later it was refined and streamlined by introducing the concept of the Learning With Errors (LWE) problem, an intermediate problem that was asymptotically at least as hard as some common worst-case lattice problems but relatively easy to use in cryptographic constructions, a significant contribution [1].

According to the LWE assumption, it is difficult to detect the distribution (A, A*s + e) from uniform, where e is a vector with random "small" coefficients selected from some distribution, s is a uniformly-random vector in $\mathbf{Z_q^n}$, and A is a uniformly-random matrix in $\mathbf{Z_q^{m\times n}}$. It has been demonstrated by Applebaum et al[10]. that the secrets in the LWE problem do not have to be selected uniformly at random[1][10]; if s is selected from the same restricted distribution as the mistakes e, the task will still be challenging.

Among the NIST-approved quantum-resistant algorithms, Kyber [9] stands out as a Key Encapsulation Mechanism (KEM) rooted in lattice-based Modular Learning with Errors (MLWE), targeting the GapSVP problem[1]. Other lattice-based cryptographic algorithms include Crystal-Dilithium [21], SABER [15], and NTRU [20], our algorithm development lies predominantly on Kyber due to its efficacy in end-to-end secure communication, Kyber is being actively implemented by various groups, although its applications remain largely un-updated. The Kyber's resilience against quantum attacks stems from its parameter sets and efficient operations like polynomial multiplication via Number Theoretic Transforms (NTT). Moreover, Kyber introduces significant noise during ciphertext generation, amplifying the complexity of attacks like BKZ[19], LLL(Lenstra–Lenstra–Lovász) reduction, BKW and primal attacks[16], rendering sieving operations exponentially more challenging. For secure messaging and communication protocols, initial endeavors have been made towards enhancing Signal's protocol [13], although updates to the application are still pending. A similar situation persists with the Messaging Layer Security (MLS) protocol developed by the Internet Engineering Task Force (IETF)[13] designed for end-to-end secure messaging across various group sizes.

In this work, we have introduced a lattice-based algorithm designed to withstand quantum attacks, inspired by the Learning With Errors (LWE) scheme and the Shortest Vector Problem (SVP). Drawing from the Odded Regev-style encryption and Kyber technique, we have developed an encryption method for generating cipher text. Notably, our algorithm focuses on optimizing space for the public key while enhancing security through increased noise in the cipher text. We generate the public key matrix as in LWE encryption with a seed and reduce the space to store the public key in less number bytes by not sending the whole matrix but rather sending the seed along with 2 more vectors through which every row and column could be generated,

We have successfully implemented this algorithm and we have built a server-client model, enabling end-to-end secure communication. As part of this implementation, we are developing a messaging application designed to operate over a quantum-resistant channel. We have made a Chosen cipher text attack(CCA)-secure Public key encryption (PKE). The random-oracle model's security reduction from the CPA-secure scheme is tight, whereas the quantum random-oracle model's security reduction is non-tight. One may build CCA-secure encryption key exchange schemes in a black-box fashion from a CCA-secure KEM.

Our algorithm demonstrates promising results in accurate encryption and decryption, we are actively addressing challenges such as reducing the size of cipher text and minimizing multiplication complexity. Our ongoing efforts aim to achieve a time complexity of $O(n \ log \ n)$ instead of $O(n^2)$, enhancing efficiency without compromising security. Our primary focus is on light and efficient implementation aspects of the algorithm, with the ultimate goal of establishing a highly secure communication channel resilient to quantum threats.

1.1 PRELIMINARIES

1.1.1 Notations

The set of integers is denoted by $\mathbb{Z}$. $\mathbb{Z}_q$ denotes the finite field of cardinality q and its elements are represented by the integers in the interval $[-[\frac{q}{2}], \frac{q}{2}]$. $B(n)$ denotes an encoded array of n integers like $\{a_1, a_2, a_3...., a_n\}$ each integer takes 4 bytes each makes the value range of 0 to $2^{32}-1$. $\mathcal{D}$ denotes the Gaussian distribution and $\mathcal{D}_\sigma$ with standard deviation σ. Here, $x \leftarrow \chi$ denotes the set of x belongs from the distributed error terms generated which is denoted by χ. $\Pr(X)$ denotes the Probability of X.B_n^k denotes an array of 32 bits representing the ASCII value for a character.

Extendable Output Function. XOF (Extendable Output Function) is a cryptographic primitive that takes a seed as input and produces output of any desired length. It's like a versatile cryptographic tool that can generate data according to specified distributions. Let's say we have a seed *seed* and we want to use XOF to create matrices and vectors. We start by initializing XOF with the seed seed, and then we use it to generate the matrices and vectors[22]. For example, we can generate a matrix A by calling XOF with a specific seed seed$_A$, and similarly, we can generate a vector b by calling XOF with a seed seed$_b$. Let seed be the seed used by the XOF. Particularly we use variants of SHA(Secure hashing algorithm)-256 and SHA-128 pseudo number generators in these functions which give us a hash value of a definite length. To generate a matrix A, we call XOF with seed seed$_A$:

$$A \leftarrow \text{XOF}(\text{seed}_A)$$

Similarly, to generate a vector b, we call XOF with seed seed$_b$:

$$b \leftarrow \text{XOF}(\text{seed}_b)$$

The process is deterministic, meaning that for a given seed, XOF will always produce the same output. However, in practice, XOF may use random oracles to achieve this determinism while statistically approximating the desired distribution.

H and G functions. G is the function that when you send the $A.s + eands\_t$ combine the function output the results as stored in p_t as $\left(\sum_{i=0}^{256} A \cdot (s_i - s_{s_{t_i}})\right)$ so that the redundant lattice dimensions which was chosen randomly from 256 dimensions will not the be added in the resultant p_t which form the public key. H function is used in the decryption algorithm where it takes the ciphertext and uses it to provide the absolute difference between the ciphertext matrix value and the value that came from putting s in those equations in ciphertext with redundant dimensions and then we check the probability that it is less than $\frac{q}{4}$ or not for decoding the bits.

1.1.2 Gaussian Distribution

As Regev [1] demonstrates a quantum reduction from worst-case lattice issues to LWE with Gaussian error, given arbitrarily many LWE samples, the Gaussian distribution is the default error distribution for classical LWE.

Definition. Let $\mathcal{D}_\sigma$ be the Gaussian distribution with standard deviation σ. The rounded Gaussian distribution function $\mathcal{D}_\sigma^{\mathcal{R}}$ is defined as follows:

For any $x \in \mathbb{R}$, the sample y from $\mathcal{D}_\sigma^{\mathcal{R}}$ is obtained by rounding x to the nearest integer and adding a sample from $\mathcal{D}_\sigma$.

In mathematical notation:

$$y = \lfloor x + z + 0.5 \rceil$$

where z is a sample from $\mathcal{D}_\sigma$. If X follows a Gaussian distribution then the rounded Gaussian of X_Z will be

$$\Pr(X_{\mathbb{Z}} = N) = \int_{N+1/2}^{N-1/2} g_{\mu,\sigma}\, dx$$

In the following μ = 0.

Goal. In here we have to find the distribution of $X_{\mathbb{Z}_q}$ = $X_{\mathbb{Z}}$ mod q.

$$\Pr(X_{\mathbb{Z}_q} = [a]) = \Pr(X_{\mathbb{Z}} = a + kq) \quad \text{for some } k \in \mathbb{Z}$$

$$\sum_{k\in\mathbb{Z}} \int_{a+kq+1/2}^{a+kq-1/2} g_{0,\sigma}(x)\,dx = \int_{a+1/2}^{a-1/2} f_{\sigma,q}(x)\,dx$$

Where

$$f_{\sigma,q}(x) : x \to \sum_{k\in\mathbb{Z}} \frac{1}{\sigma\sqrt{2\pi}} \exp \frac{-(x+kq)^2}{2\sigma^2}$$

1.1.3 LWE definitions

Learning With Errors (LWE) is a fundamental problem in lattice-based cryptography and serves as the basis for many encryption schemes and cryptographic protocols[1][3][4]. It involves the challenge of distinguishing between noisy information and random noise in a set of equations. terms e_i.

Notation. Let χ be the rounded gaussian mod q and $e_i \leftarrow \chi$.

Definition. $q > 0$ be a modulus and $m, n > 0$. Now given m samples of such equations of n dimesnions in the form of $\mathbf{a_i}, <\mathbf{a_i}, \mathbf{s_i}> + \mathbf{e_i}$ where $\mathbf{a_i}$ is uniform at random in $\mathbb{Z}_q^n$ and $e_i \leftarrow \chi$. Find the secret $s \in \mathbb{Z}_q^n$

The LWE can be formulated more compactly as a matrix-vector equation:

$$A \cdot s + e = k \mod q$$

Example equations taking 100 LWE samples with n dimensions:

$$58s_1 + 47s_2 + 63s_3 + \ldots + 29s_n = k_1 + e \mod \text{(q)}$$
$$5s_1 + 12s_2 + 74s_3 + \ldots + 6s_n = k_2 + e \mod \text{(q)}$$
$$107s_1 + 17s_2 + 21s_3 + \ldots + 27s_n = k_3 + e \mod \text{(q)}$$
$$\ldots$$
$$s_1 + 43s_2 + 116s_3 + \ldots + 25s_n = k_{100} + e \mod \text{(q)}$$

where:

- A is a known matrix with rows corresponding to randomly generated coefficients x_i.
- e is a vector of error terms e_i.
- s is a vector of known values x_i.
- k is the resultant for each equation after $\sum_{i=0}^{n} A.s_i$
- q is a prime number to module the resultant

The LWE problem is challenging because the error terms e_i are typically drawn from a random distribution, making it difficult to distinguish between the true solution and random noise. The LWE's security relies on the hardness of solving this problem, even when given multiple noisy equations. In practical lattice-based cryptography, LWE is used to build secure encryption schemes such as Ring-LWE, which is a variant of LWE, and many other cryptographic primitives. The security of these schemes is based on the assumption that solving the LWE problem is computationally infeasible, even for powerful adversaries.

Hardness. The LWE problem is difficult for several reasons. First of all, even quantum algorithms don't seem to be able to help because the most effective LWE algorithms operate in exponential time. Second, since it is a logical extension of the LPN issue, it has been well explored and is generally accepted to be a challenging topic in learning theory. Furthermore, as LPN involves the issue of decoding from random linear binary codes, any advancement in algorithmic techniques for LPN is likely to result in advances in coding theory. It is known that LWE is hard because of several assumptions about the worst-case difficulty of typical lattice problems like SIVP (the shortest independent vectors problem) and GapSVP (the decision version of the shortest vector problem) [1][4]. More specifically, the conventional assumption that GapSVP is difficult to estimate within polynomial factors underlies difficulties when the modulus q is exponential. The hardness is based on somewhat less common (but still quite credible) assumptions for polynomial moduli q, which is the more important setting for cryptographic applications.

To be more precise, this means that either GapSVP or SIVP are difficult to approximate within polynomial factors, even with a quantum hint, or that GapSVP is difficult to estimate even when provided with a short basis.[3]

CHAPTER 2

ALGORITHM AND SPECIFICATIONS

2.1 Defining the Parameters

The algorithm is based on the LWE encryption scheme which was introduced by Odded Regev. We adopted the approach [10] for the generation of the public matrix A and used the SVP problem to make the public key and secret key. The main difference in creating the lattice dimensions in our algorithm and others is that we are using the concept of reduced dimensions which means we would not take 'k' dimensions in counter to calculate the resultant lattice equations and that would work as our second secret key. It also increases the complexity of lattice reduction techniques.

Similar to other post-quantum cryptography (PQC) algorithms in which parameters play an important role in their security strength. We have also defined a set of parameters in it. The algorithm is parameterized by n, k, q, γ, C.

n	m	k	γ	q	C
256	150	25	5	3907	10

- The reason n is set to 256 is that the intention is to utilize a plaintext size of 256 bits in our algorithm in order to encapsulate keys with 256 bits of entropy. The lower value of n will reduce the noise and affect the security.
- q is chosen such that to achieve the probability of decryption failure to be negligible. As seen in Kyber also prime number approx 2^{12} could be taken to define a good security parameter,[12]
- k is the number of LWE samples chosen such that to resist the BKW, BKZ attacks with LLL reduction techniques[16][19].
- γ is chosen such that to maintain the balance between the security level and computational timing. It's the number of equations that will be selected from the public key to add for a single bit with modulo q.
- C is the compression factor for the large coeffiecnts, However in our algo, only p_t is compressed till now. Rest are not. But in further versions, ciphertext coefficients will be also compressed to reduce the size but maintain the same level of security.

.

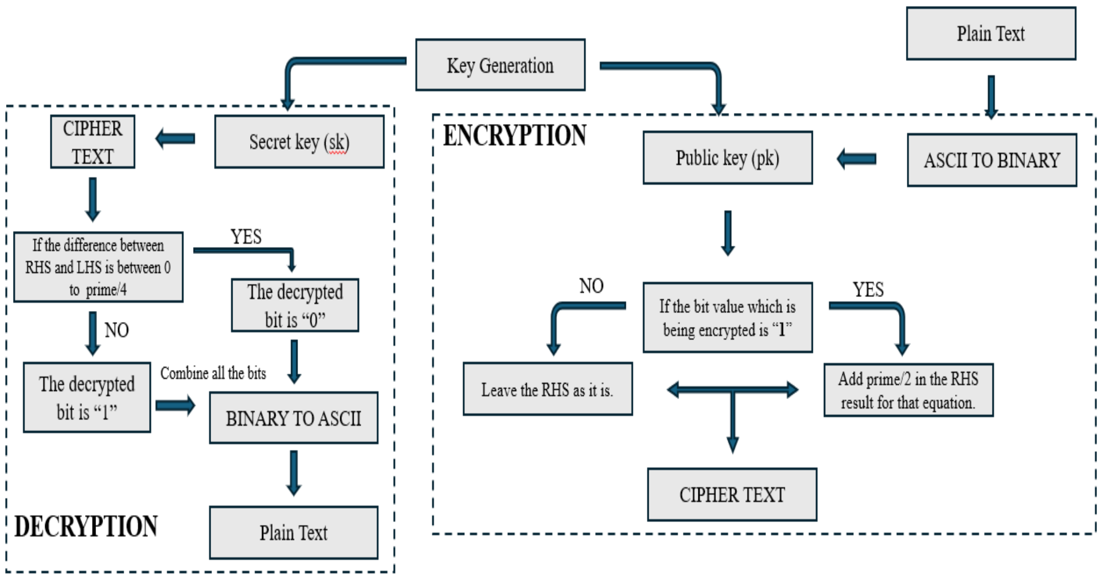

Figure 2.1: Simple flowchart run of algorithm (IND-CCA secure PKE)

2.2 Key Generation

At the beginning, several arrays (sk, sk_t, pk, pk_t, a) and a prime number (prime) are initialized. These arrays are used to store secret keys, public keys, and other parameters used in the encryption and decryption processes.The "generateUniqueRandomNumbers" function is defined to generate a specified number of unique random numbers within a given range. This function is used to generate secret keys, public keys, and other random values required for encryption by SHAKE-128 in its standard variant and AES-256 in CTR mode in its 90s variant as XOFs to generate pseudorandom output for cryptographic operations. The secret_key_gen and public_key_gen functions are defined to generate secret keys (sk, sk_t) and public keys (pk, pk_t) based on random numbers generated using the XOF. In the provided code, the equation

$$A \cdot s + e = pk\_t$$

represents the encryption process. Here:

- A is a matrix used in the encryption process made from pk,a.
- s is the secret key vector, where sk_t represents the dimensions that are excluded from multiplication with matrix A to calculate the resultant vector present in t. Excluding dimensions in s increases the hardness to reduce the lattice for short vectors.
- e is an error vector, which introduces randomness and ensures that the encryption process is secure its range is being calculated by the Gaussian function.
- pk_t is the encrypted text vector which is then compressed and sent as a public key(pk).

Algorithm 1 KeyGen(): Key generation

```
1:  $\sigma, \rho \leftarrow \{0,1\}^{256}$ and $\alpha \leftarrow \{0,1\}^{k}$
2:  $a \sim B(n) :=$ XOF$(\sigma)$
3:  $p \sim C(k) :=$ XOF$(\alpha)$
4:  $s \sim D(n) :=$ XOF$(\rho)$
5:  $e \sim \mathcal{D}_{\sigma}^{m}$
6:  $A \sim R_q^{n \times m}$
7:  for $i = 0$ to $k$ do
8:      for $j = 0$ to $n$ do
9:          $A[i][j] := (pk[i]ora[j])$
10:     end for
11: end for
12: $p\_t \sim P(m) :=$ Com$_q$(G$((A.s + e), s\_t), x)$
13: return $((pk :=$ ($\sigma$, $\alpha$, p_t) , sk:= (s, s_t))
```

2.3 Encryption and Cipher Text

The encryption function is defined to encrypt a given message. It iterates over each character of the message, converts it to binary representation, encrypts each bit using the public key, and generates encrypted text based on the encryption algorithm. Message Processing: Refer to Fig. 1. The code iterates over each character of a given message (s), converts it to binary representation, encrypts the binary representation using the encryption function, and stores the encrypted text in the vector array. The plain text “m” then first we have to convert into binary so “M” = “1 0 1 1 0 0 1” let’s suppose. Now, each of its bits will randomly take any “γ” rows of “A” and “pk_t” and add them to form a new set of equations. So, here let’s suppose at random we chose any k number of rows then for the 1st bit the equation will be. And since this bit is equal to “1” we will also add in the RHS side with q/2 else 0. For example,

Algorithm 2 Enc(pk, m): Encryption

```
1:  $M \leftarrow \mathcal{B}_m^{32}$
2:  $dim \leftarrow D(\gamma) := \beta_k^{\gamma}$
3:  $p\_t \leftarrow$ Decom$_q(p\_t, x)$
4:  $a \leftarrow$ XOF$(\sigma)$ and $p \leftarrow$ XOF$(\alpha)$
5:  $u \sim R_q^{M \times n}$ and $v \sim R_q^{M}$
6:  for $l = 0$ to size of $M$ do
7:      $u[l] = \sum_{i \in dim} \sum_{j=0}^{256} (a[j]$ or $p[l])$
8:      $v[l] = \sum_{i \in dim} p\_t[i]_q + M \times [q/2]$
9:  end for
10: return $c := (u, v)$
```

2.4 Decryption

- The decryption process attempts to recover the original bits from the encrypted data in `encrypt_txt` shown here in vectors (u,v).
- It calculates dot products between the secret key vector with u. These dot products are then compared to the final dimension coefficients to determine if the result represents a 0 or 1.
- As we have our encrypted text one user knows that the last number of every single line is the resultant within that particular line equation and for that single bit it equation coefficient is written in that line except the last number which is indeed the resultant containing error with that for that particular equation.
- The algorithm considers certain ranges for the differences between calculated results and dimension coefficients to interpret whether the original bit was a 0 or 1.

Algorithm 3 Dec(sk = s, c = (u , v)): Decryption

```
1: $m := B(\text{size of u})$
2: for $i = 0$ to size of $u$ do
3:     $m[i] = if(Pr[H(v[i] - \sum_{j=0}^{256} s.u[i][j])] < [q/2]) := 0$
4:     else $m[i] = 1$
5: end for
6: return m
```

CHAPTER 3

Correctness and Security Against Known Attacks

3.1 Correctness

In the absence of errors in the LWE samples provided, the inner product between the error $e = \langle a, s \rangle$ and the secret s would be either 0 or $\lfloor \frac{q}{2} \rfloor$, it depends on the encrypted bit. Consequently, decryption will be always correct. Hence, a decryption error occurs only if the sum of the error terms over all secrets is greater than $\frac{q}{4}$. Since we are summing at most m normal error terms, each with a standard deviation αq, the standard deviation of the sum is at most $\sqrt{m}\alpha q < \frac{q}{\log n}$. A standard calculation shows that the probability of such a normal variable being greater than $\frac{q}{4}$ is negligible.
We now illustrate the security proof, demonstrating that the system is secure according to the LWE supposition. Assume that the system is vulnerable to the selected plaintext assaults, meaning that there For a non-negligible proportion of secrets s, there exists an efficient method that, given a public key (ai, bi) mi=1 selected from the LWE distribution with some secrets and encryption of a random bit created as above, can accurately predict the encrypted bit with probability at least 1/2+1/poly(n)[1]. With Our parameter set and algorithm, many attacks seem useless.

3.2 Analysis against known attacks and hardness

3.2.1 Core-SVP Hardness

The SVP hardness depends upon how much time to solve for the given parameter set more likely the number of equations, modulus size, and the number of dimensions. Various lattice reduction techniques allow to production of the shortest vector in exponential time complexity. No such known algorithm now even with the help of quantum algorithms can solve a particular LWE problem in polynomial time. Blum, Kalai, and Wasserman's work [23] yields a far more intriguing approach that takes just $2^{O(n)}$ samples and time. The method works by breaking down the n coordinates into $\frac{1}{2}$ $logn$ blocks of size $2n/logn$ each, and then building S recursively by

finding collisions in these blocks. This is based on a clever idea that finds a small set S of equations (let's say, of size n) among $2^{O(n)}$ equations, such that $\sum_S a_i$ is, let's say, (1, 0,..., 0). By summing those equations, we get a very noisy estimate for the first coordinate of s, but a typical computation still reveals a bias of around $2^{\theta(n)}$ toward the correct value. Therefore, by repeating the preceding technique just $2^{O(n)}$ times, we have a good likelihood of recovering the value of s1 and thus all of the s.
Blum et al. approach is the most well-known technique for solving the LWE problem. The best-known algorithms for lattice problems [24], [25], take $2^{O(n)}$ time, which is closely connected to this.

3.2.2 Enumeration vs. sieving

There are two primary algorithmic approaches for solving the Shortest Vector Problem (SVP) within the Block Korkin-Zolotarev (BKZ)[27,28] lattice reduction framework: enumeration and sieving algorithms. These methods exhibit distinct performance characteristics, with sieving algorithms presenting challenges in predicting their practical scalability from smaller to larger lattice dimensions, especially in contexts like attacks against cryptographic systems such as Kyber. Enumeration algorithms demonstrate super-exponential running time, whereas sieving algorithms exhibit exponential running time. Empirical evidence drawn from typical implementations of BKZ indicates that enumeration algorithms are more efficient in smaller dimensions. Consequently, one crucial inquiry is determining the dimension at which sieving becomes more efficient. Initially, during the NIST Post-Quantum Cryptography (PQC) project's inception, the best-known sieving techniques proved slower in practice for dimensions accessible up to approximately $b \approx 130$. However, research efforts, as highlighted in[29], demonstrated practical speed enhancements for sieving techniques in exact-SVP scenarios, narrowing the efficiency gap with enumeration to less than an order of magnitude in dimensions ranging from 60 to 80. One contributing factor to this improvement is the "dimensions-for-free" technique, enabling SVP solutions in dimension b by sieving in a slightly smaller dimension $b_0 = b - d_{4f}$. Subsequent algorithmic enhancements and implementation refinements ultimately led to sieving outperforming enumeration in practice, with a crossover occurring around dimension 80.

Numerous recent studies have made significant strides in enhancing the efficiency of conventional lattice-sieving algorithms[30]. These advancements have led to improvements in heuristic complexity, reducing it from approximately $2^{0.415b+o(b)}$ to $2^{0.292b+o(b)}$ through the integration of locality-sensitive hashing (LSH) techniques.

Furthermore, research has demonstrated that many sieving algorithms benefit from Grover's quantum search algorithm, resulting in complexity reductions to $2^{0.265b+o(b)}$. For initial analysis, we'll consider $2^{0.292b}$ as the classical cost estimate, while $2^{0.265b}$ serves as the quantum cost estimate for primal with a block size (dimension) of b.

3.2.3 Primal attack

The primary attack involves creating a distinct SVP instance from the LWE problem and applying BKZ to solve it. We investigate the minimum block dimension b that BKZ needs to find the unique solution. Given LWE instance (A, b = As+e) then one make a lattice Λ = { x $\in \mathbb{Z}^{m+1}$ (A$|I_m| - b)x$ =0 mod q } and then it will generate a distinct or unique solution v = (s, e, 1) of norm $\lambda = \zeta\sqrt{kn+m}$ where the ζ shows the standard deviation[25][16][12].

Success condition. With our parameter set at first it will be very difficult to make an accurate Λ or the basis matrix

$$B = \begin{bmatrix} I_{n\times n} & \text{A'} & 0 \\ 0_{m-n\times n} & qI_{(m-n)\times(m-n)} & 0 \\ c^T & & t \end{bmatrix} \in \mathbb{Z}^{(m+1)\times(m+1)}$$

Since we have used redundant dimensions our c matrix which is (A.s + e) doesn't show the exact results for all the dimensions rather it shows the lattice of (n - M) dimensional lattice with error so that the original c matrix is hard to retrieve which will affect the lattice reduction technique to guess the correct set of vector and for even with the efficient algorithm it would take more than O(n^2) to search for each possibility. Even if it guessed the dimensions and got some c the condition of success will be Let e^* be the projection of e orthogonally onto the first $d-b$ vectors of the Gram-Schmidt basis B^*.

BKZ-like algorithms will call an SVP oracle on the last block of dimension b.

If e^*_{d-b} is the shortest vector in that block, it will be found.

If e^*_i is the shortest vector for all projections up to $d-b$, it will "travel to the front".

Assume $\|e^*_{d-b}\| \approx \sigma \cdot \sqrt{b}$.

Applying the GSA, we expect the shortest vector to be found in the last block to have a norm

$$\|b^*_{d-b+1}\| = \alpha_{d-b} \cdot \delta_d^0 \cdot \text{Vol}(B)^{\frac{1}{d}} = \delta^0_{-2(d-b)} \cdot \delta_d^0 \cdot \text{Vol}(B)^{\frac{1}{d}}$$

$$= \delta^0_{2b-d} \cdot \text{Vol}(B)^{\frac{1}{d}}.$$

Thus, we expect success if

$$\sigma \cdot \sqrt{b} \leq \delta_{2b-d}^{0} \cdot \mathrm{Vol}(B)^{\frac{1}{d}}.$$

And for the parameters, we numerically optimized it to defend against these types of attacks. Our parameter set allows security against BKZ attacks. The time complexity for these attacks is still exponential but it's one of the best-known attacks to uSVP.

3.2.4 Side-channel attacks.

The implementation of our algorithm is meticulously crafted to mitigate the risks associated with side-channel attacks, particularly timing-based attacks and differential attacks. Through careful design and rigorous testing, we've ensured that potential vulnerabilities, such as timing leakage during modulo operations, are effectively minimized. Our algorithm is engineered to operate within a tight timing bound, maintaining near-constant execution time across various inputs and conditions. While it's acknowledged that certain aspects of the implementation, such as modular reduction, may introduce potential leakage points, we've taken proactive measures to address these concerns. By leveraging device-specific optimizations and implementing countermeasures directly into the algorithm, we mitigate the risk of side-channel attacks. Furthermore, continuous evaluation and refinement of our implementation practices ensure that we maintain a robust security posture against emerging threats and attack vectors.

3.2.5 Hybrid attacks.

The hybrid attack, which merges Meet-in-the-Middle combinatorial search with lattice reduction techniques, poses a significant threat to various cryptographic algorithms. However, the intricate nature of this attack presents challenges in its analysis. Recent research suggests that its effectiveness may be less formidable than previously thought, highlighting the evolving landscape of cryptographic vulnerabilities.In our design, where errors and secrets are not ternary and are limited in occurrence, the applicability of this attack varies depending on the specific context. While it remains a pertinent consideration in certain scenarios, our design principles and mitigation strategies are tailored to address such threats effectively.

CHAPTER 4

IMPLEMENTATION

In this section, we will provide the structure of our model of end-to-end communication channel via the client-server model and how we implemented it in the device, along with the reference implementation of the algorithm and the required encoding of the algorithm in the PKE system for the E2E channel. As seen in the figure, the implementation was done on a client-server model. In which the sender and the receiver have been connected to the devices which are indeed connected with the server that has a database. A new user when comes to the server and registers with the help of the algorithm in the backend will be a unique public key and a private key will be generated on the database for each of the users. The public generated is in the form of of vector of vectors having three vectors of pk_t, pk, a, working as a seed vector that can able to generate the whole matrix A with the help of a and pk.

Now, for the generation of the cipher text from a plaintext that a sender (client 1 in fig.) wants to send to the receiver (client 2 in fig.) the encryption algorithm (Algorithm 2) works in the backend, firstly, it makes an API call to get the public key from the database. Using the public key pk of the receiver the senders create the cipher text Enc($\mathbf{pk}$) to get the cipher text and send it back to the server and store it in the database as in encrypted format. When the server realises there is a change it makes a call back to the receiver's end and the API traces the change and reflects it to the receiver's side. Now, since we got the cipher text and now with the help of Algorithm 3 (Decryption) the previously stored secret key at the receiver's end comes to use to decrypt this cipher text using Dec($\mathbf{c}, \mathbf{sk}, \mathbf{sk\_t}$)

The use of the local storage here is to keep the secret keys in it with the help of cookies, we parse the sk and sk_t with cookie tokens to the local storage. After that, we could access that token form the backend and retrieve the secret keys out of that which will able to decrypt the incoming messages and show us in the chat box. Now, since everything is running smoothly there is still an issue of storage of ciphertext since the text size is about $\mathbf{O(nm)}$ where the m denotes the number of bits per character generally '7'. So, it's approx 1800 integers each taking approx 8-12 bytes making an overall space of 7.2 KB and for 250 such characters which is the current character limit for our algorithm implementation in the chat server, it comes approx 1.8 MB which turns out to be a lot as of now. So, to reduce this we are implementing a $nlogn$ space

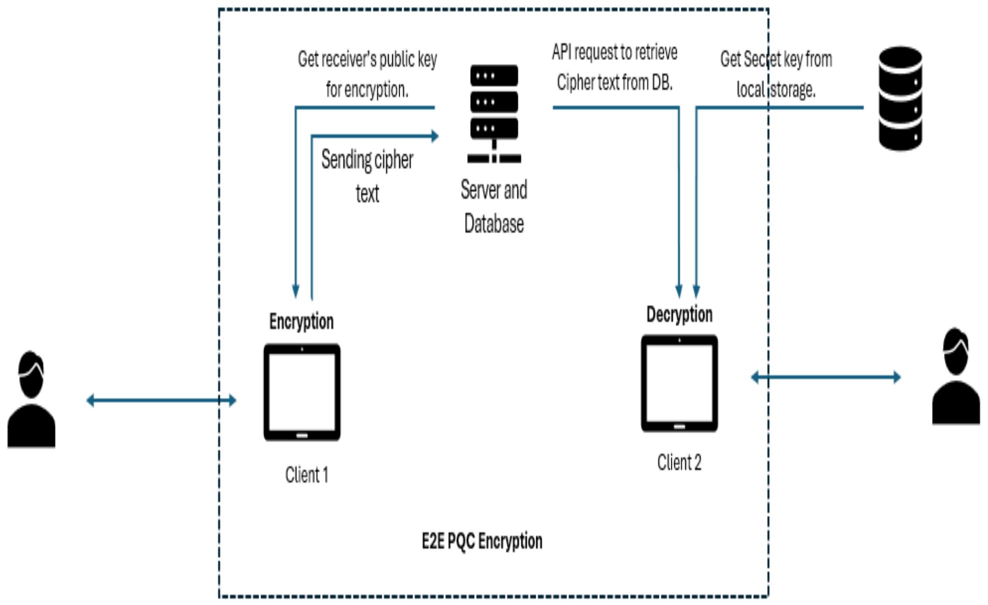
Get receiver's public key for encryption.
Sending cipher text
Server and Database
API request to retrieve Cipher text from DB.
Get Secret key from local storage.
Encryption
Decryption
Client 1
Client 2
E2E PQC Encryption

Figure 4.1: A flowchart of the client-server model implemented with this algorithm for an end-to-end secure channel with QKD protocol.

size algorithm which would encrypt multibit at the same time. To reduce space it would also make the runtime lesser.

Further work on key management systems and user and software interfaces may be necessary for encryption using QKD devices to guarantee asynchronous and long-term access to encrypted messages. The demand is presently not being met by the key rates, particularly when it comes to the integration of QKD keys into systems other than site-to-site VPNs. This issue may be resolved by deriving many keys from the original key that QKD supplied. For instance, one may utilize a single QKD key with keys that are generated for every message transmitted between two fixed participants on the same day. It could also go beyond the classical client-server model and has many use cases such as a secure quantum channel for the military or some esteemed organization. Since we are successfully able to integrate it into the system and establish secure communication between two parties. There can be still some key exchanges and session chatting features which we are introducing further.

4.1 Computational Complexity for Attacker

The difficulty of breaking an encryption scheme based on Learning With Errors (LWE) is closely tied to the challenge of solving the Shortest Vector Problem (SVP) for the lattice defined by the encryption parameters. As the dimensionality of the lattice increases, the computational complexity of solving SVP grows exponentially, making it extremely challenging for attackers.

Randomly choosing dimensions for each bit during encryption adds another layer of complexity for attackers. They would need to consider all possible combinations of dimensions for each bit, significantly increasing the computational burden.

Furthermore, the addition of random error terms during encryption adds further uncertainty. Attackers must account for these errors, making decryption even more challenging.

In practice, the hardness of breaking an LWE-based encryption scheme depends on factors such as the dimensionality of the lattice, the modulus used, and the number of random dimensions chosen during encryption. Solving SVP in high-dimensional lattices is believed to be exponentially complex, even for powerful computers.

In a client-server model where chats are stored in a database and retrieved for encryption and decryption, LWE-based encryption adds an additional layer of security. The encrypted chats stored in the database are protected by the complexity of solving SVP and the uncertainty introduced by random error terms. This makes it extremely difficult for attackers to decrypt the messages without access to the encryption keys, even if they gain access to the encrypted data in the database. The computational complexity of breaking the encryption scheme acts as a strong deterrent against unauthorized access to sensitive information. This explanation delves into the intricacies of Regev's reduction, which essentially states that a quantum solver for the Learning With Errors (LWE) problem with certain parameters can be used to solve approximate versions of lattice problems like the Shortest Vector Problem (SVP) and Shortest Independent Vectors Problem (SIVP) over lattices of dimension n. However, this reduction doesn't necessarily imply that these lattice problems are NP-Hard for linear approximation factors.

Even if a quantum solver for LWE worked for parameters very close to 1, the resulting approximation factor for lattice problems would still be greater than n. Currently, we lack proof that these approximate lattice problems are NP-Hard for linear approximation factors. Negative results even exist for the hardness of these approximate problems for approximation factors greater than or equal to $\sqrt{n}$.

Therefore, while a quantum polynomial-time algorithm for LWE would place LWE within

BQP(Bounded quantum polynomial time), it wouldn't directly impact questions related to classical hardness or NP-hardness. However, from a practical standpoint, such an algorithm would still be a significant breakthrough. The approximate versions of lattice problems aren't known to be NP-Hard, and efficient algorithms for solving them remain elusive.

CHAPTER 5

PERFORMANCE ANALYSIS

Indeed, the algorithm provided produces different encryption text vectors for the same input each time it runs due to the use of random variables. This property is crucial for the security of cryptographic systems, as it ensures that even if an attacker observes multiple ciphertexts of the same plaintext, they cannot deduce any information about the original message without knowledge of the secret key. The sizes of the keys also plays a vital role in selecting the parameters. Below is the comparison of the sizes of the secret key, public key and ciphertexts are given for different algorithms.

Algorithm	Secret Key Size	Public Key Size	Ciphertext Size
Our Algorithm	1053 bytes	965 bytes	8799 bytes
Kyber768	2400 bytes	1184 bytes	1088 bytes
Kyber1024	3168 bytes	1568 bytes	1568 bytes
FRODO	11280 bytes	11296 bytes	11288 bytes
NTRU-scheme	1422 bytes	1140 bytes	1281 bytes

Table 5.1: Comparison of Secret Key, Public Key, and Ciphertext Sizes for Different Algorithms[12]

We have taken the strings of various sizes in kilobytes (KB) of different characters in length and ran our algorithm through various processors. We took the average after "20 iterations" of each test case and found this mean data. We have taken 256 randomized integers in our secret key and each time calculated the encryption and decryption timings. The data shown here are in seconds and we are expecting more better results after optimization. These are done on the Ryzen7 4800h processor.

Size of plaintext	Encryption	Decryption
1 KB	0.02 sec	0.003 sec
4 KB	0.408 sec	0.06 sec
15 KB	1.757 sec	0.25 sec
31 KB	3.64 sec	0.518 sec
59 KB	7.44 sec	1.013 sec
118 KB	14.3 sec	1.976 sec

Table 5.2: Time taken to encrypt and decrypt the plaintext of different sizes.

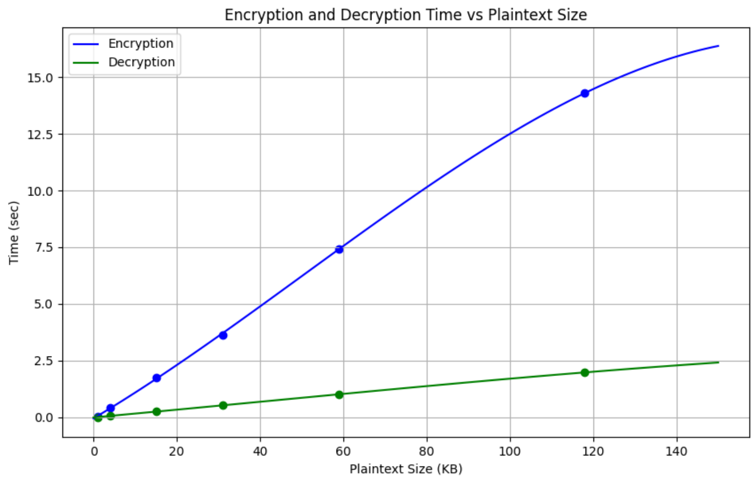

Figure 5.1: Time taken (in seconds) for encryption and decryption for different sizes of plaintext (in KB)

The graph for the same table is shown above. The graph illustrates how the encryption and decryption time gaps increase, indicating that the ciphertext computational time is a little bit longer since we are doing O(nγ) for each bit. To simultaneously minimize the size and temporal complexity, we are optimizing this timing and attempting to decrease it to O(lognγ). Also, we are trying to compress each coefficient of the equations generated by the random seed so that fewer bytes are being used in storage. Since the implementation of the client-server model requires restrictions on the size of messages sent at one time. That's why it's important to compress the coefficients but accordingly, we have to take the parameters carefully so that security won't get affected.

CHAPTER 6

Conclusion

The security of this lattice-based encryption scheme rests on the intractability of solving the SVP in high-dimensional lattices, especially when considering random dimensions and error terms. The algorithm generates a random basis base, random dimensions, and random errors for each bit, making it computationally infeasible for an attacker to recover the secret key or decrypt the ciphertext efficiently. Implementing and making a web portal for a suitable chat application secured with this algorithm for end-to-end quantum-resistant communication. There are still some queries regarding the size and storage of the ciphertext and messages and thus it creates restrictions on the length of the message sent at one time. There is more work needed to be done and can be done to optimize this without harming it's security and could introduce more parameters or error terms to increase much noise which increase the security level.

REFERENCES

[1] Oded Regev, "The Learning with Errors Problem" in *Blavatnik School of Computer Science, Tel Aviv University*.

[2] Katharina Boudgoust, Corentin Jeudy, Adeline Roux-Langlois, Weiqiang Wen "On the Hardness of Module Learning with Errors with Short Distributions," Journal of Cryptology, 2023, .

[3] David Balbás, "The Hardness of LWE and Ring-LWE: A Survey," *Universidad Politécnica de Madrid*, 8th October 2021.

[4] C. Peikert., ". Public-key cryptosystems from the worstcase shortest vector problem.," *In Proc. 41st ACM Symp. on Theory of Computing (STOC), pages 333– 342. 2009.*

[5] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, Damien Stehlé, "Algorithm Specifications And Supporting Documentation (version 3.01)," in *NIST PQC round 3*, January 31, 2021 .

[6] Daniele Micciancio Oded Regev, "Lattice-based Cryptography, IEEE, 2008.

[7] Huck Bennett†, "The Complexity of the Shortest Vector Problem," in *Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 170 (2022).*

[8] Ajtai and Dwork, "A public-key cryptosystem with worst-case/average-case equivalence" in *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*.

[9] Joppe Bos, Léo Ducas†, Eike Kiltz‡, Tancrède Lepoint§, Vadim Lyubashevsky¶, John M. Schanck, Peter Schwabe, Gregor Seiler††, Damien Stehlé‡‡, , "CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM" in *2018 IEEE European Symposium on Security and Privacy*.

[10] Applebaum, B.; Cash, D.; Peikert, C.; Sahai , "A. Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems" in *Proceedings of the Advances in Cryptology-Crypto, International Cryptology Conference, Santa Barbara, CA, USA, 16–20 August 2009.*

[11] Lei Bi, Xianhui Lu, Junjie Luo, Kunpeng Wang Zhenfei Zhang , "Hybrid dual attack on LWE with arbitrary secrets" https://doi.org/10.1186/s42400-022-00115-y.

[12] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, Damien Stehlé , "ACRYSTALS-Kyber Algorithm Specifications And Supporting Documentation (version 3.01)" in *NIST round 3 submission https://pq-crystals.org/kyber/data/kyber-specification-round3-20210131.pdf.*

[13] Christoph Döberl, Wolfgang Eibner ,Simon Gärtner, Manuela Kos, Florian Kutschera Sebastian Ramacher "Quantum-resistant End-to-End Secure Messaging and Email Communication" in *ARES '23: Proceedings of the 18th International Conference on Availability, Reliability and Security*.

[14] Uddipana Dowerah, Srinivasan Krishnaswamy "Towards an efficient LWE-based fully homomorphic encryption scheme", https://doi.org/10.1049/ise2.12052.

[15] UAndrea Basso "Lattice-based cryptography and SABER", in *Quantum CS seminar, Budapest, 25 March 2021.*

[16] Yu Wei1,2, Lei Bi1,2* , Xianhui Lu1,2 and Kunpeng Wang1,2 "LSecurity estimation of LWE via BKW algorithms", in https://cybersecurity.springeropen.com/articles/10.1186/s42400-023-00158-9.

[17] Miklós Ajtai, Ravi Kumar, D. Sivakumar "A sieve algorithm for the shortest lattice vector problem", in *STOC '01: Proceedings of the thirty-third annual ACM symposium on Theory of computing July 2001.*

[18] Andreas Hülsing, Joost Rijneveld, John Schanck, and Peter Schwabe. "High-speed key encapsulation from NTRU". In Wieland Fischer and Naofumi Homma, editors, in *Cryptographic Hardware and Embedded Systems – CHES 2017, LNCS. Springer, 2017.*

[19] Ziyu Zhao, Jintai Ding. "Practical Improvements on BKZ Algorithm", in *NIST ://csrc.nist.gov/csrc/media/Events/2022/fourth-pqc-standardization-conference/documents/papers/practical-improvements-on-bkz-algorithm-pqc2022.pdf.*

[20] Jeffrey Hoffstein, Jill Pipher, Joseph H. Silverman. "NTRU: A Ring-Based Public Key Cryptosystem ", https://www.ntru.org/f/hps98.pdf.

[21] Shi Bai, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler and Damien Stehlé "CRYSTALS-Dilithium Algorithm Specifications and

Supporting Documentation", *https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf.*

[22] Shay Gueron1,2 and Fabian Schlieker3

"Speeding up R-LWE post-quantum key exchange ", *https://eprint.iacr.org/2016/467.pdf.*

[23] A. Blum, A. Kalai, and H. Wasserman, "Noise-tolerant learning, the parity problem, and the statistical query model ", in *Journal of the ACM, 50(4):506–519, 2003..*

[24] M. Ajtai, R. Kumar, and D. Sivakumar "A sieve algorithm for the shortest lattice vector problem. ", in *In Proc. 33rd Annual ACM Symp. on Theory of Computing (STOC), pages 601–610. 2001.*

[25] D. Micciancio and P. Voulgaris. "A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations. ", in *In STOC. 2010.*

[26] Xue Zhang, Zhongxiang Zheng Xiaoyun Wang "A detailed analysis of primal attack and its variants. ", *https://link.springer.com/article/10.1007/s11432-020-2958-9*

[27] Yuanmi Chen and Phong Q. Nguyen "BKZ 2.0: Better lattice security estimates" in,*In Dong Hoon Lee and Xiaoyun Wang, editors, Advances in Cryptology – ASIACRYPT 2011, volume 7073 of LNCS, pages 1–20. Springer, 2011.*

[28] Nicolas Gama, Phong Q Nguyen, and Oded Regev. "Lattice enumeration using extreme pruning". in , *Henri Gilbert, editor, Advances in Cryptology – EUROCRYPT 2010, volume 6110 of LNCS, pages 257–278. Springer, 2010.*

[29] Léo Ducas. "Shortest vector from lattice sieving: a few dimensions for free" in, *Annual International Conference on the Theory and Applications of Cryptographic Techniques, pages 125–145. Springer, 2018.*

[30] Phong Q. Nguyen and Thomas Vidick. ". Sieve algorithms for the shortest vector problem are practical." , in *Journal of Mathematical Cryptology, 2(2):181–207, 2008.*