2208636

Cryptanalysis of Lattice Based Post-Quantum Encryption Schemes

Submitted as part of the requirements for the award of the
MSc in Information Security
at Royal Holloway, University of London.

Information Security Group
Royal Holloway, University of London
August 2022

Contents

1 Preliminaries

1.1 Abbreviations and Acronyms

BDD bounded distance decoding 9

BKZ Block Korkine-Zolotarev, Schnorr and Euchner, 1994 7

CCA chosen ciphertext attack 4

CCA2 adaptive chosen ciphertext attack 4

CPA chosen plaintext attack 4

CVP closest vector problem 6

FFT fast fourier transform 5

IND indistinguishability of ciphertexts 4

LLL Lenstra, Lenstra and Lovász, 1982 5

LWE learning with errors 3

NIST National Institute of Standards and Technology 3

NTT number theoretic transform 7

PKE public-key encryption 3

SVP shortest vector problem 7

Abstract

Many post-quantum encryption schemes rely on the hardness of the learning with errors (LWE) problem as the foundational building block for the schemes. This work aims to discuss the conjectured hardness of the LWE problem with an angle towards the NIST nominated scheme KYBER, looking at both the primal and dual attacks to solve lattice-based problems - with a specific interest towards the tailored improvements presented by MATZOV. The work aims to present a comprehensive and rigorous definition of lattice reduction algorithms (BKZ, LLL), lattice attacks (primal and (improved) dual attacks), and encryption schemes (KYBER), as well as the mathematical backgrounds of these concepts before discussing open questions uncovered during the discussion of said topics whilst raising our own for further debate.

1.2 Introduction

Since Regev detailed the LWE problem [19], many public-key encryption (PKE) schemes have relied on the hardness of the worst-case lattice problem reductions to average-case LWE for secure and efficient cryptography. Interest in this area has grown due to lattice problems' conjectured resilience to quantum computers, in light of Shor's proof of the weaknesses of classical cryptographic primitives [22]. The latest efforts in light of this work as well as that in furthering quantum computing has prompted a call for proposals for standardisation by the National Institute of Standards and Technology (NIST) of post-quantum cryptographic algorithms [23].

In this work, we shall detail both the LWE and the lattice problems it relies upon for its believed hardness, before understanding the current attacks and their applications to candidates for standardisation, with the goal of introducing new readers to modern post-quantum cryptography, discussing open questions and presenting new ones, hopefully for further research.

1.3 Definitions

Def 1.1 (PKE Scheme)**.** Let Σ be a PKE scheme, consisting of the following three algorithms:

- $KeyGen$
  Output: (pk, sk), where pk is the public key and sk, the private key.
- Enc
  Input: pk and m, where pk is as defined above and m is the plaintext message to be encrypted.
  Output: c, the ciphertext.
- Dec
  Input: sk and c as defined above.
  Output: m, the plaintext.

Σ must also satisfy a correctness condition:

$$\forall m \in \mathcal{M} \text{ and } \forall (pk, sk) \leftarrow \Sigma.KeyGen$$

$$\Sigma.Dec_{sk}(\Sigma.Enc_{pk}(m)) = m$$

(with probability 1 over the randomness of Enc)

Note: All security properties discussed will be for PKE schemes.

Def 1.2 (chosen plaintext attack (CPA) security, indistinguishability of ciphertexts (IND)-CPA)**.** "Indistinguishability of ciphertexts under chosen plaintext attack" for an encryption scheme Σ.

$$(pk, sk) \leftarrow \Sigma.KeyGen$$

The adversary $\mathcal{A}$ is given pk and outputs two messages m_0, m_1 (of equal length), and is also given a challenge ciphertext:

$$c = \Sigma.Enc_{pk}(m_b) \qquad \text{for a chosen } b \in \{0, 1\}.$$

$\mathcal{A}$ now generates a bit $a \in \{0, 1\}$, and if $a = b$ then $\mathcal{A}$ has succeeded.
The encryption scheme has CPA security if:

$$\mathbb{P}(\mathcal{A} \text{ succeeds}) = \frac{1}{2} + \varepsilon \qquad \text{where } \varepsilon \text{ is negligible.}$$

This can intuitively be thought of as if an attacker is given access to the public key (therefore able to encrypt plaintext's of their choice **but not decrypt**), then if given the encryption of one of two plaintexts, the attacker has negligible advantage over guessing.

Def 1.3 (chosen ciphertext attack (CCA) security, IND-CCA)**.** "Indistinguishability of ciphertexts under a chosen ciphertext attack" for an encryption scheme Σ.

$$(pk, sk) \leftarrow \Sigma.KeyGen$$

The adversary $\mathcal{A}$ is given pk and a decryption oracle $\mathcal{O}_{\Sigma.Dec_{sk}}$, and outputs m_0, m_1 (of equal length). The adversary is only able to query this oracle up until it receives the challenge ciphertext,

$$c = \Sigma.Enc_{pk}(m_b) \qquad \text{for a chosen } b \in \{0, 1\}.$$

$\mathcal{A}$ then generates a bit $a \in \{0, 1\}$, and if $a = b$ then $\mathcal{A}$ has succeeded. The encryption scheme has CCA security if:

$$\mathbb{P}(\mathcal{A} \text{ succeeds}) = \frac{1}{2} + \varepsilon \qquad \text{where } \varepsilon \text{ is negligible.}$$

Intuitively, this is if an adversary is able to ask for decryptions before given the challenge, once given the challenge ciphertext they have negligible advantage over guessing.

Def 1.4 (adaptive chosen ciphertext attack (CCA2) security, IND-CCA2)**.** "Indistinguishability of ciphertexts under an adaptive chosen ciphertext attack" for an encryption scheme Σ.

$$(pk, sk) \leftarrow \Sigma.KeyGen$$

The adversary $\mathcal{A}$ is given pk and a decryption oracle $\mathcal{O}_{Dec_{sk}}$, and outputs m_0, m_1 (of equal length). The adversary then receives the challenge ciphertext,

$$c = \Sigma.Enc_{pk}(m_b) \qquad \text{for a chosen } b \in \{0, 1\}.$$

but may continue to query $\mathcal{O}_{Dec_{sk}}$ provided the requested decryption is not of c. $\mathcal{A}$ then generates a bit $a \in \{0, 1\}$, and if $a = b$ then $\mathcal{A}$ has succeeded. The encryption scheme has CCA2 security if:

$$\mathbb{P}(\mathcal{A} \text{ succeeds}) = \frac{1}{2} + \varepsilon \qquad \text{where } \varepsilon \text{ is negligible.}$$

Intuitively, this can be thought of as if an adversary has the ability to decrypt any ciphertext of their choice, other than the challenge; the adversary still does not have enough information to decrypt the challenge.

1.4 Notation

We shall use $\mathcal{B}$ to denote the set of 8-bit unsigned integers (or bytes), ie. the set $\{0, ..., 255\}$. Reals and integers will be denoted by lowercase letters, distributions and sets by capital letters. Bold letters denote vectors (lower case)[1] or matrices (capital), with the ith element of a vector $\mathbf{v}$ being denoted by $\mathbf{v}_i$, and the jth column of a matrix $\mathbf{M}$ by $\mathbf{m}_j$. $\mathbf{M}[i][j]$ denotes the entry in row i and column j (all vectors will be column vectors unless otherwise stated). Let the concatenation of two (row) vectors $\mathbf{v}, \mathbf{w}$ be denoted by $(\mathbf{v}||\mathbf{w})$.

Def 1.5 (Linearly independent)**.** The set of vectors $\{\mathbf{v}_1, ..., \mathbf{v}_m\}$ are linearly independent if $\Sigma_{i=1}^{m} \lambda_i \mathbf{v}_i = 0$ only has the trivial solution, $\lambda_1 = ... = \lambda_m = 0$.

Def 1.6 (Span)**.** Let the span of a set of vectors $\{\mathbf{v}_1, ..., \mathbf{v}_m\}$ be the set of all linear combinations $\Sigma_{i=1}^{m} \lambda_i \mathbf{v}_i$, where $\lambda_i \in \mathbb{R}$

Def 1.7 (Inner product and norm)**.** Let $\mathbf{v}, \mathbf{w} \in \mathbb{R}^n$. The inner product of $\mathbf{v}$ and $\mathbf{w}$, $\langle \mathbf{v}, \mathbf{w} \rangle = \Sigma_{i=1}^{n} \mathbf{v}_i \mathbf{w}_i$ and the norm of $\mathbf{v}$, $|\mathbf{v}| = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}$.

1.4.1 Lattices

Let $\mathbf{B} \in \mathbb{R}^{d \times m}$ where the columns, $\mathbf{b}_i$, are linearly independent basis vectors[2]. This can also be thought of as a set $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_m\} \subset \mathbb{R}^d$, where as before, $\mathbf{b}_i$ are linearly independent basis vectors. The lattice generated by $\mathbf{B}$ is defined as $\mathcal{L} = \mathcal{L}(\mathbf{B}) := \{\mathbf{Bx} | \mathbf{x} \in \mathbb{Z}^d\}$. We say $Vol(\mathcal{L}) = \sqrt{\det(\mathbf{B}\mathbf{B}^T)}$.

Def 1.8 (Shortest Vector Problem, SVP)**.** Given a lattice $\mathcal{L}$ with basis matrix $\mathbf{B}$, find a non-zero vector $\mathbf{v} \in \mathcal{L}$ such that the length, $|\mathbf{v}|$, is minimal.

$\lambda_i(\mathcal{L}) = \min\{\max\{|\mathbf{x}_1|, ..., |\mathbf{x}_i|\} : \mathbf{x}_1, ..., \mathbf{x}_i \in \mathcal{L} \text{ are lineraly independent}\}$ - in other words, $\lambda_1(\mathcal{L})$ is the shortest non-zero vector in $\mathcal{L}$.

Def 1.9 (Closest Vector Problem, CVP)**.** Given a lattice $\mathcal{L}$ with basis matrix $\mathbf{B}$, and a vector $\mathbf{v} \in \mathbb{R}^n$, find a vector $\mathbf{w} \in \mathcal{L}$ such that $|\mathbf{v} - \mathbf{w}|$ is minimal.

Def 1.10 (Bounded Distance Decoding, BDD)**.** Given a lattice $\mathcal{L}$ with basis matrix $\mathbf{B}$, and a vector $\mathbf{v} \in \mathbb{R}^n$ such that its distance from the lattice is at most $(\min\{|\mathbf{w}| : \mathbf{w} \in \mathcal{L} \backslash \{\mathbf{0}\}\})/2$, find the closest vector $\mathbf{w} \in \mathcal{L}$ to $\mathbf{v}$. In other words, given a vector such that its distance from any lattice point is less than half the shortest vector of the lattice, find the closest vector to that given.

1.5 Standard Algorithms

Def 1.11 (fast fourier transform (FFT))**.** Given $f : G \to \mathbb{C}$ where G is an abelian group, FFT will evaluate $\hat{f}(\chi) := \Sigma_{g \in G} f(g)\chi(g)$ for all $\chi \in \hat{G}$ where $\hat{G}$ is the dual group of G. $\hat{G}$ is defined as the set of all homomorphisms from G into the multiplicative group of roots of unity in $\mathbb{C}$.

For $f : (\mathbb{Z}/q\mathbb{Z})^n \to \mathbb{C}$, the Fourier transform is $\hat{f}(\mathbf{v}) = \Sigma_{\mathbf{u}} e^{\frac{2\pi i}{q} \mathbf{v}^T \mathbf{u}} f(\mathbf{u})$.

Def 1.12 (Lenstra, Lenstra and Lovász, 1982 (LLL) [25])**.** Let $\mathbf{B}$ be the basis matrix for a lattice $\mathcal{L}$, with the corresponding Gram-Schmidt orthogonal basis matrix $\mathbf{B}^*$. This is found by first computing the Gram-Schmidt coefficients

$$\mu_{ij} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle} \qquad , 1 \leq j \leq i \leq n$$

[1] with 0 denoting the zero vector.

[2] We will assume that $d = m$ unless otherwise stated.

and set

$$\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^*$$

with $\mathbf{b}_1^* = \mathbf{b}_1$. Then $\mathbf{B}$ is δ-LLL-reduced if both conditions below hold.

1. $|\mu_{ij}| \leq \frac{1}{2}$ for $1 \leq j \leq i \leq n$.
2. $|\mathbf{b}_i^*|^2 \geq (\delta - \mu_{i,i-1}^2)|\mathbf{b}_{i_1}^*|^2$ for $2 \leq i \leq n$.

This gives us algorithm 1.

Algorithm 1 LLL

1: **Input:** Lattice basis $\mathbf{B}$, $\delta \in (\frac{1}{4}, 1)$ a hermite constant
2: $k = 2$
3: $\mathbf{b}_1^* = \mathbf{b}_1$
4: $c_1 = |\mathbf{b}_1^*|^2$
5: **if** $k > n$ **then**
6: **return B**
7: **end if**
8: **for** $k \leftarrow (k-1), 1$ **do** ▷ Reduce $\mathbf{b}_k$ using previous $\mathbf{b}_j$'s.
9: $\mu_{ij} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle}$, $1 \leq j \leq i \leq n$
10: $\mathbf{b}_k = \mathbf{b}_k - \lceil \mu_{kj} \mathbf{b}_j \rceil$
11: $\mathbf{b}_k^* = \mathbf{b}_k - \sum_{i=1}^{k-1} \mu_{k,i} \mathbf{b}_i^*$
12: $c_k = |\mathbf{b}_k^*|^2$
13: $\mu_{ij} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle}$, $1 \leq j \leq i \leq n$
14: **end for**
15: **if** $c_k \geq (\delta - \mu_{k,k-1}^2) c_{k-1}$ **then**
16: $k = k + 1$
17: **Go to line 3**
18: **else**
19: $temp = \mathbf{b}_k$
20: $\mathbf{b}_k = \mathbf{b}_{k-1}$
21: $\mathbf{b}_{k-1} = temp$
22: **Go to line 3**
23: **end if**

Def 1.13 (Babai's Rounding and Nearest Plane Algorithm [3])**.** Babai's rounding is a method used to solve closest vector problem (CVP) for a lattice that has a good basis, where basis vectors are orthogonal or close to orthogonal. Let us have a lattice $\mathcal{L}$ with basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$, then given a vector $\mathbf{v} \in \mathbb{R}^n$, we can express this as a unique linear combination

$$\mathbf{w} = \sum_{i=1}^{n} \lambda_i \mathbf{b}_i \qquad \text{for } \lambda_i \in \mathbb{R}$$

where $\{\lambda_1, ..., \lambda_n\}$ can be found from $\mathbf{v}\mathbf{B}^{-1}$. Babai's rounding then says to find the closest vector in our lattice to $\mathbf{v}$, $\mathbf{v}'$, we do the following

$$\mathbf{v}' = \sum_{i=1}^{n} \lfloor \lambda_i \rceil \, \mathbf{b}_i.$$

Babai's second algorithm is the nearest plane algorithm which solves the same problem utilising a recursive search and hyperplanes. It is shown below:

Algorithm 2 Babai's nearest plane algorithm

Input: Reduced lattice basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$, target vector $\mathbf{v} \in \mathbb{R}^n$
$\mathbf{s} \leftarrow \mathbf{v}$
for $i \leftarrow n, 1$ **do**
    $\lambda = [\frac{\mathbf{s} \cdot \mathbf{b}_i^*}{\mathbf{b}_i^* \cdot \mathbf{b}_i^*}]$ ▷ Where $\mathbf{b}_i^*$ are the Gram-Schmidt vectors.
    $\mathbf{s} \leftarrow \mathbf{s} - \lambda \mathbf{b}_i$
end for
return $\mathbf{v}' = \mathbf{v} - \mathbf{s}$

Def 1.14 (number theoretic transform (NTT))**.** NTT is a method that allows cheap and efficient multiplications in R_q, a polynomial ring. Let $g = \sum_{i=0}^{n-1} g_i X^i$ be a polynomial in R_q, then let

$$NTT(g) = \hat{g} = \sum_{i=0}^{n-1} \hat{g}_i X^i, \qquad \text{where } \hat{g}_i = \sum_{j=0}^{n-1} \psi^j g_j \omega^{ij}$$

and gives

$$NTT(\hat{g}) = g = \sum_{i=0}^{n-1} g_i X^i, \qquad \text{where } g_i = n^{-1} \psi^{-i} \sum_{j=0}^{n-1} \hat{g}_j \omega^{-ij}$$

Note, that this is done so that both NTT and NTT^{-1} can be computed with very similar functions and the computation is almost identical, besides the difference in coefficient. Also, note that this allows us to easily and efficiently compute $f\dot{g}$, $f, g \in R_q$, using $f\dot{g} = NTT^{-}1(NTT(f) \circ NTT(g))$, where $\circ$ is pointwise multiplication. We also define the application of NTT or NTT^{-1} to a vector or matrix with elements in R_q as NTT or NTT^{-1} being applied to each individual entry. $\circ$ is defined as being applied to vectors or matrices as usual multiplication but where the individual products of entries are pointwise multiplications of coefficients.

Def 1.15 (Block Korkine-Zolotarev, Schnorr and Euchner, 1994 (BKZ) [21])**.** BKZ relies on utilising an shortest vector problem (SVP) oracle in order to KZ-reduce the basis of a lattice in a block-wise manner. If it can be ensured that each consecutive block of basis vectors of a lattice is KZ-reduced, then strong bounds on the first basis vector's length is more easily provable. A lattice $\mathcal{L}$ with basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$ and corresponding Gram-Schmidt orthognalised basis $\mathbf{B}^*$, is defined as being KZ-reduced if

$$|\mathbf{b}_i^*| = \lambda_1(\pi_i(\mathcal{L})) \qquad \text{for all } i$$

where $\pi_i(\mathbf{x}) = \sum_{j \geq i} \frac{\langle \mathbf{x}, \mathbf{b}_j^* \rangle}{|\mathbf{b}_j^*|^2} \mathbf{b}_j^*$ is a projection function to project $\mathbf{x}$ orthogonally onto the span of $\{\mathbf{b}_i^*, ..., \mathbf{b}_n^*\}$.

Algorithm 3 KZ-reduction

Input: Basis $\mathbf{B}$ of $\mathcal{L}$, $\mathcal{O}$ an SVP oracle
for $i \leftarrow 1, k$ **do**
    Call $\mathcal{O}$ to find a $\mathbf{b}_i^* \in \pi_i(\mathcal{L})$ with $|\mathbf{b}_i^*| = \lambda_1(\pi_i(\mathcal{L}))$
    Lift $\mathbf{b}_i^*$ into a $\mathbf{b}_i \in \mathbf{B}$ s.t. $\{\mathbf{b}_1, ...\mathbf{b}_i\}$ is size reduced. ▷
    To 'lift' a vector in $\pi_i(\mathcal{L})$ to a lattice vector is to add the necessary amount of $\mathbf{b}_{i-1}$ as in general, before lifting, $\mathbf{b}_i^*$ is not in our lattice. Lifting in this manner ensures our basis remains size-reduced.
    Replace our basis vectors $\{\mathbf{b}_{i+1}, ..., \mathbf{b}_k\}$ with lattice vectors $\{\mathbf{b}_{i+1}, ..., \mathbf{b}_k\}$ s.t. $\{\mathbf{b}_1, ..., \mathbf{b}_k\}$ is a basis for $\mathcal{L}$.
end for
return KZ-reduced basis $\mathbf{B}$

With this, we can construct a further algorithm utilising KZ-reduction as a subroutine.

Algorithm 4 BKZ algorithm

Input: Basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_d\}$ of $\mathcal{L}$, $\mathcal{O}$ an SVP oracle for k dimensions, blocksize k, hermite constant $\delta \in (\frac{1}{4}, 1)$
while the basis changes **do**
    for $i \leftarrow 1, d-k+1$ **do**
        KZ-reduce basis $\pi_i(\mathbf{b}_i, ..., \mathbf{b}_{i+k-1})$ using algorithm 3
        for $j \leftarrow 2, d$ **do** ▷ size reducing $\mathbf{B}$
            for $j \leftarrow i-1, 1$ **do**
                $\mathbf{b}_i \leftarrow \mathbf{b}_i - [\mu_{i,j}]\,\mathbf{b}_j$ ▷ $\mu_{i,j}$ is the Gram-Schmidt coefficient.
                for $k \leftarrow 1, j$ **do**
                    $\mu_{i,k} \leftarrow \mu_{i,k} - [\mu_{i,j}]\,\mu_{j,k}$
                end for
            end for
        end for
    end for
end while
return δ-LLL-reduced basis $\mathbf{B}$ where $|\mathbf{b}_i^*| = \lambda_1(\pi_i(\mathbf{b}_i, ..., \mathbf{b}_{i+k-1}))$ for all $i \in \{1, ..., d-k+1\}$.

2 Understanding the LWE problem

KYBER.PKE is a module LWE based encryption scheme; relying on the hardness of the of the LWE problem - believed to be hard for both classical and quantum computers, first introduced by O. Regev [19]. Below is an informal mod q set-up for the LWE problem for a prime q:

1. Let us chose an n dimensional vector $\mathbf{s} \in \mathbb{F}_q^n$. This is our secret.
2. Let us randomly and uniformly generate an $m \times n$ matrix $\mathbf{A}$ over $\mathbb{F}_q$ from elements in $\mathbb{F}_q$.
3. Let us generate an m dimensional vector, $\mathbf{e}$, s.t. $\mathbf{e}_i \sim \chi$ for all $i \in 1, ..., m$ independently for the distribution χ on $\mathbb{F}_q$ centred on 0 with a small variance.
4. Let $\mathbf{b} = \mathbf{As} + \mathbf{e}$ over $\mathbb{F}_q^m$

Now, given $(\mathbf{A}, \mathbf{b})$, find $\mathbf{s}$. Alternatively, the problem may be stated as given $(\mathbf{A}, \mathbf{b})$, determine if $(\mathbf{A}, \mathbf{b})$ was generated from our LWE set-up or uniformly at random. Formally defined, this is as follows:

Def 2.1 (LWE)**.** Let $n, m, q \in \mathbb{N}$. Let $\chi_{\mathbf{s}}$, $\chi_{\mathbf{e}}$ be distributions over $\mathbb{Z}/q\mathbb{Z}$. Further, let $LWE_{n,m,q,\chi_{\mathbf{s}},\chi_{\mathbf{e}}}$ be the probability distribution on $(\mathbb{Z}/q\mathbb{Z})^{m\times n} \times (\mathbb{Z}/q\mathbb{Z})^m$ from independently and uniformly sampling coordinates of $\mathbb{A} \in (\mathbb{Z}/q\mathbb{Z})^{m\times n}$ over $\mathbb{Z}/q\mathbb{Z}$, independently sampling coordinates of $\mathbf{s} \in (\mathbb{Z}/q\mathbb{Z})^n$, $\mathbf{e} \in (\mathbb{Z}/q\mathbb{Z})^m$ from $\chi_{\mathbf{e}}$ and $\chi_{\mathbf{e}}$ respectively. This outputs $(\mathbf{A}, \mathbf{As} + \mathbf{e})$.

- **Decision-LWE:** Distinguish $LWE_{n,m,q,\chi_{\mathbf{s}},\chi_{\mathbf{e}}}$ from the uniform distribution over $(\mathbb{Z}/q\mathbb{Z})^{m\times n} \times (\mathbb{Z}/q\mathbb{Z})^m$.
- **Search-LWE:** Given a sample from $LWE_{n,m,q,\chi_{\mathbf{s}},\chi_{\mathbf{e}}}$, recover $\mathbf{s}$.

This problem can be reduced to solving the lattice problems SVP, CVP or bounded distance decoding (BDD).

2.1 Lattices (revisited)

For the reduction of LWE to lattice problems, let us first consider the q-ary lattice[3]

$$\mathcal{L}_{Im(\mathbf{A})} = \{y \in \mathbb{Z}^m | y = \mathbf{Az} \mod q \text{ for some } \mathbf{z} \in \mathbb{Z}^n\}$$

from the image of our matrix $\mathbf{A}$, $Im(\mathbf{A}) = \mathbf{A}x | x \in \mathbb{F}_q^n$, which we can see has the volume

$$Vol(\mathcal{L}_{Im(\mathbf{A})}) = q^{m-n}.$$

This lattice is generated by the column vectors of our matrix $\mathbf{A} \mod q$. From here, we can find our $m \times m$ basis matrix, $\mathbf{B}_{Im(\mathbf{A})}$ through the reduction of the $(n+m) \times m$ matrix

$$\mathbf{A}_q^T = (\frac{\mathbf{A}^T}{q\mathbf{I}_m})$$

Therefore, we can see that given $(\mathbf{A}, \mathbf{b})$ from an LWE problem, we can use these to generate a lattice $\mathcal{L}_{Im(\mathbf{A})}$. Then by finding the closest point to $\mathbf{b} = \mathbf{As} + \mathbf{e}$, we can find $\mathbf{s}$ (the closest vector to $\mathbf{b}$ should be $\mathbf{As}$ given $\mathbf{e}$ is small enough) - solving the LWE problem if CVP can be solved.

[3] A q-ary lattice generated by $\mathbf{B} \in \mathbb{Z}_q^{d\times n}$ is defined as $\mathcal{L}_q(\mathbf{B}) = \{\mathbf{y} \in \mathbb{Z}^d | \mathbf{y} = \mathbf{Bz} \mod q, \mathbf{z} \in \mathbb{Z}^n\}$.

3 The Dual and Primal Attacks

Both the dual and primal attacks are approaches widely believed to the most promising in solving the hard lattice problems we have just discussed. Primal attacks rely on embedding techniques followed by lattice reductions (such as BKZ or LLL) in order to solve the posed problems, or alternatively, the use of Voronoi cells and sieving if preprocessing is possible. Dual attacks, meanwhile, utilise the dual lattice, taking short vectors and computing dot products with the target vector to gain evidence to see whether the target vector lies close to the lattice. This is then repeated with many dual vectors, a distinguisher is constructed and gradient ascent search algorithms used to solve our problems. In this paper, we will focus on the dual attack, due to the recent advancements relative to KYBER [15].

3.1 Primal Attack

The primal attack relies on the transformation of a search-LWE instance into SVP, via embedding and solving this by lattice reduction, enumeration, or sieving. Lattice reduction relies on the reduction of a lattice with basis matrix $\mathbf{B}$ to a lattice with basis vectors that are orthogonal or close to orthogonal, from which we can solve SVP much easier. Enumeration utilises an exhaustive search of integer combinations of basis vectors to find the shortest; note that a bound of some sort must be placed else our problem becomes intractable due to an infinite number of possibilities to try. Sieving aims to reduce the high run times that enumeration suffers from, however itself comes with the problem of needing exponential space - meaning that the dimensions of the lattice we are searching must be considered when deciding what method one should use. Sieving relies on first beginning with an estimate for the shortest vector, and running the algorithm many times in order to make this estimate more accurate.

3.1.1 Kannan's Embedding

Kannan's embedding [8] is a method that allows the reduction of BDD to unique-SVP, where given an LWE instance $(\mathbf{A}, \mathbf{b} = \mathbf{As} + \mathbf{e})$, we can embed the corresponding q-ary LWE lattice, $\mathcal{L}(\mathbf{A}) = \{\mathbf{y} \in \mathbb{Z}^m | \mathbf{y} = \mathbf{Az} \mod q, \forall \mathbf{z} \in \mathbb{Z}^n\}$, into $\mathcal{L}(\mathbf{B})$, a lattice with uSVP structure and higher dimension, where

$$\mathbf{B} = \begin{pmatrix} \bar{\mathbf{A}} & \mathbf{b} \\ 0 & t \end{pmatrix}$$

and $\bar{\mathbf{A}}$ is the basis of the $\mathcal{L}(\mathbf{A})$. Then, if $t \leq \frac{\lambda_1(\mathcal{L}(\mathbf{A}))}{2} = \text{dist}(\mathbf{c}, \mathcal{L}(\mathbf{A}))$, [4] then $\mathcal{L}(\mathbf{B})$ contains a unique shortest vector $\mathbf{w} = (\mathbf{e}^T || -t)$, allowing the easy recovery of the error and consequently the secret also. As such, our focus is now on solving SVP.

3.1.2 Lattice Reduction

Once we have embedded the lattice properly, the first method we can use to solve LWE is by lattice reduction. This will generally utilise LLL or BKZ, algorithms 1 and 4 respectively. These algorithms will return a whole basis of short vectors, and so we can take the shortest, generally $\mathbf{b}_1$, as our solution - however this will not always be the true shortest vector and so they solve approximate SVP. In some cases, if the approximation factor is low enough, this may be sufficient to break an encryption scheme. Alternatively, lattice reduction can be used in order to solve CVP.

[4] Where $\text{dist}(\mathbf{c}, \mathcal{L}(\mathbf{A}))$ is the distance from b to the lattice, $|\mathbf{b} - \mathbf{v}|$ where $\mathbf{v} \in \mathcal{L}(\mathbf{A})$ is the closest lattice point to $\mathbf{b}$

It can be seen that both SVP and BDD can be reduced to the more general problem of CVP. The reduction of BDD is simple due to the only difference being the added condition to the input vector. However, SVP to CVP is not necessarily as simple, as using an oracle for CVP with input $\mathbf{0}$ will return $\mathbf{0}$, because this is also a lattice vector. Instead, we can do the following. Let $\mathcal{L}$ be a lattice and $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_m\}$ be the basis matrix, and let $\mathbf{B}^{(i)} = \{\mathbf{b}_1, ..., \mathbf{b}_{i-1}, 2\mathbf{b}_i, \mathbf{b}_{i+1}, ..., \mathbf{b}_m\}$. Then, for each $1 \leq i \leq m$, call our oracle for CVP with inputs $\mathbf{B}^{(i)}$ and $\mathbf{b}_i$, returning $\mathbf{x}_i$. Then, our shortest vector of $\mathcal{L}$ is $\min\{|\mathbf{x}_i - \mathbf{b}_i| : 1 \leq i \leq n\}$. As such, one may use a lattice reduction algorithm followed by Babai's rounding method or nearest plane algorithm in order to find the closest vector, and thus shortest vector also.

3.1.3 Enumeration

Lattice enumeration is akin to a brute force attack, however, as the number of vectors in a lattice is infintite, we must place a bound on the vectors we will enumerate ($R > 0$, and $R \geq \lambda_1(\mathcal{L})$), which we then use to enumerate all vectors shorter than R. Generally, R is chosen to be the length of our first basis vector. Again, we must reduce and orthogonalise our basis $\mathbf{B}$ to get $\mathbf{B}^*$ and choose a vector $\mathbf{v} \in \mathcal{L}$ s.t. $\lambda_1(\mathcal{L}) \leq \mathbf{v} \leq R$, for a lattice $\mathcal{L}$ of dimension d. Note that all basis vectors may be written as a linear combination of Gram-Schmidt vectors:

$$\mathbf{b}_i = \mathbf{b}_i^* + \sum_{j=1}^{i-1} \mu_{i,j}\mathbf{b}_j^*$$

but also that since $\mathbf{v}$ is a lattice vector, it can be expressed as a linear combination of basis vectors:

$$\mathbf{v} = \sum_{i=1}^{d} v_i \mathbf{b}_i$$

Therefore, we can rewrite $\mathbf{v}$ as

$$\sum_{i=1}^{d} v_i(\mathbf{b}_i^* + \sum_{j=1}^{i-1} \mu_{i,j}\mathbf{b}_j^*) = \sum_{j=1}^{d}(v_j + \sum_{i=j+1}^{d} v_i\mu_{i,j})\mathbf{b}_j^*$$

which also simplifies the projection of $\mathbf{v}$:

$$\pi_k(\mathbf{v}) = \pi_k(\sum_{j=1}^{d}(v_j + \sum_{i=j+1}^{d} v_i\mu_{i,j})\mathbf{b}_j^*) = \sum_{j=k}^{d}(v_j + \sum_{i=j+1}^{d} v_i\mu_{i,j})\mathbf{b}_j^*$$

and given the Gram-Schmidt vectors are orthogonal,

$$\left|\sum_{j=k}^{d}(v_j + \sum_{i=j+1}^{d} v_i\mu_{i,j})\mathbf{b}_j^*\right|^2 = \sum_{j=k}^{d}(v_j + \sum_{i=j+1}^{d} v_i\mu_{i,j})^2|\mathbf{b}_j^*|^2$$

Note also, that the projection of a vector cannot be shorter than the vector itself, and so we can create the following inequality:

$$|\pi_k(\mathbf{v})|^2 = \sum_{j=k}^{d}(v_j + \sum_{i=j+1}^{d} v_i\mu_{i,j})^2|\mathbf{b}_j^*|^2 \leq R^2 \quad \forall k \in [1, ..., d]$$

We are now able to use this inequality for our enumeration. We begin by enumerating all vectors $\mathbf{w} \in \pi_d(\mathcal{L})$ of length at most R, and then for all potential candidates,

enumerate all vectors in $\pi_{d-1}(\mathcal{L})$ of length at most R that project to $\mathbf{v}$. This is then repeated all the way down in a recursive fashion, until all vectors in $\pi_1(\mathcal{L})$, $\mathcal{L}$, have been enumerated.

Therefore, if we try all possible linear combinations that satisfy our inequality, we can create a list of all lattice vectors shorter than R, and thus finding the shortest from a list is simple.

3.1.4 Sieving

Sieving aims to reduce the run-time of the very greedy approach of enumeration, however they also require exponential space - meaning that depending on the dimension of the lattice, they may not be the most efficient. First, we must take an estimate for the length of the shortest vector[5] in $\mathcal{L}$, call this $\tilde{\lambda}$. This can be increased if we find no vectors of at most this length, or decreased if our estimate was too generous.

The algorithm first must collect a list of small error vectors $\mathbf{e}_i$, and an associated lattice vector $\mathbf{v}_i$. Then, letting $\mathbf{r}_i = \mathbf{v}_i + \mathbf{e}_i$, rewrite this as a shorter vector $\mathbf{r}'_i = \mathbf{v}'_i + \mathbf{e}_i$ with $\mathbf{r}'_i$ that have already been found, and add $\mathbf{v}'_i$ to our list Q. This is then repeated such that we are adding vectors of norm no greater that $|\mathbf{B}|$, and at least a distance $\tilde{\lambda}$ apart pairwise. We do this until we find a $\mathbf{v}, \mathbf{w} \in Q$ that are at most $\tilde{\lambda}$ distance apart, or until we run out of space. Below, is the Micciancio-Voulgaris sieving algorithm.

Algorithm 5 Micciancio-Voulgaris sieving algorithm [17]

Input: A reduced basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_d\}$ of a lattice $\mathcal{L}$, and an estimate $\tilde{\lambda}$ for $\lambda_1(\mathcal{L})$.
$Q \leftarrow \emptyset$
$\xi \leftarrow 0.685$
$N \leftarrow \text{poly}(d) \cdot 2^{3.199d}$
for $i \leftarrow 1, N$ **do**
    $\mathbf{e}_i \in_R B_d(\mathbf{0}, \xi\tilde{\lambda})$
    $\mathbf{r}_i \leftarrow \mathbf{e}_i \mod B$
    while $\exists \mathbf{v}_j \in Q : |\mathbf{r}_i - \mathbf{v}_j| \leq (1 - \frac{1}{d})|\mathbf{v}_i|$ **do**
        $\mathbf{r}_i \leftarrow \mathbf{r}_i \mathbf{v}_j$
    end while
    $\mathbf{v}_i \leftarrow \mathbf{r}_i \mathbf{e}_i$
    if $\mathbf{v}_i \notin Q$ **then**
        if $\exists \mathbf{v}_j \in Q : |\mathbf{v}_i - \mathbf{v}_j| < \tilde{\lambda}$ **then**
            return $\mathbf{v}_i - \mathbf{v}_j$
        end if
        $Q \leftarrow Q \cup \{\mathbf{v_i}\}$
    end if
end for

Therefore, if we find two vectors $\mathbf{v}, \mathbf{w} \in \mathcal{L}$ which are at most $\tilde{\lambda}$ apart, then $\mathbf{u} = \mathbf{v} - \mathbf{w}$ is also in our lattice and has length at most $\tilde{\lambda}$ and $\mathbf{u}$ is returned by the algorithm.

3.1.5 Voronoi Cells

Def 3.1 ((Open) Voronoi cell)**.** Given a lattice $\mathcal{L}$, then the (open) Voronoi cell be the set

$$\mathcal{V}(\mathcal{L}) = \{\mathbf{x} \in \mathbb{R}^n : \forall \mathbf{v} \in \mathcal{L} \backslash \{\mathbf{0}\}, |\mathbf{x}| < |\mathbf{x} - \mathbf{v}|\}$$

[5]The true shortest vector $\lambda_1(\mathcal{L})$ is not generally known

This is the set of all points closer to the origin than any other lattice vector. The closed Voronoi cell, $\bar{\mathcal{V}}(\mathcal{L})$ is defined as the topological closure of the above set. Similarly, the Voronoi cell of a lattice point $\mathbf{w} \in \mathcal{L}$ is defined as

$$\mathcal{V}(\mathcal{L})_{\mathbf{w}} = \{\mathbf{x} \in \mathbb{R}^n : \forall \mathbf{v} \in \mathcal{L} \backslash \{\mathbf{0}\}, |\mathbf{x} - \mathbf{w}| < |\mathbf{x} - \mathbf{v}|\}$$

and is equal to $\mathbf{v} + \mathcal{V}(\mathcal{L})$. We also define the half-space

$$H_{\mathbf{v}} = \{\mathbf{x} : |\mathbf{x}| < |\mathbf{x} - \mathbf{v}|\}$$

and define the set of relevant vectors V as being the minimal set of lattice vectors such that $\mathcal{V}(\mathcal{L}) = \bigcap_{\mathbf{v} \in V} H_{\mathbf{v}}$.

When looking at utilising Voronoi cells to solve CVP or SVP, we will rely on the following equivalences:

1. $\mathbf{t}_s$ is the shortest vector of the coset $\mathcal{L} + \mathbf{t}$
2. $\mathbf{t}_s \in \mathcal{L} + \mathbf{t} \cap \mathcal{V}(\mathcal{L})$
3. $\text{CVP}(\mathbf{t}) = \mathbf{t} - \mathbf{t}_s$

for $\mathbf{t}, \mathbf{t}_s$ vectors in the span of $\mathcal{L}$. We will also use Voronoi's theorem:

Let $\mathcal{L}$ be a lattice and $\mathbf{v} \in \mathcal{L}$ a non-zero vector. Then $\mathbf{v} \in V$ if and only if the pair $\pm\mathbf{v}$ are unique shortest vectors of the coset $2\mathcal{L} + \mathbf{v}$.

Now, to solve CVP, we will detail an algorithm to first solve CVP for a special case where our target vector is an element of $2\bar{\mathcal{V}}(\mathcal{L})$, and then solving CVP by making calls to this algorithm. This first algorithm makes use of the above equivalences and finds a shortest vector $\mathbf{t}_s \in \bar{\mathcal{V}}(\mathcal{L}) \cap (\mathcal{L} + \mathbf{t})$ and outputs $\mathbf{t} - \mathbf{t}_s$, the closest lattice vector.

We first take our target vector $\mathbf{t}$ and relevant vectors V of our lattice $\mathcal{L}$ as inputs and begin to iteratively calculate a $\mathbf{t}_i \in \mathcal{L} + \mathbf{t}$, such that each successive $\mathbf{t}_i$ is shorter than the last, until there are no changes and so we have found the shortest. This is done by testing if $\mathbf{t}_i \in \bar{\mathcal{V}}(\mathcal{L})$, and if not, then there exists at least one $H_{\mathbf{v}_i}$ s.t. $\mathbf{t}_i \notin H_{\mathbf{v}_i}$, as $\bar{\mathcal{V}}(\mathcal{L}) = \bigcap_{\mathbf{v} \in V} H_{\mathbf{v}}$. Then, if we set $\mathbf{t}_{i+1} = \mathbf{t}_i - \mathbf{v}_i$, then $\mathbf{t}_{i+1} \in \mathcal{L} + \mathbf{t}_i$ and $|\mathbf{t}_{i+1}| < |\mathbf{t}_i|$. Then, once we have found a shortest $\mathbf{t}_i$, we can return the closest lattice vector as described previously. This gives us the below algorithm:

Algorithm 6 Voronoi cell CVP solver for $2\bar{\mathbf{V}}(\mathcal{L})$

Input: Target vector $\mathbf{t}$, and relevant vectors V
$i \leftarrow 0$
$\mathbf{t}_0 \leftarrow \mathbf{t}$
while $\mathbf{t}_i \notin \bar{\mathcal{V}}(\mathcal{L})$ **do**
    Find a $\mathbf{v} \in V$ s.t $\frac{\langle \mathbf{t}_i, \mathbf{v}_i \rangle}{|\mathbf{v}_i|^2}$ is maximal
    $\mathbf{t}_{i+1} \leftarrow \mathbf{t}_i - \mathbf{v}_i$
    $i \leftarrow i + 1$
end while
return $\mathbf{t} - \mathbf{t}_i$

We now detail an algorithm that can take this and utilise it even when the special condition is not met:

Algorithm 7 Voronoi cell CVP solver

Input: Target vector $\mathbf{t}$, basis $\mathbf{B}$, and relevant vectors V
$\mathbf{t} \leftarrow$ Project $\mathbf{t}$ onto the linear span of $\mathbf{B}$
$p \leftarrow \min\{p \in \mathbb{Z} | \mathbf{t} \in 2^p \bar{\mathcal{V}}(\mathcal{L})\}$
$\mathbf{t}_p \leftarrow \mathbf{t}$
for $i \leftarrow p, 1$ **do**
    $\mathbf{t}_{i-1} \leftarrow \mathbf{t}_i - \text{Alg.6}(\mathbf{t}_i, 2^{i-1}V)$
end for
return $\mathbf{t} - \mathbf{t}_0$

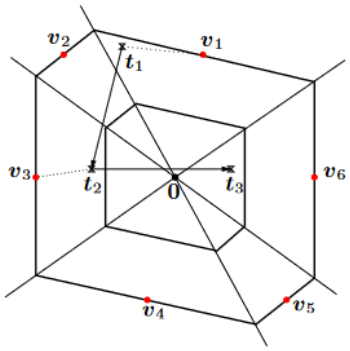


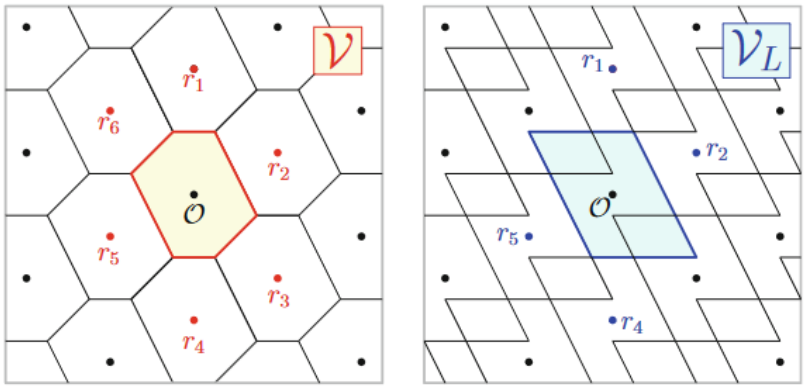
Figure 1: A graphical representation of the Voronoi cell CVP algorithm [7].

Now, we must understand how to compute such a list V efficiently. We first note that if our lattice is of only one dimension, then the V is simply $\pm\mathbf{b}_1$, and so using this information, we can build up our list of relevant vectors by utilising the fact that if we know the relevant vectors of $\mathcal{L}(\{\mathbf{b}_1, ..., \mathbf{b}_{n-1}\})$, we can easily search by exhaustion for V of $\mathcal{L}(\{\mathbf{b}_1, ..., \mathbf{b}_n\})$ as the extra relevant vectors will not be far, allowing us to bound our search.

Algorithm 8 Compute relevant vectors

Input: Basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$
$\mathbf{B} \leftarrow \text{LLL}(\mathbf{B})$
$V_1 = \{\mathbf{b}_1, -\mathbf{b}_1\}$
for $i \leftarrow 2, n$ **do**
  $V_i \leftarrow \{\}$
  $\mathbf{B}_i \leftarrow \{\mathbf{b}_1, ..., \mathbf{b}_i\}$
  for $\mathbf{c} \in \{\{0,1\}^i \backslash \mathbf{0}\}$ **do**
    $\mathbf{t} \leftarrow -\frac{\mathbf{B}_i \mathbf{c}}{2}$
    $h_t \leftarrow \frac{\langle \mathbf{t}, \mathbf{b}_i^* \rangle}{|\mathbf{b}_i^*|^2}$
    $\mathbf{best} \leftarrow \text{NearestPlaneCVP}(\mathbf{t}, \mathbf{B}_i)$
    for h s.t. $|h - h_t| \cdot |\mathbf{b}_i^*| < |\mathbf{best}|$ **do**
      $\mathbf{v} \leftarrow \text{Alg.7}(\mathbf{t} - h\mathbf{b}_i, V_{i-1}) + h\mathbf{b}_i$
      if $|\mathbf{best}| < |\mathbf{v}|$ **then**
        $\mathbf{best} \leftarrow \mathbf{v}$
      end if
    end for
    $V_i \leftarrow V_i \cup \{\pm 2(\mathbf{best} - \mathbf{t})\}$
  end for
  for $\mathbf{v}_j \in V_i$ **do**
    if $\exists \mathbf{v}_n \in V_k : \mathbf{v}_n \neq \mathbf{v}_j, \left|\frac{\mathbf{v}_j}{2} - \mathbf{v}_n\right| = \left|\frac{\mathbf{v}_j}{2}\right|$ **then**
      $V_k \leftarrow V_k \backslash \{\mathbf{v}_j -, -\mathbf{v}_j\}$
    end if
  end for
end for
return V_n

It can be seen from this, that computing exact Voronoi cells can be computationally expensive, and so we can use approximate Voronoi cells, cutting down much of the computation.



(a) A tiling of $\mathbb{R}^2$ with exact Voronoi cells $\mathcal{V}$ of a lattice $\mathcal{L}$ (red/black points), generated by the set $\mathcal{R} = \{r_1, \ldots, r_6\}$ of all *relevant vectors* of $\mathcal{L}$. Here $\text{vol}(\mathcal{V}) = \det(\mathcal{L})$.

(b) An overlapping tiling of $\mathbb{R}^2$ with approximate Voronoi cells $\mathcal{V}_L$ of the same lattice $\mathcal{L}$, generated by a subset of the relevant vectors, $L = \{r_1, r_2, r_4, r_5\} \subset \mathcal{R}$.

Figure 2: A graphical representation Voronoi cells against approximate voronoi cells [4].

3.2 Dual Attack

Def 3.2 (Dual lattice)**.** Given a lattice $\mathcal{L}$, let $\mathcal{L}^*$ be the corresponding dual lattice, where $\mathcal{L}^*$ contains all vectors $\mathbf{v} \in span(\mathcal{L})$ such that $\langle \mathbf{v}, \mathbf{w} \rangle \in \mathbb{Z}$ for all $\mathbf{w} \in \mathcal{L}$. For a full rank lattice with basis $\mathbf{B}$, its dual is given by $\mathbf{B}^{-T}$.

The dual attack relies on the following concept; given $(\mathbf{A}, \mathbf{b}) \in (\mathbb{Z}/q\mathbb{Z})^{m \times n} \times (\mathbb{Z}/q\mathbb{Z})^m$, find many small vectors v_i such that if our sample is an LWE sample, we have $\langle \mathbf{b},\ \mathbf{v}_i \rangle$ is also small, and is distributed according to a modular Gaussian distribution. If this is not the case, and our sample is random, then these values will be distributed randomly (and not necessarily small).

We can use this concept to develop an algorithm that allows us to enumerate the secret from a given LWE sample. First, we must find many vectors $(\mathbf{x}_j, \mathbf{y}_j)$ in such a way that $\mathbf{x}_j^T \mathbf{A} = (\mathbf{y}_{j,1}^T || \mathbf{y}_{j,2}^T)$ and $(\mathbf{x}_j, \mathbf{y}_{j,2})$ is short, where $\mathbf{y}_{j,1}$ and $\mathbf{y}_{j,2}$ is partitioned in the same way that we split $\mathbf{s}$ into two components, $\mathbf{s}_1$ and $\mathbf{s}_2$ such that $\mathbf{s}^T = (\mathbf{s}_1^T || \mathbf{s}_2^T)$, and similarly, $\mathbf{A} = (\mathbf{A}_1 || \mathbf{A}_2)$ such that $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e} = \mathbf{A}_1\mathbf{s}_1 + \mathbf{A}_2\mathbf{s}_2 + \mathbf{e}$.

We are then able to notice that we may create a new sample, $(\mathbf{A}_2, \mathbf{b} - \mathbf{A}_1\bar{\mathbf{s}}_1)$, where $\bar{\mathbf{s}}_1$ is our guess for the first part of the secret. If our guess is correct, then this new sample will act like a genuine LWE sample, and we can use the distinguishing methods we earlier described, and will describe this in more depth later.

Algorithm 9 Dual Attack in general

Input: $(n, m, q, \chi_{\mathbf{s}}, \chi_{\mathbf{e}})$ LWE parameters, $k_1, k_2 \in \mathbb{Z}$ s.t. $k_1 + k_2 = n$, and $(\mathbf{A}, \mathbf{b}) \in (\mathbb{Z}/q\mathbb{Z})^{m \times n} \times (\mathbb{Z}/q\mathbb{Z})^m$ LWE pair.
Find many vectors $(\mathbf{x}_j, \mathbf{y}_j)$ s.t. $\mathbf{x}_j^T \mathbf{A} = (\mathbf{y}_{j,1}^T | \mathbf{y}_{j,2}^T)$ and $(\mathbf{x}_j, \mathbf{y}_{j,2})$ are short. ▷ Finding these short vectors is usually done by reducing the lattice using BKZ and sieving as the SVP oracle. The output from the first block will normally suffice.
for all possible $\tilde{\mathbf{s}}_1$ for $\mathbf{s}_1$ **do**
    $X = \{(\mathbf{x}_j^T(\mathbf{b} - \mathbf{A}_1\tilde{\mathbf{s}}_1)) | \forall \mathbf{x}_j\}$
    if $X \sim N$ **then** ▷ As opposed to distributed uniformly.
        return $\tilde{\mathbf{s}}_1$
    end if
end for

4 Algorithm specification of Kyber.PKE

Note: We will present a simplified and abstracted version of the algorithm, for example, omitting 'compress', 'encode' and other such functions in order to more easily analyse the algorithm as opposed to it's implementation.

KYBER.PKE is defined over the ring $R \equiv \mathbb{Z}/(X^n + 1)$ and $R_q \equiv \mathbb{Z}_q[X]/(X^n + 1)$ where $n = 2^{n'-1}$ s.t. $X^n + 1$ is the $2^{n'\text{th}}$ cyclotomic polynomial [citecrystals]. For this chapter, our notation will be slightly different, with regular upper and lowercase letters denoting elements in R or R_q, and our notation for vectors instead denoting vectors with coefficients in R or R_q. In relation to our earlier explained LWE problem, R is $\mathbb{F}$ and R_q is $\mathbb{F}_q$.

4.1 Generate Keys

We begin, by generating our matrix $\mathbf{A}$ using the following algorithm:

Algorithm 10 Generate keys.

$N := 0$
$(\rho, \sigma) := G(d)$ ▷ Where G is a hash function s.t. $G : \mathcal{B}^* \to \mathcal{B}^{32} \times \mathcal{B}^{32}$
for $i \leftarrow 0, k-1$ **do**
    for $j \leftarrow 0, k-1$ **do**
        $\mathbf{A} := Parse(XOF(\rho, j, i))$ ▷ This essentially generates a random $k \times k$ matrix over R_q as ρ is pseudorandom from the hash function $G()$.
    end for
end for
for $i \leftarrow 0, k-1$ **do**
    $\mathbf{s} := CBD(PRF(\sigma, N))$ ▷ Where CBD is a function outputting a polynomial in R_q with the coefficients distributed central-binomially. PRF is a pseudorandom function, $PRF : \mathcal{B}^{32} \times \mathcal{B} \to \mathcal{B}^*$.
    $N := N + 1$
end for
for $i \leftarrow 0, k-1$ **do**
    $\mathbf{e} := CBD(PRF(\sigma, N))$
    $N := N + 1$
end for
$\mathbf{s} := NTT(\mathbf{s})$ ▷ Where NTT is a bijection mapping $f \in R_q$ to a polynomial with the coefficient vector.
$\mathbf{e} := NTT(\mathbf{e})$
$\mathbf{b} := \mathbf{As} + \mathbf{e}$
return $\mathbf{A}$, $\mathbf{s}$, $\mathbf{b}$, $\mathbf{e}$

From this, we have our public and private keys, $pk := (\mathbf{b} \mod q) \| \rho$ and $sk := \mathbf{s} \mod q$ (both encoded).

4.2 Encryption

Algorithm 11 Encryption

Input: pk, $m \in \mathcal{B}^{32}$
first we must extract $\mathbf{A}$ and $\mathbf{b}$ from pk.
$\rho := pk + 12 \cdot k \cdot \frac{n}{8}$ ▷ ρ was simply appended to the end of $\mathbf{b}$ so we can extract it simply. As we now have ρ we can re-construct $\mathbf{A}$ like we did in key generation.
for $i \leftarrow 0, k-1$ **do**
    for $j \leftarrow 0, k-1$ **do**
        $\mathbf{A}^T := Parse(XOF(\rho, i, j))$
    end for
end for
for $i \leftarrow 0, k-1$ **do**
    $\mathbf{r} := CBD(PRF(r, N))$ ▷ Where $r \in \mathcal{B}^{32}$ is a random coin.
    $N := N + 1$
end for
for $i \leftarrow 0, k-1$ **do**
    $\mathbf{e}_1 := CBD(PRF(r, N))$
    $N := N + 1$
end for
$e_2 := CBD(PRF(r, N))$
$\mathbf{r} := NTT((r))$
$\mathbf{u} := NTT^{-1}(\mathbf{A}^T \circ \mathbf{r}) + \mathbf{e}_1$
$v := NTT^{-1}(\mathbf{b}^T \circ \mathbf{r}) + e_2 + m$
return $(\mathbf{u} \| v)$

It is important that the ciphertext composes of two parts, only one of which is dependent on the message, so that the receiver has enough information in order to decrypt correctly.

4.3 Decryption

Algorithm 12 Decryption

Input: sk, c
$m := v - NTT^{-1}(\mathbf{s}^T \circ NTT(\mathbf{u}))$
return m

It may not be readily obvious why it is this decryption works:

$$\begin{aligned} v - NTT^{-1}(\mathbf{s}^T \circ NTT(\mathbf{u})) &= NTT^{-1}(\mathbf{b}^T \circ \mathbf{r} + e_2 + m) - NTT^{-1}(\mathbf{s}^T \circ NTT(\mathbf{u})) \\ &= NTT^{-1}(\mathbf{b}^T \circ \mathbf{r} + e_2 + m) - NTT^{-1}(\mathbf{s}^T \circ \mathbf{A}^T \circ \mathbf{r} + \mathbf{e}_1) \\ &= NTT^{-1}(\mathbf{b}^T \circ \mathbf{r} + e_2 + m - \mathbf{s}^T \circ \mathbf{A}^T \circ \mathbf{r} + \mathbf{e}_1) \\ &= NTT^{-1}((\mathbf{A} \circ \mathbf{s})^T \circ \mathbf{r} - (\mathbf{s}^T \circ \mathbf{A}^T) \circ \mathbf{r} + \mathbf{e}^T + \mathbf{e}_1 + e_2 + m) \\ &= NTT^{-1}(m + \mathbf{e}^T + \mathbf{e}_1 + e_2) \end{aligned}$$

Therefore, we decrypt by rounding to the nearest $\lfloor c \cdot \frac{q}{32} \mod q \rfloor$ for some $c \in \mathbb{N}$.

5 Applying our security notions to LWE and Kyber.PKE

It can be seen that the security of KYBER.PKE can be reduced to the hard problem of LWE; in other words, given $LWE_{n,m,q,\chi_{\mathbf{s}},\chi_{\mathbf{e}}}$ is hard then KYBER.PKE is LWE-CPA secure. In both our security notions, we rely on the statement

$$\mathbb{P}(\mathcal{A} \text{ succeeds}) = 1/2 + \varepsilon \qquad \text{where } \varepsilon \text{ is negligible.}$$

and so we can define what $\mathbb{P}(\mathcal{A}$ succeeds) is in order to begin to analyse the security of KYBER.PKE, and so we have $\mathbb{P}(\mathcal{A}$ succeeds) =

$$\mathbb{P}(a = i|\begin{smallmatrix}(pk,sk)\leftarrow\text{KYBER.PKE}.KeyGen(),\\(m_0,m_1)\leftarrow\mathcal{A}(pk),\\c\leftarrow\text{KYBER.PKE}.Enc(pk,m_i),\\a\leftarrow\mathcal{A}(s,c)\end{smallmatrix}) \qquad \text{for } i \in \{0,1\}.$$

Therefore, the advantage of the adversary (in our earlier definition written as ε), $\mathbf{Adv}^{\text{CPA}}_{\text{KYBER.PKE}}(\mathcal{A}) =$

$$|\mathbb{P}(a = i|\begin{smallmatrix}(pk,sk)\leftarrow\text{KYBER.PKE}.KeyGen(),\\(m_0,m_1)\leftarrow\mathcal{A}(pk),\\c\leftarrow\text{KYBER.PKE}.Enc(pk,m_i),\\a\leftarrow\mathcal{A}(s,c)\end{smallmatrix}) - \frac{1}{2}|$$

In other words, if $\mathbf{Adv}^{\text{CPA}}_{\text{KYBER.PKE}} \approx 0$ (or the adversary has no advantage), then KYBER.PKE is CPA secure.

We can also do the same thing for an LWE problem, and define the advantage of our adversary over this problem. The LWE challenge that our adversary must solve is slightly different to our challenge that we gave them for KYBER.PKE, and instead of distinguishing which message created the challenge ciphertext, they must determine if the challenge was generated via the LWE scheme, or if it was generated randomly. Thus, our $\mathbf{Adv}^{\text{LWE}}(\mathcal{A}) =$

$$|\mathbb{P}(b' = 1|\begin{smallmatrix}\mathbf{A}\leftarrow R_q^{m\times k},\\(\mathbf{s},\mathbf{e})\leftarrow\mathcal{B}^k\times\mathcal{B}^m,\\\mathbf{b}=\mathbf{As}+\mathbf{e},\\b'\leftarrow\mathcal{A}(\mathbf{A},\mathbf{b})\end{smallmatrix}) - \mathbb{P}(b' = 1|\begin{smallmatrix}\mathbf{A}\leftarrow R_q^{m\times k},\\\mathbf{b}\leftarrow R_q^m,\\b'\leftarrow\mathcal{A}(\mathbf{A},\mathbf{b})\end{smallmatrix})|$$

It can be seen that $\mathbf{Adv}^{\text{CPA}}_{\text{KYBER.PKE}}(\mathcal{A})$ can be re-written as:

$$|\mathbb{P}(b' = 1|\begin{smallmatrix}\mathbf{A}\leftarrow R_q^{m\times k},\\(\mathbf{s},\mathbf{e})\leftarrow\{0,1\}^k\times\{0,1\}^m,\\\mathbf{b}=\mathbf{As}+\mathbf{e}+m_i,\\a\leftarrow\mathcal{A}(\mathbf{A},\mathbf{b})\end{smallmatrix}) - \frac{1}{2}|$$

by using the definitions of KYBER.PKE's algorithms. Thus, we can show that the IND-CPA security of KYBER.PKE is reducible to that of the LWE scheme, $\mathbf{Adv}^{\text{LWE}}(\mathcal{A}) \approx$

$$|\mathbb{P}(b' = 1|\begin{smallmatrix}\mathbf{A}\leftarrow R_q^{m\times k},\\(\mathbf{s},\mathbf{e})\leftarrow\{0,1\}^k\times\{0,1\}^m,\\\mathbf{b}=\mathbf{As}+\mathbf{e}+m_i,\\a\leftarrow\mathcal{A}(\mathbf{A},\mathbf{b})\end{smallmatrix}) - \frac{1}{2}|$$

as $\mathbb{P}(b' = 1|\begin{smallmatrix}\mathbf{A}\leftarrow R_q^{m\times k},\\\mathbf{b}\leftarrow R_q^m,\\b'\leftarrow\mathcal{A}(\mathbf{A},\mathbf{b})\end{smallmatrix})|$ can simply be reduced to the expected value $\frac{1}{2}$.

$$\Rightarrow \mathbf{Adv}^{\text{CPA}}_{\text{KYBER.PKE}}(\mathcal{A}) \leq_T \mathbf{Adv}^{\text{LWE}}(\mathcal{A})$$

6 Tailoring the Dual Attack to Kyber

MATZOV published a paper in which they detail improvements to the dual attack in order to further increase its efficacy at enumerating secrets [15], something that proves very interesting for those interested in the security levels of KYBER. This comes largely through the use of FFT during the distinguishing process, allowing us to check all values at once instead of one at a time, as well as improved methods to sample short vectors. The formal definition of the improved algorithm is as follows:

Algorithm 13 MATZOV Improved Dual Attack [15]

Input: $(n, m, q, \chi_{\mathbf{s}}, \chi_{\mathbf{e}})$ LWE parameters, $\beta_1, \beta_2 \leq d$ where $\beta_1, \beta_2 \in \mathbb{Z}$, $k_{enum}, k_{fft}, k_{last}$ s.t. $k_{enum} + k_{fft} + k_{last} = n$, $p \in \mathbb{Z}$ s.t. $p \leq q$, $D \in \mathbb{Z}$, $C \in \mathbb{R}$, and $(\mathbf{A}, \mathbf{b}) \in (\mathbb{Z}/q\mathbb{Z})^{m \times n} \times (\mathbb{Z}/q\mathbb{Z})^m$ LWE pair.

$\mathbf{A}_{last} := (\mathbf{a}_{n-k_{last}} ... \mathbf{a}_n)$ ▷ The last k_{last} columns of $\mathbf{A}$.

$\alpha = \frac{\sigma_{\mathbf{e}}}{\sigma_{\mathbf{s}}}$

$\mathbf{B} = \begin{pmatrix} \alpha \mathbf{I}_m & 0 \\ \mathbf{A}_{last}^T & q\mathbf{I}_{k_{last}} \end{pmatrix}$

$L = \text{shortVectors}(\mathbf{B}, \beta_1, \beta_2, D)$ ▷ Where L is a list of D short vectors.

for all possible $\bar{\mathbf{s}}_{enum}$ **do** ▷ taken in decreasing order of probability according to $\chi_{\mathbf{s}}$.

  Let T be a table of dimension $p \times ... \times p$ (k_{fft} times).

  for all short vectors $(\alpha \mathbf{x}_j, \mathbf{y}_{last}) \in L$ **do**

    $\mathbf{y}_{j,fft} = \mathbf{x}_j^T \mathbf{A}_{fft}$

    $\mathbf{y}_{j,enum} = \mathbf{x}_j^T \mathbf{A}_{enum}$

    Add $e^{(\mathbf{x}_j^T \mathbf{b} - \mathbf{y}_{j,enum}^T \bar{\mathbf{s}}_{enum}) \frac{2\pi i}{q}}$ to the $\left[\frac{p}{q}\mathbf{y}_{j,fft}\right]$th cell of T.

  end for

  FFT(T)

  if $Re(\frac{1}{\psi(\bar{\mathbf{s}}_{fft})} T[\mathbf{s}_{fft}]) > C$ **then**

    return $\bar{\mathbf{s}}_{enum}$ ▷ Where $\psi(\bar{\mathbf{s}}_{fft})$ is a constant for centering our distribution.

  end if

end for

Algorithm 14 Short Vectors [15]

Input: Basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_d\}$ for lattice $\mathcal{L}$, $D \in \mathbb{N}$, and $\beta_1, \beta_2 \in \mathbb{Z}$ s.t. $\beta_1, \beta_2 \leq d$

for $i \leftarrow 1, \left\lceil \frac{D}{N_{sieve}(\beta_2)} \right\rceil$ **do** ▷ Where $N_{sieve}(d)$ is the number of vectors outputted by a sieving algorithm on a lattice of dimension d, and we are rounding with ties to rounding up.

  Randomise the basis $\mathbf{B}$

  Run BKZ on our basis using blocksize β_1 on $\mathbf{B}$ to obtain a reduced basis $\mathbf{B}'$

  Run a sieving algorithm of dimension β_2 on the sublattice $\{\mathbf{b}'_1, ..., \mathbf{b}'_{\beta_2}\}$ and add the outputted vectors to a list H.

end for

return H

The lengths of these short vectors are at most:

$$\det(\mathcal{L})^{\frac{1}{d}} \cdot N_{sieve}(\beta_2)^{\frac{1}{\beta_2}} \cdot \sqrt{\frac{\beta_2}{2\pi e} \cdot (\pi\beta_2)^{\frac{1}{\beta_2}}} \cdot \delta(\beta_1)^{\frac{d-\beta_2}{2}}$$

Where $\delta(\beta_1) := \frac{|\mathbf{b}_i^*|}{|\mathbf{b}_{i+1}^*|}$ for the first $k < d - 2\beta_1$ vectors of the Gram-Schmidt orthognalised basis.

The inputs to this algorithm are:

- D: The number of short vectors to be found.
- C: The boundary by which guesses of the secret are judged.
- β_1, β_2: Parameters for $shortVectors$.
- k_{enum}: The number of secret coordinates we directly search.
- k_{fft}: The number of secret coordinates enumerated using FFT.
- k_{last}: Remaining secrets.
- p: Modulus for FFT.

We also have $\mathbf{A}, \mathbf{b}$ from an LWE sample, where $\mathbf{s}$ and $mathbfe$ are sampled with known distributions $\chi_\mathbf{s}$ and $\chi_\mathbf{e}$ that have small variances $\sigma_\mathbf{s}^2$ and $\sigma_\mathbf{e}^2$ respectively. We then split $\mathbf{s}$ as dictated by our k values, and also $\mathbf{A}$, thus giving us

$$\mathbf{As} = \mathbf{A}_{enum}\mathbf{s}_{enum} + \mathbf{A}_{fft}\mathbf{s}_{fft} + \mathbf{A}_{last}\mathbf{s}_{last}$$

.

We can then find

$$B = \begin{pmatrix} \alpha\mathbf{I}_m & 0 \\ \mathbf{A}_{last}^T & q\mathbf{I}_{k_{last}} \end{pmatrix}$$

where $\alpha = \frac{\sigma_\mathbf{e}}{\sigma_\mathbf{s}}$, and is only in place to account for the cases where $\chi_\mathbf{s} \neq \chi_\mathbf{e}$.

We can now use $shortVectors$ to find D short vectors of $\mathbf{B}$, which can all be separated into the following form:

$$\mathbf{v} \equiv_q \begin{pmatrix} \alpha\mathbf{x} \\ \mathbf{A}_{last}^T\mathbf{x} \end{pmatrix}$$

and for each of these vectors

$$\mathbf{x}^T\mathbf{b} = \mathbf{x}^T\mathbf{As} + \mathbf{x}^T\mathbf{e}.$$

Setting $\mathbf{y}^T = \mathbf{x}^T\mathbf{A}$, we can rewrite the above to give

$$\mathbf{x}^T\mathbf{b} = \mathbf{y}^T\mathbf{s} + \mathbf{x}^T\mathbf{e},$$

which when we split up $\mathbf{y}$ and $\mathbf{s}$ into the three constituent parts as described earlier, we have

$$\mathbf{x}^T\mathbf{b} = \mathbf{y}_{enum}^T\mathbf{s}_{enum} + \mathbf{y}_{fft}^T\mathbf{s}_{fft} + \mathbf{y}_{last}^T\mathbf{s}_{last} + \mathbf{x}^T\mathbf{e}$$
$$\mathbf{y}_{enum}^T\mathbf{s}_{enum} + \mathbf{y}_{fft}^T\mathbf{s}_{fft} - \mathbf{x}^T\mathbf{b} = -\mathbf{y}_{last}^T\mathbf{s}_{last} - \mathbf{x}^T\mathbf{e}$$

Now, we apply the change of modulus by multiplying by $\frac{p}{q}$ and rounding the y_{fft} component of our equation.

$$\frac{p}{q}\mathbf{y}_{enum}^T\mathbf{s}_{enum} + \frac{p}{q}\mathbf{y}_{fft}^T\mathbf{s}_{fft} - \frac{p}{q}\mathbf{x}^T\mathbf{b} = -\frac{p}{q}\mathbf{y}_{last}^T\mathbf{s}_{last} - \frac{p}{q}\mathbf{x}^T\mathbf{e}$$
$$\frac{p}{q}\mathbf{y}_{enum}^T\mathbf{s}_{enum} + (\left[\frac{p}{q}\mathbf{y}_{fft}^T\right] - \left[\frac{p}{q}\mathbf{y}_{fft}^T\right] + \frac{p}{q}\mathbf{y}_{fft}^T)\mathbf{s}_{fft} - \frac{p}{q}\mathbf{x}^T\mathbf{b} = -\frac{p}{q}\mathbf{y}_{last}^T\mathbf{s}_{last} - \frac{p}{q}\mathbf{x}^T\mathbf{e}$$

$$\frac{p}{q}\mathbf{y}_{enum}^T\mathbf{s}_{enum} + \left[\frac{p}{q}\mathbf{y}_{fft}^T\right]\mathbf{s}_{fft} - \frac{p}{q}\mathbf{x}^T\mathbf{b} = -\frac{p}{q}\mathbf{y}_{last}^T\mathbf{s}_{last} - \frac{p}{q}\mathbf{x}^T\mathbf{e} - (\frac{p}{q}\mathbf{y}_{fft}^T - \left[\frac{p}{q}\mathbf{y}_{fft}^T\right])\mathbf{s}_{fft}$$

The right hand side of this expression is approximately distributed with modular Gaussian distribution, and so if we have the correct guesses, $(\bar{\mathbf{s}}_{enum}, \bar{\mathbf{s}}_{fft}) = (\mathbf{s}_{enum}, \mathbf{s}_{fft})$, then

$$\frac{p}{q}\mathbf{y}_{enum}^T\bar{\mathbf{s}}_{enum} + \left[\frac{p}{q}\mathbf{y}_{fft}^T\right]\bar{\mathbf{s}}_{fft} - \frac{p}{q}\mathbf{x}^T\mathbf{b}$$

will also be distributed with a modular Gaussian distribution, whilst if not, it will be distributed uniformly. Finally, we must distinguish between those with the correct distribution, and so let us consider the below expression.

$$\text{FFT}((\bar{\mathbf{s}}_{enum}, \bar{\mathbf{s}}_{fft})) = Re(\frac{1}{\psi(\bar{\mathbf{s}}_{fft})}\sum_j e^{(\frac{p}{q}\mathbf{y}_{j,enum}^T\bar{\mathbf{s}}_{enum} + \left[\frac{p}{q}\mathbf{y}_{j,fft}^T\right]\bar{\mathbf{s}}_{fft} - \frac{p}{q}\mathbf{x}_j^T\mathbf{b})\frac{2\pi i}{p}})$$

where $\psi(\bar{\mathbf{s}}_{fft}) = e^{\frac{2\pi i c}{p}\sum_t \mathbf{s}_t}$, then if this value is approximately zero, our guess $\bar{\mathbf{s}}_{enum}$ is incorrect, and if above the cutoff C, then $\bar{\mathbf{s}}_{enum} = \mathbf{s}_{enum}$ (with overwhelming probability)[6].

Let us write $\bar{\mathbf{s}}_{enum}$ as $\mathbf{s}_{enum} + \Delta\mathbf{s}_{enum}$, and $\bar{\mathbf{s}}_{ftt}$ as $\mathbf{s}_{ftt} + \Delta\mathbf{s}_{ftt}$ to give

$$\text{FFT}((\bar{\mathbf{s}}_{enum}, \bar{\mathbf{s}}_{fft})) = \sum_j Re(\frac{1}{\psi(\bar{\mathbf{s}}_{fft})}e^{(\frac{p}{q}\mathbf{y}_{j,enum}^T(\mathbf{s}_{enum}+\Delta\mathbf{s}_{enum}) + \left[\frac{p}{q}\mathbf{y}_{j,fft}^T\right](\mathbf{s}_{ftt}+\Delta\mathbf{s}_{ftt}) - \frac{p}{q}\mathbf{x}^T\mathbf{b})\frac{2\pi i}{p}})$$

Let us first assume that our guess is incorrect, in other words$\Delta\mathbf{s}_{enum}$ and $\Delta\mathbf{s}_{ftt}$ are non-zero.

$$\text{FFT}((\bar{\mathbf{s}}_{enum}, \bar{\mathbf{s}}_{fft})) = \sum_j Re(\frac{1}{\psi(\bar{\mathbf{s}}_{fft})}e^{(\frac{p}{q}\mathbf{y}_{j,enum}^T(\mathbf{s}_{enum}+\Delta\mathbf{s}_{enum}) + \left[\frac{p}{q}\mathbf{y}_{j,fft}^T\right](\mathbf{s}_{ftt}+\Delta\mathbf{s}_{ftt}) - \frac{p}{q}\mathbf{x}^T\mathbf{b})\frac{2\pi i}{p}})$$

$$= \sum_j Re(\frac{1}{\psi(\bar{\mathbf{s}}_{fft})}e^{\frac{p}{q}\mathbf{y}_{j,enum}^T\mathbf{s}_{enum} + \left[\frac{p}{q}\mathbf{y}_{j,fft}^T\right]\mathbf{s}_{fft} - \frac{p}{q}\mathbf{x}^T\mathbf{b} + \frac{p}{q}\mathbf{y}_{j,enum}^T\Delta\mathbf{s}_{enum} + \left[\frac{p}{q}\mathbf{y}_{j,fft}^T\right]\Delta\mathbf{s}_{fft}}$$

$$= \sum_j Re(\frac{1}{\psi(\bar{\mathbf{s}}_{fft})}e^{-\mathbf{y}_{j,enum}^T\Delta\mathbf{s}_{enum}\frac{2\pi i}{q}}e^{(\frac{p}{q}\mathbf{y}_{last}^T\mathbf{s}_{last} + \frac{p}{q}\mathbf{x}^T\mathbf{e} + (\frac{p}{q}\mathbf{y}_{j,fft}^T - \left[\frac{p}{q}\mathbf{y}_{fft}^T\right])\mathbf{s}_{fft} - \left[\frac{p}{q}\mathbf{y}_{j,fft}^T\right]\Delta\mathbf{s}_{fft})\frac{2\pi i}{q}})$$

Then, as the vectors $\mathbf{y}_{j,last}, \mathbf{y}_{j,enum}, \mathbf{y}_{j,fft}, \mathbf{x}_j$ are independent, with $\mathbf{y}_{j,enum}, \mathbf{y}_{j,fft}$ distributed uniformly, and $/mathbf{y}_{j,last}, \mathbf{x}_j$ normally distributed around 0, we can rewrite the second exponential and beginning reciprocal in terms of a new distribution:

$$\sum_j Re(e^{-\mathbf{y}_{j,enum}^T\Delta\mathbf{s}_{enum}\frac{2\pi i}{q}}e^{2\pi i W_j}$$

Where

$$\mathbb{E}(\sum_j Re(e^{-\mathbf{y}_{j,enum}^T\Delta\mathbf{s}_{enum}\frac{2\pi i}{q}}e^{2\pi i W_j}) = 0$$

Now, assume that our guess is correct.

[6]The higher our value for C, the more likely our guess is correct.

$$\mathrm{FFT}((\bar{\mathbf{s}}_{enum}, \bar{\mathbf{s}}_{fft})) = \sum_j Re(\frac{1}{\psi(\bar{\mathbf{s}}_{fft})} e^{(\frac{p}{q}\mathbf{y}_{j,enum}^T(\mathbf{s}_{enum}+\Delta\mathbf{s}_{enum})+\left[\frac{p}{q}\mathbf{y}_{j,fft}^T\right](\mathbf{s}_{ftt}+\Delta\mathbf{s}_{ftt})-\frac{p}{q}\mathbf{x}^T\mathbf{b})\frac{2\pi i}{p}})$$

$$= \mathrm{FFT}((\mathbf{s}_{enum}, \mathbf{s}_{fft})) = \sum_j Re(\frac{1}{\psi(\bar{\mathbf{s}}_{fft})} e^{(\frac{p}{q}\mathbf{y}_{j,last}^T\mathbf{s}_{last}+\frac{p}{q}\mathbf{x}_j^T\mathbf{e}+(\frac{p}{q}\mathbf{y}_{j,fft}^T-\left[\frac{p}{q}\mathbf{y}_{j,fft}^T\right])\mathbf{s}_{fft})\frac{2\pi i}{p}}$$

Where again, we can split this into parts of similar distribution to get

$$\sum_j Re(\varepsilon_{j,eq} \cdot \varepsilon_{j,round})$$

Which, when D is sufficiently large, will be approximately normally distributed; the proofs for the mean and variance are too long and not particularly relevant to our goals in this paper and so are omitted. Using this, we have therefore found the first k_{enum} digits of the key, and we can repeat the algorithm lengthening the portion of the key we are guessing whilst using the already enumerated part.

6.1 Impact on Kyber

Below is verbatim the results presented by MATZOV [15] for interest of the reader.

Table 3: Evaluation of the security level using [AGPS20] sieve costs and the asymptotic model [Duc18].

Candidate	Required Security Level [Nat16]	Estimated Security Level [DKL+21] [ABD+21] [BMD+20]	This Work	Parameters					
				m	p	β_1	β_2	k_{enum}	k_{fft}
Kyber512	143	151.5	**143.8**	474	5	377	380	19	34
Kyber768	207	215.1	**200.5**	659	4	576	570	30	59
Kyber1024	272	287.3	**266.0**	836	4	809	790	40	82
Dilithium2	146	159	**153.4**	1002	6	409	406	21	31
Dilithium3	207	217	**210.5**	1280	12	609	601	21	34
Dilithium5	272	285	**273.3**	1679	6	832	814	36	64
LightSaber	143	Unspecified	**144.8**	512	7	380	383	16	29
Saber	207	Unspecified	**210.4**	712	6	612	603	25	48
FireSaber	272	Unspecified	**273.4**	885	5	835	816	36	72

Table 4: Evaluation of the security level using [AGPS20] sieve costs and the G6K model [ADH+19].

Candidate	Required Security Level [Nat16]	Estimated Security Level [DKL+21] [ABD+21] [BMD+20]	This Work	Parameters					
				m	p	β_1	β_2	k_{enum}	k_{fft}
Kyber512	143	151.5	**143.1**	474	6	380	378	19	30
Kyber768	207	215.1	**199.5**	655	4	580	566	31	58
Kyber1024	272	287.3	**264.4**	839	4	816	786	39	82
Dilithium2	146	159	**152.2**	1002	6	409	406	21	31
Dilithium3	207	217	**208.9**	1280	12	613	594	20	34
Dilithium5	272	285	**270.9**	1707	5	837	805	36	71
LightSaber	143	Unspecified	**144.1**	512	7	383	381	17	27
Saber	207	Unspecified	**209.3**	708	6	617	599	24	49
FireSaber	272	Unspecified	**271.7**	897	5	843	811	35	72

Table 5: Evaluation of the security level using the sieve costs from Section 6 and the asymptotic model [Duc18].

Candidate	Required Security Level [Nat16]	Estimated Security Level [DKL+21] [ABD+21] [BMD+20]	This Work	Parameters					
				m	p	β_1	β_2	k_{enum}	k_{fft}
Kyber512	143	151.5	**138.2**	485	5	379	383	17	33
Kyber768	207	215.1	**194.5**	652	5	580	574	27	51
Kyber1024	272	287.3	**259.3**	834	4	813	794	36	82
Dilithium2	146	159	**147.3**	989	6	409	411	17	33
Dilithium3	207	217	**203.7**	1279	11	611	603	18	35
Dilithium5	272	285	**266.2**	1663	6	834	816	32	65
LightSaber	143	Unspecified	**139.1**	512	7	382	386	15	27
Saber	207	Unspecified	**203.9**	714	6	614	605	23	48
FireSaber	272	Unspecified	**266.5**	878	5	840	818	32	73

Table 6: Evaluation of the security level using the sieve costs from Section 6 and the G6K model [ADH+19].

Candidate	Required Security Level [Nat16]	Estimated Security Level [DKL+21] [ABD+21] [BMD+20]	This Work	Parameters					
				m	p	β_1	β_2	k_{enum}	k_{fft}
Kyber512	143	151.5	**137.5**	492	5	381	381	17	33
Kyber768	207	215.1	**193.5**	651	5	585	571	27	50
Kyber1024	272	287.3	**257.8**	844	4	820	789	37	80
Dilithium2	146	159	**146.3**	1008	7	410	409	17	30
Dilithium3	207	217	**202.0**	1274	12	615	594	18	34
Dilithium5	272	285	**263.6**	1680	6	840	804	32	64
LightSaber	143	Unspecified	**138.4**	512	7	385	383	15	27
Saber	207	Unspecified	**202.7**	714	6	619	602	23	47
FireSaber	272	Unspecified	**264.9**	905	5	846	814	32	72

Table 7: Evaluation of the security level across models.

Candidate	Required Security Level [Nat16]	Estimated Security Level [DKL+21] [ABD+21] [BMD+20]	This Work			
			G6K [ADH+19]		Asymptotic [Duc18]	
			[AGPS20]	Section 6	[AGPS20]	Section 6
Kyber512	143	151.5	**143.1**	**137.5**	**143.8**	**138.2**
Kyber768	207	215.1	**199.5**	**193.5**	**200.5**	**194.5**
Kyber1024	272	287.3	**264.4**	**257.8**	**266.0**	**259.3**
Dilithium2	146	159	**152.2**	**146.3**	**153.4**	**147.3**
Dilithium3	207	217	**208.9**	**202.0**	**210.5**	**203.7**
Dilithium5	272	285	**270.9**	**263.6**	**273.3**	**266.2**
LightSaber	143	Unspecified	**144.1**	**138.4**	**144.8**	**139.1**
Saber	207	Unspecified	**209.3**	**202.7**	**210.4**	**203.9**
FireSaber	272	Unspecified	**271.7**	**264.9**	**273.5**	**266.5**

7 Open Questions

"To simplify the analysis, we do not assume the recovery of $\mathbf{s}_{fft}$*. Although the above sum is expected to have large real value when* $\mathbf{s}_{fft}$ *is guessed correctly, it may also be large for certain wrong guesses of* $\mathbf{s}_{fft}$*, provided* $\mathbf{s}_{enum}$ *is guessed correctly. This does not concern us since we only wish to recover* $\mathbf{s}_{enum}$*."* [15]

This is possibly the most interesting paragraph of the whole MATZOV paper, and opens up the question; is it possible to distinguish between a wrong $\mathbf{s}_{fft}$ and a correct $\mathbf{s}_{fft}$, given that $\mathbf{s}_{enum}$ is correct? If so, it is likely that this would greatly increase the efficacy of the attack due to the fact we must use p^k guesses.

We can also see that our fully constructed and tailored dual attack still relies on earlier mentioned techniques, specifically the use of sieving and the BKZ algorithm when generating our list of small vectors. Due to the fact that our second use of sieving must return all found short lattice vectors, we are unable to utilise 'dimensions-for-free' [6], however, is there a method by which we can manipulate the lattice used for this in order to allow us to utilise such technique? Alternatively, any improvements of sieving algorithms or BKZ is likely to increase the potency of this dual attack. Also, is it possible to implement pre-processing somewhere in this attack? Perhaps the same techniques we use when utilising Voronoi cells can be implemented elsewhere, and on the topic of Voronoi cells, can we reduce the space required by somehow compressing our requirements of Voronoi cells - approximate Voronoi cells, and generally how to choose the parameters, such that our cells are accurate enough yet still much easier to compute.

More generally, however, the ideal choice for a lot of variables - such as how many bits we should enumerate at one time, and β_1, β_2 - is yet to be fully solved. For example, it can be seen that by reducing the number of bits we are enumerating, we decrease our search space for that run but in doing so we reduce the potential advantage we are aiming to achieve over simply guessing by brute force.

References

[1] Martin R. Albrecht. "On Dual Lattice Attacks Against Small-Secret LWE and Parameter Choices in HElib and SEAL". In: *Advances in Cryptology – EUROCRYPT 2017*. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Cham: Springer International Publishing, 2017, pp. 103–129. ISBN: 978-3-319-56614-6.

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

[3] László Babai. "On Lovász' lattice reduction and the nearest lattice point problem". In: *STACS 85*. Ed. by K. Mehlhorn. Berlin, Heidelberg: Springer Berlin Heidelberg, 1985, pp. 13–20. ISBN: 978-3-540-39136-4.

[4] Emmanouil Doulgerakis, Thijs Laarhoven, and Benne de Weger. "Finding Closest Lattice Vectors Using Approximate Voronoi Cells". In: *Post-Quantum Cryptography*. Ed. by Jintai Ding and Rainer Steinwandt. Cham: Springer International Publishing, 2019, pp. 3–22. ISBN: 978-3-030-25510-7.

[5] Emmanouil Doulgerakis, Thijs Laarhoven, and Benne de Weger. "Sieve, Enumerate, Slice, and Lift:" in: *Progress in Cryptology - AFRICACRYPT 2020*. Ed. by Abderrahmane Nitaj and Amr Youssef. Cham: Springer International Publishing, 2020, pp. 301–320. ISBN: 978-3-030-51938-4.

[6] Leo Ducas. "Shortest Vector from Lattice Sieving: A Few Dimensions for Free". In: Jan. 2018, pp. 125–145. ISBN: 978-3-319-78380-2. DOI: 10.1007/978-3-319-78381-9_5.

[7] Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. "Algorithms for the Shortest and Closest Lattice Vector Problems". In: *Coding and Cryptology*. Ed. by Yeow Meng Chee et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 159–190. ISBN: 978-3-642-20901-7.

[8] Ravi Kannan. "Minkowski's Convex Body Theorem and Integer Programming". In: *Math. Oper. Res.* 12 (1987), pp. 415–440.

[9] Jonathan Katz. *Introduction to modern cryptography*. eng. Second edition. Chapman and hall/crc cryptography and network security series. 2015. ISBN: 9781466570269.

[10] Thijs Laarhoven, Joop van de Pol, and Benne de Weger. *Solving Hard Lattice Problems and the Security of Lattice-Based Cryptosystems*. Cryptology ePrint Archive, Paper 2012/533. https://eprint.iacr.org/2012/533. 2012. URL: https://eprint.iacr.org/2012/533.

[11] Thijs Laarhoven and Michael Walter. *Dual lattice attacks for closest vector problems (with preprocessing)*. Cryptology ePrint Archive, Paper 2021/557. https://eprint.iacr.org/2021/557. 2021. URL: https://eprint.iacr.org/2021/557.

[12] Jianwei Li and Phong Q. Nguyen. *A Complete Analysis of the BKZ Lattice Reduction Algorithm*. Cryptology ePrint Archive, Paper 2020/1237. https://eprint.iacr.org/2020/1237. 2020. URL: https://eprint.iacr.org/2020/1237.

[13] Richard Lindner and Chris Peikert. "Better Key Sizes (and Attacks) for LWE-Based Encryption". In: *Topics in Cryptology – CT-RSA 2011*. Ed. by Aggelos Kiayias. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 319–339. ISBN: 978-3-642-19074-2.

[14] Mingjie Liu and Phong Q. Nguyen. "Solving BDD by Enumeration: An Update". In: *Topics in Cryptology – CT-RSA 2013*. Ed. by Ed Dawson. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 293–309. ISBN: 978-3-642-36095-4.

[15] MATZOV. *Report on the Security of LWE: Improved Dual Lattice Attack*. Apr. 2022. DOI: 10.5281/zenodo.6412487. URL: https://doi.org/10.5281/zenodo.6412487.

[16] Daniele Micciancio and Panagiotis Voulgaris. "A Deterministic Single Exponential Time Algorithm for Most Lattice Problems Based on Voronoi Cell Computations". In: *SIAM Journal on Computing* 42.3 (2013), pp. 1364–1391. DOI: 10.1137/100811970. eprint: https://doi.org/10.1137/100811970. URL: https://doi.org/10.1137/100811970.

[17] Daniele Micciancio and Panagiotis Voulgaris. "Faster Exponential Time Algorithms for the Shortest Vector Problem." In: Jan. 2010, pp. 1468–1480. DOI: 10.1137/1.9781611973075.119.

[18] Daniele Micciancio and Michael Walter. *Fast Lattice Point Enumeration with Minimal Overhead*. Cryptology ePrint Archive, Paper 2014/569. https://eprint.iacr.org/2014/569. 2014. DOI: 10.1137/1.9781611973730.21. URL: https://eprint.iacr.org/2014/569.

[19] Oded Regev. "On Lattices, Learning with Errors, Random Linear Codes, and Cryptography". In: *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*. STOC '05. Baltimore, MD, USA: Association for Computing Machinery, 2005, pp. 84–93. ISBN: 1581139608. DOI: 10.1145/1060590.1060603. URL: https://doi.org/10.1145/1060590.1060603.

[20] Mike Rosulek. *The Joy of Cryptography*. https://joyofcryptography.com. URL: https://joyofcryptography.com.

[21] C. P. Schnorr and M. Euchner. "Lattice basis reduction: Improved practical algorithms and solving subset sum problems". In: *Fundamentals of Computation Theory*. Ed. by L. Budach. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 68–85. ISBN: 978-3-540-38391-8.

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

[23] *Submission require- ments and evaluation criteria for the post-quantum cryptography standardization process*. 2016. URL: https://csrc.nist.gov/.

[24] Masaya Yasuda. "A Survey of Solving SVP Algorithms and Recent Strategies for Solving the SVP Challenge". In: *International Symposium on Mathematics, Quantum Theory, and Cryptography*. Ed. by Tsuyoshi Takagi et al. Singapore: Springer Singapore, 2021, pp. 189–207. ISBN: 978-981-15-5191-8.

[25] Xue Zhang, Zhongxiang Zheng, and Xiaoyun Wang. "A detailed analysis of primal attack and its variants". In: *Science China Information Sciences* 65 (Mar. 2022), p. 132301. DOI: 10.1007/s11432-020-2958-9.