100912024
Joshua Limbrey

Cryptanalysis of Lattice Based Post-Quantum Encryption Schemes

June 2022

Supervisor: Dr Rachel Player

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

Contents

1 Preliminaries

1.1 Abbreviations and Acronyms

CCA2 adaptive chosen ciphertext attack 3

CCA chosen ciphertext attack 3

CPA chosen plaintext attack 3

CVP closest vector problem 6

FFT fast fourier transform 4

IND indistinguishability of ciphertexts 3

LLL Lenstra, Lenstra and Lovász, 1982 4

LWE learning with errors 1

MLWE module learning with errors 1

NTT number theoretic transform 5

PKE public-key encryption 1

SVP shortest vector problem 6

1.2 Definitions

Def 1.1 (PKE Scheme)**.** Let Σ be a PKE encryption 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 other than the challenge, can they decrypt the challenge.

1.3 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 by capital letters. Bold letters denote vectors (lower case) 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 $\mathbf{m}_j$, and $\mathbf{M}[i][j]$ denotes the entry in row i and column j (all vectors will be column vectors unless otherwise stated).

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 Standard Algorithms

Def 1.8 (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. 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 1.9 (Lenstra, Lenstra and Lovász, 1982 (LLL))**.** 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

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

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 LLL-reduced if both conditions below hold.

1. $|\mu_{ij} \leq \frac{1}{2}$ for $1 \leq j \leq i \leq n$.
2. $|\mathbf{b}_i^*|^2 \geq (\frac{3}{4} - \mu_{i,i-1}^2)|\mathbf{b}_{i_1}^*|^2$ for $2 \leq i \leq n$.

This gives us algorithm 1.

Algorithm 1 LLL

1: **Input: B**
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** $c_k \geq (\frac{3}{4} - \mu_{k,k-1}^2) c_{k-1}$ **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 1.10 (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. We also define $\circ$ being applied to vectors or matrices as being usual multiplication but where the individual products of entries are pointwise multiplications of coefficients.

Def 1.11 (BKZ)**.** test

[BKZ]

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 ***cite regev lwe****. 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 shortest vector problem (SVP) or closest vector problem (CVP).

2.1 Lattices

Let $\mathbf{B} \in \mathbb{R}^{d\times m}$ where the columns, $\mathbf{b}_i$, are linearly independent basis vectors[1]. 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 $Vol(\mathcal{L}) = \sqrt{det(\mathbf{BB}^T)}$.

For this reduction to lattice problems, let us first consider the q-ary lattice

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

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

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

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

Def 2.2 (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.

Def 2.3 (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 $\mathbf{v} \in \mathcal{L}$ such that $|\mathbf{v} - \mathbf{w}|$ is minimal.

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 The Dual and Primal Attacks

Both the dual and primal attacks are approaches widely believed to the most promising approaches to 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 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 ***cite matzot***.

3.1 Primal Attack

3.2 Dual Attack

Def 3.1 (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 $\mathbf{b} = \mathbf{As} + \mathbf{e} = \mathbf{A}_1\mathbf{s}_1 + \mathbf{A}_2\mathbf{s}_2 + \mathbf{e}$.

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 more in-depth later.

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 with coefficients in R or R_q. In relation to our earlier explained LWE problem, R is $\mathbb{F}$ and R_q is $\mathbb{F}_q$. We begin, by generating our matrix $\mathbf{A}$ using the following algorithm:

Algorithm 2 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).

Algorithm 3 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.

Algorithm 4 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 MLWE and Kyber.PKE

It can be seen that the security of KYBER.PKE can be reduced to the hard problem of MLWE; 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) =

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

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

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

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 MLWE problem, and define the advantage of our adversary over this problem. The MLWE 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 instead determine if the challenge was generated via the MLWE scheme, or if it was instead generated randomly. Thus, our $\mathbf{Adv}^{\text{MLWE}}(\mathcal{A}) =$

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

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 MLWE scheme, $\mathbf{Adv}^{\text{MLWE}}(\mathcal{A}) \approx$

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

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 $\frac{1}{2}$.

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

6 Tailoring the Dual Attack to Kyber

Matzot published a paper in which they detail improvements to the dual attack in order to further increase its efficacy at enumerating secrets, something 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 5 MATZOT Improved Dual Attack

Input: $(n, m, q, \chi_{\mathbf{s}}, \chi_{\mathbf{e}})$ LWE parameters, $\beta_1, \beta_2 \leq d$ where $\beta_1, \beta_2 \in \mathbb{Z}$, $k_{enum}, k_{fft}, k_{lat}$ s.t. $k_{enum} + k_{fft} + k_{lat} = n$, $p \in \mathbb{Z}$ 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.

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

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

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

$L = shortVectors(\mathbf{B}, \beta_1, \beta_2, D)$ ▷ 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 $(\alpha \mathbf{x}_j, \mathbf{y}_{lat}) \in L$ **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 $[\frac{p}{q}\mathbf{y}_{j,fft}]$th cell of T.

  end for$FFT(T)$

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

    return $\bar{\mathbf{s}}_{enum}$

  end if

end for

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_{lat}: 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

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

.

We can then find

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

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:

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

and for each, $\mathbf{x}^T \mathbf{b}$

7 Open Questions

MATZOT: To simplify the analysis, we do not assume the recovery of sfft. Although the above sum is expected to have large real value when sfft is guessed correctly, it may also be large for certain wrong guesses of sfft, provided senum is guessed correctly. This does not concern us since we only wish to recover senum.

is there a way to distinguish wrong sfft from correct sfft given senum correct?

other questions from MATZOT paper...

How should variables be chosen?

Time mem trade off possible in matzot dual attack?