A Novel CCA Attack for NTRU+ KEM*

Joohee Lee[1], Minju Lee[1], and Jaehui Park[2]

[1] Sungshin Women's University, Seoul, Republic of Korea
{jooheelee, 20211082}@sungshin.ac.kr
[2] Seoul National University, Seoul, Republic of Korea
hiems1855@gmail.com

Abstract. The KpqC competition has begun in 2022, that aims to standardize Post-Quantum Cryptography (PQC) in the Republic of Korea. Among the 16 submissions of the KpqC competition, the lattice-based schemes exhibit the most promising and balanced features in performance. In this paper, we propose an effective *classical* CCA attack to recover the transmitted session key for NTRU+, one of the lattice-based Key Encapsulation Mechanisms (KEM) proposed in the KpqC competition, for the first time. With the proposed attacks, we show that all the suggested parameters of NTRU+ do not satisfy the claimed security. We also suggest a way to modify the NTRU+ scheme to defend our attack.

Keywords: Post-Quantum Cryptography, KpqC Competition, Key Encapsulation Mechanism, NTRU+

1 Introduction

Recently, there have been increasing research efforts from both industry and academia on Post-Quantum Cryptography (PQC), following the initiation of a standardization project by the National Institute of Standards and Technology (NIST) aimed at developing new standards for public-key cryptosystems resilient to quantum attacks [13]. In this circumstance, the KpqC research center has initiated a competition for standardization of PQC algorithms in the Republic of Korea, called KpqC competition [2], since 2022.

Among the various PQC schemes, the lattice-based schemes using structured lattices [6, 10, 11, 7, 16, 1, 9, 4, 14] exhibit the most promising features in performance. Especially, the lattice-based KEMs enjoy not only fast encryption, decryption speeds better than the former standards such as RSA encryption, but also balanced sizes of public keys and ciphertexts which are around 1 KB, respectively, for the security level corresponding to AES-128.

The NTRU encryption [6] was suggested as a first encryption scheme based on the structured lattices, and has been analyzed without severe security degradation since proposed. In this regard, NTRU-type schemes have been also proposed to the NIST PQC standardization project [16, 7, 3]. Recently, Lyubashevsky and

* The attack has been firstly reported in the KpqC bulletin in June 14, 2023.

Seiler [12] proposed an NTRU-based KEM that enables the number theoretic transform (NTT) by introducing cyclotomic ring $\mathbb{Z}_{7681}[X]/(X^{768} - X^{384} + 1)$, which results several times to several dozen times faster key generation, encapsulation, and decapsulation than the NTRU KEM submitted to NIST. Also, Duman et al. [5] suggest a generic transformation with new message encoding called generalized one-time pad (GOTP) to make the decryption failure rate independent from the message, and achieve more compact parameters than those of [12].

NTRU+ KEM [8, 2] is one of the successors of this line of works with additional advantages, and is one of the lattice-based KEMs selected as the 1st round candidates of the KpqC competition. NTRU+ takes the NTT-friendly rings as the base rings to exploit NTT, and suggests a new generic transformation with a new encoding method called Semi-generalized One Time Pad (SOTP). They also suggest a CCA transformation without re-encryption to integrate their CPA-secure NTRU-type encryption into CCA-secure KEM.

In this paper, we suggest a novel CCA attack for NTRU+, which exploits the features of SOTP. Since SOTP combines addition over $\mathbb{Z}$ and the bit-wise XOR operation, we introduce an unreported ambiguity in between, by maliciously modifying the ciphertext. Using such situations on the attacker's side, we show that an attacker can retrieve a transmitted session key encapsulated in the challenge ciphertext in the CCA security game of CCA-NTRU+. This breaks OW-CCA security of CCA-NTRU+, and hence the claimed IND-CCA security does not hold for all suggested parameters. We also suggest a way to modify the CCA-NTRU+ algorithm to defend our attack. In future works, our attack algorithm for special encoding would provide useful insights for both theorist and practitioner to design and implement NTRU-type schemes securely.

Paper Organization In Section 2, we explain the basic notation, necessary backgrounds on key encapsulation mechanism, and the NTRU+ KEM in its CCA version. In Section 3, we present the CCA attack algorithm on NTRU+ KEM with concrete examples, and discuss how to defend such an attack. In Section 4, we conclude the paper.

2 Preliminaries

NOTATIONS. Throughout the paper, we denote the security parameter as $\lambda > 0$. We denote $\mathcal{T}$ as the set of ternary values -1, 0, and 1, i.e., $\mathcal{T} := \{-1, 0, 1\} \subset \mathbb{Z}$, and $\mathcal{T}^n = \{-1, 0, 1\}^n$. Also, we identify n-bit string with an n-dimensional vector in $\{0, 1\}^n$. We define n-bit string $\vec{1}$ as an n-dimensional vector of which components are all 1's, i.e., $\vec{1} := (1, 1, \cdots, 1) \in \{0, 1\}^n$. For $i \in \{1, \cdots, n\}$ and an n-dimensional vector b, we define $\pi_i(b)$ as an i-th component of b. We denote e_i as an n-bit string of which the i-th component is 1 and the other components are all 0's. We denote $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n - X^{n/2} + 1)$ for $n = 2^i 3^j$.

2.1 Key Encapsulation Mechanism

Definition 1. *A key encapsulation mechanism (KEM) with a key space $\mathcal{K}$ consists of three algorithms, key generation* KeyGen*, encapsulation* Encaps*, and decapsulation* Decaps *algorithms, defined as follows.*

- $\mathsf{KeyGen}(1^\lambda)$*: the key generation algorithm* KeyGen *is a probabilistic algorithm that takes a security parameter λ as an input, and outputs a pair of public key and secret key (pk, sk).*
- $\mathsf{Encaps}(pk)$*: the encapsulation algorithm is a probabilistic algorithm that inputs a public key pk and retrieves a pair of a ciphertext c and a key $K \in \mathcal{K}$.*
- $\mathsf{Decaps}(sk, c)$*: the decapsulation algorithm is a deterministic algorithm that takes a secret key sk and a ciphertext c as inputs and outputs a key $K \in \mathcal{K} \cup \{\perp\}$ where $K = \perp$ if c is an invalid ciphertext.*

Correctness. KEM is defined to be δ-correct if

$$Pr[K \neq K' | K' \leftarrow \mathsf{Decaps}(sk, c), (c, K) \leftarrow \mathsf{Encaps}(pk)] \leq \delta,$$

where the probability is taken over $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and the randomness in Encaps algorithm.

Security. We define two notions OW-CCA (one-way against chosen ciphertext attack) security and IND-CCA security of KEM, respectively.

Game OW-CCA	$\mathcal{O}_{dec}(c)$
1: $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$	1: **if** $c = c^*$
2: $(K^*, c^*) \leftarrow \mathsf{Encaps}(pk)$	2: **return** $\perp$
3: $K' \leftarrow \mathcal{A}^{\mathcal{O}_{dec}(\cdot)}(pk, c^*)$	3: **else return**
4: **return** $[K' = K^*]$	4: $K \leftarrow \mathsf{Decaps}(sk, c)$

Table 1: OW-CCA game for KEM

Definition 2 (OW-CCA Security of KEM). *Let* $\mathsf{KEM} = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ *be a KEM scheme with a key space $\mathcal{K}$. OW-CCA (one-way against chosen ciphertext attacks) is defined via the OW-CCA game in Table 1 and the advantage of adversary $\mathcal{A}$ is defined by*

$$\mathcal{A}dv^{\mathsf{OW\text{-}CCA}}_{\mathsf{KEM}}(\mathcal{A}) := \left| Pr[OW\text{-}CCA^{\mathcal{A}}_{\mathsf{KEM}} = 1] - \frac{1}{2} \right|.$$

Definition 3 (IND-CCA Security of KEM). *Let* $\mathsf{KEM} = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ *be a KEM scheme with a key space $\mathcal{K}$. IND-CCA (indistinguishability under*

Game IND-CCA	$\mathcal{O}_{dec}(c)$
1: $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$	1: **if** $c = c^*$
2: $(K_0, c^*) \leftarrow \mathsf{Encaps}(pk)$	2: **return** $\perp$
3: $K_1 \leftarrow \mathcal{K}$	3: **else return**
4: $b \leftarrow \{0, 1\}$	4: $K \leftarrow \mathsf{Decaps}(sk, c)$
5: $b' \leftarrow \mathcal{A}^{\mathcal{O}_{dec}(\cdot)}(pk, c^*, K_b)$	
6: **return** $[b = b']$	

Table 2: IND-CCA game for KEM

chosen ciphertext attacks) is defined via the IND-CCA game in Table 2 and the advantage of an adversary $\mathcal{A}$ is defined by

$$\mathcal{A}dv^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}}(\mathcal{A}) := \left| Pr[\mathit{IND\text{-}CCA}^{\mathcal{A}}_{\mathsf{KEM}} = 1] - \frac{1}{2} \right|.$$

We remark that if a KEM achieves IND-CCA security, i.e., the advantage of IND-CCA security is negligible in security parameter λ, then the OW-CCA security also holds.

2.2 NTRU+

In this section, we briefly explain the NTRU+ KEM [8, 2], of which security is based on the hardness assumptions of the NTRU [6] and Learning with Errors (LWE) [15, 11] problems.

In NTRU+, they suggest new message encoding and decoding algorithms called SOTP and Inv, respectively. Designing new encoding and decoding algorithms is a crucial part of NTRU+, since it allows messages to be sampled from the bit string space without any constraints on distribution, and also guarantees cryptographically negligible correctness errors in the worst case. We review the definitions of

$$\begin{aligned} \mathsf{SOTP} &: \{0,1\}^n \times \{0,1\}^{2n} \to \mathcal{T}^n \text{ and} \\ \mathsf{Inv} &: \quad \mathcal{T}^n \times \{0,1\}^{2n} \to \{0,1\}^n, \end{aligned}$$

as follows.

NTRU+ also adopted an NTT-friendly ring over cyclotomic polynomial $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n - X^{n/2} + 1)$ where q is a prime and $n = 2^i 3^j$ following the strategies in [12, 5] to speed up the algorithms in its AVX2 optimizations. Hence, the AVX2 optimized implementation shows the quite fast running time results which are from 14 to 43 kcycles for key generation, from 14 to 26 kcycles for encapsulation, and from 12 to 24 kcycles for decapsulation.

We review the IND-CCA KEM $\mathsf{CCA\text{-}NTRU+}$ in the NTRU+ proposal as follows. Let $\mathcal{K}$ be a key space, ψ_1 be a centered binomial distribution over $\mathbb{Z}$

Definition of SOTP and Inv in [8, 2]
$\mathsf{SOTP}(x \in \{0,1\}^n, u \in \{0,1\}^{2n})$
1: $u = (u_1, u_2) \in \{0,1\}^n \times \{0,1\}^n$
2: $y = (x \oplus u_1) - u_2 \in \mathcal{T}^n$
3: **return** y
$\mathsf{Inv}(y \in \mathcal{T}^n, u \in \{0,1\}^{2n})$
1: $u = (u_1, u_2) \in \{0,1\}^n \times \{0,1\}^n$
2: $x = (y + u_2) \oplus u_1 \in \{0,1\}^n$
3: **return** x

obtained by subtracting two random bits from each other, and $G : \{0,1\}^* \to \{0,1\}^{2n}$ and $H : \{0,1\}^* \to \{0,1\}^n \times \mathcal{K}$ are cryptographic hash functions.

- KeyGen$(1^\lambda) \to (pk, sk)$
  1) $f', g \leftarrow \psi_1^n$
  2) $f = 3f' + 1$
  3) If the inverses of f and g do not exist in R_q, start over from the beginning.
  4) $(pk, sk) = \big(h = 3g \cdot f^{-1} \bmod q, f\big)$
- Encaps$(pk) \to (K, c)$
  1) $m \leftarrow \{0,1\}^n$
  2) $(r, K) = H(m)$
  3) $M = \mathsf{SOTP}(m, G(r))$
  4) $c = h \cdot r + M \bmod q$
- Decaps$(pk) \to K$or $\perp$
  1) $M = (c \cdot f \bmod q) \bmod 3$
  2) $r = (c - M) \cdot h^{-1} \bmod q$
  3) $m = \mathsf{Inv}(M, G(r))$
  4) $(r', K) = H(m)$
  5) if $r = r'$, return K, and otherwise, return $\perp$.

3 CCA Attack for NTRU+

In this section, we demonstrate our attack idea for NTRU+ with an example, and explain how to launch the attack for the NTRU+ implementation.

3.1 Overview

In the SOTP algorithm, an n-bit bit string m with another $2n$-bit bit string $u = (u_1, u_2)$ are encoded into $\mathcal{T}^n$, say $M := \mathsf{SOTP}(m, u = (u_1, u_2)) = (m \oplus u_1) - u_2 \in \mathcal{T}^n$. In contrast, the Inv algorithm inputs an n-dimensional ternary vector $M \in \mathcal{T}^n$ together with $2n$-bit bit string $u = (u_1, u_2)$ and computes a binary string $m' := \mathsf{Inv}(M, u = (u_1, u_2)) = (M + u_2) \oplus u_1$. We remark that the correctness holds, i.e., $\mathsf{Inv}(\mathsf{SOTP}(m, u), u) = m$.

We observed that if an input for Inv is maliciously modified, i.e., the input is not legitimately constructed using the SOTP algorithm, then Inv may have to deal with non-binary n-dimensional vectors, rather than an n-bit bit strings as intermediate value $M + u_2$. In their theoretical definition of Inv, they assumed the output is a binary bit string for any ternary inputs (See Figure 13 from the NTRU+ proposal [8, 2]), and they did not consider the case that an intermediate value or an output of Inv can be non-binary.

Moreover, in their implementation of the Inv algorithm, they enforced the output of Inv to be a binary n-dimensional vector by putting `&0x1` at the end of the computation for each component. For example, one can check this in line 313, 337 (*resp.* 284) in `poly.c` file of `ntruplus576` (*resp.* `ntruplus768`) of the reference implementation of NTRU+ implementation published at [2]. Since there is an ambiguity in the theoretical definition of Inv when the intermediate value or output is non-binary, we follow the implementational definition that computes

$$\mathsf{Inv}(M, u = (u_1, u_2)) = ((M + u_2) \oplus u_1) \ \ \& \ \vec{1} \in \{0,1\}^n. \tag{1}$$

We use the above observations and definition of Inv in (1) to show that the ciphertext of $\mathsf{CCA\text{-}NTRU+}$ is malleable. More precisely, we show that it is possible to modify the challenge ciphertext of $\mathsf{CCA\text{-}NTRU+}$ in its security game, and then ask decapsulation oracle for a few times to achieve the secret key corresponding to the challenge ciphertext. This breaks the OW-CCA security of $\mathsf{CCA\text{-}NTRU+}$, and hence the claimed IND-CCA security does not hold.

3.2 Example

In this section, we show an example case in which, given a ciphertext $\vec{c}$ of $\mathsf{CCA\text{-}NTRU+}$, an attacker successfully generates a modified ciphertext $\vec{c}'$. Here, we assume $n = 4$ for simplicity.

Counterexample of injectivity of the Inv *algorithm* Suppose $m = (1, 0, 1, 1)$ and $u = (u_1, u_2) = (1, 1, 0, 1, 1, 0, 1, 0)$. Let $M := \mathsf{SOTP}(m, u) \in \mathcal{T}^n$. We will show that one can produce $M' \neq M$ in $\mathcal{T}^n$ such that $\mathsf{Inv}(M', u) = m = \mathsf{Inv}(M, u)$.

In this case,

$$\begin{aligned}
\mathsf{SOTP}(m, u) &= (m \oplus u_1) - u_2 \\
&= ((1, 0, 1, 1) \oplus (1, 1, 0, 1)) - (1, 0, 1, 0) \\
&= (0, 1, 1, 0) - (1, 0, 1, 0) \\
&= (-1, 1, 0, 0) = M.
\end{aligned}$$

For example, consider $M' := M + (2,0,0,0) = (1,1,0,0) \in \mathcal{T}^n$. Then,

$$\begin{aligned}
\mathsf{Inv}(M', u) &= ((M' + u_2) \oplus u_1) \ \& \ \vec{1} \\
&= (((1,1,0,0) + (1,0,1,0)) \oplus (1,1,0,1)) \ \& \ \vec{1} \\
&= ((2,1,1,0) \oplus (1,1,0,1)) \ \& \ (1,1,1,1) \\
&= (3,0,1,1) \ \& \ (1,1,1,1) \\
&= (1,0,1,1) = m.
\end{aligned} \tag{2}$$

We note that $3\&1 = 11_{(2)} \ \& \ 01_{(2)} = 1$ in the first component of the fourth line of the above equation array, since & denotes the bit-wise AND operation.

Remark 1. Our counterexample shows that, for $M' \in \mathcal{T}^n$ and $u \in \{0,1\}^{2n}$, $\mathsf{Inv}(M', u) = m$ does not imply $\mathsf{SOTP}(m, u) = M'$ since $\mathsf{SOTP}(m, u) = M$ which is the contrast of the claim of the NTRU+ submission (See Section 6.1., Message-Hiding and Rigidity Properties of SOTP in [8, 2]).

Generating a Modified Ciphertext We will use the above counterexample to show an example to generate a modified ciphertext for a given ciphertext of CCA-NTRU+. Suppose that $c = h \cdot r + M$ is a ciphertext of CCA-NTRU+, where h is a public key, and the encapsulation sets $m = (1,0,1,1) \in \{0,1\}^n$, $(r, K) \leftarrow H(m)$, $G(r) = u = (1,1,0,1,1,0,1,0)$, and $M = \mathsf{SOTP}(m, G(r))$ to generate c.

Consider $c' := c + (2,0,0,0)$. We remark that $c' = h \cdot r + M'$, since $M' = M + (2,0,0,0)$. When we decapsulate c', the following holds.

1. Since $M' = (1,1,0,0) \in \mathcal{T}^n$,
$$(c' \cdot f \bmod q) \bmod 3 = M'.$$
2. Since $c' - M' = (c + (2,0,0,0)) - (M + (2,0,0,0)) = c - M$, the same $r = (c' - M') \cdot h^{-1} = (c - M) \cdot h^{-1}$ value as in the encapsulation is recovered.
3. $\mathsf{Inv}(M', G(r))$ produces $m = \mathsf{Inv}(M, G(r))$ as shown in (2).
4. Hence, for $(r', K) \leftarrow H(m)$, $r' = r$ holds since m and r are the same values as in the encapsulation.

Hence, the decapsulation successfully outputs K which is the secret key encapsulated in the ciphertext c, despite we decapsulated c'. This can be generalized and exploited as a crucial part of our attack to recover the secret key corresponding to the challenge ciphertext for CCA-NTRU+ described in the next section.

3.3 Attack Algorithm

In this section, we explain our attack algorithm to recover the secret key corresponding to the challenge ciphertext of CCA-NTRU+ in the CCA security game. Suppose that the challenge ciphertext c^* in the CCA security game satisfies

$$c^* = h \cdot r^* + M^* \mod q,$$

where h is a public key, $(r^*, K^*) \leftarrow H(m^*) \in \{0,1\}^n$ for $m^* \in \{0,1\}^n$.

We first consider the case that the first component of $M^* \in \mathcal{T}^n$ is -1, i.e., $\pi_1(M^*) = -1$. Since the challenge ciphertext c^* is honestly generated by the challenger and we assumed $\pi_1(M^*) = -1$, it should be the case that the first component of $G(r^*) \in \{0,1\}^{2n}$ is 1, i.e., $\pi_1(G(r^*)) = 1$, since otherwise $\pi_1(M^* + G(r^*)) = -1 \notin \{0,1\}^n$.

Then, we define $c' := c^* + 2 \cdot e_1$ and send it to the decapsulation oracle of CCA-NTRU+. The decapsulation oracle first gets

$$M' := (c' \cdot f \bmod q) \bmod 3 = M^* + 2 \cdot e_1, \tag{3}$$

since $M^* + 2 \cdot e_1 \in \mathcal{T}^n$. It then recovers the same r^* used in the encapsulation since

$$\begin{aligned}(c' - M') \cdot h^{-1} &= ((c^* + 2 \cdot e_1) - (M^* + 2 \cdot e_1)) \cdot h^{-1} \\ &= (c^* - M^*) \cdot h^{-1} = r^*.\end{aligned}$$

Then, the decapsulation oracle computes

$$\begin{aligned}\mathsf{Inv}(M', G(r^*)) &= ((M' + u_1) \oplus u_2) \;\&\; \vec{1} \\ &= (((M^* + 2 \cdot e_1 + u_1) \oplus u_2) \;\&\; \vec{1} \\ &= ((M^* + u_1) \oplus u_2) \;\&\; \vec{1} = m^*,\end{aligned}$$

where $\pi_1(u_1) = \pi_1(G(r^*)) = 1$ and $m^* := \mathsf{Inv}(M^*, G(r^*))$. We note that, under the assumption $\pi_1(M^*) = -1$, the first component of $M' + u_1$ becomes 2, and it is canceled out to be zero by $\&\vec{1}$ operation at the end so that it is equal to the first component of $M^* + u_1$ again. This implies the decapsulation oracle successfully recover the initial randomness m^* for the challenge ciphertext, and hence it produces the values $(r', K') \leftarrow H(m^*)$ in which $r' = r^*$ and $K' = K^*$, the correct decapsulation result of the challenge ciphertext. To sum up, the decapsulation oracle does not abort and successfully returns K^* which is a decapsulation result of the challenge ciphertext c^*.

We remark that $\pi_1(M^*) = -1$ does not always hold by its definition. More precisely, if legitimately generated, $(\pi_1(M^*), \pi_1(G(r^*)), \pi_1(M^* + G(r^*)))$ should be one of the four cases corresponding to the rows in Table 3, and the probability that each case happens is 1/4. In Case II, III, and IV, the relation (3) does not hold after modulo 3 operation since $M^* + 2 \cdot e_1 \notin \mathcal{T}^n$, so that it produces $r := (c' - M') \cdot h^{-1} \neq r^*$ and $m := \mathsf{Inv}(M', G(r)) \neq m^*$. Hence, in these three cases, the decapsulation fails with overwhelming probability.

This implies that the decapsulation oracle is expected to produce a decapsulation result which equals to the decapsulation of the challenge ciphertext in Case I in Table 3, and abort otherwise. Also, the same holds for the i-th component in general, for $i = 1, \cdots, n$. Hence, the attacker can proceed with the same strategy for the i-th component of the challenge ciphertext by sending $c^* + 2 \cdot e_i$ to the decapsulation oracle increasing $i = 1, \cdots n$, until she gets the decapsulation result of the challenge ciphertext. The attack succeeds in the fourth trial in average. We present the pseudocode for our attack algorithm in Algorithm 1.

	$\pi_i(M^*)$	$\pi_i(G(r^*))$	$\pi_i(M^* + G(r^*))$
I	-1	1	0
II	1	0	1
III	0	1	1
IV	0	0	0

Table 3: Possible cases of $(\pi_i(M^*), \pi_i(G(r^*)), \pi_i(M^* + G(r^*)))$ for $i = 1, \cdots, n$

Algorithm 1 Pseudocode for our attack algorithm

Require: a challenge ciphertext $c^* \in \mathcal{R}_q$
Ensure: a secret key $K \in \{0,1\}^{2\lambda}$
  for $i \in \{1, \cdots, n\}$ **do**
    $c' \leftarrow c^* + 2 \cdot e_i$ (Note that $c' \neq c^*$)
    Send c' to the decapsulation oracle $\mathcal{O}_{dec}$
    if $\mathcal{O}_{dec}$ outputs K' **then**
      Output K' as a decapsulation for c^*
      break;
    end if
  end for

Theorem 1. *For* CCA-NTRU+*, an attacker described in Algorithm 1 terminates in polynomial time in λ, and wins the OW-CCA game with an overwhelming probability.*

Proof. First, Algorithm 1 terminates in time $\mathcal{O}(n)$, hence in polynomial time in λ. If an attacker loses, then it splits into two cases :

i) the decapsulation oracle $\mathcal{O}_{dec}$ aborts for all $i \in \{1, \cdots, n\}$.

ii) the decapsulation oracle $\mathcal{O}_{dec}$ outputs a wrong output $K' \neq K^*$ for some $i \in \{1, \cdots, n\}$.

The first case occurs only if Case I never happens for all $i = 1, \cdots, n$. Hence, the probability of the first case is upper-bounded by $(1-1/4)^n$ which is negligible in λ.

The latter happens with probability $1/2^n$, assuming the hash functions are the random oracles, and hence the probability it happens is negligible in λ.

To sum up, we showed that i) and ii) happen with negligible probability in λ so that the adversary finds the correct secret key corresponding to the challenge ciphertext and wins the OW-CCA game in polynomial time for all but a negligible probability in λ.

Remark 2. As noted, the CCA-NTRU+ KEM algorithm does not achieve IND-CCA security, since it is not OW-CCA secure as shown in Theorem 1.

3.4 Discussion

In this section, we provide some insights on why this attack can take place, and how one can prevent it when designing KEMs based on the NTRU assumption.

The attack exploits that the intermediate value of the computation of Inv can be non-binary which is not considered in the theoretical definition throughout the NTRU+ document. However, we emphasize that the vulnerability comes not only from the implementation aspects, but also from the theoretical definition itself, since the theoretical definition does not deal with the non-binary intermediate values in Inv. The ambiguity of the theoretical definition for the non-binary intermediate values led us to follow the description of Inv in the implementation as well.

We remark that, to defend the attack, the simplest way is to check if each component of the intermediate value $M+u_2$ of Inv is binary, and reject otherwise. The rejection can also be implicitly held by generating and outputting a random session key instead. This way may create other vulnerabilities against physical attacks, such as simple power analysis, due to the process of checking whether the intermediate value is binary or not. Hence, it would have to be designed carefully concerning secure implementation, and the change should be addressed in the theoretical specification (and also in the security proofs) as well as in the implementation.

4 Conclusion

In this paper, we suggest a CCA attack for NTRU+, using the features of the encoding method SOTP. We explain our observation and the attack idea with an example, and present an attack algorithm to recover a key corresponding to the challenge ciphertext of CCA-NTRU+. We also suggest a countermeasure to modify CCA-NTRU+ KEM to make it secure against our attack.

Acknowledgments. This work is the result of commissioned research project supported by the affiliated institute of ETRI [2023-080]. This work was partly supported by the Sungshin Women's University Research Grant of 2023 (Grant No. H20230056). We thank Suhri Kim, Hansol Ryu, and Kyung Chul Jeong for fruitful discussion and comments.

References

[1] Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: Crystals-kyber: a cca-secure module-lattice-based kem. In: 2018 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 353–367. IEEE (2018)

[2] Center, K.R.: Kpqc competition round 1, available from: `https://www.kpqc.or.kr/competition.html` [last accessed June 2023]

[3] Chen, C., Danba, O., Hoffstein, J., Hülsing, A., Rijneveld, J., Schanck, J.M., Schwabe, P., Whyte, W., Zhang, Z.: Algorithm specifications and supporting documentation. Brown University and Onboard security company, Wilmington USA (2019)
[4] Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: Crystals-dilithium: A lattice-based digital signature scheme. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 238–268 (2018)
[5] Duman, J., Hövelmanns, K., Kiltz, E., Lyubashevsky, V., Seiler, G., Unruh, D.: A thorough treatment of highly-efficient ntru instantiations. In: IACR International Conference on Public-Key Cryptography. pp. 65–94. Springer (2023)
[6] Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: A ring-based public key cryptosystem. In: International algorithmic number theory symposium. pp. 267–288. Springer (1998)
[7] Hülsing, A., Rijneveld, J., Schanck, J.M., Schwabe, P.: NTRU-HRSS-KEM. NIST submissions (2017)
[8] Kim, J., Park, J.H.: Ntru+: Compact construction of ntru using simple encoding method. Cryptology ePrint Archive (2022)
[9] Lee, J., Kim, D., Lee, H., Lee, Y., Cheon, J.H.: Rlizard: Post-quantum key encapsulation mechanism for iot devices. IEEE Access **7**, 2080–2091 (2018)
[10] Lindner, R., Peikert, C.: Better key sizes (and attacks) for LWE-based encryption. In: Topics in Cryptology–CT-RSA 2011: The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings. pp. 319–339. Springer (2011)
[11] Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. Journal of the ACM (JACM) **60**(6), 1–35 (2013)
[12] Lyubashevsky, V., Seiler, G.: Nttru: truly fast ntru using ntt. Cryptology ePrint Archive (2019)
[13] NIST: Post-quantum cryptography, available from: `https://csrc.nist.gov/projects/post-quantum-cryptography` [last accessed June 2023]
[14] Prest, T., Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: Falcon. Post-Quantum Cryptography Project of NIST (2020)
[15] Regev, O.: The learning with errors problem. Invited survey in CCC **7**(30), 11 (2010)
[16] Zhang, Z., Chen, C., Hoffstein, J., Whyte, W., Schanck, J.M., Hulsing, A., Rijneveld, J., Schwabe, P., Danba, O.: NTRUEncrypt. Tech. Rep. (2019)