NTRU+

Algorithm Specifications And Supporting Documentation

(version 1.1)

NTRU+: Compact Construction of NTRU Using Simple Encoding Method[*][†]

Jonghyun Kim[‡] Jong Hwan Park[§]

September 16, 2023

Abstract

NTRU was the first practical public key encryption scheme constructed on a lattice over a polynomial-based ring and has been considered secure against significant cryptanalytic attacks over the past few decades. However, NTRU and its variants suffer from several drawbacks, including difficulties in achieving worst-case correctness error in a moderate modulus, inconvenient sampling distributions for messages, and relatively slower algorithms compared to other lattice-based schemes.

In this work, we propose a new NTRU-based key encapsulation mechanism (KEM), called $\mathsf{NTRU+}$, which overcomes nearly all existing drawbacks. $\mathsf{NTRU+}$ is constructed based on two new generic transformations: ACWC_2 and $\overline{\mathsf{FO}}^{\perp}$ (a variant of the Fujisaki-Okamoto transform). ACWC_2 is used to easily achieve worst-case correctness error, while $\overline{\mathsf{FO}}^{\perp}$ is used to achieve chosen-ciphertext security without re-encryption. Both ACWC_2 and $\overline{\mathsf{FO}}^{\perp}$ are defined using a randomness-recovery algorithm and an encoding method. In particular, our simple encoding method, the semi-generalized one-time pad (SOTP), allows us to sample a message from a natural bit-string space with an arbitrary distribution. We provide four parameter sets for $\mathsf{NTRU+}$ and present implementation results using NTT-friendly rings over cyclotomic trinomials.

Keywords: NTRU, RLWE, Lattice-based cryptography, Post-quantum cryptography.

1 Introduction

The NTRU encryption scheme [15] was introduced in 1998 by Hoffstein, Pipher, and Silverman as the first practical public key encryption scheme using lattices over polynomial rings. The hardness of NTRU is crucially based on the NTRU problem [15], which has withstood significant cryptanalytic attacks over the past few decades. This longer history, compared to other lattice-based problems (such as ring/module-LWE), has been considered an important factor in selecting NTRU as a finalist in the NIST PQC standardization process. While the finalist NTRU [6] has not been chosen by NIST as one of the first four quantum-resistant cryptographic algorithms, it still has several distinct advantages over other lattice-based competitive schemes such as KYBER [25] and Saber [9]. Specifically, the advantages of NTRU include: (1) the compact structure

[*]This work is submitted to 'Korean Post-Quantum Cryptography Competition' (`www.kpqc.or.kr`).

[†]This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2021-00518, Blockchain privacy preserving techniques based on data encryption).

[‡]Korea University, Seoul, Korea. Email: `yoswuk@korea.ac.kr`.

[§]Sangmyung University, Seoul, Korea. Email: `jhpark@smu.ac.kr`.

of a ciphertext consisting of a single polynomial, and (2) (possibly) faster encryption and decryption without the need to sample the coefficients of a public key polynomial.

The central design principle of NTRU is described over a ring $R_q = \mathbb{Z}_q[x]/\langle f(x)\rangle$, where q is a positive integer and $f(x)$ is a polynomial. The public key is generated as $\mathbf{h} = p\mathbf{g}/(p\mathbf{f}' + 1)$[1], where $\mathbf{g}$ and $\mathbf{f}'$ are sampled according to a narrow distribution ψ, p is a positive integer that is coprime with q and smaller than q (e.g., 3), and the corresponding private key is $\mathbf{f} = p\mathbf{f}' + 1$. To encrypt a message m sampled from the message space $\mathcal{M}'$, one creates two polynomials $\mathbf{r}$ and $\mathbf{m}$, with coefficients drawn from a narrow distribution ψ, and computes the ciphertext $\mathbf{c} = \mathbf{hr} + \mathbf{m}$ in R_q. An (efficient) encoding method may be used to encode $m \in \mathcal{M}'$ into $\mathbf{m}$ and $\mathbf{r} \in R_q$. Alternatively, it is possible to directly sample $\mathbf{m}$ and $\mathbf{r}$ from ψ, where $\mathbf{m}$ is considered as the message to be encrypted. To decrypt the ciphertext $\mathbf{c}$, one computes $\mathbf{cf}$ in R_q, recovers $\mathbf{m}$ by deriving the value $\mathbf{cf}'$ modulo p, and (if necessary) decodes $\mathbf{m}$ to obtain the message m. The decryption of NTRU works correctly if all the coefficients of the polynomial $p(\mathbf{gr} + \mathbf{f}'\mathbf{m}) + \mathbf{m}$ are less than $q/2$. Otherwise, the decryption fails, and the probability that it fails is called a *correctness (or decryption) error*.

In the context of chosen-ciphertext attacks, NTRU, like other ordinary public key encryption schemes, must guarantee an extremely negligible worst-case correctness error. This is essential to prevent the leakage of information about the private key through adversarial decryption queries, such as attacks against lattice-based encryption schemes [18, 8]. Roughly speaking, the worst-case correctness error refers to the probability that decryption fails for any ciphertext that can be generated with all possible messages and randomness in their respective spaces. The worst-case correctness error considers that an adversary, $\mathcal{A}$, can *maliciously* choose messages and randomness without sampling normally according to their original distributions (if possible). In the case of NTRU, the failure to decrypt a specific ciphertext $\mathbf{c} = \mathbf{hr} + \mathbf{m}$ provides $\mathcal{A}$ with the information that one of the coefficients of $p(\mathbf{gr} + \mathbf{f}'\mathbf{m}) + \mathbf{m}$ is larger than or equal to $q/2$. If $\mathcal{A}$ has control over the choice of $\mathbf{r}$ and $\mathbf{m}$, even one such decryption failure may open a path to associated decryption queries to obtain more information about secret polynomials $\mathbf{g}$ and $\mathbf{f}$.

When designing NTRU, two approaches can be used to achieve worst-case correctness error. One approach is to draw $\mathbf{m}$ and $\mathbf{r}$ directly from ψ, while setting the modulus q to be relatively large. The larger q guarantees a high probability that all coefficients of $p(\mathbf{gr}+\mathbf{f}'\mathbf{m})+\mathbf{m}$ are less than $q/2$ for *nearly all possible* $\mathbf{m}$ and $\mathbf{r}$ in their spaces, although it causes inefficiency in terms of public key and ciphertext sizes. Indeed, this approach has been used by the third-round finalist NTRU [6], wherein all recommended parameters provide *perfect* correctness error (i.e., the worst-case correctness error becomes zero for all possible $\mathbf{m}$ and $\mathbf{r}$). By contrast, the other approach [12] is to use an encoding method by which a message $m \in \mathcal{M}'$ is used as a randomness to sample $\mathbf{m}$ and $\mathbf{r}$ according to ψ. Under the Fujisaki-Okamoto (FO) transform [13], decrypting a ciphertext $\mathbf{c}$ requires re-encrypting m by following the same sampling process as encryption. Thus, an ill-formed ciphertext that does not follow the sampling rule will always fail to be successfully decrypted, implying that $\mathbf{m}$ and $\mathbf{r}$ should be *honestly* sampled by $\mathcal{A}$ according to ψ. Consequently, by disallowing $\mathcal{A}$ to have control over $\mathbf{m}$ and $\mathbf{r}$, the NTRU with an encoding method has a worst-case correctness error that is close to an average-case error.

Based on the aforementioned observation, [12] proposed generic (average-case to worst-case) transformations[2] that make the average-case correctness error of an underlying scheme nearly close to the worst-case error of a transformed scheme. One of their transformations (denoted by ACWC) is based on an encoding method called the generalized one-time pad (denoted by GOTP). Roughly speaking, GOTP works as fol-

[1]There is another way of creating the public key as $\mathbf{h} = p\mathbf{g}/\mathbf{f}$, but we focus on setting $\mathbf{h} = p\mathbf{g}/(p\mathbf{f}' + 1)$ for a more efficient decryption process.

[2]They proposed two transformations called ACWC_0 and ACWC. In this paper, we focus on ACWC that does not expand the size of a ciphertext.

Scheme	NTRU[6]	NTRU-B [12]	NTRU+
NTT-friendly	No	Yes	Yes
Correctness error	Perfect	Worst-case	Worst-case
$(\mathbf{m}, \mathbf{r})$-encoding	No	Yes	Yes
Message set	$\mathbf{m}, \mathbf{r} \leftarrow \{-1,0,1\}^n$	$m \leftarrow \{-1,0,1\}^\lambda$	$m \leftarrow \{0,1\}^n$
Message distribution	Uniform/Fixed-weight	Uniform	Arbitrary
CCA transform	DPKE + SXY variant	ACWC + $\mathsf{FO}^\perp$	ACWC_2 + $\overline{\mathsf{FO}}^\perp$
Assumptions	NTRU, RLWE	NTRU, RLWE	NTRU, RLWE
Tight reduction	Yes	No	Yes

n: polynomial degree of the ring. λ: length of the message. DPKE: deterministic public key encryption.
SXY variant: SXY transformation [24] described in the finalist NTRU.

Table 1: Comparison to previous NTRU constructions

lows: a message $m \in \mathcal{M}'$ is first used to sample $\mathbf{r}$ and $\mathbf{m}_1$ according to ψ, and $\mathbf{m}_2 = \mathsf{GOTP}(m, \mathsf{G}(\mathbf{m}_1))$ using a hash function G, and then $\mathbf{m}$ is constructed as $\mathbf{m}_1||\mathbf{m}_2$. If the GOTP acts as a sampling function wherein the output follows ψ, $\mathbf{m}$ and $\mathbf{r}$ are created from m following ψ, which can be verified in decryption using the FO transform. Specifically, for two inputs m and $\mathsf{G}(\mathbf{m}_1)$ that are sampled from $\{-1,0,1\}^\lambda$ for some integer λ, $\mathbf{m}_2 \in \{-1,0,1\}^\lambda$ is computed by doing the component-wise exclusive-or modulo 3 of two ternary strings m and $\mathsf{G}(\mathbf{m}_1)$. Thus, if $\mathsf{G}(\mathbf{m}_1)$ follows a uniformly random distribution ψ over $\{-1,0,1\}^\lambda$, m is hidden from $\mathbf{m}_2$ because of the one-time pad property.

However, an ACWC based on the GOTP has two disadvantages in terms of security reduction and message distribution. First, [12] showed that ACWC converts a one-way CPA (OW-CPA) secure underlying scheme into a transformed one that is still OW-CPA secure, besides the fact that their security reduction is loose[3] by causing a security loss factor of q_G, the number of random oracle queries. Second, ACWC forces even a message $m \in \mathcal{M}'$ to follow a specific distribution because their security analysis of ACWC requires GOTP to have the additional randomness-hiding property, meaning that $\mathsf{G}(\mathbf{m}_1)$ should also be hidden from the output $\mathbf{m}_2$. Indeed, the NTRU instantiation from ACWC, called 'NTRU-B' [12], requires that m should be chosen uniformly at random from $\mathcal{M}' = \{-1,0,1\}^\lambda$. Notably, it is difficult to generate exactly uniformly random numbers from $\{-1,0,1\}$ in constant time due to rejection sampling. Therefore, it was an open problem [12] to construct a new transformation that permits a different, more easily sampled distribution of a message while relying on the same security assumptions.

1.1 Our Results

We propose a new practical NTRU construction called "NTRU+" that addresses the two drawbacks of the previous ACWC. To achieve this, we introduce a new generic ACWC transformation, denoted as ACWC_2, which utilizes a simple encoding method. By using ACWC_2, NTRU+ achieves a worst-case correctness error close to the average-case error of the underlying NTRU. Additionally, NTRU+ requires the message m to be drawn from $\mathcal{M}' = \{0,1\}^n$ (for a polynomial degree n), following an *arbitrary* distribution with high min-entropy, and is proven to be *tightly* secure under the same assumptions as NTRU-B, the NTRU and RLWE assumptions. To achieve chosen-ciphertext security, NTRU+ relies on a novel FO-equivalent

[3][12] introduced a new security notion, q-OW-CPA, which states that an adversary outputs a set Q with a maximum size of q and wins if the correct message corresponding to a challenged ciphertext belongs to Q. We believe that q-OW-CPA causes a security loss of q.

	ACWC_0[12]	ACWC[12]	ACWC_2
Message encoding	No	GOTP	SOTP
Message distribution	Arbitrary	Uniform	Arbitrary
Ciphertext expansion	Yes	No	No
Transformation	OW-CPA $\rightarrow$ IND-CPA	OW-CPA $\rightarrow$ OW-CPA	OW-CPA $\rightarrow$ IND-CPA
Tight reduction	No	No	Yes
Underlying PKE	Any	Any	Injective + MR + RR

MR: message-recoverable. RR: randomness-recoverable.

Table 2: Comparison to previous ACWC transformations

transform without re-encryption, which makes the decryption algorithm of NTRU+ faster than in the ordinary FO transform. In terms of efficiency, we use the idea from [23] to apply the Number Theoretic Transform (NTT) to NTRU+ and therefore instantiate NTRU+ over a ring $R_q = \mathbb{Z}_q[x]/\langle f(x)\rangle$, where $f(x) = x^n - x^{n/2} + 1$ is a cyclotomic trinomial. By selecting appropriate (n, q) and ψ, we suggest four parameter sets for NTRU+ and provide the implementation results for NTRU+ in each parameter set. Table 1 lists the main differences between the previous NTRU constructions [6, 12] and NTRU+. In the following section, we describe our technique, focusing on these differences.

ACWC_2 Transformation with Tight Reduction. ACWC_2 is a new generic transformation that allows for the aforementioned average-case to worst-case correctness error conversion. However, to apply ACWC_2, the underlying scheme is required to have injectivity, randomness-recoverable (RR), and message-recoverable (MR) properties, which are typical of NTRU.[4] Additionally, ACWC_2 involves an encoding method called semi-generalized one-time pad (denoted by SOTP). In contrast to the GOTP in [12], SOTP works in a generic sense as follows: first, a message $m \in \mathcal{M}'$ is used to sample $\mathbf{r}$ based on ψ, and then $\mathbf{m} = \mathsf{SOTP}(m, \mathsf{G}(\mathbf{r}))$ is computed, where the coefficients follow ψ, using a hash function G. When decrypting a ciphertext $\mathbf{c} = \mathsf{Enc}(pk, \mathbf{m}; \mathbf{r})$ under a public key pk, $\mathbf{m}$ is recovered by a normal decryption algorithm, and using $\mathbf{m}$, $\mathbf{r}$ is also recovered by a randomness-recovery algorithm. Finally, an inverse of SOTP using $\mathsf{G}(\mathbf{r})$ and $\mathbf{m}$ yields m.

The MR property of an underlying scheme allows us to show that, without causing any security loss, ACWC_2 transforms an OW-CPA secure scheme into a chosen-plaintext (IND-CPA) secure scheme. The proof idea is simple: unless an IND-CPA adversary $\mathcal{A}$ queries $\mathbf{r}$ to a (classical) random oracle G, $\mathcal{A}$ does not obtain any information on m_b (that $\mathcal{A}$ submits) for $b \in \{0, 1\}$ because of the basic message-hiding property of SOTP. However, whenever $\mathcal{A}$ queries $\mathbf{r}_i$ to G for $i = 1, \cdots, q_\mathsf{G}$, a reductionist can check whether each $\mathbf{r}_i$ is the randomness used for its OW-CPA challenge ciphertext using a message-recovery algorithm. Therefore, the reductionist can find the exact $\mathbf{r}_i$ among the q_G number of queries if $\mathcal{A}$ queries $\mathbf{r}_i$ (with respect to its IND-CPA challenge ciphertext) to G. In this security analysis, it is sufficient for SOTP to have the message-hiding property, which makes SOTP simpler than GOTP because GOTP must have both message-hiding and randomness-hiding properties.

Table 2 presents a detailed comparison between previous ACWC transformations and our new ACWC_2. Unlike the previous ACWC based on GOTP, [12] proposed another generic ACWC transformation (denoted by ACWC_0) without using any message-encoding method. In ACWC_0, a (bit-string) message m is encrypted with a ciphertext $\mathbf{c} = (\mathsf{Enc}(pk, \mathbf{m}; \mathbf{r}), \mathsf{F}(\mathbf{m}) \oplus m)$ using a hash function F, which causes the ciphertext

[4]In the decryption of NTRU with $pk = \mathbf{h}$, given $(pk, \mathbf{c}, \mathbf{m})$, $\mathbf{r}$ is recovered as $\mathbf{r} = (\mathbf{c} - \mathbf{m})\mathbf{h}^{-1}$. Similarly, given $(pk, \mathbf{c}, \mathbf{r})$, $\mathbf{m}$ is recovered as $\mathbf{m} = \mathbf{c} - \mathbf{hr}$.

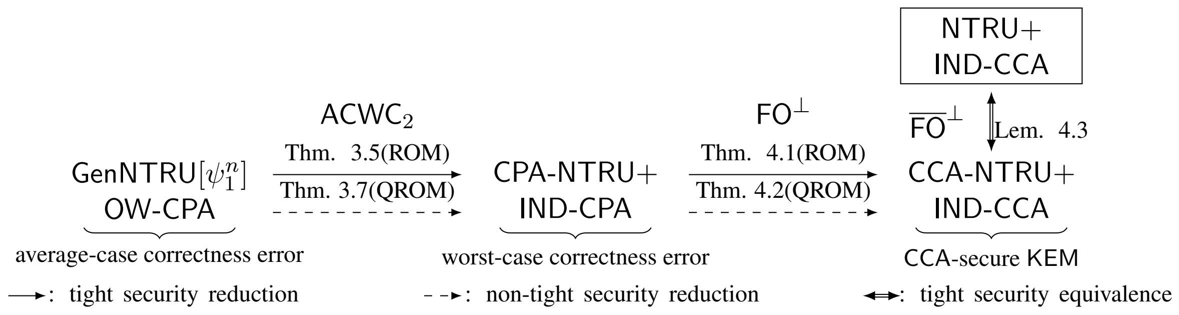

Figure 1: Overview of security reductions

expansion of $\mathsf{F}(\mathbf{m}) \oplus m$, whereas such a ciphertext redundancy does not occur in ACWC and ACWC_2. Like ACWC_2, ACWC_0 transforms any OW-CPA secure scheme into an IND-CPA secure one, but their security reduction is not tight as in ACWC. ACWC_0 and ACWC_2 requires no specific message distribution, whereas ACWC requires $m \in \mathcal{M}'$ to be sampled according to a uniformly random distribution from $\mathcal{M}'$. ACWC_0 and ACWC work for any OW-CPA secure scheme, but ACWC_2 works for any OW-CPA secure scheme satisfying injectivity, MR, and RR properties.

FO-Equivalent Transform without Re-encryption. To achieve chosen-ciphertext (IND-CCA) security, we apply the generic transform $\mathsf{FO}^{\perp}$ to the ACWC_2-derived scheme, which is IND-CPA secure. As with other FO-transformed schemes, the resulting scheme from ACWC_2 and $\mathsf{FO}^{\perp}$ is still required to perform re-encryption in the decryption process to check if (1) $(\mathbf{m}, \mathbf{r})$ are correctly generated from m and (2) a (decrypted) ciphertext $\mathbf{c}$ is correctly encrypted from $(\mathbf{m}, \mathbf{r})$. However, by using the RR property of the underlying scheme, we further remove the re-encryption process from $\mathsf{FO}^{\perp}$. Instead, the more advanced transform (denoted by $\overline{\mathsf{FO}}^{\perp}$) simply checks whether $\mathbf{r}$ from the randomness-recovery algorithm is the same as the (new) randomness $\mathbf{r}'$ created from m. We show that $\overline{\mathsf{FO}}^{\perp}$ is functionally identical to $\mathsf{FO}^{\perp}$ by proving that the randomness-checking process in $\overline{\mathsf{FO}}^{\perp}$ is equivalent to the re-encryption process $\mathsf{FO}^{\perp}$. The equivalence proof relies mainly on the injectivity [16, 5] and rigidity [4] properties of the underlying schemes. As a result, although the RR property seems to incur some additional decryption cost, it ends up making the decryption algorithm faster than the original FO transform. Figure 1 presents an overview of security reductions from OW-CPA to IND-CCA.

Simple SOTP Instantiation with More Convenient Sampling Distributions. As mentioned previously, ACWC_2 is based on an efficient construction of SOTP that takes m and $\mathsf{G}(\mathbf{r})$ as inputs and outputs $\mathbf{m} = \mathsf{SOTP}(m, \mathsf{G}(\mathbf{r}))$. In particular, computing $\mathbf{m} = \mathsf{SOTP}(m, \mathsf{G}(\mathbf{r}))$ requires that each coefficient of $\mathbf{m}$ should follow ψ, while preserving the message-hiding property. To achieve this, we set ψ as the centered binomial distribution (CBD) ψ_k with $k = 1$, which is easily obtained by subtracting two uniformly random bits from each other. For a polynomial degree n and hash function $\mathsf{G} : \{0,1\}^* \to \{0,1\}^{2n}$, m is chosen from the message space $\mathcal{M}' = \{0,1\}^n$ for an arbitrary distribution (with high min-entropy) and $\mathsf{G}(\mathbf{r}) = y_1 || y_2 \in \{0,1\}^n \times \{0,1\}^n$. SOTP then computes $\tilde{m} = (m \oplus y_1) - y_2$ by bitwise subtraction and assigns each subtraction value of $\tilde{m}$ to the coefficient of $\mathbf{m}$. By the one-time pad property, it is easily shown that $m \oplus y_1$ becomes uniformly random in $\{0,1\}^n$ (and thus message-hiding) and each coefficient of $\mathbf{m}$ follows ψ_1. Since $\mathbf{r}$ is also sampled from a hash value of m according to ψ_1, all sampling distributions in NTRU+ are easy to implement. We can also expect that, similar to the case of ψ_1, the SOTP is expanded to sample a centered binomial distribution reduced modulo 3 (i.e., $\overline{\psi}_2$) by summing up and subtracting more uniformly random bits.

NTT-Friendly Rings Over Cyclotomic Trinomials. $\mathsf{NTRU+}$ is instantiated over a polynomial ring $R_q = \mathbb{Z}_q[x]/\langle f(x)\rangle$, where $f(x) = x^n - x^{n/2} + 1$ is a cyclotomic trinomial of degree $n = 2^i 3^j$. [23] showed that, with appropriate parameterization of n and q, such a ring can also provide NTT operation essentially as fast as that over a ring $R_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$. Moreover, because the choice of a cyclotomic trinomial is moderate, it provides more flexibility to satisfy a certain level of security. Based on these results, we choose four parameter sets for $\mathsf{NTRU+}$, where the polynomial degree n of $f(x) = x^n - x^{n/2} + 1$ is set to be 576, 768, 864, and 1152, and the modulus q is 3457 for all cases. Table 7 lists the comparison results between finalist NTRU [6], KYBER, KYBER-90s [25], and NTRU+ in terms of security and efficiency. To estimate the concrete security level of $\mathsf{NTRU+}$, we use the Lattice estimator [1] for the RLWE problem and the NTRU estimator [6] for the NTRU problem, considering that all coefficients of each polynomial $\mathbf{f}'$, $\mathbf{g}$, $\mathbf{r}$, and $\mathbf{m}$ are drawn according to the centered binomial distribution ψ_1. The implementation results in Table 7 are estimated with reference and AVX2 optimizations. We can observe that $\mathsf{NTRU+}$ outperforms NTRU at a similar security level.

1.2 Related Works

The first-round NTRUEncrypt [26] submission to the NIST PQC standardization process was an NTRU-based encryption scheme with the NAEP padding method [19]. Roughly speaking, NAEP is similar to our SOTP, but the difference is that it does not completely encode $\mathbf{m}$ to prevent an adversary $\mathcal{A}$ from choosing $\mathbf{m}$ maliciously. This is due to the fact that $\mathbf{m} := \mathsf{NAEP}(m, \mathsf{G}(\mathbf{hr}))$ is generated by subtracting two n-bit strings m and $\mathsf{G}(\mathbf{hr})$ from each other, i.e., $m - \mathsf{G}(\mathbf{hr})$ by bitwise subtraction, and then assigning them to the coefficients of $\mathbf{m}$. Since m can be maliciously chosen by $\mathcal{A}$ in NTRUEncrypt, $\mathbf{m}$ can also be maliciously chosen, regardless of $\mathsf{G}(\mathbf{hr})$.

The finalist NTRU [6] was submitted as a key encapsulation mechanism (KEM) that provides four parameter sets for perfect correctness. To achieve chosen-ciphertext security, [6] relied on a variant of the SXY [24] conversion, which also avoids re-encryption during decapsulation. Similar to $\mathsf{NTRU+}$, the SXY variant requires the rigidity [4] of an underlying scheme and uses the notion of deterministic public key encryption (DPKE) where $(\mathbf{m}, \mathbf{r})$ are all recovered as a message during decryption. In the NTRU construction, the recovery of $\mathbf{r}$ is conceptually the same as the existence of the randomness-recovery algorithm RRec. Instead of removing re-encryption, the finalist NTRU needs to check whether $(\mathbf{m}, \mathbf{r})$ are selected correctly from predefined distributions.

In 2019, Lyubashevsky et al. [23] proposed an efficient NTRU-based KEM called NTTRU by applying NTT to the ring defined by a cyclotomic trinomial $\mathbb{Z}_q[x]/\langle x^n - x^{n/2} + 1\rangle$. NTTRU was based on the Dent [10] transformation without any encoding method, which resulted in an approximate worst-case correctness error of 2^{-13}, even with an average-case error of 2^{-1230}. To overcome this significant difference, NTTRU was modified to reduce the message space of the underlying scheme, while increasing the size of the ciphertext. This modification was later generalized to ACWC_0 in [12].

In 2021, Duman et al. [12] proposed two generic transformations, ACWC_0 and ACWC, which aim to make the average-case correctness error of an underlying scheme nearly equal to the worst-case error of the transformed scheme. Specifically, ACWC introduced GOTP as an encoding method to prevent $\mathcal{A}$ from adversarially choosing $\mathbf{m}$. While ACWC_0 is simple, it requires a ciphertext expansion of 32 bytes. On the other hand, ACWC does not requires an expansion of the ciphertext size. The security of ACWC_0 and ACWC was analyzed in both the classical and quantum random oracle models [12]. However, their NTRU instantiation using ACWC has the drawback of requiring the message m to be chosen from a uniformly random distribution over $\mathcal{M}' = \{-1, 0, 1\}^{\lambda}$.

2 Preliminaries

2.1 Public Key Encryption and Related Properties

Definition 2.1 (Public Key Encryption)**.** A public key encryption scheme $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with a message space $\mathcal{M}$ and a randomness space $\mathcal{R}$ consists of the following three algorithms:

- $\mathsf{Gen}(1^\lambda)$: The key generation algorithm Gen is a randomized algorithm that takes a security parameter 1^λ as input and outputs a pair of public/secret keys (pk, sk).
- $\mathsf{Enc}(pk, m)$: The encryption algorithm Enc is a randomized algorithm that takes a public key pk and a message $m \in \mathcal{M}$ as input and outputs a ciphertext c. If necessary, we make the encryption algorithm explicit by writing $\mathsf{Enc}(pk, m; r)$, where $r \in \mathcal{R}$ denotes the used randomness.
- $\mathsf{Dec}(sk, c)$: The decryption algorithm Dec is a deterministic algorithm that takes a secret key sk and a ciphertext c as input and outputs a message $m \in \mathcal{M}$.

Correctness. We say that PKE has a (worst-case) correctness error δ [16] if

$$\mathbb{E}\left[\max_{m\in\mathcal{M}} \Pr[\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) \neq m]\right] \leq \delta,$$

where the expectation is taken over $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$ and the choice of the random oracles involved (if any). We say that PKE has an average-case correctness error δ relative to the distribution $\psi_\mathcal{M}$ over $\mathcal{M}$ if

$$\mathbb{E}\left[\Pr\left[\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) \neq m\right]\right] \leq \delta,$$

where the expectation is taken over $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$, the choice of the random oracles involved (if any), and $m \leftarrow \psi_\mathcal{M}$.

Injectivity. [16, 5] We say that PKE is μ-injective if for all $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$ and $m, m' \in \mathcal{M}$ and $r, r' \in \mathcal{R}$, we have that

$$\Pr[c = c' \wedge (m, r) \neq (m', r') \mid c \leftarrow \mathsf{Enc}(pk, m; r) \wedge c' \leftarrow \mathsf{Enc}(pk, m'; r')] \leq \mu,$$

where the probability is taken over $c \leftarrow \mathsf{Enc}(pk, m; r)$ and $c' \leftarrow \mathsf{Enc}(pk, m'; r')$.

Spreadness. For $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$ and $m \in \mathcal{M}$, we define the min-entropy [13] of $\mathsf{Enc}(pk, m)$ as

$$\gamma(pk, m) := -\log \max_{c\in\mathcal{C}} \Pr_{r\leftarrow\psi_\mathcal{R}}[c = \mathsf{Enc}(pk, m; r)].$$

Then, we say that PKE is γ-spread [13] if for every key pair $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$ and every message $m \in \mathcal{M}$,

$$\gamma(pk, m) \geq \gamma.$$

In particular, this implies that for every possible ciphertext $c \in \mathcal{C}$, $\Pr_{r\leftarrow\psi_\mathcal{R}}[c = \mathsf{Enc}(pk, m; r)] \leq 2^{-\gamma}$.

Randomness Recoverability. We say that PKE is randomness-recoverable (RR) if there exists an algorithm RRec such that for all $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$, $m \in \mathcal{M}$, and $r \in \mathcal{R}$, we have that

$$\Pr\left[\forall m' \in \mathsf{Pre}^m(pk, c) : \mathsf{RRec}(pk, m', c) \notin \mathcal{R} \vee \mathsf{Enc}(pk, m'; \mathsf{RRec}(pk, m', c)) \neq c | c \leftarrow \mathsf{Enc}(pk, m; r)\right] = 0,$$

where the probability is taken over $c \leftarrow \mathsf{Enc}(pk, m; r)$ and $\mathsf{Pre}^m(pk, c) := \{m \in \mathcal{M} | \, \exists \, r \in \mathcal{R} : \mathsf{Enc}(pk, m; r) = c\}$. Additionally, it is required that RRec returns $\perp$ if $\mathsf{RRec}(pk, m', c) \notin \mathcal{R}$ or $\mathsf{Enc}(pk, m'; \mathsf{RRec}(pk, m', c)) \neq c$.

Message Recoverability. We say that PKE is message-recoverable (MR) if there exists an algorithm MRec such that for all $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$, $m \in \mathcal{M}$, and $r \in \mathcal{R}$, we have that

$$\Pr\left[\forall r' \in \mathsf{Pre}^r(pk, c) : \mathsf{MRec}(pk, r', c) \notin \mathcal{M} \vee \mathsf{Enc}(pk, \mathsf{MRec}(pk, r', c); r') \neq c | c \leftarrow \mathsf{Enc}(pk, m; r)\right] = 0,$$

where the probability is taken over $c \leftarrow \mathsf{Enc}(pk, m; r)$ and $\mathsf{Pre}^r(pk, c) := \{r \in \mathcal{R} | \exists \, m \in \mathcal{M} : \mathsf{Enc}(pk, m; r) = c\}$. Additionally, it is required that MRec returns $\perp$ if $\mathsf{MRec}(pk, r', c) \notin \mathcal{M}$ or $\mathsf{Enc}(pk, \mathsf{MRec}(pk, r', c); r') \neq c$.

Rigidity. Under the assumption that PKE is RR, we say that PKE is δ-rigid if for all $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$, $m \in \mathcal{M}$, and $r \in \mathcal{R}$, we have

$$\Pr\left[\mathsf{Enc}\big(pk, \mathsf{Dec}(sk, c); \mathsf{RRec}(pk, \mathsf{Dec}(sk, c), c)\big) \neq c | c \leftarrow \mathsf{Enc}(pk, m; r)\right] \leq \delta,$$

where the probability is taken over $c \leftarrow \mathsf{Enc}(pk, m; r)$.

2.2 Security

Definition 2.2 (OW-CPA Security of PKE)**.** Let PKE = (Gen, Enc, Dec) be a public key encryption scheme with message space $\mathcal{M}$. Onewayness under chosen-plaintext attacks (OW-CPA) for message distribution $\psi_\mathcal{M}$ is defined via the game OW-CPA, which is shown in Figure 2, and the advantage function of adversary $\mathcal{A}$ is

$$\mathsf{Adv}^{\mathsf{OW\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A}) := \Pr\left[\mathsf{OW\text{-}CPA}^{\mathcal{A}}_{\mathsf{PKE}} \Rightarrow 1\right].$$

Definition 2.3 (IND-CPA Security of PKE)**.** Let PKE = (Gen, Enc, Dec) be a public key encryption scheme with message space $\mathcal{M}$. Indistinguishability under chosen-plaintext attacks (IND-CPA) is defined via the game IND-CPA, as shown in Figure 2, and the advantage function of adversary $\mathcal{A}$ is

$$\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A}) := \left|\Pr\left[\mathsf{IND\text{-}CPA}^{\mathcal{A}}_{\mathsf{PKE}} \Rightarrow 1\right] - \frac{1}{2}\right|.$$

```
Game OW-CPA
  1: $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$
  2: $m \leftarrow \psi_\mathcal{M}$
  3: $c^* \leftarrow \mathsf{Enc}(pk, m)$
  4: $m' \leftarrow \mathcal{A}(pk, c^*)$
  5: return $[\![m = m']\!]$
```

```
Game IND-CPA
  1: $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$
  2: $(m_0, m_1) \leftarrow \mathcal{A}_0(pk)$
  3: $b \leftarrow \{0, 1\}$
  4: $c^* \leftarrow \mathsf{Enc}(pk, m_b)$
  5: $b' \leftarrow \mathcal{A}_1(pk, c^*)$
  6: return $[\![b = b']\!]$
```

Figure 2: Game OW-CPA and Game IND-CPA for PKE

2.3 Key Encapsulation Mechanism

Definition 2.4 (Key Encapsulation Mechanism)**.** A key encapsulation mechanism $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Encap}, \mathsf{Decap})$ with a key space $\mathcal{K}$ consists of the following three algorithms:

- $\mathsf{Gen}(1^\lambda)$: The key generation algorithm Gen is a randomized algorithm that takes a security parameter λ as input and outputs a pair of public key and secret key, (pk, sk).
- $\mathsf{Encap}(pk)$: The encapsulation algorithm Encap is a randomized algorithm that takes a public key pk as input, and outputs a ciphertext c and a key $K \in \mathcal{K}$.
- $\mathsf{Decap}(sk, c)$: The decryption algorithm Decap is a deterministic algorithm that takes a secret key sk and ciphertext c as input, and outputs a key $K \in \mathcal{K}$.

Correctness. We say that KEM has a correctness error δ if

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

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

Definition 2.5 (IND-CCA Security of KEM)**.** Let $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Encap}, \mathsf{Decap})$ be a key encapsulation mechanism with a key space $\mathcal{K}$. Indistinguishability under chosen-ciphertext attacks (IND-CCA) is defined via the game IND-CCA, as shown in Figure 3, and the advantage function of adversary $\mathcal{A}$ is as follows:

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

```
Game IND-CCA
  1: $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$
  2: $(K_0, c^*) \leftarrow \mathsf{Encap}(pk)$
  3: $K_1 \leftarrow \mathcal{K}$
  4: $b \leftarrow \{0, 1\}$
  5: $b' \leftarrow \mathcal{A}^{\mathsf{Decap}}(pk, c^*, K_b)$
  6: return $[\![b = b']\!]$

Decap($c \neq c^*$)
  1: return $\mathsf{Decap}(sk, c)$
```

Figure 3: Game IND-CCA for KEM

3 ACWC$_2$ Transformation

We introduce our new ACWC transformation ACWC_2 by describing $\mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$ for a hash function G, as shown in Figure 4. Let $\mathsf{PKE}' = \mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$ be the resulting encryption scheme. By applying ACWC_2 to an underlying PKE, we prove that (1) PKE' has a worst-case correctness error that is essentially close to the average-case error of PKE, and (2) PKE' is tightly IND-CPA secure if PKE is OW-CPA secure.

3.1 SOTP

Definition 3.1. Function $\mathsf{SOTP} : \mathcal{X} \times \mathcal{U} \to \mathcal{Y}$ is called a semi-generalized one-time pad (relative to distributions $\psi_{\mathcal{U}}$ and $\psi_{\mathcal{Y}}$) if

1. Decoding: There exists an efficient algorithm Inv such that for all $x \in \mathcal{X}$, $u \in \mathcal{U}$, $\mathsf{Inv}(\mathsf{SOTP}(x,u),u) = x$.
2. Message-hiding: For all $x \in \mathcal{X}$, the random variable $\mathsf{SOTP}(x,u)$, for $u \leftarrow \psi_{\mathcal{U}}$, has the same distribution as $\psi_{\mathcal{Y}}$.
3. Rigid: For all $u \in \mathcal{U}$ and all $y \in \mathcal{Y}$ encoded with respect to u, it holds that $\mathsf{SOTP}(\mathsf{Inv}(y,u),u) = y$.

In contrast to the GOTP defined in [12], SOTP does not need to have an additional *randomness-hiding* property, which requires that the output $y = \mathsf{SOTP}(x,u)$ follows the distribution $\psi_{\mathcal{Y}}$ and simultaneously does not leak any information about the randomness u. The absence of such an additional property allows us to design SOTP more flexibly and efficiently than GOTP. Instead, SOTP is required to be *δ_s-rigid*, which means that for all $u \in \mathcal{U}$ and all $y \in \mathcal{Y}$ encoded with respect to u, $\mathsf{Inv}(y,u) = x$ implies that $\mathsf{SOTP}(x,u) = y$, except with a probability of at most δ_s.

3.2 ACWC$_2$

Let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an underlying public key encryption scheme with message space $\mathcal{M}$ and randomness space $\mathcal{R}$, where a message $M \in \mathcal{M}$ and randomness $r \in \mathcal{R}$ are drawn from the distributions $\psi_{\mathcal{M}}$ and $\psi_{\mathcal{R}}$, respectively. Similarly, let $\mathsf{PKE}' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ be a transformed encryption scheme with message space $\mathcal{M}'$ and randomness space $\mathcal{R}'$, where $\psi_{\mathcal{M}'}$ and $\psi_{\mathcal{R}'}$ are associated distributions. Let $\mathsf{SOTP} : \mathcal{M}' \times \mathcal{U} \to \mathcal{M}$ be a semi-generalized one-time pad for distributions $\psi_{\mathcal{U}}$ and $\psi_{\mathcal{M}}$, and let $\mathsf{G} : \mathcal{R} \to \mathcal{U}$ be a hash function such that every output is independently $\psi_{\mathcal{U}}$-distributed. Assuming that $\mathcal{R} = \mathcal{R}'$ and $\psi_{\mathcal{R}} = \psi_{\mathcal{R}'}$, then $\mathsf{PKE}' = \mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$ is described in Figure 4.

```
Gen'(1^λ)
  1: $(pk, sk) := \mathsf{Gen}(1^\lambda)$
  2: return $(pk, sk)$

Enc'($pk, m \in \mathcal{M}'; r \leftarrow \psi_{\mathcal{R}}$)
  1: $M := \mathsf{SOTP}(m, \mathsf{G}(r))$
  2: $c := \mathsf{Enc}(pk, M; r)$
  3: return $c$

Dec'($sk, c$)
  1: $M := \mathsf{Dec}(sk, c)$
  2: $r := \mathsf{RRec}(pk, M, c)$
  3: $m := \mathsf{Inv}(M, \mathsf{G}(r))$
  4: return $m$
```

Figure 4: $\mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$

Under the condition that $\mathsf{Dec}(sk,c)$ in Dec' yields the same M as in Enc, the deterministic RRec and Inv functions do not affect the correctness error of PKE'. Thus, the factor that determines the success or failure of $\mathsf{Dec}'(\mathsf{sk}, \mathsf{c})$ is the result of $\mathsf{Dec}(sk,c)$ in Dec'. This means that the correctness error of PKE is straightforwardly transferred to that of PKE', and eventually determined by how randomness $r \in \mathcal{R}$ and message $M \in \mathcal{M}$ are sampled in PKE'. We see that r is drawn according to the distribution $\psi_{\mathcal{R}}$ and M is an SOTP-encoded element in $\mathcal{M}$. Because every output of G is independently $\psi_{\mathcal{U}}$-distributed, we can expect that the message-hiding property of SOTP makes M follow the distribution $\psi_{\mathcal{M}}$ while hiding m. Eventually, both M and r are chosen according to their respective initially-intended distributions.

However, since the choice of the random oracle G can affect the correctness error of PKE', we need to include this observation in the analysis of the correctness error. Theorem 3.2 shows that for all but a negligible fraction of random oracles G, the worst-case correctness of PKE' (transformed by ACWC_2) is close to the average-case correctness of PKE. This is the same idea as in ACWC, and the proof strategy of Theorem 3.2 is essentially the same as that of [12] (Lemma 3.6 therein), except for slight modifications to the message distribution.

Theorem 3.2 (Average-Case to Worst-Case Correctness error)**.** Let PKE be RR and have a randomness space $\mathcal{R}$ relative to the distribution $\psi_{\mathcal{R}}$. Let $\mathsf{SOTP} : \mathcal{M}' \times \mathcal{U} \to \mathcal{M}$ be a semi-generalized one-time pad (for distributions $\psi_{\mathcal{U}}$, $\psi_{\mathcal{M}}$), and let $\mathsf{G} : \mathcal{R} \to \psi_{\mathcal{U}}$ be a random oracle. If PKE is δ-average-case-correct, then $\mathsf{PKE}' := \mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$ is δ'-worst-case-correct for

$$\delta' = \delta + \|\psi_{\mathcal{R}}\| \cdot \left(1 + \sqrt{(\ln|\mathcal{M}'| - \ln\|\psi_{\mathcal{R}}\|)/2}\right),$$

where $\|\psi_{\mathcal{R}}\| := \sqrt{\sum_r \psi_{\mathcal{R}}(r)^2}$.

Proof. With the expectation over the choice of G and $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$, the worst-case correctness of the PKE' is

$$\begin{aligned}\delta' &= \mathbb{E}\left[\max_{m\in\mathcal{M}'} \Pr[\mathsf{Dec}'(sk, \mathsf{Enc}'(pk, m)) \neq m]\right]\\ &= \mathbb{E}[\delta'(pk, sk)],\end{aligned}$$

where $\delta'(pk, sk) := \mathbb{E}[\max_{m\in\mathcal{M}'} \Pr[\mathsf{Dec}'(sk, \mathsf{Enc}'(pk, m)) \neq m]$ is the expectation taken over the choice of G, for a fixed key pair (pk, sk). For any fixed key pair and any positive real $t \in \mathbb{R}^+$, we have

$$\begin{aligned}\delta'(pk, sk) &= \mathbb{E}[\max_{m\in\mathcal{M}'} \Pr\left[\mathsf{Dec}'(sk, \mathsf{Enc}'(pk, m)) \neq m\right]]\\ &\leq t + \Pr_{\mathsf{G}}\left[\max_{m\in\mathcal{M}'} \Pr[\mathsf{Dec}'(sk, \mathsf{Enc}'(pk, m)) \neq m] \geq t\right]\\ &\leq t + \Pr_{\mathsf{G}}\left[\max_{m\in\mathcal{M}'} \Pr_r[\mathsf{Dec}'(sk, \mathsf{Enc}(pk, \tilde{m}); r) \neq m] \geq t\right], \qquad (1)\end{aligned}$$

where $\tilde{m} = \mathsf{SOTP}(m, \mathsf{G}(r))$. Note that the first inequality holds by Lemma 3.3.

For any fixed key pair and any real c, let $t(pk, sk) := \mu(pk, sk) + \|\psi_{\mathcal{R}}\| \cdot \sqrt{(c + \ln|\mathcal{M}'|)/2}$, where $\mu(pk, sk) := \Pr_{M,r}[\mathsf{Dec}(sk, \mathsf{Enc}(pk, M; r)) \neq M]$. Then, we can use the helper Lemma 3.4 to argue that

$$\begin{aligned}&\Pr_{\mathsf{G}}\left[\max_{m\in\mathcal{M}'} \Pr_r[\mathsf{Dec}'(sk, \mathsf{Enc}(pk, \tilde{m}; r)) \neq m] > t(pk, sk)\right]\\ &\qquad\qquad \leq e^{-c}. \qquad (2)\end{aligned}$$

To this end, we define $g(m, r, u)$ and B as $g(m, r, u) = (\mathsf{SOTP}(m, u), r)$ and $B = \{(M, r) \in |\mathsf{Dec}(sk, \mathsf{Enc}(pk, M; r)) \neq M\}$, which will be used in Lemma 3.4. Note that $\Pr_{r\leftarrow\psi_{\mathcal{R}}, u\leftarrow\psi_{\mathcal{U}}}[g(m, r, u) \in$

$B] = \mu(pk, sk)$ holds for all $m \in \mathcal{M}'$ by the message-hiding property of the SOTP. For all $m \in \mathcal{M}'$,

$$
\begin{aligned}
&\Pr_{r \leftarrow \psi_{\mathcal{R}}, u \leftarrow \psi_{\mathcal{U}}}[g(m,r,u) \in B] \\
&\quad = \Pr_{r \leftarrow \psi_{\mathcal{R}}, u \leftarrow \psi_{\mathcal{U}}}[(\mathsf{SOTP}(m,u), r) \in B] \\
&\quad = \Pr_{r \leftarrow \psi_{\mathcal{R}}, M \leftarrow \psi_{\mathcal{M}}}[(M, r) \in B] \\
&\quad = \Pr_{r \leftarrow \psi_{\mathcal{R}}, M \leftarrow \psi_{\mathcal{M}}}[\mathsf{Dec}(sk, \mathsf{Enc}(pk, M; r) \neq M] \\
&\quad = \mu(pk, sk).
\end{aligned}
$$

Combining Equation (2) with Equation (1) and taking the expectation yields

$$
\begin{aligned}
\delta' &\leq \mathbb{E}\left[\mu(pk, sk) + \|\psi_{\mathcal{R}}\| \cdot \sqrt{(c + \ln|\mathcal{M}'|)/2} + e^{-c}\right] \\
&= \delta + \|\psi_{\mathcal{R}}\| \cdot \sqrt{(c + \ln|\mathcal{M}'|)/2} + e^{-c},
\end{aligned}
$$

and setting $c := -\ln\|\psi_{\mathcal{R}}\|$ yields the claim in the theorem. □

Lemma 3.3. Let X be a random variable and let f be a non-negative real-valued function with $f(X) \leq 1$. Then,

$$\mathbb{E}[f(X)] \leq t + \Pr[f(X) \geq t]$$

for all positive real $t \in \mathbb{R}^+$.

Proof. By using the law of total probability and by partitioning all possible values of x into conditions satisfying either $f(x) < t$ or $f(x) \geq t$, we can achieve the required inequality as follows:

$$
\begin{aligned}
&\mathbb{E}[f(X)] = \sum f(x)\Pr[X = x] \\
&= \sum_{f(x)<t} f(x)\Pr[X = x] + \sum_{f(x)\geq t} f(x)\Pr[X = x] \\
&\leq \sum_{f(x)<t} t\Pr[X = x] + \sum_{f(x)\geq t} f(x)\Pr[X = x] \\
&\leq t + \sum_{f(x)\geq t} f(x)\Pr[X = x] \\
&\leq t + \sum_{f(x)\geq t} \Pr[X = x] = t + \Pr[f(X) \geq t]
\end{aligned}
$$

The last equality can be checked by $\sum_{f(x)\geq t}\Pr[X = x] = \Pr[f(X) \geq t]$. □

Lemma 3.4 (Adapting Lemma 3.7 from [12])**.** Let g be a function, and B be some set such that

$$\forall m \in \mathcal{M}', \Pr_{r \leftarrow \psi_{\mathcal{R}}, u \leftarrow \psi_{\mathcal{U}}}[g(m,r,u) \in B] \leq \mu \tag{3}$$

for some $\mu \in [0,1]$. Let $\mathsf{G} : \mathcal{R} \to \mathcal{U}$ be a random function such that every output is independently $\psi_{\mathcal{U}}$-distributed. Define $\|\psi_{\mathcal{R}}\| = \sqrt{\sum_r \psi_{\mathcal{R}}(r)^2}$. Then, for all but an e^{-c} fraction of random functions G, we have that $\forall m \in \mathcal{M}'$,

$$\Pr_{r \leftarrow \psi_{\mathcal{R}}}[g(m,r,\mathsf{G}(r)) \in B] \\ \leq \mu + \|\psi_{\mathcal{R}}\| \cdot \sqrt{(c + \ln |\mathcal{M}'|)/2} \tag{4}$$

for some positive $c \in \mathbb{R}^+$.

Proof. Let us fix a specific $m \in \mathcal{M}'$, and for each $r \in \mathcal{R}$, define $p_r := \Pr_{u \leftarrow \psi_{\mathcal{U}}}[g(m,r,u) \in B]$. By the assumption of g in Equation (3), we know that $\sum_r \psi_{\mathcal{R}}(r) p_r \leq \mu$. For each r, define a random variable X_r whose value is determined as follows: G chooses a random $u = \mathsf{G}(r)$ and then checks whether $g(m,r,\mathsf{G}(r)) \in B$; if it does, then we set $X_r = 1$; otherwise we set it to zero. Because G is a random function, the probability that $X_r = 1$ is exactly p_r.

The probability of Equation (4) for our particular m is the same as the sum $\sum_r \psi_{\mathcal{R}}(r) X_r$, and we use the Hoeffding bound to show that this value is not significantly larger than μ. We define the random variable $Y_r = \psi_{\mathcal{R}}(r) X_r$. Notice that $Y_r \in [0, \psi_{\mathcal{R}}(r)]$, and $\mathbb{E}[\sum Y_r] = \mathbb{E}[\sum_r \psi_{\mathcal{R}}(r) X_r] = \sum_r \psi_{\mathcal{R}}(r) p_r \leq \mu$. By the Hoeffding bound, we have for all positive t,

$$\Pr[\sum_r Y_r > \mu + t] \leq \exp\left(\frac{-2t^2}{\sum \psi_{\mathcal{R}}(r)^2}\right) = \exp\left(\frac{-2t^2}{\|\psi_{\mathcal{R}}\|^2}\right). \tag{5}$$

By setting $t \geq \|\psi\| \cdot \sqrt{(c + \ln |\mathcal{M}'|)/2}$, for a fixed m, Equation (4) holds for all but an $e^{-c} \cdot |\mathcal{M}'|^{-1}$ fraction of random functions G. Applying the union bound yields the claim in the lemma. $\square$

Theorem 3.5 (OW-CPA of PKE $\overset{\text{ROM}}{\Longrightarrow}$ IND-CPA of $\mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$)**.** Let PKE be a public key encryption scheme with RR and MR properties and be μ-injective. For any adversary $\mathcal{A}$ against the IND-CPA security of $\mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$, making at most q_{G} random oracle queries, there exists an adversary $\mathcal{B}$ against the OW-CPA security of PKE with

$$\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{ACWC}_2[\mathsf{PKE},\mathsf{SOTP},\mathsf{G}]}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{OW\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{B}) + \mu,$$

where the running time of $\mathcal{B}$ is about $\mathrm{Time}(\mathcal{A}) + O(q_{\mathsf{G}})$.

```
Game $G_0$
1: $\mathsf{G} \leftarrow (\mathcal{R} \to \mathcal{U})$
2: $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$
3: $(m_0, m_1) \leftarrow \mathcal{A}_0^{\mathsf{G}}(pk)$
4: $b \leftarrow \{0,1\}$
5: $r^* \leftarrow \psi_{\mathcal{R}}$
6: $M^* = \mathsf{SOTP}(m_b, \mathsf{G}(r^*))$
7: $c^* \leftarrow \mathsf{Enc}(pk, M^*; r^*)$
8: $b' \leftarrow \mathcal{A}_1^{\mathsf{G}}(pk, c^*)$
9: return $[\![b = b']\!]$
```

Figure 5: Game G_0 of Theorems 3.5 and 3.7

```
$\underline{\mathcal{B}(pk, c^*)}$
1: $\mathcal{L}_{\mathsf{G}}, \mathcal{L}_{\mathsf{r}} := \emptyset$
2: $b \leftarrow \{0,1\}$
3: $(m_0, m_1) \leftarrow \mathcal{A}_0^{\mathsf{G}}(pk)$
4: $b' \leftarrow \mathcal{A}_1^{\mathsf{G}}(pk, c^*)$
5: for $r \in \mathcal{L}_r$ do
6:     $M := \mathsf{MRec}(pk, r, c^*)$
7:     if $M \in \mathcal{M}$
8:         return $M$
9: return $M \leftarrow \psi_{\mathcal{M}}$
```

```
$\underline{\mathsf{G}(r)}$
1: if $\exists(r, u) \in \mathcal{L}_{\mathsf{G}}$
2:     return $u$
3: else
4:     $u \leftarrow \psi_{\mathcal{U}}$
5:     $\mathcal{L}_{\mathsf{G}} := \mathcal{L}_{\mathsf{G}} \cap \{(r, u)\}$
6:     $\mathcal{L}_{\mathsf{r}} := \mathcal{L}_{\mathsf{r}} \cap \{r\}$
7: return $u$
```

Figure 6: Adversary $\mathcal{B}$ for the proof of Theorem 3.5

Proof. We show that there exists an algorithm $\mathcal{B}$ (see Figure 6) which breaks the $\mathsf{OW\text{-}CPA}$ security of PKE using an algorithm $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ that breaks the $\mathsf{IND\text{-}CPA}$ security of $\mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$.

GAME G_0. G_0 (see Figure 5) is the original $\mathsf{IND\text{-}CPA}$ game with $\mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$. In G_0, $\mathcal{A}$ is given the challenge ciphertext $c^* := \mathsf{Enc}(pk, M^*; r^*)$ for some unknown message M^* and randomness r^*. By the definition of the $\mathsf{IND\text{-}CPA}$ game, we have

$$\left|\Pr[G_0^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right| = \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{ACWC}_2[\mathsf{PKE},\mathsf{SOTP},\mathsf{G}]}(\mathcal{A}).$$

GAME G_1. G_1 is the same as G_0, except that we abort G_1 when $\mathcal{A}$ queries two distinct r_1^* and r_2^* to G, such that $\mathsf{MRec}(pk, r_1^*, c^*)$ and $\mathsf{MRec}(pk, r_2^*, c^*) \in \mathcal{M}$. This leads to breaking the injectivity of the PKE. Thus, we have

$$\left|\Pr[G_1^{\mathcal{A}} \Rightarrow 1] - \Pr[G_0^{\mathcal{A}} \Rightarrow 1]\right| \le \mu.$$

GAME G_2. Let QUERY be an event that $\mathcal{A}$ queries G on r^*. G_2 is the same as G_1, except that we abort G_2 in the QUERY event. In this case, we have

$$\left|\Pr[G_2^{\mathcal{A}} \Rightarrow 1] - \Pr[G_1^{\mathcal{A}} \Rightarrow 1]\right| \le \Pr[\mathsf{QUERY}].$$

Unless QUERY occurs, $\mathsf{G}(r^*)$ is a uniformly random value that is independent of $\mathcal{A}$s view. In this case, $M^* := \mathsf{SOTP}(m_b, \mathsf{G}(r^*))$ does not leak any information about m_b by the message-hiding property of the SOTP, meaning that $\Pr[G_2^{\mathcal{A}} \Rightarrow 1] = 1/2$. By contrast, if QUERY occurs, $\mathcal{B}$ (defined in Figure 6) can find $r^* \in \mathcal{L}_r$ such that $c^* := \mathsf{Enc}(pk, M^*; r^*)$, using the algorithm MRec. Indeed, for each query r to G, $\mathcal{B}$ checks whether $\mathsf{MRec}(pk, r, c^*) \in \mathcal{M}$. In the QUERY event, there exists $M^* := \mathsf{MRec}(pk, r^*, c^*) \in \mathcal{M}$ which can be the solution to its challenge ciphertext c^*. It follows that

$$\Pr[\mathsf{QUERY}] \le \mathsf{Adv}^{\mathsf{OW\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{B}),$$

which concludes the proof. □

Lemma 3.6 (Classical O2H, Theorem 3 from the eprint version of [3])**.** Let $S \subset \mathcal{R}$ be random. Let G and F be random functions satisfying $\forall r \notin S : \mathsf{G}(r) = \mathsf{F}(r)$. Let z be a random classical value (S, G, F, z may have an arbitrary joint distribution). Let $\mathcal{C}$ be a quantum oracle algorithm with query depth q_{G}, expecting

```
Games $G_1$-$G_5$
1:  $\mathsf{G} \leftarrow (\mathcal{R} \to \mathcal{U})$          // $G_1$
2:  $r \leftarrow \mathcal{R}$
3:  $u := \mathsf{G}(r)$                                            // $G_1$
4:  $\mathsf{F} \leftarrow (\mathcal{R} \to \mathcal{U})$          // $G_2$-$G_5$
5:  $u \leftarrow \psi_{\mathcal{U}}$                               // $G_2$-$G_5$
6:  $\mathsf{G} := \mathsf{F}(r := u)$                              // $G_2$-$G_5$
7:  $w \leftarrow \mathcal{C}^{\mathsf{G}}(r, u)$                   // $G_1$-$G_2$
8:  $w \leftarrow \mathcal{C}^{\mathsf{F}}(r, u)$                   // $G_3$
9:  $T \leftarrow \mathcal{D}^{\mathsf{F}}(r, u)$                   // $G_4$-$G_5$
10: return $w$                                                      // $G_1$-$G_3$
11: return $r \in T$                                                // $G_4$-$G_5$

$\mathcal{C}^{\mathsf{G}}(r, u)$
1: $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$
2: $(m_0, m_1) \leftarrow \mathcal{A}_0^{\mathsf{G}}(pk)$
3: $b \leftarrow \{0, 1\}$                                          // $G_1$-$G_4$
4: $M = \mathsf{SOTP}(m_b, u)$                                      // $G_1$-$G_4$
5: $M \leftarrow \psi_{\mathcal{M}}$                                // $G_5$
6: $c^* \leftarrow \mathsf{Enc}(pk, M; r)$
7: $b' \leftarrow \mathcal{A}_1^{\mathsf{G}}(pk, c^*)$
8: return $[\![b = b']\!]$

$\mathcal{D}^{\mathsf{F}}(r, u)$
1: $i \leftarrow \{1, \cdots, q_{\mathsf{G}}\}$
2: Run $\mathcal{C}^{\mathsf{F}}(r, u)$ till $i$-th query
3: $T \leftarrow$ measure $\mathsf{F}$-query
4: return $T$
```

Figure 7: Games G_1-G_5 for the proof of Theorem 3.7

input z. Let $\mathcal{D}$ be the algorithm that, on input z, samples a uniform i from $\{1, ..., q_{\mathsf{G}}\}$, runs $\mathcal{C}$ right before its i-th query to F, measures all query input registers, and outputs the set T of measurement outcomes. Then

$$\left|\Pr[\mathcal{C}^{\mathsf{G}}(z) \Rightarrow 1] - \Pr[\mathcal{C}^{\mathsf{F}}(z) \Rightarrow 1]\right|$$
$$\leq 2q_{\mathsf{G}}\sqrt{\Pr[S \cap T \neq \emptyset : T \leftarrow \mathcal{D}^{\mathsf{F}}(z)]}.$$

Theorem 3.7 (OW-CPA of PKE $\overset{\mathsf{QROM}}{\Longrightarrow}$ IND-CPA of $\mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$)**.** Let PKE be a public key encryption scheme with RR and MR properties and be μ-injective. For any quantum adversary $\mathcal{A}$ against the IND-CPA security of $\mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$ with a query depth at most q_{G}, there exists a quantum adversary $\mathcal{B}$ against the OW-CPA security of PKE with

$$\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{ACWC}_2[\mathsf{PKE},\mathsf{SOTP},\mathsf{G}]}(\mathcal{A}) \leq 2q_{\mathsf{G}}\sqrt{\mathsf{Adv}^{\mathsf{OW\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{B}) + \mu},$$

and the running time of $\mathcal{B}$ is about that of $\mathcal{A}$.

Proof. To prove this theorem, we use a sequence of games G_0 to G_7 defined in Figures 5, 7, and 8, and Lemma 3.6. Before applying Lemma 3.6, we change G_0 to G_2. Subsequently, we apply Lemma 3.6 to G_2 and G_3. A detailed explanation of the security proof is provided in the following.
GAME G_0. G_0 (see Figure 5) is the original IND-CPA game with $\mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$. By definition, we have

$$\left|\Pr[G_0^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right| = \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{ACWC}_2[\mathsf{PKE},\mathsf{SOTP},\mathsf{G}]}(\mathcal{A}).$$

GAME G_1. We define G_1 by moving part of G_0 inside an algorithm $\mathcal{C}^{\mathsf{G}}$. In addition, we query $u := \mathsf{G}(r)$ before algorithm $\mathcal{C}^{\mathsf{G}}$ runs adversary $\mathcal{A}$. As the changes are only conceptual, we have

$$\Pr[G_0^{\mathcal{A}} \Rightarrow 1] = \Pr[G_1^{\mathcal{A}} \Rightarrow 1].$$

```
Game $G_6$-$G_7$
1: $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$
2: $r \leftarrow \psi_{\mathcal{R}}$
3: $M \leftarrow \psi_{\mathcal{M}}$
4: $c^* \leftarrow \mathsf{Enc}(pk, M; r)$
5: $T \leftarrow \mathcal{E}(pk, c^*)$                // $G_6$
6: $M' \leftarrow \mathcal{B}(pk, c^*)$               // $G_7$
7: return $r \in T$                       // $G_6$
8: return $[\![M = M']\!]$                 // $G_7$

$\mathcal{E}(pk, c^*)$
1: $i \leftarrow \{1, \cdots, q_{\mathsf{G}}\}$
2: Run until $i$-th $\mathsf{F}$-query:
3:     $\mathcal{A}_1^{\mathsf{F}}(pk)$
4:     $\mathcal{A}_2^{\mathsf{F}}(pk, c^*)$
5: $T \leftarrow$measure $\mathsf{F}$-query
6: return $T$

$\mathcal{B}(pk, c^*)$
1: $T \leftarrow \mathcal{E}(pk, c^*)$
2: for $r \in T$ do
3:     if $M = \mathsf{MRec}(pk, r, c^*) \in \mathcal{M}$
4:         return $M$
5: return $M \leftarrow \psi_{\mathcal{M}}$
```

Figure 8: Games G_6-G_7 for the proof of Theorem 3.7

GAME G_2. We change the way G is defined in G_2. Rather than choosing G uniformly, we choose F and u uniformly and then set $\mathsf{G} := \mathsf{F}(r := u)$. Here, $\mathsf{G} = \mathsf{F}(r := u)$ is the same function as F, except that it returns u on input r. Because the distributions of G and u remain unchanged, we have

$$\Pr[G_1^{\mathcal{A}} \Rightarrow 1] = \Pr[G_2^{\mathcal{A}} \Rightarrow 1].$$

GAME G_3. We define G_3 by providing function F to algorithm $\mathcal{C}$ instead of G. By applying Lemma 3.6 with $\mathcal{C}$, $S := \{r\}$, and $z := (r, u)$, we obtain the following:

$$\left|\Pr[G_2^{\mathcal{A}} \Rightarrow 1] - \Pr[G_3^{\mathcal{A}} \Rightarrow 1]\right| \leq 2q_{\mathsf{G}}\sqrt{\Pr[G_4 \Rightarrow 1]}.$$

In addition, since the uniformly random value u is only used in the $\mathsf{SOTP}(m_b, u)$, by the message-hiding property of the SOTP, M is independent of m_b. Thus, $b = b'$ with a probability of $1/2$. Therefore,

$$\Pr[G_3^{\mathcal{A}} \Rightarrow 1] = \frac{1}{2}.$$

GAME G_4 and G_5. We define G_4 according to Lemma 3.6. In addition, we define G_5 by changing the way M is calculated. Instead of computing $M = \mathsf{SOTP}(m_b, u)$, we sample $M \leftarrow \psi_{\mathcal{M}}$. By contrast, in G_4, since u is sampled from $\psi_{\mathcal{U}}$ and used only for computing $\mathsf{SOTP}(m_b, u)$, the message-hiding property of SOTP shows that $M = \mathsf{SOTP}(m_b, u)$ follows the distribution $\psi_{\mathcal{M}}$. Therefore,

$$\Pr[G_4^{\mathcal{A}} \Rightarrow 1] = \Pr[G_5^{\mathcal{A}} \Rightarrow 1].$$

GAME G_6. We define G_6 by rearranging G_5, as shown in Figure 8. As the changes are only conceptual, we have

$$\Pr[G_5^{\mathcal{A}} \Rightarrow 1] = \Pr[G_6^{\mathcal{A}} \Rightarrow 1].$$

GAME G_7. G_7 is defined by Algorithm $\mathcal{B}$, as shown in Figure 8, moving from G_6. G_7 is the same as G_6, except for the case in which there are two distinct $r, r' \in T$ such that $\mathsf{MRec}(pk, r, c^*)$, $\mathsf{MRec}(pk, r', c^*) \in \mathcal{M}$. If this occurs, the injectivity of PKE is broken. Thus, we have

$$\left|\Pr[G_6^{\mathcal{A}} \Rightarrow 1] - \Pr[G_7^{\mathcal{A}} \Rightarrow 1]\right| \leq \mu.$$

We can observe that in G_7, $\mathcal{B}$ wins if there exists $r \in T$ such that $m^* := \mathsf{MRec}(pk, r, c^*) \in \mathcal{M}$, as the solution of its challenge ciphertext c^*. Therefore, we have

$$\mathsf{Adv}_{\mathsf{PKE}}^{\mathsf{OW\text{-}CPA}}(\mathcal{B}) = \Pr[G_7^{\mathcal{A}} \Rightarrow 1].$$

Combining all (in)equalities and bounds, we have

$$\mathsf{Adv}_{\mathsf{ACWC_2[PKE,SOTP,G]}}^{\mathsf{IND\text{-}CPA}}(\mathcal{A}) \leq 2q_{\mathsf{G}}\sqrt{\mathsf{Adv}_{\mathsf{PKE}}^{\mathsf{OW\text{-}CPA}}(\mathcal{B}) + \mu},$$

which concludes the proof. □

Lemma 3.8. If PKE is γ-spread, then so is $\mathsf{PKE}' = \mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$.

Proof. For a fixed key pair (pk, sk) and a fixed m (with respect to PKE'), we consider the probability that $\Pr_{r \leftarrow \psi_{\mathcal{R}}}[c = \mathsf{Enc}'(pk, m; r)]$ for every possible ciphertext c. Whenever $r \leftarrow \psi_{\mathcal{R}}$, the equation $c = \mathsf{Enc}'(pk, m; r)$ is equivalently transformed into $c = \mathsf{Enc}(pk, M; r)$, where $M = \mathsf{SOTP}(m, \mathsf{G}(r))$ is a message and c is a possible ciphertext with respect to PKE. Since PKE is γ-spread, we observe that $\Pr_{r \leftarrow \psi_{\mathcal{R}}}[c = \mathsf{Enc}(pk, M; r)] \leq 2^{-\gamma}$, which yields $\Pr_{r \leftarrow \psi_{\mathcal{R}}}[c = \mathsf{Enc}'(pk, m; r)] \leq 2^{-\gamma}$. By averaging over (pk, sk) and $m \in \mathcal{M}'$, the proof is completed. □

4 IND-CCA Secure KEM from ACWC$_2$

4.1 FO Transform with Re-encryption

One can apply the Fujisaki-Okamoto transformation $\mathsf{FO}^{\perp}$ to the IND-CPA secure PKE', as shown in Figure 4, to obtain an IND-CCA secure KEM. Figure 9 shows the resultant $\mathsf{KEM} := \mathsf{FO}^{\perp}[\mathsf{PKE}', \mathsf{H}] = (\mathsf{Gen}, \mathsf{Encap}, \mathsf{Decap})$, where H is a hash function (modeled as a random oracle). Regarding the correctness error of KEM, KEM preserves the worst-case correctness error of PKE', as Decap works correctly as long as Dec' is performed correctly. Regarding the IND-CCA security of KEM, we can use the previous results [16] and [11], which are stated in Theorems 4.1 and 4.2, respectively. By combining these results with Theorems 3.5 and 3.7, we can achieve the IND-CCA security of KEM in the classical/quantum random oracle model. In the case of the quantum random oracle model (QROM), we need to further use the fact that IND-CPA generically implies OW-CPA.

```
Gen(1^λ)
  1: $(pk, sk) := \mathsf{Gen}'(1^\lambda)$
  2: return $(pk, sk)$

Encap(pk)
  1: $m \leftarrow \mathcal{M}$
  2: $(r, K) := \mathsf{H}(m)$
  3: $c := \mathsf{Enc}'(pk, m; r)$
       - $M := \mathsf{SOTP}(m, \mathsf{G}(r))$
       - $c := \mathsf{Enc}(pk, M; r)$
  4: return $(K, c)$

Decap(sk, c)
  1: $m' := \mathsf{Dec}'(sk, c)$
       - $M' = \mathsf{Dec}(sk, c)$
       - $r' = \mathsf{RRec}(pk, M', c)$
       - $m' = \mathsf{Inv}(M', \mathsf{G}(r'))$
  2: $(r'', K') := \mathsf{H}(m')$
  3: if $m' = \perp$ or $\boxed{c \neq \mathsf{Enc}'(pk, m'; r'')}$
  4:     return $\perp$
  5: else
  6:     return $K'$
```

Figure 9: $\mathsf{KEM} = \mathsf{FO}^{\perp}[\mathsf{PKE}', \mathsf{H}]$

Theorem 4.1 ($\mathsf{IND\text{-}CPA}$ of $\mathsf{PKE}' \overset{\mathsf{ROM}}{\Longrightarrow}$ $\mathsf{IND\text{-}CCA}$ of KEM [16]). Let PKE' be a public key encryption scheme with a message space $\mathcal{M}$. Let PKE' has (worst-case) correctness error δ and is (weakly) γ-spread. For any adversary $\mathcal{A}$ making at most q_D decapsulation and q_H hash queries, against the $\mathsf{IND\text{-}CCA}$ security of KEM, there exists an adversary $\mathcal{B}$ against the $\mathsf{IND\text{-}CPA}$ security of PKE' with

$$\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}}(\mathcal{A}) \le 2(\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}'}(\mathcal{B}) + \frac{q_\mathsf{H}}{|\mathcal{M}|}) + q_\mathsf{D}2^{-\gamma} + q_\mathsf{H}\delta,$$

where the running time of $\mathcal{B}$ is about that of $\mathcal{A}$.

Theorem 4.2 ($\mathsf{OW\text{-}CPA}$ of $\mathsf{PKE}' \overset{\mathsf{QROM}}{\Longrightarrow}$ $\mathsf{IND\text{-}CCA}$ of KEM [11]). Let PKE' have (worst-case) correctness error δ and be (weakly) γ-spread. For any quantum adversary $\mathcal{A}$, making at most q_D decapsulation and q_H (quantum) hash queries against the $\mathsf{IND\text{-}CCA}$ security of KEM, there exists a quantum adversary $\mathcal{B}$ against the $\mathsf{OW\text{-}CPA}$ security of PKE' with

$$\begin{aligned}\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}}(\mathcal{A}) \le & 2q\sqrt{\mathsf{Adv}^{\mathsf{OW\text{-}CPA}}_{\mathsf{PKE}'}(\mathcal{B})} \\ & + 24q^2\sqrt{\delta} + 24q\sqrt{qq_D}\cdot 2^{-\gamma/4},\end{aligned}$$

where $q := 2(q_\mathsf{H} + q_\mathsf{D})$ and $\mathbf{Time}(\mathcal{B}) \approx \mathbf{Time}(\mathcal{A}) + O(q_\mathsf{H}\cdot q_\mathsf{D}\cdot \mathbf{Time}(\mathsf{Enc}) + q^2)$.

4.2 FO-Equivalent Transform Without Re-encryption

The aforementioned $\mathsf{FO}^\perp$ requires the Decap algorithm to perform re-encryption to check if ciphertext c is well-formed. Using m' as the result of $\mathsf{Dec}'(sk, c)$, a new randomness r'' is obtained from $\mathsf{H}(m')$, and $\mathsf{Enc}'(pk, m'; r'')$ is computed and compared with the (decrypted) ciphertext c. However, even if m' is the same as m used in Encap, it does not guarantee that $\mathsf{Enc}'(pk, m'; r'') = c$ without performing re-encryption. In other words, there could exist many other ciphertexts $\{c_i\}$ (including c as one of them), all of which are decrypted into the same m' but generated with distinct randomness $\{r''\}$. In $\mathsf{FO}^\perp$ (and other FO transformations), there is still no way to find the same c (honestly) generated in Encap other than by comparing $\mathsf{Enc}'(pk, m'; r'')$ and c. In the context of chosen-ciphertext attacks, it is well known that decapsulation queries using $\{c_i\}$ can leak information on sk, particularly in lattice-based encryption schemes.

```
Gen(1^λ)
  1: $(pk, sk) := \mathsf{Gen}'(1^\lambda)$
  2: return $(pk, sk)$

Encap(pk)
  1: $m \leftarrow \mathcal{M}$
  2: $(r, K) := \mathsf{H}(m)$
  3: $c := \mathsf{Enc}'(pk, m; r)$
       - $M := \mathsf{SOTP}(m, \mathsf{G}(r))$
       - $c := \mathsf{Enc}(pk, M; r)$
  4: return $(K, c)$

Decap(sk, c)
  1: $m' := \mathsf{Dec}'(sk, c)$
       - $M' = \mathsf{Dec}(sk, c)$
       - $r' = \mathsf{RRec}(pk, M', c)$
       - $m' = \mathsf{Inv}(M', \mathsf{G}(r'))$
  2: $(r'', K') := \mathsf{H}(m')$
  3: if $m' = \perp$ or $\boxed{r' \neq r''}$
  4:     return $\perp$
  5: else
  6:     return $K'$
```

Figure 10: $\mathsf{KEM} = \overline{\mathsf{FO}}^\perp[\mathsf{PKE}', \mathsf{H}]$

However, we demonstrate that $\mathsf{FO}^{\perp}$ based on ACWC_2 can eliminate the need for ciphertext comparison $c = \mathsf{Enc}'(pk, m'; r'')$ in Decap, and instead replace it with a simpler and more efficient comparison $r' = r''$. We denote the new $\mathsf{FO}^{\perp}$ based on ACWC_2 as $\overline{\mathsf{FO}}^{\perp}$, as shown in Figure 10. In $\overline{\mathsf{FO}}^{\perp}$, r' and r'' are values generated during the execution of Decap, where r' is the output of $\mathsf{RRec}(pk, M', c)$ and r'' is computed from $\mathsf{H}(m')$. The only change compared to $\mathsf{FO}^{\perp}$ in Figure 9 is the boxed area, replacing $c \neq \mathsf{Enc}'(pk, m'; r'')$ with $r' \neq r''$, while the remaining parts remain the same. By proving that the equality $c = \mathsf{Enc}'(pk, m'; r'')$ is equivalent to the equality $r' = r''$, we can show that both $\mathsf{FO}^{\perp}$ and $\overline{\mathsf{FO}}^{\perp}$ work identically and achieve the same level of IND-CCA security.

Lemma 4.3. Let PKE' be μ'-injective and PKE be μ-injective, and let PKE be δ-rigid and SOTP be δ_s-rigid. Then, with probability at least $1 - (\mu' + \mu + \delta + \delta_s)$, $c = \mathsf{Enc}'(pk, m'; r'')$ in $\mathsf{FO}^{\perp}$ if and only if $r' = r''$ in $\overline{\mathsf{FO}}^{\perp}$.

Proof. For the sake of simplicity, we assume that events that break injectivity and rigidity never occurs.

Assume that $c = \mathsf{Enc}'(pk, m'; r'')$ holds in the Decap of $\mathsf{FO}^{\perp}$. Because PKE' is injective, the pair (m, r) used in Encap is the same as (m', r''). Therefore, ciphertext c generated by Encap is expressed as $c = \mathsf{Enc}(pk, \mathsf{SOTP}(m', \mathsf{G}(r'')); r'')$. Furthermore, because PKE is rigid, for a ciphertext c given to Decap, the two equations $M' = \mathsf{Dec}(sk, c)$ and $r' = \mathsf{RRec}(pk, M', c)$ lead to $\mathsf{Enc}(pk, \mathsf{Dec}(sk, c); r') = c$. In addition, because of the rigidity of the SOTP, the equation $m' = \mathsf{Inv}(M', \mathsf{G}(r'))$ implies $M' = \mathsf{SOTP}(m', \mathsf{G}(r'))$. Thus, using $\mathsf{Dec}(sk, c) = M' = \mathsf{SOTP}(m', \mathsf{G}(r'))$, we can express the ciphertext c in Decap as $\mathsf{Enc}(pk, \mathsf{SOTP}(m', \mathsf{G}(r')); r') = c$. We now have two equations with respect to c generated by Enc. Because PKE is also injective, we observe that $\mathsf{SOTP}(m', \mathsf{G}(r')) = \mathsf{SOTP}(m', \mathsf{G}(r''))$, and $r' = r''$, as required.

Conversely, assume that $r' = r''$ holds in the Decap of $\overline{\mathsf{FO}}^{\perp}$. The rigidity of the SOTP means that $m' = \mathsf{Inv}(M', \mathsf{G}(r'))$ implies $M' = \mathsf{SOTP}(m', \mathsf{G}(r'))$, and thus $M' = \mathsf{SOTP}(m', \mathsf{G}(r''))$. Also, the rigidity of PKE means that for a ciphertext c given to Decap, the two equations $M' = \mathsf{Dec}(sk, c)$ and $r' = \mathsf{RRec}(pk, M', c)$ lead to $\mathsf{Enc}(pk, \mathsf{Dec}(sk, c); r') = c$ and thus $\mathsf{Enc}(pk, \mathsf{Dec}(sk, c); r'') = c$. Because $\mathsf{Dec}(sk, c) = M' = \mathsf{SOTP}(m', \mathsf{G}(r''))$, we see that $\mathsf{Enc}(pk, \mathsf{SOTP}(m', \mathsf{G}(r'')); r'') = c$. Then, $\mathsf{Enc}(pk, \mathsf{SOTP}(m', \mathsf{G}(r'')); r'')$ can be expressed as $\mathsf{Enc}'(pk, m'; r'')$, which implies $\mathsf{Enc}'(pk, m'; r'') = c$. □

5 GenNTRU$[\psi_1^n]$ (=PKE)

5.1 Notations

5.1.1 Centered Binomial Distribution ψ_k

The Centered Binomial Distribution (CBD) ψ_k is a distribution over $\mathbb{Z}$, defined as follows:

- $b_1, \cdots, b_k \leftarrow \{0, 1\}, b'_1, \cdots, b'_k \leftarrow \{0, 1\}$.
- Return $\sum_{i=1}^{k} (b_i - b'_i)$.

Hereafter, in our NTRU construction, we use ψ_1 over the set $\{-1, 0, 1\}$. For a positive integer n, the distribution ψ_1^n is defined over the set $\{-1, 0, 1\}^n$, where each element is selected according to ψ_1.

```
Gen(1^λ)
1: $\mathbf{f}', \mathbf{g} \leftarrow \psi_1^n$
2: $\mathbf{f} = 3\mathbf{f}' + 1$
3: if $\mathbf{f}, \mathbf{g}$ is not invertible in $R_q$
4:     restart
5: $\mathbf{h} = 3\mathbf{g}\mathbf{f}^{-1}$
6: return $(pk, sk) = (\mathbf{h}, \mathbf{f})$

Enc($\mathbf{h}, \mathbf{m} \leftarrow \psi_1^n; \mathbf{r} \leftarrow \psi_1^n$)
1: return $\mathbf{c} = \mathbf{h}\mathbf{r} + \mathbf{m}$
Dec($\mathbf{f}, \mathbf{c}$)
1: return $\mathbf{m} = (\mathbf{c}\mathbf{f} \bmod q) \bmod 3$
RRec($\mathbf{h}, \mathbf{m}, \mathbf{c}$)
1: return $\mathbf{r} = (\mathbf{c} - \mathbf{m})h^{-1}$
MRec($\mathbf{h}, \mathbf{r}, \mathbf{c}$)
1: return $\mathbf{m} = \mathbf{c} - \mathbf{h}\mathbf{r}$
```

Figure 11: $\mathsf{GenNTRU}[\psi_1^n]$ with average-case correctness error

5.1.2 Other Notations

Let $R_q := \mathbb{Z}_q[x]/\langle x^n - x^{n/2} + 1\rangle$ be a ring, where q is a modulus and $n = 2^i 3^j$ for some positive integers i and j. For a polynomial $f \in R_q$, we use the notation '$\mathbf{f} \leftarrow \psi_1^n$' to represent that each coefficient of $\mathbf{f}$ is drawn according to the distribution ψ_1. In addition, we use the notation '$\mathbf{h} \leftarrow R_q$' to show that polynomial $\mathbf{h}$ is chosen uniformly at random from R_q. Let U be a uniformly random distribution over $\{0, 1\}$. We denote U^ℓ as the uniformly random distribution over the set $\{0, 1\}^\ell$. We use the notation '$u \leftarrow U^\ell$' to represent that each bit of u is drawn according to the distribution U. Let a and q be positive integers, and q be an odd integer. We denote $y = a \bmod q$ as the unique integer $y \in \{-(q-1)/2, \cdots, (q-1)/2\}$ that satisfies $q|x - a$.

5.2 Description of GenNTRU$[\psi_1^n]$

Figure 11 defines $\mathsf{GenNTRU}[\psi_1^n]$ relative to the distribution ψ_1^n over R_q. Since $\mathsf{GenNTRU}[\psi_1^n]$ should be MR and RR for our ACWC_2, Figure 11 shows two additional algorithms RRec and MRec.

We notice that $\mathsf{RRec}(\mathbf{h}, \mathbf{m}, \mathbf{c})$ is necessary for performing ACWC_2 where $\mathbf{r}$ should be recovered from $\mathbf{c}$ once $\mathbf{m}$ is obtained. The RR property guarantees that such a randomness-recovery process works well, because for a ciphertext $\mathbf{c} = \mathsf{Enc}(\mathbf{h}, \mathbf{m}, \mathbf{r})= \mathbf{h}\mathbf{r} + \mathbf{m}$ we see that $\mathsf{RRec}(\mathbf{h}, \mathbf{m}, \mathbf{c}) = (\mathbf{c} - \mathbf{m})\mathbf{h}^{-1} = \mathbf{r} \in \mathcal{R}$. On the other hand, $\mathsf{MRec}(\mathbf{h}, \mathbf{r}, \mathbf{c})$ is only used for proving IND-CPA security of the ACWC_2-transformed scheme. The security analysis requires that for a challenge ciphertext $\mathbf{c}^* = \mathsf{Enc}(\mathbf{h}, \mathbf{m}^*, \mathbf{r}^*)= \mathbf{h}\mathbf{r}^* + \mathbf{m}^*$ the algorithm $\mathsf{MRec}(\mathbf{h}, \mathbf{r}^*, \mathbf{c}^*)$ returns the corresponding message $\mathbf{m}^*$ if a queried $\mathbf{r}^*$ was used for $\mathbf{c}^*$. The MR property guarantees that once $\mathbf{r}^*$ is given, $\mathsf{MRec}(\mathbf{h}, \mathbf{r}^*, \mathbf{c}^*) = \mathbf{c}^* - \mathbf{h}\mathbf{r}^* = \mathbf{m}^* \in \mathcal{M}$.

5.3 Security and Other Properties

5.3.1 Cryptographic Assumptions

Definition 5.1 (The NTRU problem)**.** Let ψ be a distribution over R_q. The NTRU problem $\mathsf{NTRU}_{n,q,\psi}$ is to distinguish $\mathbf{h} = \mathbf{g}(p\mathbf{f}' + 1)^{-1} \in R_q$ from $\mathbf{u} \in R_q$, where $\mathbf{f}', \mathbf{g} \leftarrow \psi$ and $\mathbf{u} \leftarrow R_q$. The advantage of adversary $\mathcal{A}$ in solving $\mathsf{NTRU}_{n,q,\psi}$ is defined as follows:

$$\mathsf{Adv}^{\mathsf{NTRU}}_{n,q,\psi}(\mathcal{A}) = \Pr[\mathcal{A}(\mathbf{h}) = 1] - \Pr[\mathcal{A}(\mathbf{u}) = 1].$$

Definition 5.2 (The RLWE problem)**.** Let ψ be a distribution over R_q. The RLWE problem $\mathsf{RLWE}_{n,q,\psi}$ is to find $\mathbf{s}$ from $(\mathbf{a}, \mathbf{b} = \mathbf{a}\mathbf{s} + \mathbf{e}) \in R_q \times R_q$, where $\mathbf{a} \leftarrow R_q$, $\mathbf{s}, \mathbf{e} \leftarrow \psi$. The advantage of an adversary $\mathcal{A}$ in solving $\mathsf{RLWE}_{n,q,\psi}$ is defined as follows:

$$\mathsf{Adv}^{\mathsf{RLWE}}_{n,q,\psi}(\mathcal{A}) = \Pr[\mathcal{A}(\mathbf{a}, \mathbf{b}) = \mathbf{s}].$$

5.3.2 Security Proofs

Theorem 5.3 (OW-CPA security of $\mathsf{GenNTRU}[\psi_1^n]$)**.** For any adversary $\mathcal{A}$, there exist adversaries $\mathcal{B}$ and $\mathcal{C}$ such that

$$\mathsf{Adv}^{\mathsf{OW\text{-}CPA}}_{\mathsf{GenNTRU}[\psi_1^n]}(\mathcal{A}) \le \mathsf{Adv}^{\mathsf{NTRU}}_{n,q,\psi_1^n}(\mathcal{B}) + \mathsf{Adv}^{\mathsf{RLWE}}_{n,q,\psi_1^n}(\mathcal{C}).$$

Proof. We complete our proof through a sequence of games G_0 to G_1. Let $\mathcal{A}$ be the adversary against the OW-CPA security experiment.
GAME G_0. In G_0, we have the original OW-CPA game with $\mathsf{GenNTRU}[\psi_1^n]$. By the definition of the advantage function of the adversary $\mathcal{A}$ against the OW-CPA game, we have that

$$\mathsf{Adv}^{\mathsf{OW\text{-}CPA}}_{\mathsf{GenNTRU}[\psi_1^n]}(\mathcal{A}) = \Pr[G_0^{\mathcal{A}} \Rightarrow 1].$$

GAME G_1. In G_1, the public key $\mathbf{h}$ in Gen is replaced by $\mathbf{h} \leftarrow R_q$. Therefore, distinguishing G_1 from G_0 is equivalent to solving the $\mathsf{NTRU}_{n,q,\psi_1^n}$ problem. More precisely, there exists an adversary $\mathcal{B}$ with the same running time as that of $\mathcal{A}$ such that

$$\left|\Pr[G_0^{\mathcal{A}} \Rightarrow 1] - \Pr[G_1^{\mathcal{A}} \Rightarrow 1]\right| \le \mathsf{Adv}^{\mathsf{NTRU}}_{n,q,\psi_1^n}(\mathcal{B}).$$

Since $\mathbf{h} \leftarrow R_q$ is now changed to a uniformly random polynomial from R_q, G_1 is equivalent to solving an $\mathsf{RLWE}_{n,q,\psi_1^n}$ problem. Therefore,

$$\Pr[G_1^{\mathcal{A}} \Rightarrow 1] = \mathsf{Adv}^{\mathsf{RLWE}}_{n,q,\psi_1^n}(\mathcal{C}).$$

Combining all the probabilities completes the proof. □

Lemma 5.4 (Spreadness). $\mathsf{GenNTRU}[\psi_1^n]$ is n-spread.

Proof. For a fixed message $\mathbf{m}$ and ciphertext $\mathbf{c}$, there exists at most one $\mathbf{r}$ such that $\mathbf{c} = \mathsf{Enc}(\mathbf{h}, \mathbf{m}; \mathbf{r})$. Suppose there exist $\mathbf{r}_1$ and $\mathbf{r}_2$ such that $c = \mathsf{Enc}(\mathbf{h}, \mathbf{m}; \mathbf{r}_1) = \mathsf{Enc}(\mathbf{h}, \mathbf{m}; \mathbf{r}_2)$. Based on this assumption, $\mathbf{h}\mathbf{r}_1 + \mathbf{m} = \mathbf{h}\mathbf{r}_2 + \mathbf{m}$ holds. By subtracting $\mathbf{m}$ and multiplying $\mathbf{h}^{-1}$ on both sides of the equation, we obtain $\mathbf{r} = \mathbf{r}'$. Therefore, there exists at most one $\mathbf{r}$ such that $\mathbf{c} = \mathsf{Enc}(\mathbf{h}, \mathbf{m}; \mathbf{r})$.

For fixed $\mathbf{m}$, to maximize $\Pr[\mathsf{Enc}(\mathbf{h}, \mathbf{m}; \mathbf{r}) = \mathbf{c}]$, we need to choose c such that $\mathbf{c} = \mathsf{Enc}(\mathbf{h}, \mathbf{m}; \mathbf{r})$ for $\mathbf{r} = \mathbf{0}$. Since there exists only one $\mathbf{r}$ such that $\mathbf{c} = \mathsf{Enc}(\mathbf{h}, \mathbf{m}; \mathbf{r})$, we have $\Pr[\mathsf{Enc}(\mathbf{h}, \mathbf{m}; \mathbf{r}) = \mathbf{c}] = 2^{-n}$. Since this holds for any $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$ and $m \in \mathcal{M}$, $\mathsf{GenNTRU}[\psi_1^n]$ is n-spread. □

5.3.3 Average-Case Correctness Error

We analyze the average-case correctness error δ relative to the distribution $\psi_{\mathcal{M}} = \psi_{\mathcal{R}} = \psi_1^n$ using the template provided in [23]. We can expand $\mathbf{cf}$ in the decryption algorithm as follows:

$$\mathbf{cf} = (\mathbf{hr} + \mathbf{m})\mathbf{f} = (3\mathbf{g}\mathbf{f}^{-1}\mathbf{r} + \mathbf{m})(3\mathbf{f}' + 1) = 3(\mathbf{gr} + \mathbf{mf}') + \mathbf{m}.$$

For a polynomial $\mathbf{p}$ in R_q, let $\mathbf{p}_i$ be the i-th coefficient of $\mathbf{p}$, and $|\mathbf{p}_i|$ be the absolute value of $\mathbf{p}_i$. Then, $((\mathbf{cf})_i \mod q) \mod 3 = \mathbf{m}_i$ if the following inequality holds:

$$\left|3(\mathbf{gr} + \mathbf{mf'}) + \mathbf{m}\right|_i \leq \frac{q-1}{2},$$

where all coefficients of each polynomial are distributed according to ψ_1^n. Let ϵ_i be

$$\epsilon_i = \Pr\left[\left|3(\mathbf{gr} + \mathbf{mf'}) + \mathbf{m}\right|_i \leq \frac{q-1}{2}\right].$$

Then, assuming that each coefficient is independent,

$$\Pr\left[\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) \neq m\right] = 1 - \prod_{i=0}^{n-1} \epsilon_i. \tag{6}$$

Because the coefficients of $\mathbf{m}$ have a size at most one,

$$\begin{aligned}
\epsilon_i &= \Pr\left[\left|3(\mathbf{gr} + \mathbf{mf'}) + \mathbf{m}\right|_i \leq \frac{q-1}{2}\right] \\
&\geq \Pr\left[\left|3(\mathbf{gr} + \mathbf{mf'})\right|_i + |\mathbf{m}|_i \leq \frac{q-1}{2}\right] \\
&\geq \Pr\left[\left|3(\mathbf{gr} + \mathbf{mf'})\right|_i + 1 \leq \frac{q-1}{2}\right] \\
&= \Pr\left[\left|\mathbf{gr} + \mathbf{mf'}\right|_i \leq \frac{q-3}{6}\right] := \epsilon'_i.
\end{aligned}$$

Therefore,

$$\Pr\left[\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) \neq m\right] = 1 - \prod_{i=0}^{n} \epsilon_i \leq 1 - \prod_{i=0}^{n} \epsilon'_i := \delta.$$

Now, we analyze $\epsilon'_i = \Pr\left[|\mathbf{gr} + \mathbf{mf'}|_i \leq \frac{q-3}{6}\right]$. To achieve this, we need to analyze the distribution of $\mathbf{gr} + \mathbf{mf'}$. By following the analysis in [23], we can check that for $i \in [n/2, n]$, the degree-i coefficient of $\mathbf{gr} + \mathbf{mf'}$ is the sum of n independent random variables:

$$c = ba + b'(a + a') \in \{0, \pm1, \pm2, \pm3\}, \text{ where } a, b, a, b \leftarrow \psi_1. \tag{7}$$

Additionally, for $i \in [0, n/2 - 1]$, the degree-i coefficient of $\mathbf{gr} + \mathbf{mf'}$ is the sum of $n - 2i$ random variables c (as in Equation (7)), and $2i$ independent random variables c' of the form:

$$c' = ba + b'a' \in \{0, \pm1, \pm2\} \text{ where } a, b, a', b' \leftarrow \psi_1. \tag{8}$$

Computing the probability distribution of this sum can be done via a convolution (i.e. polynomial multiplication). Define the polynomial:

$$\rho_i(X) = \begin{cases} \sum_{j=-3n}^{3n} \rho_{i,j} X^j = \left(\sum_{j=-3}^{3} \theta_j X^j\right)^n \text{ for } i = [n/2, n-1], \\ \sum_{j=-(3n-2i)}^{3n-2i} \rho_{i,j} X^j = \left(\sum_{j=-3}^{3} \theta_j X^j\right)^{n-2i} \left(\sum_{j=-2}^{2} \theta'_j X^j\right)^{2i} \text{ for } i = [0, n/2 - 1], \end{cases} \tag{9}$$

± 3	± 2	± 1	0
1/128	1/32	23/128	9/16

Table 3: Probability distribution of $c = ab + b'(a + a')$

± 2	± 1	0
1/64	3/16	19/32

Table 4: Probability distribution of $c' = ab + a'b'$

where $\theta_j = \Pr[c = j]$ (distribution is shown in Table 3) and $\theta'_j = \Pr[c' = j]$ (distribution is shown in Table 4). Let $\rho_{i,j}$ be the probability that the degree-i coefficient of $\mathbf{gr} + \mathbf{mf'}$ is j. Then, ϵ'_i can be computed as:

$$\epsilon'_i = \begin{cases} 2 \cdot \sum_{j=(q+3)/6}^{3n} \rho_{i,j} & \text{for } i \in [n/2, n-1], \\ 2 \cdot \sum_{j=(q+3)/6}^{3n-2i} \rho_{i,j} & \text{for } i \in [0, n/2-1], \end{cases}$$

where we used the symmetry $\rho_{i,j} = \rho_{i,-j}$. Putting ϵ'_i into Equation (6), we compute the average-case correctness error δ of $\mathsf{GenNTRU}[\psi_1^n]$.

5.3.4 Injectivity and rigidity

The injectivity of $\mathsf{GenNTRU}[\psi_1^n]$ can be easily shown as follows: if there exist two inputs $(\mathbf{m}_1, \mathbf{r}_1)$ and $(\mathbf{m}_2, \mathbf{r}_2)$ such that $\mathsf{Enc}(\mathbf{h}, \mathbf{m}_1; \mathbf{r}_1) = \mathsf{Enc}(\mathbf{h}, \mathbf{m}_2; \mathbf{r}_2)$, the equality indicates that $(\mathbf{r}_1 - \mathbf{r}_2)\mathbf{h} + (\mathbf{m}_1 - \mathbf{m}_2) = 0$, where $\mathbf{r}_1 - \mathbf{r}_2$ and $\mathbf{m}_1 - \mathbf{m}_2$ still have small coefficients of length, at most $2\sqrt{n}$. For a lattice set

$$\mathcal{L}_0^{\perp} := \{(\mathbf{v}, \mathbf{w}) \in R_q \times R_q : \mathbf{hv} + \mathbf{w} = 0 \text{ (in } R_q)\},$$

$(\mathbf{r}_1 - \mathbf{r}_2, \mathbf{m}_1 - \mathbf{m}_2)$ becomes an approximate shortest vector in $\mathcal{L}_0^{\perp}$. Thus, if the injectivity is broken against $\mathsf{GenNTRU}[\psi_1^n]$, we can solve the approximate shortest vector problem (SVP) (of length at most $2\sqrt{n}$) over $\mathcal{L}_0^{\perp}$. It is well-known [12] that the approximate SVP over $\mathcal{L}_0^{\perp}$ is at least as hard as the $\mathsf{NTRU}_{n,q,\psi_1^n}$ problem (defined above). Hence, if the $\mathsf{NTRU}_{n,q,\psi_1^n}$ assumption holds, then the injectivity of $\mathsf{GenNTRU}[\psi_1^n]$ also holds.

Furthermore, under the assumption that $\mathsf{GenNTRU}[\psi_1^n]$ is μ-injective, it is easy to show that $\mathsf{GenNTRU}[\psi_1^n]$ is μ-rigid, because the injectivity guarantees that decrypting $\mathbf{c} = \mathbf{hr} + \mathbf{m}$ yields the same $\mathbf{m}$, and thus the same $\mathbf{r}$ via the deterministic RRec shown in Figure 11.

6 NTRU+

6.1 Instantiation of SOTP

We introduce $\mathsf{SOTP} : \mathcal{M}' \times \mathcal{U} \to \mathcal{M}$, where $\mathcal{M}' = \{0,1\}^n, \mathcal{U} = \{0,1\}^{2n}$, and $\mathcal{M} = \{-1,0,1\}^n$ relative to distributions $\psi_{\mathcal{U}} = U^{2n}$ and $\psi_{\mathcal{M}} = \psi_1^n$. Figure 12 shows SOTP used for ACWC_2. We notice that, following [21], the values of $y + u_2$ generated by the Inv algorithm should be checked to determine whether they are 0 or 1.

```
SOTP($x \in \mathcal{M}', u \leftarrow U^{2n}$)
  1: $u = (u_1, u_2) \in \{0,1\}^n \times \{0,1\}^n$
  2: $y = (x \oplus u_1) - u_2 \in \{-1,0,1\}^n$
  3: return $y$

Inv($y \in \mathcal{M}, u \in U^{2n}$)
  1: $u = (u_1, u_2) \in \{0,1\}^n \times \{0,1\}^n$
  2: if $y + u_2 \notin \{0,1\}^n$, return $\perp$
  3: $x = (y + u_2) \oplus u_1 \in \{0,1\}^n$
  4: return $x$
```

Figure 12: SOTP instantiation for NTRU+

Message-Hiding and Rigidity Properties of SOTP**.** It is easily shown that SOTP is message-hiding because of the one-time pad property, particularly for part $x \oplus u_1$. That is, unless u_1 is known, the message $x \in \mathcal{M}'$ is unconditionally hidden from $y \in \mathcal{M}$. Similarly, $x \oplus u_1$ becomes uniformly random over $\{0,1\}^n$, regardless of the message distribution $\psi_{\mathcal{M}'}$, and thus the resulting y follows ψ_1^n. In addition, the rigidity of the SOTP is trivial because $\mathsf{Inv}(y,u) = x$ implies that $\mathsf{SOTP}(x,u) = y$.

6.2 CPA-NTRU+ (=PKE ′)

We obtain $\mathsf{CPA\text{-}NTRU+} := \mathsf{ACWC}_2$ [$\mathsf{GenNTRU}$ $[\psi_1^n]$,SOTP, G] by applying ACWC_2 from Section 3 to $\mathsf{GenNTRU}[\psi_1^n]$. Because the underlying $\mathsf{GenNTRU}[\psi_1^n]$ provides injectivity, MR, and RR properties, Theorems 3.5 and 3.7 provide us with the $\mathsf{IND\text{-}CPA}$ security of the resulting $\mathsf{CPA\text{-}NTRU+}$ in the classical and quantum random oracle models, respectively. Regarding the correctness error, Theorem 3.2 shows that the worst-case correctness error of $\mathsf{CPA\text{-}NTRU+}$ and the average-case correctness error of $\mathsf{GenNTRU}[\psi_1^n]$ differ by the amount of $\Delta = \|\psi_{\mathcal{R}}\|\left(1 + \sqrt{(\ln|\mathcal{M}'| - \ln\|\psi_{\mathcal{R}}\|)/2}\right)$, where $\psi_{\mathcal{R}}$ and $\mathcal{M}'$ are specified by ψ_1^n and $\{0,1\}^n$, respectively. For instance, when $n = 768$, we obtain about $\Delta = 2^{-1083}$.

```
Gen'(1^λ)
  1: $(pk, sk) := \mathsf{GenNTRU}[\psi_1^n].\mathsf{Gen}(1^\lambda)$
       - $\mathbf{f}', \mathbf{g} \leftarrow \psi_1^n$
       - $\mathbf{f} = 3\mathbf{f}' + 1$
       - if $\mathbf{f}$, $\mathbf{g}$ are not invertible in $R_q$, restart
       - $(pk, sk) = (\mathbf{h} = 3\mathbf{g}\mathbf{f}^{-1} \bmod q, \mathbf{f})$
  2: return $(pk, sk)$

Enc'(pk, m ∈ {0,1}^n; r ← ψ_1^n)
  1: $\mathbf{m} = \mathsf{SOTP}(m, \mathsf{G}(\mathbf{r}))$
  2: $\mathbf{c} = \mathsf{GenNTRU}[\psi_1^n].\mathsf{Enc}(pk, \mathbf{m}; \mathbf{r})$
       - $\mathbf{c} = \mathbf{h}\mathbf{r} + \mathbf{m}$
  3: return $\mathbf{c}$

Dec'(sk, c)
  1: $\mathbf{m} = \mathsf{GenNTRU}[\psi_1^n].\mathsf{Dec}(sk, \mathbf{c})$
       - $\mathbf{m} = (\mathbf{c}\mathbf{f} \bmod q) \bmod 3$
  2: $\mathbf{r} = \mathsf{RRec}(pk, \mathbf{c}, \mathbf{m})$
       - $\mathbf{r} = (\mathbf{c} - \mathbf{m})\mathbf{h}^{-1}$
  3: $m = \mathsf{Inv}(\mathbf{m}, \mathsf{G}(\mathbf{r}))$
  4: return $m$
```

Figure 13: CPA-NTRU+

Spreadness and Injectivity Properties of $\mathsf{CPA\text{-}NTRU+}$**.** To achieve $\mathsf{IND\text{-}CCA}$ security of the transformed KEM via $\overline{\mathsf{FO}}^{\perp}$, we need to show the spreadness and injectivity of $\mathsf{CPA\text{-}NTRU+}$. The spreadness can be easily obtained by combining Lemma 3.8 with Lemma 5.4. Next, the injectivity of $\mathsf{CPA\text{-}NTRU+}$ can also be proven under the assumption that the $\mathrm{NTRU}_{n,q,\psi_1^n}$ problem is infeasible, analogous to $\mathsf{GenNTRU}[\psi_1^n]$. Specifically, if there exist two distinct pairs $(m_1, \mathbf{r}_1)$ and $(m_2, \mathbf{r}_2)$ such that $\mathsf{Enc}'(pk, m_1; \mathbf{r}_1) = \mathsf{Enc}'(pk, m_2; \mathbf{r}_2)$, this results in the equation $\mathbf{h}\mathbf{r}_1 + \mathbf{m}_1 = \mathbf{h}\mathbf{r}_2 + \mathbf{m}_2$, where $\mathbf{m}_1 = \mathsf{SOTP}(m_1, \mathsf{G}(\mathbf{r}_1))$ and $\mathbf{m}_2 = \mathsf{SOTP}(m_2, \mathsf{G}(\mathbf{r}_2))$. In this case, we have two short polynomials: $\mathbf{r}_1 - \mathbf{r}_2$ and $\mathbf{m}_1 - \mathbf{m}_2$, which can be a solution of the approximate SVP (of length at most $2\sqrt{n}$) over $\mathcal{L}_0^{\perp}$.

6.3 NTRU+ (=KEM)

Finally, we achieve $\mathsf{IND\text{-}CCA}$ secure KEM by applying $\overline{\mathsf{FO}}^{\perp}$ to $\mathsf{CPA\text{-}NTRU+}$. We denote such KEM by $\mathsf{NTRU+} := \overline{\mathsf{FO}}^{\perp}[\mathsf{CPA\text{-}NTRU+}, \mathsf{H}]$. Figure 14 shows the resultant $\mathsf{NTRU+}$, which is the basis of our implementation in the next section. By combining Theorems 4.1, 4.2, and Lemma 4.3, we can achieve

IND-CCA security of NTRU+. As for the correctness error, NTRU+ preserves the worst-case correctness error of the underlying CPA-NTRU+.

```
Gen(1^λ)
1: $\mathbf{f}', \mathbf{g} \leftarrow \psi_1^n$
2: $\mathbf{f} = 3\mathbf{f}' + 1$
3: if $\mathbf{f}$, $\mathbf{g}$ are not invertible in $R_q$, restart
4: return $(pk, sk) = (\mathbf{h} = 3\mathbf{g}\mathbf{f}^{-1}, \mathbf{f})$

Encap(pk)
1: $m \leftarrow \{0,1\}^n$
2: $(\mathbf{r}, K) = \mathsf{H}(m)$
3: $\mathbf{m} = \mathsf{SOTP}(m, \mathsf{G}(\mathbf{r}))$
4: $\mathbf{c} = \mathbf{hr} + \mathbf{m}$
5: return $(\mathbf{c}, K)$

Decap(sk, c)
1: $\mathbf{m} = (\mathbf{cf} \bmod q) \bmod 3$
2: $\mathbf{r} = (\mathbf{c} - \mathbf{m})\mathbf{h}^{-1}$
3: $m = \mathsf{Inv}(\mathbf{m}, \mathsf{G}(\mathbf{r}))$
4: $(\mathbf{r}', K) = \mathsf{H}(m)$
5: if $\mathbf{r} = \mathbf{r}'$
6:     return $K$
7: else
8:     return $\perp$
```

Figure 14: NTRU+

7 Algorithm Specification

7.1 Preliminaries and notation

Symmetric primitives. NTRU+ uses three different hash functions: F, G, and H. To instantiate these functions, we use the hash functions SHA256 and SHA512, and we use AES256-CTR with nonce 0 as an extendable output function (XOF). Algorithms 1, 2, and 3 describe the details of F, G, and H.

```
Algorithm 1 F
Require: Byte array $m = (m_0, m_1, \cdots, m_{3n/2-1})$
Ensure: Byte array $B = (b_0, b_1, \cdots, b_{31})$
1: $(b_0, \cdots, b_{31}) := \mathsf{SHA256}((0, m_0, m_1, \cdots, m_{3n/2-1}), 3n/2 + 1)$;
2: return $(b_0, \cdots b_{31})$
```

```
Algorithm 2 G
Require: Byte array $m = (m_0, m_1, \cdots, m_{n/8-1})$
Ensure: Byte array $B = (b_0, b_1, \cdots, b_{n/8+31})$
1: $(b_0, \cdots, b_{31}) := \mathsf{SHA256}((1, m_0, m_1, \cdots, m_{n/8-1}), n/8 + 1)$;
2: $(b_0, \cdots b_{n/8-1}) = \mathsf{XOF}((b_0, \cdots, b_{31}), n/4)$
3: return $(b_0, \cdots b_{n/8-1})$
```

```
Algorithm 3 H
Require: Byte array $m = (m_0, m_1, \cdots, m_{n/8-1})$
Ensure: Byte array $B = (b_0, b_1, \cdots, b_{n/8+31})$
1: $(b_0, \cdots, b_{31}, b_{32}, \cdots b_{63}) := \mathsf{SHA512}(m, n/8)$;
2: $(b_{32}, \cdots b_{n/8+31}) = \mathsf{XOF}((b_{32}, \cdots, b_{63}), n/4)$
3: return $(b_0, \cdots b_{n/8+31})$
```

n	q	Radix-2 for cyclotomic trinomial	Radix-3	Radix-2	d	ζ	$\ell = 3n/d$
576	3457	1	1	5	3	361	576
768	3457	1	1	6	2	19	1152
864	3457	1	2	4	3	9	864
1152	3457	1	1	6	3	19	1152

w : primitive ℓ-th root of unity modulo q

Table 5: Combinations of NTT layers

Modular reductions. Let a and q be positive integers, where q is an odd integer. We denote $y = a \bmod q$ as the unique integer y in the set $\{0, 1, \ldots, q-1\}$ such that q divides $x - a$. Additionally, we denote $y = a \bmod^{\pm} q$ as the unique integer y in the set $\{-(q-1)/2, \cdots, (q-1)/2\}$ such that q divides $x - a$.
Polynomial rings and Number Theoretic Transform. We define two quotient rings: $R = \mathbb{Z}[x]/\langle x^n - x^{n/2} + 1\rangle$ and $R_q = \mathbb{Z}_q[x]/\langle x^n - x^{n/2} + 1\rangle$, where $n = 2^a 3^b$ with $a, b \in \mathbb{N} \cup \{0\}$ such that $x^n - x^{n/2} + 1$ is the $3n$-th cyclotomic polynomial. To efficiently perform computations within the ring R_q, we reduce the computations to the product of smaller rings, denoted as $\prod_{i=0}^{n/d-1} \mathbb{Z}_q[x]/\langle x^d - \zeta_i\rangle$, using the Number Theoretic Transform (NTT). To implement NTT efficiently, we combine three different NTT layers in the following sequence: Radix-2 NTT layer for the cyclotomic trinomial, Radix-3 NTT layer, and then Radix-2 NTT layer[5]. The initial Radix-2 NTT layer for the cyclotomic trinomial, as introduced by [23], establishes a ring isomorphism from $\mathbb{Z}_q[x]/\langle x^n - x^{n/2} + 1\rangle$ to the product ring $\mathbb{Z}_q[x]/\langle x^{n/2} - \zeta\rangle \times \mathbb{Z}_q[x]/\langle x^{n/2} - \zeta^5\rangle$, where ζ denotes a primitive sixth root of unity modulo q. Subsequently, we use Radix-3 NTT layers to establish isomorphisms from $\mathbb{Z}_q[x]/\langle x^n - \alpha^3\rangle$ to the product ring $\mathbb{Z}_q[x]/\langle x^{n/3} - \alpha\rangle \times \mathbb{Z}_q[x]/\langle x^{n/3} - \alpha\omega\rangle \times \mathbb{Z}_q[x]/\langle x^{n/3} - \alpha\omega^2\rangle$, where ω denotes a primitive third root of unity modulo q. In the final step, we use Radix-2 NTT layers to establish isomorphisms from $\mathbb{Z}_q[x]/\langle x^n - \zeta^2\rangle$ to the product ring $\mathbb{Z}_q[x]/\langle x^{n/2} - \zeta\rangle \times \mathbb{Z}_q[x]/\langle x^{n/2} + \zeta\rangle$. Table 5 presents comprehensive information, including the number of applied NTT layers and the resulting degree d of component rings in the product rings for various parameter sets. Note that, for the successful implementation of NTT, it requires a primitive ℓ-th root of unity ζ modulo q, where $\ell = 3n/d$. The values of ℓ and ζ for each parameter are also included in Table 5.

Considering efficient implementation of the NTT, we assume the use of an in-place implementation that does not require reordering of the output values. For clarity, we define NTT as follows:

$$\begin{aligned}\hat{f} = \mathsf{NTT}(f) &= (f \bmod x^d - \zeta^{\texttt{index[0]}}, \cdots, f \bmod x^d - \zeta^{\texttt{index[}n/d-1\texttt{]}}) \\ &= (\sum_{i=0}^{d-1} \hat{f}_i x^i, \sum_{i=0}^{d-1} \hat{f}_{3+i} x^i, \cdots, \sum_{i=0}^{d-1} \hat{f}_{n-d+i} x^i) = (\hat{f}_0, \hat{f}_1, \cdots, \hat{f}_{n-1})\end{aligned}$$

where the array `index` is defined in Figure 15. In this document, we denote NTT as the number theoretic transform function and NTT^{-1} as the inverse number theoretic transform function.
Multiplication in NTT domain. After we transform polynomials in R_q into elements of the product rings, multiplication must be performed in each component ring $\mathbb{Z}_q[x]/\left\langle x^d - \zeta_i\right\rangle$. In the case of $d = 2$, multiplication is carried out as follows:

$$c(x) = a(x)b(x) = (a_0 b_0 + a_1 b_1 \zeta_i) + (a_0 b_1 + a_1 b_0)x$$

[5]We choose to use Radix-3 NTT layers before Radix-2 NTT layers to minimize the size of pre-computation table.

- ntruplus576

```
index[192] = {1, 289, 145, 433, 73, 361 , 217, 505, 37, 325, 181, 469, 109, 397, 253, 541,
19, 307, 163, 451, 91, 379, 235, 523, 55, 343, 199, 487, 127, 415, 271, 559, 7, 295, 151,
439, 79, 367, 223, 511, 43, 331, 187, 475, 115, 403, 259, 547, 25, 313, 169, 457, 97, 385,
241, 529, 61, 349, 205, 493, 133, 421, 277, 565, 13, 301, 157, 445, 85, 373, 229, 517, 49,
337, 193, 481, 121, 409, 265, 553, 31, 319, 175, 463, 103, 391, 247, 535, 67, 355, 211, 499,
139, 427, 283, 571, 5, 293, 149, 437, 77, 365, 221, 509, 41, 329, 185, 473, 113, 401, 257,
545, 23, 311, 167, 455, 95, 383, 239, 527, 59, 347, 203, 491, 131, 419, 275, 563, 11, 299,
155, 443, 83, 371, 227, 515, 47, 335, 191, 479, 119, 407, 263, 551, 29, 317, 173, 461, 101,
389, 245, 533, 65, 353, 209, 497, 137, 425, 281, 569, 17, 305, 161, 449, 89, 377, 233, 521,
53, 341, 197, 485, 125, 413, 269, 557, 35, 323, 179, 467, 107, 395, 251, 539, 71, 359, 215,
503, 143, 431, 287, 575};
```

- ntruplus768 and ntruplus1152

```
index[384] = {1, 577, 289, 865, 145, 721, 433, 1009, 73, 649, 361, 937, 217, 793, 505, 1081,
37, 613, 325, 901, 181, 757, 469, 1045, 109, 685, 397, 973, 253, 829, 541, 1117, 19, 595,
307, 883, 163, 739, 451, 1027, 91, 667, 379, 955, 235, 811, 523, 1099, 55, 631, 343, 919,
199, 775, 487, 1063, 127, 703, 415, 991, 271, 847, 559, 1135, 7, 583, 295, 871, 151, 727,
439, 1015, 79, 655, 367, 943, 223, 799, 511, 1087, 43, 619, 331, 907, 187, 763, 475, 1051,
115, 691, 403, 979, 259, 835, 547, 1123, 25, 601, 313, 889, 169, 745, 457, 1033, 97, 673,
385, 961, 241, 817, 529, 1105, 61, 637, 349, 925, 205, 781, 493, 1069, 133, 709, 421, 997,
277, 853, 565, 1141, 13, 589, 301, 877, 157, 733, 445, 1021, 85, 661, 373, 949, 229, 805,
517, 1093, 49, 625, 337, 913, 193, 769, 481, 1057, 121, 697, 409, 985, 265, 841, 553, 1129,
31, 607, 319, 895, 175, 751, 463, 1039, 103, 679, 391, 967, 247, 823, 535, 1111, 67, 643,
355, 931, 211, 787, 499, 1075, 139, 715, 427, 1003, 283, 859, 571, 1147, 5, 581, 293, 869,
149, 725, 437, 1013, 77, 653, 365, 941, 221, 797, 509, 1085, 41, 617, 329, 905, 185, 761,
473, 1049, 113, 689, 401, 977, 257, 833, 545, 1121, 23, 599, 311, 887, 167, 743, 455, 1031,
95, 671, 383, 959, 239, 815, 527, 1103, 59, 635, 347, 923, 203, 779, 491, 1067, 131, 707,
419, 995, 275, 851, 563, 1139, 11, 587, 299, 875, 155, 731, 443, 1019, 83, 659, 371, 947,
227, 803, 515, 1091, 47, 623, 335, 911, 191, 767, 479, 1055, 119, 695, 407, 983, 263, 839,
551, 1127, 29, 605, 317, 893, 173, 749, 461, 1037, 101, 677, 389, 965, 245, 821, 533, 1109,
65, 641, 353, 929, 209, 785, 497, 1073, 137, 713, 425, 1001, 281, 857, 569, 1145, 17, 593,
305, 881, 161, 737, 449, 1025, 89, 665, 377, 953, 233, 809, 521, 1097, 53, 629, 341, 917,
197, 773, 485, 1061, 125, 701, 413, 989, 269, 845, 557, 1133, 35, 611, 323, 899, 179, 755,
467, 1043, 107, 683, 395, 971, 251, 827, 539, 1115, 71, 647, 359, 935, 215, 791, 503, 1079,
143, 719, 431, 1007, 287, 863, 575, 1151};
```

- ntruplus864

```
index[288] = {1, 433, 217, 649, 109, 541, 325, 757, 55, 487, 271, 703, 163, 595, 379, 811,
19, 451, 235, 667, 127, 559, 343, 775, 73, 505, 289, 721, 181, 613, 397, 829, 37, 469, 253,
685, 145, 577, 361, 793, 91, 523, 307, 739, 199, 631, 415, 847, 7, 439, 223, 655, 115, 547,
331, 763, 61, 493, 277, 709, 169, 601, 385, 817, 25, 457, 241, 673, 133, 565, 349, 781, 79,
511, 295, 727, 187, 619, 403, 835, 43, 475, 259, 691, 151, 583, 367, 799, 97, 529, 313, 745,
205, 637, 421, 853, 13, 445, 229, 661, 121, 553, 337, 769, 67, 499, 283, 715, 175, 607, 391,
823, 31, 463, 247, 679, 139, 571, 355, 787, 85, 517, 301, 733, 193, 625, 409, 841, 49, 481,
265, 697, 157, 589, 373, 805, 103, 535, 319, 751, 211, 643, 427, 859, 5, 437, 221, 653, 113,
545, 329, 761, 59, 491, 275, 707, 167, 599, 383, 815, 23, 455, 239, 671, 131, 563, 347, 779,
77, 509, 293, 725, 185, 617, 401, 833, 41, 473, 257, 689, 149, 581, 365, 797, 95, 527, 311,
743, 203, 635, 419, 851, 11, 443, 227, 659, 119, 551, 335, 767, 65, 497, 281, 713, 173, 605,
389, 821, 29, 461, 245, 677, 137, 569, 353, 785, 83, 515, 299, 731, 191, 623, 407, 839, 47,
479, 263, 695, 155, 587, 371, 803, 101, 533, 317, 749, 209, 641, 425, 857, 17, 449, 233,
665, 125, 557, 341, 773, 71, 503, 287, 719, 179, 611, 395, 827, 35, 467, 251, 683, 143, 575,
359, 791, 89, 521, 305, 737, 197, 629, 413, 845, 53, 485, 269, 701, 161, 593, 377, 809, 107,
539, 323, 755, 215, 647, 431, 863};
```

Figure 15: index for the NTT

We can easily express the multiplication in the matrix form as follows:

$$c(x) = \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} = \begin{pmatrix} a_0 & a_1\zeta_i \\ a_1 & a_0 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \end{pmatrix}.$$

In the case of $d = 3$, multiplication is carried out as follows:

$$a(x)b(x) = (a_0b_0 + (a_2b_1 + a_1b_2)\zeta_i) + (a_1b_0 + a_0b_1 + a_2b_2\zeta_i)x + (a_2b_0 + a_1b_1 + a_0b_2)x^2$$

Similarly, we can express the multiplication in the matrix form as follows:

$$c(x) = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} a_0 & a_2\zeta_i & a_1\zeta_i \\ a_1 & a_0 & a_2\zeta_i \\ a_2 & a_1 & a_0 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix}.$$

Inversion in NTT domain. In the NTT domain, inversion must be performed in each component ring $\mathbb{Z}_q[x]/\langle x^d - \zeta_i \rangle$ similar to multiplication. We can easily derive the formula for the inversion considering the matrix form of multiplication. In the case of $d = 2$, we can compute the inverse of $f(x) = f_0 + f_1x \in \mathbb{Z}_q[x]/\langle x^2 - \zeta_i \rangle$ as

$$f(x)^{-1} = \begin{pmatrix} f_0 & f_1\zeta_i \\ f_1 & f_0 \end{pmatrix}^{-1} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = d^{-1} \begin{pmatrix} f_0 & -f_1\zeta_i \\ -f_1 & f_0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = d^{-1} \begin{pmatrix} f_0 \\ -f_1 \end{pmatrix}$$

where $d = (f_0^2 - f_1^2\zeta_i)$. In the case of $d = 3$, we can compute the inverse of $f(x) = f_0 + f_1x + f_2x^2 \in \mathbb{Z}_q[x]/\langle x^3 - \zeta_i \rangle$ as

$$f^{-1}(x) = \begin{pmatrix} f_0 & f_2\zeta & f_1\zeta \\ f_1 & f_0 & f_2\zeta \\ f_2 & f_1 & f_0 \end{pmatrix}^{-1} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = d^{-1} \begin{pmatrix} f_0' \\ f_1' \\ f_2' \end{pmatrix}$$

where

$$f_0' = f_0^2 - \zeta_i f_1 f_2, \qquad f_1' = \zeta_i f_2^2 - f_0 f_1, \qquad f_2' = f_1^2 - f_0 f_2$$

and

$$d = f_0(f_0^2 - \zeta_i f_1 f_2) + \zeta_i f_1(f_1^2 - f_0 f_2) + \zeta_i f_2(\zeta_i f_2^2 - f_0 f_1) = f_0 f_1' + \zeta_i(f_1 f_2' + f_2 f_1').$$

In both cases, we need to compute the inverse of the determinant d modulo q. To mitigate the risk of side-channel attacks, we opt for Fermat's Little Theorem rather than the extended Euclidean algorithm. Fermat's Little Theorem states that if a is co-prime with q, then $a^{q-1} \equiv 1 (\mathrm{mod}\ q)$ holds true. Using this theorem, we can compute the inverse of a by calculating $a^{q-2} \mathrm{mod}\ q$.

Sampling from a Binomial distribution. NTRU+ employs a centered binomial distribution with $\eta = 1$ for sampling the coefficients of polynomials, as defined in Algorithm 5. Additionally, we introduce the BytesToBits function in Algorithm 4, which determines the order of sampled coefficients. BytesToBits plays a crucial role in the efficient implementation of CBD_1 and SOTP using AVX2 instructions. We also define BitsToBytes as the inverse function of BytesToBits.

Algorithm 4 BytesToBits

```
Require: Byte array $B = (b_0, b_1, \cdots, b_{n/8-1}) \in \mathcal{B}^{n/8}$
Ensure: Bit array $f = (f_0, \cdots, f_{n-1}) \in \{0,1\}^n$
 1: $s = \lfloor n/256 \rfloor$
 2: $r = n - 256s$
 3: $(r_0, r_1, r_2, r_4, r_5, r_6, r_7) :=$ bit-decompose$(r)$        // $r = r_0 2^0 + \cdots r_7 2^7$
 4: for $i$ from 0 to $s-1$ do
 5:     for $j$ from 0 to 7 do
 6:         $t = b_{32i+4j+3} | b_{32i+4j+2} | b_{32i+4j+1} | b_{32i+4j}$
 7:         for $k$ from 0 to 1 do
 8:             for $l$ from 0 to 15 do
 9:                 $f_{256i+16l+2j+k} = t\&1;$
10:                 $t = t >> 1;$
11: $c_1 = 256s$, $c_2 = 32s$
12: if $r_7 = 1$
13:     for $j$ from 0 to 3 do
14:         $t = b_{c_2+4j+3} | b_{c_2+4j+2} | b_{c_2+4j+1} | b_{c_2+4j}$
15:         for $k$ from 0 to 1 do
16:             for $l$ from 0 to 16 do
17:                 $f_{c_1+8l+2j+k} = t\&1;$
18:                 $t = t >> 1;$
19: $c_1 = c_1 + 128r_7$, $c_2 = c_2 + 16r_7$
20: if $r_6 = 1$
21:     for $j$ from 0 to 1 do
22:         $t = b_{c_2+4j+3} | b_{c_2+4j+2} | b_{c_2+4j+1} | b_{c_2+4j}$
23:         for $k$ from 0 to 1 do
24:             for $l$ from 0 to 15 do
25:                 $f_{c_1+4l+2j+k} = t\&1;$
26:                 $t = t >> 1;$
27: $c_1 = c_1 + 64r_6$, $c_2 = c_2 + 8r_6$
28: if $r_5 = 1$
29:     $t = b_{c_2+3} | b_{c_2+2} | b_{c_2+1} | b_{c_2}$
30:     for $k$ from 0 to 1 do
31:         for $l$ from 0 to 15 do
32:             $f_{c_1+2l+k} = t\&1;$
33:             $t = t >> 1;$
34: return $f = (f_0, \cdots, f_{n-1})$
```

Algorithm 5 $\mathsf{CBD}_1 : \mathcal{B}^{n/4} \to R_q$

```
Require: Byte array $B = (b_0, b_1, \cdots, b_{n/4-1})$
Ensure: Polynomial $\mathbf{f} \in R_q$
1: $(\beta_0, \cdots, \beta_{n-1}) := \mathsf{BytesToBits}((b_0, \cdots, b_{n/8-1}))$
2: $(\beta_n, \cdots, \beta_{2n-1}) := \mathsf{BytesToBits}((b_{n/8}, \cdots, b_{n/4-1}))$
3: for $i$ from 0 to $n-1$ do
4:     $f_i := \beta_i - \beta_{i+n}$
5: return $\mathbf{f} = f_0 + f_1 x + f_2 x^2 + \cdots + f_{n-1} x^{n-1}$
```

Semi-generalized one time pad The SOTP function is nearly identical to CBD_1, differing only in that it applies an exclusive OR operation to the first half of the random bytes and the message before sampling from the centered binomial distribution. Consequently, SOTP, as defined in Algorithm 6, also utilizes the BytesToBits function, just like CBD_1. Additionally, we introduce the Inv function in Algorithm 7, which serves as the inverse of the SOTP function and utilizes the BitsToBytes function for byte recovery.

Algorithm 6 SOTP

```
Require: Message Byte array $m = (m_0, m_1, \cdots, m_{31})$
Require: Byte array $B = (b_0, b_1, \cdots, b_{n/4-1})$
Ensure: Polynomial $\mathbf{f} \in R_q$
1: $(\beta_0, \cdots, \beta_{n-1}) := \mathsf{BytesToBits}((b_0, \cdots, b_{n/8-1}))$
2: $(\beta_n, \cdots, \beta_{2n-1}) := \mathsf{BytesToBits}((b_{n/8}, \cdots, b_{n/4-1}))$
3: $(m_0, \cdots, m_{n-1}) := \mathsf{BytesToBits}(m)$
4: for $i$ from 0 to $n-1$ do
5:     $f_i := (m_i \oplus \beta_i) - \beta_{i+n}$
6: return $\mathbf{f} = f_0 + f_1 x + f_2 x^2 + \cdots + f_{n-1} x^{n-1}$
```

Algorithm 7 Inv

```
Require: Polynomial $\mathbf{f} \in R_q$
Require: Byte array $B = (b_0, b_1, \cdots, b_{n/4-1})$
Ensure: Message Byte array $m = (m_0, m_1, \cdots, m_{31})$
1: $(\beta_0, \cdots, \beta_{n-1}) := \mathsf{BytesToBits}((b_0, \cdots, b_{n/8-1}))$
2: $(\beta_n, \cdots, \beta_{2n-1}) := \mathsf{BytesToBits}((b_{n/8}, \cdots, b_{n/4-1}))$
3: for $i$ from 0 to $n-1$ do
4:     if $f_i + \beta_{i+n} \notin \{0, 1\}$, return $\perp$
5:     $m_i := ((f_i + \beta_{i+n}) \& 1) \oplus \beta_i$
   $m = \mathsf{BitsToBytes}((m_0, \cdots, m_{n-1}))$
6: return $m$
```

Encoding and Decoding. To encode polynomials in R_q into a $3n/2$ byte array, we introduce the Encode_q function in Algorithm 8 and 9. This function assumes that each coefficient of the polynomial belongs to the set $\{0, \ldots, q-1\}$ and is stored as a 16-bit data. The design concept behind Encode_q aims to ensure efficiency when implemented with the AVX2 instruction set. Additionally, we define the Decode_q function in Algorithm 10 and 11 as the inverse of Encode_q.

Algorithm 8 Encode$_q$ for ntruplus576, ntruplus768, and ntruplus1152

```
Require: Polynomial $\mathbf{f} \in R_q$
Ensure: Byte array $B = (b_0, \cdots, b_{3n/2-1})$
 1: $max_j = 8$ for ntruplus576, $max_j = 11$ for ntruplus768, $max_j = 17$ for ntruplus1152
 2: for $i$ from 0 to 15 do
 3:     for $j$ from 0 to $max_j$ do
 4:         for $k$ from 0 to 3 do
 5:             $t_k = f_{64j+i+16k}$
 6:         $b_{96j+2i} = t_0$
 7:         $b_{96j+2i+1} = (t_0 >> 8) + (t_1 << 4)$
 8:         $b_{96j+2i+32} = t_1 >> 4$
 9:         $b_{96j+2i+33} = t_2$
10:         $b_{96j+2i+64} = (t_2 >> 8) + (t_3 << 4)$
11:         $b_{96j+2i+65} = t_3 >> 4$
12: return $(b_0, \cdots, b_{3n/2-1})$
```

Algorithm 9 Encode$_q$ for ntruplus864

```
Require: Polynomial $\mathbf{f} \in R_q$
Ensure: Byte array $B = (b_0, \cdots, b_{3n/2-1})$
 1: for $i$ from 0 to 15 do
 2:     for $j$ from 0 to 12 do
 3:         for $k$ from 0 to 3 do
 4:             $t_k = f_{64j+i+16k}$
 5:         $b_{96j+2i} = t_0$
 6:         $b_{96j+2i+1} = (t_0 >> 8) + (t_1 << 4)$
 7:         $b_{96j+2i+32} = t_1 >> 4$
 8:         $b_{96j+2i+33} = t_2$
 9:         $b_{96j+2i+64} = (t_2 >> 8) + (t_3 << 4)$
10:         $b_{96j+2i+65} = t_3 >> 4$
11: for $i$ from 0 to 7 do
12:     for $k$ from 0 to 3 do
13:         $t_k = f_{832+i+8k}$
14:     $b_{1248+2i} = t_0$
15:     $b_{1248+2i+1} = (t_0 >> 8) + (t_1 << 4)$
16:     $b_{1248+2i+16} = t_1 >> 4$
17:     $b_{1248+2i+17} = t_2$
18:     $b_{1248+2i+32} = (t_2 >> 8) + (t_3 << 4)$
19:     $b_{1248+2i+33} = t_3 >> 4$
20: return $(b_0, \cdots, b_{3n/2-1})$
```

Algorithm 10 Decode$_q$ for ntruplus576, ntruplus768, and ntruplus1152

```
Require: Byte array $B = (b_0, \cdots, b_{3n/2-1})$
Ensure: Polynomial $\mathbf{f} \in R_q$
 1: $max_j = 8$ for ntruplus576, $max_j = 11$ for ntruplus768, $max_j = 17$ for ntruplus1152
 2: for $i$ from 0 to 15 do
 3:     for $j$ from 0 to $max_j$ do
 4:         $t_0 = b_{96j+2i}$
 5:         $t_1 = b_{96j+2i+1}$
 6:         $t_2 = b_{96j+2i+32}$
 7:         $t_3 = b_{96j+2i+33}$
 8:         $t_4 = b_{96j+2i+64}$
 9:         $t_5 = b_{96j+2i+65}$
10:         $f_{64j+i} = t_0|(t_1\&0\text{xf}) << 8$
11:         $f_{64j+i+16} = t_1 >> 4|t_2 << 4$
12:         $f_{64j+i+32} = t_3|(t_4\&0\text{xf}) << 8$
13:         $f_{64j+i+48} = t_4 >> 4|t_5 << 4$
14: return $\mathbf{f} = (f_0, \cdots, f_{n-1})$
```

Algorithm 11 Decode$_q$ for ntruplus864

```
Require: Byte array $B = (b_0, \cdots, b_{3n/2-1})$
Ensure: Polynomial $\mathbf{f} \in R_q$
 1: for $i$ from 0 to 15 do
 2:     for $j$ from 0 to 12 do
 3:         $t_0 = b_{96j+2i}$
 4:         $t_1 = b_{96j+2i+1}$
 5:         $t_2 = b_{96j+2i+32}$
 6:         $t_3 = b_{96j+2i+33}$
 7:         $t_4 = b_{96j+2i+64}$
 8:         $t_5 = b_{96j+2i+65}$
 9:         $f_{64j+i} = t_0|(t_1\&0\text{xf}) << 8$
10:         $f_{64j+i+16} = t_1 >> 4|t_2 << 4$
11:         $f_{64j+i+32} = t_3|(t_4\&0\text{xf}) << 8$
12:         $f_{64j+i+48} = t_4 >> 4|t_5 << 4$
13: for $i$ from 0 to 15 do
14:     $t_0 = b_{1248+2i}$
15:     $t_1 = b_{1248+2i+1}$
16:     $t_2 = b_{1248+2i+16}$
17:     $t_3 = b_{1248+2i+17}$
18:     $t_4 = b_{1248+2i+32}$
19:     $t_5 = b_{1248+2i+33}$
20:     $f_{832+i} = t_0|(t_1\&0\text{xf}) << 8$
21:     $f_{832+i+8} = t_1 >> 4|t_2 << 4$
22:     $f_{832+i+16} = t_3|(t_4\&0\text{xf}) << 8$
23:     $f_{832+i+24} = t_4 >> 4|t_5 << 4$
24: return $\mathbf{f} = (f_0, \cdots, f_{n-1})$
```

7.2 Specification of NTRU+

In this section, we specify our IND-CCA secure KEM for the KpqC competition called $\mathsf{NTRU+}$. Unlike $\mathsf{NTRU+}$ in section 6.3, we applied a slightly tweaked $\overline{\mathsf{FO}}^{\perp}$ that is resistant to the multi-target attack. Algorithms 12, 13, and 14 define the key generation, encapsulation, and decapsulation of $\mathsf{NTRU+}$. Note that, in the key generation algorithm, we multiplied $\hat{\mathbf{h}}$ and $\hat{\mathbf{h}}^{-1}$ by 2^{16} before encoding to account for Montgomery reduction.

Algorithm 12 $\mathsf{Gen}(1^\lambda)$: key generation

```
Ensure: Public key $pk \in \mathcal{B}^{\lceil \log_2 q \rceil \cdot n/8}$
Ensure: Secret key $sk \in \mathcal{B}^{\lceil \log_2 q \rceil \cdot n/4}$
 1: $d \leftarrow \mathcal{B}^{32}$
 2: $(f, g) := \mathsf{XOF}(d, n/2)$
 3: $\mathbf{f}' := \mathsf{CBD}_1(f)$
 4: $\mathbf{g}' := \mathsf{CBD}_1(g)$
 5: $\mathbf{f} = 3\mathbf{f}' + 1$
 6: $\mathbf{g} = 3\mathbf{g}'$
 7: $\hat{\mathbf{f}} = \mathsf{NTT}(\mathbf{f})$
 8: $\hat{\mathbf{g}} = \mathsf{NTT}(\mathbf{g})$
 9: if $\mathbf{f}$ or $\mathbf{g}$ is not invertible in $R_q$, restart
10: $\hat{\mathbf{h}} = \hat{\mathbf{g}} \circ \hat{\mathbf{f}}^{-1}$
11: $pk := \mathsf{Encode}_q(2^{16} \cdot \hat{\mathbf{h}})$
12: $sk := \mathsf{Encode}_q(\hat{\mathbf{f}}) || \mathsf{Encode}_q(2^{16} \cdot \hat{\mathbf{h}}^{-1}) || \mathsf{F}(pk)$
13: return $(pk, sk)$
```

Algorithm 13 $\mathsf{Encap}(pk)$: encapsulation

```
Require: Public key $pk \in \mathcal{B}^{\lceil \log_2 q \rceil \cdot n/8}$
Ensure: Ciphertext $c \in \mathcal{B}^{\lceil \log_2 q \rceil \cdot n/8}$
 1: $m \leftarrow \mathcal{B}^{n/8}$
 2: $(K, r) := \mathsf{H}(m, \mathsf{F}(pk))$
 3: $\mathbf{r} := \mathsf{CBD}_1(r)$
 4: $\hat{\mathbf{r}} = \mathsf{NTT}(\mathbf{r})$
 5: $\mathbf{m} = \mathsf{SOTP}(m, \mathsf{G}(\mathsf{Encode}_q(\hat{\mathbf{r}})))$
 6: $\hat{\mathbf{m}} = \mathsf{NTT}(\mathbf{m})$
 7: $2^{16} \cdot \hat{\mathbf{h}} := \mathsf{Decode}_q(pk)$
 8: $\hat{\mathbf{c}} = \hat{\mathbf{h}} \circ \hat{\mathbf{r}} + \hat{\mathbf{m}}$
 9: $c := \mathsf{Encode}_q(\hat{\mathbf{c}})$
10: return $(c, K)$
```

Algorithm 14 $\mathsf{Decap}(sk, c)$: decapsulation

```
Require: Secret key $sk \in \mathcal{B}^{\lceil \log_2 q \rceil \cdot n/4+32}$
Require: Ciphertext $c \in \mathcal{B}^{\lceil \log_2 q \rceil \cdot n/8}$
Ensure: Shared key $m \in \mathcal{B}^{32}$
 1: Parse $sk = (sk_1, sk_2, sk_3) \in \mathcal{B}^{\lceil \log_2 q \rceil \cdot n/8} \times \mathcal{B}^{\lceil \log_2 q \rceil \cdot n/8} \times \mathcal{B}^{32}$
 2: $\hat{\mathbf{f}} = \mathsf{Decode}_q(sk_1)$
 3: $\hat{\mathbf{c}} = \mathsf{Decode}_q(c)$
 4: $\mathbf{m} = \mathsf{NTT}^{-1}(\hat{\mathbf{c}} \circ \hat{\mathbf{f}}) \mod {}^{\pm}3$
 5: $\hat{\mathbf{m}} = \mathsf{NTT}(\mathbf{m})$
 6: $2^{16} \cdot \hat{\mathbf{h}}^{-1} = \mathsf{Decode}_q(sk_2)$
 7: $\hat{\mathbf{r}} = (\hat{\mathbf{c}} - \hat{\mathbf{m}}) \circ \hat{\mathbf{h}}^{-1}$                // Recover$^r$
 8: $m' := \mathsf{Inv}(\mathbf{m}, \mathsf{G}(\mathsf{Encode}_q(\hat{\mathbf{r}})))$
 9: if $m' = \perp$, return $\perp$
10: $(K', r') := \mathsf{H}(m', sk_3)$
11: $\mathbf{r}' := \mathsf{CBD}_1(r')$
12: $\hat{\mathbf{r}}' = \mathsf{NTT}(\mathbf{r}')$
13: if $\hat{\mathbf{r}} = \hat{\mathbf{r}}'$, return $K'$. Else, return $\perp$
```

8 Parameters and Security Analysis

We define four parameters sets for NTRU+, which are listed in Table 7. We call them NTRU+{576, 768, 864, 1152}, respectively, depending on the degree of the polynomial $x^n - x^{n/2} + 1$. In all parameter sets, the modulus q is set to 3457, and the coefficients of $\mathbf{m}$ and $\mathbf{r}$ are sampled according to the distribution ψ_1^n (i.e., $\psi_{\mathcal{R}} = \psi_{\mathcal{M}} = \psi_1^n$). For each set of $(n, q, \psi_1^n, \mathcal{M}' = \{0,1\}^n)$, the worst-case correctness error δ' is calculated by adding the average-case correctness error δ of $\mathsf{GenNTRU}[\psi_1^n]$ and the value $\Delta = \|\psi_{\mathcal{R}}\|\Big(1 + \sqrt{(\ln|\mathcal{M}'| - \ln\|\psi_{\mathcal{R}}\|)/2}\Big)$ using the equation from Theorem 3.2. Since Δ is negligible for all parameter sets, the worst-case correctness error of NTRU+ is almost equal to the average-case correctness error of each corresponding $\mathsf{GenNTRU}[\psi_1^n]$ as expected.

Scheme	classical		quantum	
	LWE	NTRU	LWE	NTRU
NTRU+576	115	114	105	105
NTRU+768	164	164	150	149
NTRU+864	189	189	172	172
NTRU+1152	263	265	241	241

Table 6: Concrete Security Level relative to LWE and NTRU problems

To estimate the concrete security level of NTRU+, we analyze the hardness of the two problems $\mathsf{RLWE}_{n,q,\psi_1^n}$ and $\mathsf{NTRU}_{n,q,\psi_1^n}$ based on each parameter set. For the RLWE problem, we employ the Lattice estimator [1], which uses the BKZ lattice reduction algorithm [7] for the best-known lattice attacks such as the primal [2] and dual [22] attacks. Next, for the NTRU problem, we use the NTRU estimator provided by the finalist NTRU [6], which is based on the primal attack and Howgrave-Graham's hybrid attack [17] over the NTRU lattice. The primal attack over the NTRU lattice is essentially the same as the attack using the BKZ algorithm, and Howgrave-Graham's hybrid attack is also based on the BKZ algorithm combined with Odlyzko's Meet-in-the-Middle (MitM) attack [20] on a (reduced) sub-lattice. As a result, the concrete security level of the NTRU problem is almost the same as that of the RLWE problem. Table 6 shows the resulting security levels relative to the RLWE and NTRU problems, depending on each $\mathsf{NTRU+}$ parameter set.

9 Performance Analysis

All benchmarks were obtained on a single core of an Intel Core i7-8700K (Coffee Lake) processor clocked at 3700 MHz. The benchmarking machine was equipped with 16 GB of RAM. Implementations were compiled using gcc version 9.4.0. Table 7 lists the execution time of the reference and AVX2 implementations of $\mathsf{NTRU+}$, NTRU, KYBER, and KYBER-90s, along with the security level, the size of the secret key, public key, and ciphertext. The execution time was measured as the average cycle counts of 100,000 executions for the respective algorithms. The source code of $\mathsf{NTRU+}$ can be downloaded from `https://github.com/ntruplus/ntruplus`.

When comparing NTRU and $\mathsf{NTRU+}$, Table 7 shows that both schemes have similar bandwidth (consisting of a public key and a ciphertext) at comparable security levels. For instance, $\mathsf{NTRU+}864$ at the 189-bit security level requires a bandwidth of 2592 bytes, and ntruhps4096821 at the 178-bit security level requires a bandwidth of 2460 bytes. In terms of storage cost with respect to the secret key, $\mathsf{NTRU+}$ requires almost twice as much storage cost as NTRU. This is because $\mathsf{NTRU+}$ stores $(\mathbf{f}, \mathbf{h}^{-1}, \mathsf{F}(pk))$ as a secret key rather than only $\mathbf{f}$. However, in terms of execution time, $\mathsf{NTRU+}$ outperforms NTRU, primarily depending on whether NTT-friendly rings are used.

When comparing KYBER (KYBER-90s) and $\mathsf{NTRU+}$, the bandwidth of $\mathsf{NTRU+}$ is slightly larger than that of KYBER at similar security levels. This is because KYBER uses a rounding technique to reduce the size of a ciphertext. In terms of efficiency, it would be fairer to compare KYBER-90s (rather than KYBER) and $\mathsf{NTRU+}$, because both schemes commonly use AES256-CTR as an eXtendable-Output Function (XOF) to expand randomness from a seed. We notice that KYBER uses SHAKE-128 as its XOF. Generally, SHAKE-128 is faster than AES256-CTR in the reference implementation, but the situation is reversed in the AVX2 implementation due to the existence of assembly instructions designed for AES. Table 7 shows that, at similar security levels, the key generation of $\mathsf{NTRU+}$ is slower than that of KYBER-90s in the reference implementation. However, the encapsulation and decapsulation of $\mathsf{NTRU+}$ is faster than that of KYBER-90s in both the reference and AVX2 implementations.

Table 7: Comparison between the finalist NTRU, KYBER (KYBER-90s) and NTRU+

Scheme	security level		n	q	pk	ct	sk	$\log_2 \delta'$	reference			AVX2		
	classical	quantum							Gen	Encap	Decap	Gen	Encap	Decap
NTRU+576	114	105	576	3457	864	864	1760	-487	285	106	135	20	20	12
NTRU+768	164	149	768	3457	1152	1152	2336	-379	325	137	177	24	26	17
NTRU+864	189	172	864	3457	1296	1296	2624	-340	324	162	217	23	28	18
NTRU+1152	263	241	1152	3457	1728	1728	3488	-260	770	204	288	45	36	24
KYBER512	118	108	512	3329	800	768	1632	-139	103	129	156	27	35	26
KYBER512-90s									181	211	238	17	22	15
KYBER768	182	165	768	3329	1184	1088	2400	-164	184	217	253	43	55	42
KYBER768-90s									333	371	405	24	31	22
KYBER1024	255	231	1024	3329	1568	1568	3168	-174	283	317	362	61	80	64
KYBER1024-90s									526	563	604	33	43	31
ntruhps2048509	106	96	509	2048	699	699	935	$-\infty$	8428	596	1435	195	84	33
ntruhrss701	137	124	701	8192	1138	1138	1450	$-\infty$	15603	938	2655	257	60	51
ntruhps2048677	144	131	677	2048	930	930	1234	$-\infty$	14461	993	2478	307	114	49
ntruhps4096821	178	162	821	4096	1230	1230	1590	$-\infty$	21403	1401	3597	417	136	62

n: polynomial degree of the ring. q: modulus. (pk, ct, sk): bytes. δ': worst-case (or perfect) correctness error.

(Gen, Encap, Decap): K cycles of reference or AVX2 implementations.

References

[1] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *J. Math. Cryptol.*, 9(3):169–203, 2015.

[2] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In Thorsten Holz and Stefan Savage, editors, *USENIX Security 2016: 25th USENIX Security Symposium*, pages 327–343, Austin, TX, USA, August 10–12, 2016. USENIX Association.

[3] Andris Ambainis, Mike Hamburg, and Dominique Unruh. Quantum security proofs using semi-classical oracles. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 269–295, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.

[4] Daniel J. Bernstein and Edoardo Persichetti. Towards KEM unification. Cryptology ePrint Archive, Report 2018/526, 2018. `https://eprint.iacr.org/2018/526`.

[5] Nina Bindel, Mike Hamburg, Kathrin Hövelmanns, Andreas Hülsing, and Edoardo Persichetti. Tighter proofs of CCA security in the quantum random oracle model. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019: 17th Theory of Cryptography Conference, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 61–90, Nuremberg, Germany, December 1–5, 2019. Springer, Heidelberg, Germany.

[6] Cong Chen, Oussama Danba, Jeffrey Hoffstein, Andreas Hulsing, Joost Rijneveld, John M. Schanck, Peter Schwabe, William Whyte, Zhenfei Zhang, Tsunekazu Saito, Takashi Yamakawa, and Keita Xagawa. NTRU. Technical report, National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions`.

[7] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 1–20, Seoul, South Korea, December 4–8, 2011. Springer, Heidelberg, Germany.

[8] Jan-Pieter D'Anvers, Qian Guo, Thomas Johansson, Alexander Nilsson, Frederik Vercauteren, and Ingrid Verbauwhede. Decryption failure attacks on IND-CCA secure lattice-based schemes. In Dongdai Lin and Kazue Sako, editors, *PKC 2019: 22nd International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 11443 of *Lecture Notes in Computer Science*, pages 565–598, Beijing, China, April 14–17, 2019. Springer, Heidelberg, Germany.

[9] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, Frederik Vercauteren, Jose Maria Bermudo Mera, Michiel Van Beirendonck, and Andrea Basso. SABER. Technical report, National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions`.

[10] Alexander W. Dent. A designer's guide to KEMs. In Kenneth G. Paterson, editor, *9th IMA International Conference on Cryptography and Coding*, volume 2898 of *Lecture Notes in Computer Science*, pages 133–151, Cirencester, UK, December 16–18, 2003. Springer, Heidelberg, Germany.

[11] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Online-extractability in the quantum random-oracle model. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part III*, volume 13277 of *Lecture Notes in Computer Science*, pages 677–706, Trondheim, Norway, May 30 – June 3, 2022. Springer, Heidelberg, Germany.

[12] Julien Duman, Kathrin Hövelmanns, Eike Kiltz, Vadim Lyubashevsky, Gregor Seiler, and Dominique Unruh. A thorough treatment of highly-efficient NTRU instantiations. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023: 26th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 13940 of *Lecture Notes in Computer Science*, pages 65–94, Atlanta, GA, USA, May 7–10, 2023. Springer, Heidelberg, Germany.

[13] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology*, 26(1):80–101, January 2013.

[14] Chenar Abdulla Hassan and Oğuz Yayla. Radix-3 NTT-based polynomial multiplication for lattice-based cryptography. Cryptology ePrint Archive, Report 2022/726, 2022. `https://eprint.iacr.org/2022/726`.

[15] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In *Third Algorithmic Number Theory Symposium (ANTS)*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer, Heidelberg, Germany, June 1998.

[16] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 341–371, Baltimore, MD, USA, November 12–15, 2017. Springer, Heidelberg, Germany.

[17] Nick Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 150–169, Santa Barbara, CA, USA, August 19–23, 2007. Springer, Heidelberg, Germany.

[18] Nick Howgrave-Graham, Phong Q. Nguyen, David Pointcheval, John Proos, Joseph H. Silverman, Ari Singer, and William Whyte. The impact of decryption failures on the security of NTRU encryption. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 226–246, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany.

[19] Nick Howgrave-Graham, Joseph H. Silverman, Ari Singer, and William Whyte. NAEP: Provable security in the presence of decryption failures. Cryptology ePrint Archive, Report 2003/172, 2003. `https://eprint.iacr.org/2003/172`.

[20] Nick Howgrave-Graham, Joseph H. Silverman, and William Whyte. A meet-in-the-middle attack on an ntru private key. Technical report, NTRU Cryptosystems, 2003. available at `https://ntru.org/f/tr/tr004v2.pdf`.

[21] Joohee Lee, Minju Lee, and Jaehui Park. A Novel CCA Attack for NTRU+ KEM. Cryptology ePrint Archive, Report 2023/1188, 2023. `https://eprint.iacr.org/2023/1188`.

[22] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In Aggelos Kiayias, editor, *Topics in Cryptology – CT-RSA 2011*, volume 6558 of *Lecture Notes in Computer Science*, pages 319–339, San Francisco, CA, USA, February 14–18, 2011. Springer, Heidelberg, Germany.

[23] Vadim Lyubashevsky and Gregor Seiler. NTTRU: Truly fast NTRU using NTT. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(3):180–201, 2019. `https://tches.iacr.org/index.php/TCHES/article/view/8293`.

[24] Tsunekazu Saito, Keita Xagawa, and Takashi Yamakawa. Tightly-secure key-encapsulation mechanism in the quantum random oracle model. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 520–551, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

[25] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehlé. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions`.

[26] Zhenfei Zhang, Cong Chen, Jeffrey Hoffstein, and William Whyte. NTRUEncrypt. Technical report, National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions`.

A Factorizing the Polynomial

For a better understanding of applying NTT, we will describe how to factor the polynomial ring $\mathbb{Z}_{3457}[x]/\langle x^{576} - x^{288} + 1\rangle$. By utilizing the Radix-2 NTT layer for the cyclotomic trinomial, we can factorize $x^{576} - x^{288} + 1$ as follows:

$$x^{576} - x^{288} + 1 = (x^{288} - \zeta^{96})(x^{288} - \zeta^{480}).$$

Here, $\zeta^{\ell/6} = \zeta^{96}$ represents a primitive sixth root of unity modulo q. Consequently, we can observe that we can apply a Radix-3 NTT layer because both $x^{288} - \zeta^{96}$ and $x^{288} - \zeta^{480}$ can be factorized as:

$$x^{288} - \zeta^{96} = (x^{96} - \zeta^{32})(x^{96} - \zeta^{32}\omega)(x^{96} - \zeta^{32}\omega^2) = (x^{96} - \zeta^{32})(x^{96} - \zeta^{224})(x^{96} - \zeta^{416})$$
$$x^{288} - \zeta^{480} = (x^{96} - \zeta^{160})(x^{96} - \zeta^{160}\omega)(x^{96} - \zeta^{160}\omega^2) = (x^{96} - \zeta^{160})(x^{96} - \zeta^{352})(x^{96} - \zeta^{544}).$$

Here, $\omega = \zeta^{\ell/3} = \zeta^{192}$ is a primitive third root of unity modulo q. Similarly, we can observe that we can apply a Radix-2 NTT layer because both $x^{96} - \zeta^{32}$ and $x^{96} - \zeta^{480}$ can be further factorized by half. For example, $x^{96} - \zeta^{32}$ can be factorized as:

$$x^{96} - \zeta^{32} = (x^{48} - \zeta^{16})(x^{48} + \zeta^{16}) = (x^{48} - \zeta^{16})(x^{48} - \zeta^{16}\zeta^{\ell/2}) = (x^{48} - \zeta^{16})(x^{48} - \zeta^{336})$$

Here, $\zeta^{\ell/2} = \zeta^{288}$ is a primitive second root of unity modulo q. If we continue this process, we can factor the polynomial $x^{576} - x^{288} + 1$ all the way down to the degree $d = 3$.

B Radix-3 NTT layer

For a clearer understanding, we describe the Radix-3 NTT layer used in our implementation. The Radix-3 NTT layer establishes a ring isomorphism between $\mathbb{Z}_q[x]/\langle x^n - \alpha^3\rangle$ and the product ring $\mathbb{Z}_q[x]/\langle x^{n/3} - \alpha\rangle \times \mathbb{Z}_q[x]/\langle x^{n/3} - \beta\rangle \times \mathbb{Z}_q[x]/\langle x^{n/3} - \gamma\rangle$, where $\beta = \alpha\omega$, and $\gamma = \alpha\omega^2$ (with ω representing a primitive third root of unity modulo q). To transform a polynomial $a(x) = a_0(x) + a_1(x)x^{n/3} + a_2(x)x^{2n/3} \in \mathbb{Z}_q[x]/\langle x^n - \alpha^3\rangle$ (where $a_0(x)$, $a_1(x)$, and $a_2(x)$ are polynomials with a maximum degree of $n/3 - 1$) into the form $(\hat{a}_0(x), \hat{a}_1(x), \hat{a}_2(x)) \in \mathbb{Z}_q[x]/\langle x^{n/3} - \alpha\rangle \times \mathbb{Z}_q[x]/\langle x^{n/3} - \beta\rangle \times \mathbb{Z}_q[x]/\langle x^{n/3} - \gamma\rangle$, the following equations must be computed.

$$\begin{aligned}\hat{a}_0(x) &= a_0(x) + a_1(x)\alpha + a_2(x)\alpha^2,\\ \hat{a}_1(x) &= a_0(x) + a_1(x)\beta + a_2(x)\beta^2,\\ \hat{a}_2(x) &= a_0(x) + a_1(x)\gamma + a_2(x)\gamma^2.\end{aligned}$$

Naively, these equations might appear to require $2n$ multiplications and $2n$ additions, using six predefined values: α, α^2, β, β^2, γ, and γ^2. Nevertheless, by following the techniques in [14], we can significantly reduce this computational load to n multiplications, n additions, and $4n/3$ subtractions, by using only three predefined values: α, α^2, and ω, as described in Algorithm 15.

$$\begin{aligned}\hat{a}_0(x) &= a_0(x) + a_1(x)\alpha + a_2(x)\alpha^2\\ \hat{a}_1(x) &= a_0(x) - a_2(x)\alpha^2 + \omega(a_1(x)\alpha - a_2(x)\alpha^2)\\ \hat{a}_2(x) &= a_0(x) - a_1(x)\alpha - \omega(a_1(x)\alpha - a_2(x)\alpha^2)\end{aligned}$$

Algorithm 15 Radix-3 NTT layer

```
Require: $a(x) = a_0(x) + a_1(x)x^{n/3} + a_2(x)x^{2n/3} \in \mathbb{Z}_q[x]/\langle x^n - \zeta^3 \rangle$
Ensure: $(\hat{a}_0(x), \hat{a}_1(x), \hat{a}_2(x)) \in \mathbb{Z}_q[x]/\langle x^{n/3} - \alpha \rangle \times \mathbb{Z}_q[x]/\langle x^{n/3} - \beta \rangle \times \mathbb{Z}_q[x]/\langle x^{n/3} - \gamma \rangle$
1: $t_1(x) = a_1(x)\alpha$
2: $t_2(x) = a_2(x)\alpha^2$
3: $t_3(x) = (t_1(x) - t_2(x))w$
4: $\hat{a}_2(x) = a_0(x) - t_1(x) + t_3(x)$
5: $\hat{a}_1(x) = a_0(x) - t_1(x) + t_3(x)$
6: $\hat{a}_0(x) = a_0(x) - t_1(x) + t_3(x)$
7: return $(\hat{a}_0(x), \hat{a}_1(x), \hat{a}_2(x))$
```

Considering the aforementioned Radix-3 NTT layer, we need to compute the following equations to recover $a(x) \in \mathbb{Z}_q[x]/\langle x^n - \zeta^3 \rangle$ from $(\hat{a}_0(x), \hat{a}_1(x), \hat{a}_2(x)) \in \mathbb{Z}_q[x]/\langle x^{n/3} - \alpha \rangle \times \mathbb{Z}_q[x]/\langle x^{n/3} - \beta \rangle \times \mathbb{Z}_q[x]/\langle x^{n/3} - \gamma \rangle$.

$$\begin{aligned}
3a_0(x) &= \hat{a}_0(x) + \hat{a}_1(x) + \hat{a}_2(x), \\
3a_1(x) &= \hat{a}_0(x)\alpha^{-1} + \hat{a}_1(x)\beta^{-1} + \hat{a}_2(x)\gamma^{-1}, \\
3a_2(x) &= \hat{a}_0(x)\alpha^{-2} + \hat{a}_1(x)\beta^{-2} + \hat{a}_2(x)\gamma^{-2}.
\end{aligned}$$

Naively, these equations might appear to require $2n$ multiplications and $2n$ additions, using six predefined values: α^{-1}, α^{-2}, β^{-1}, β^{-2}, γ^{-1}, and γ^{-2}. Nevertheless, by following the techniques in [14], we can significantly reduce this computational load to n multiplications, n additions, and $4n/3$ subtractions, by employing only four predefined values: α^{-1}, α^{-2}, and ω, as described in in Algorithm 16.

$$\begin{aligned}
3a_0(x) &= \hat{a}_0(x) + \hat{a}_1(x) + \hat{a}_2(x) \\
3a_1(x) &= \alpha^{-1}(\hat{a}_0(x) - \hat{a}_1(x) - w(\hat{a}_1(x) - \hat{a}_2(x))) \\
3a_2(x) &= \alpha^{-2}(\hat{a}_0(x) - \hat{a}_2(x) + w(\hat{a}_1(x) - \hat{a}_2(x)))
\end{aligned}$$

Algorithm 16 Radix-3 Inverse NTT layer

```
Require: $(\hat{a}_0(x), \hat{a}_1(x), \hat{a}_2(x)) \in \mathbb{Z}_q[x]/\langle x^{n/3} - \alpha \rangle \times \mathbb{Z}_q[x]/\langle x^{n/3} - \beta \rangle \times \mathbb{Z}_q[x]/\langle x^{n/3} - \gamma \rangle$
Ensure: $3a(x) = 3a_0(x) + 3a_1(x)x^{n/3} + 3a_2(x)x^{2n/3} \in \mathbb{Z}_q[x]/\langle x^n - \alpha^3 \rangle$
1: $t_1(x) = w(\hat{a}_1(x) - \hat{a}_2(x))$
2: $t_2(x) = \hat{a}_0(x) - \hat{a}_1(x) - t_1(x)$
3: $t_3(x) = \hat{a}_0(x) - \hat{a}_2(x) + t_1(x)$
4: $3a_0(x) = \hat{a}_0(x) + \hat{a}_1(x) + \hat{a}_2(x)$
5: $3a_1(x) = t_2(x)\alpha^{-1}$
6: $3a_2(x) = t_3(x)\alpha^{-2}$
7: return $3a(x) = 3a_0(x) + 3a_1(x)x^{n/3} + 3a_2(x)x^{2n/3}$
```

Note that we can reuse the predefined table used for NTT in the computation of Inverse NTT.

$$\begin{aligned}
3a_0(x) &= \hat{a}_0(x) + \hat{a}_1(x) + \hat{a}_2(x) \\
3a_1(x) &= (w^{-1}\alpha^{-1})(\hat{a}_1(x) - \hat{a}_2(x) - (\hat{a}_2(x) - \hat{a}_0(x))w) \\
3a_2(x) &= (w^{-2}\alpha^{-2})(\hat{a}_1(x) - \hat{a}_0(x) + (\hat{a}_2(x) - \hat{a}_0(x))w)
\end{aligned}$$