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 List of Definitions, Notation, Abbreviations and Acronyms

LWE learning with errors 4

PKE public-key encryption 2

CVP closest vector problem 4

SVP shortest vector problem 4

Def 1.1 (public-key encryption (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:

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

$$\Sigma.Dec_sk(\Sigma.Enc_pk(m)) = m$$

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

Def 1.2 (Indistinguishable)**.**

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

An adversary $\mathcal{A}$ produces two messages $m_0, m_1 \in \mathcal{M}$ (of equal length). We choose a bit $b \in \{0, 1\}$.

$$c = \Sigma.Enc_{pk}(m_b)$$

Give the adversary (c, pk), and allow them to generate a bit $a \in \{0, 1\}$. If $a = b$, then the adversary has succeeded.
We say an encryption scheme Σ is indistinguishable if the following holds:

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

Intuitively, an encryption scheme has indistinguishability if an adversary is given a challenge ciphertext c, they cannot tell if it is from m_0 or m_1.

Def 1.3 (IND-CPA or CPA security)**.** "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 1.4 (IND-CCA or CCA security)**.** "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 1.5 (IND-CCA2 or CCA2 security)**.** "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 CCA 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.

Note: We shall use $\mathcal{B}$ to denote the set of 8-bit unsigned integers (or bytes), ie. the set $\{0, ..., 255\}$

2 Understanding the LWE problem

KYBER.PKE is a module learning with errors (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 $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}$ 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. This problem can be reduced to solving the lattice problems shortest vector problem (SVP) or closest vector problem (CVP).

First we 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

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

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

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

Def 2.1 (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 2.2 (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 $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}$, we can find $\mathbf{s}$ (the closest vector to $\mathbf{b}$ should be $\mathbf{A}\mathbf{s}$ given $\mathbf{e}$ is small enough) - solving the LWE problem if CVP can be solved.

3 Algorithm specification of Kyber.PKE

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]. 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 1 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 2 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 3 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 each bit as 0 if it is closer to 0 than $\lfloor \frac{q}{2} \mod q \rceil$, ****check and update**** otherwise we decrypt as 1.