NTRU+: Compact Construction of NTRU Using Simple Encoding Method*

Jonghyun Kim† Jong Hwan Park‡

November 21, 2022

Abstract

NTRU was the first practical public-key encryption scheme constructed on a lattice over a polynomial-based ring, and has been still considered secure against significant cryptanalytic attacks in a few decades. Despite such a long history, NTRU and its variants proposed to date suffer from several drawbacks, such as the difficulty of achieving worst-case correctness error in a moderate modulus, inconvenient sampling distributions for messages, and relatively slower algorithms than other lattice-based schemes.

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

1 Introduction

The NTRU encryption scheme [13] was published by Hoffstein, Pipher, and Silverman in 1998 as the first practical public-key encryption scheme using lattices over polynomial rings. The hardness of the NTRU is crucially based on the NTRU problem [13], which has withstood significant cryptanalytic attacks over a few decades. Such a longer history than other lattice-based problems (such as Ring/Module-LWE) has been considered as being an important factor in selecting the NTRU as a finalist in the NIST PQC standardization process. While the finalist NTRU [4], which was a merger of two submissions NTRU-HRSS [21] and NTRU-HPS [23], has not been chosen by NIST as one of the first four quantum-resistant cryptographic algorithms, the NTRU has several distinct advantages over other lattice-based competitive schemes like Kyber [22] and Saber [6]. Indeed, the advantages of the NTRU include (1) compact structure of a ciphertext consisting of a single polynomial, and (2) (possibly) faster encryption and decryption without need to sample coefficients of a public key polynomial.

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

†Korea University, Seoul, Korea. Email: `yoswuk@korea.ac.kr`.

‡Sangmyung University, Seoul, Korea. Email: `jhpark@smu.ac.kr`.

The central design principle of the 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 and p is a positive integer smaller than q (e.g., 3), and the corresponding private key is then $\mathbf{f} = p\mathbf{f}' + 1$. To encrypt a message m sampled from a message space $\mathcal{M}'$, one creates two polynomials $\mathbf{r}$ and $\mathbf{m}$ whose coefficients are also drawn from a narrow distribution ψ, and computes a ciphertext $\mathbf{c} = \mathbf{hr} + \mathbf{m}$ in R_q. There could exist an (efficient) encoding method by which $m \in \mathcal{M}'$ is encoded into $\mathbf{m}$ and $\mathbf{r} \in R_q$. Alternatively, it is also possible to directly sample $\mathbf{m}$ and $\mathbf{r}$ from ψ, where $\mathbf{m}$ is considered as a message to be encrypted. To decrypt the ciphertext $\mathbf{c}$, one computes $\mathbf{cf}$ in R_q, recovers $\mathbf{m}$ by taking the value $\mathbf{cf}'$ modulo p, and (if necessary) decodes $\mathbf{m}$ to obtain the message m. The decryption of the 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, in which case the probability that the decryption fails is called a *correctness (or decryption) error*.

In the context of chosen-ciphertext attacks, the NTRU (as ordinary public-key encryption schemes) must guarantee a very negligible worst-case correctness error, because otherwise information about the private key may be leaked by adversarial decryption queries (like attacks [15, 5] against lattice-based encryption schemes). Roughly speaking, a worst-case correctness error means the probability that a decryption fails for any ciphertext that can be generated with all possible messages and randomnesses in their respective spaces. Naturally, the worst-case correctness error considers that an adversary, $\mathcal{A}$, is able to *maliciously* choose messages and randomnesses, without sampling normally according to their original distributions (if possible). In case of the NTRU, the failure to decrypt a specific ciphertext $\mathbf{c} = \mathbf{hr} + \mathbf{m}$ gives $\mathcal{A}$ 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 the secret polynomials $\mathbf{g}$ and $\mathbf{f}$.

When designing the NTRU, there have been two approaches for achieving worst-case correctness error. One is to draw $\mathbf{m}$ and $\mathbf{r}$ directly from ψ, while setting the modulus q to be relatively large. The larger q guarantees with a high probability that all coefficients of $p(\mathbf{gr} + \mathbf{f}'\mathbf{m}) + \mathbf{m}$ are less than $q/2$ for *almost all possible* $\mathbf{m}$ and $\mathbf{r}$ in their spaces, but causes inefficiency in terms of public-key and ciphertext sizes. Indeed, this approach has been taken by the third-round finalist NTRU [4], where all recommended parameters provide *perfect* correctness error (i.e., the worst-case correctness error becomes zero for all possible $\mathbf{m}$ and $\mathbf{r}$). On the other hand, the other approach [9] 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 [11], decrypting a ciphertext $\mathbf{c}$ requires to re-encrypt m by following the same sampling process as in encryption. Thus, an ill-formed ciphertext especially not following the sampling rule will always fail to be successfully decrypted, resulting in the fact 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 one.

Based on the above observation, [9] proposed generic (average-case to worst-case) transformations[2] that make an average-case correctness error of an underlying scheme almost close to a worst-case one of a transformed scheme. One of their transformations (denoted by ACWC) is based on an encoding method called a generalized one-time pad (denoted by GOTP). Roughly speaking, GOTP works as follows: when $\mathbf{m}$ is split into two polynomials $\mathbf{m}_1||\mathbf{m}_2$, a message $m \in \mathcal{M}'$ is first used to sample $\mathbf{r}$ and $\mathbf{m}_1$ according

[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 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[4]	NTRU-B [9]	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 a ring. λ: length of a message. DPKE: deterministic public key encryption. SXY variant: SXY transformation [20] described in the finalist NTRU.

Table 1: Comparison to previous NTRU Constructions

to ψ, and then $\mathbf{m}_2 = \mathsf{GOTP}(m, \mathsf{G}(\mathbf{m}_1))$ using a hash function G. If GOTP acts as a sampling function whose output follows ψ, $\mathbf{m}$ and $\mathbf{r}$ are created from m following ψ, which can be verified in decryption by using the FO transform. However, ACWC based on GOTP has two disadvantages in terms of security reduction and message distribution. Firstly, [9] showed that ACWC converts a oneway (OW-CPA) secure underlying scheme into a transformed scheme that is also OW-CPA secure, but their security reduction is loose[3] by causing a security loss factor of q_G, the number of random oracle queries. Secondly, ACWC requires m to be sampled from the message space $\mathcal{M}'$ according to a special distribution ψ'. Indeed, the NTRU instantiation from ACWC, called 'NTRU-B' [9], requires that m should be chosen uniformly at random from $\mathcal{M}' = \{-1, 0, 1\}^\lambda$ for some integer λ. It is still known that it is not easy to generate exactly uniform numbers in constant time when a modulus is not a power-of-2, because of rejection sampling.

1.1 Our Results

We present a new practical NTRU construction, called 'NTRU+', which overcomes the two drawbacks raised in the previous ACWC. To achieve our goal, we provide a new generic ACWC transformation (denoted by ACWC_2) that works with a simple encoding method. Using ACWC_2, NTRU+ achieves a worst-case correctness error that is almost close to an average-case one of an underlying NTRU. In addition, NTRU+ requires 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 of NTRU and RLWE. To achieve chosen-ciphertext security, NTRU+ relies on a novel FO-equivalent transform without re-encryption, which makes decryption algorithm of NTRU+ faster than in the ordinary FO transform. In terms of efficiency, we use the idea of [18] to apply NTT (Number Theoretic Transform) [17] 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 give implementation results for NTRU+ in each parameter set. Table 1 shows the main differences between the previous NTRU constructions [4, 9] and NTRU+. In the following, we describe our technique focusing on those differences.

ACWC_2 Transformation with Tight Reduction. ACWC_2 is a new generic transformation that allows for the average-case to worst-case correctness error conversion mentioned above. However, to apply ACWC_2,

[3] [9] introduced a new security notion, q-OW-CPA, which captures that an adversary output a set Q of size at most q and wins if a correct message corresponding to a challenged ciphertext belongs to Q. We think that q-OW-CPA causes a security loss of q.

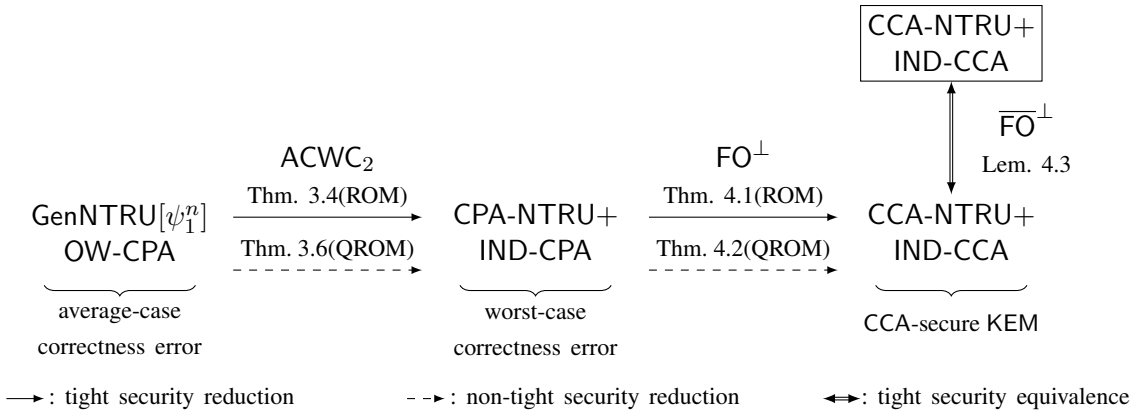

Figure 1: Overview of security reductions

an underlying scheme is required to have so-called randomness- and message-recoverable properties, which are typical of the NTRU.[4] Additionally, ACWC$_2$ involves an encoding method, called semi-generalized one-time pad (denoted by SOTP). Unlike GOTP in [9], SOTP works as follows: a message $m \in \mathcal{M}'$ is first used to sample $\mathbf{r}$ according to ψ, and then $\mathbf{m} = \mathsf{SOTP}(m, \mathsf{G}(\mathbf{r}))$ whose coefficients follow ψ, using a hash function G. In 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, and then an inverse of SOTP using $\mathsf{G}(\mathbf{r})$ and $\mathbf{m}$ gives m.

The message-recoverable property of an underlying scheme allows us to show that, without causing security loss, ACWC$_2$ transforms a OW-CPA secure scheme into a chosen-plaintext (IND-CPA) secure one. 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 gain any information on m_b (that $\mathcal{A}$ submits) for $b \in \{0, 1\}$ because of the basic message-hiding property of SOTP. However, importantly, whenever $\mathcal{A}$ queries $\mathbf{r}_i$ to G for $i = 1, \cdots, q_{\mathsf{G}}$, a reductionist can check if each $\mathbf{r}_i$ is the randomness used for its OW-CPA challenge ciphertext by using a message-recovery algorithm. Therefore, the reductionist can find the exact $\mathbf{r}_i$ among q_{G} number of queries, as long as $\mathcal{A}$ 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- and randomness-hiding properties.

FO-Equivalent Transform without Re-encryption. To achieve chosen-ciphertext (IND-CCA) security, we apply the generic transform FO$^{\perp}$ to the ACWC$_2$-derived scheme that is IND-CPA-secure. As other FO-transformed schemes, the resulting scheme from ACWC$_2$ and FO$^{\perp}$ is still required to perform re-encryption in decryption process in order to check if (1) $(\mathbf{m}, \mathbf{r})$ are correctly generated from m and also (2) a (decrypted) ciphertext $\mathbf{c}$ is correctly encrypted from $(\mathbf{m}, \mathbf{r})$. However, by using the randomness-recoverable property of an underlying scheme, we further proceed to remove the re-encryption process from FO$^{\perp}$. Instead, the more advanced transform (denoted by $\overline{\mathsf{FO}}^{\perp}$) simply checks if $\mathbf{r}$ from the randomness-recovery algorithm is the same as the (new) randomness $\mathbf{r}'$ created from m. We show that $\overline{\mathsf{FO}}^{\perp}$ works functionally identical to the FO$^{\perp}$ by proving that the randomness-checking process in $\overline{\mathsf{FO}}^{\perp}$ is equivalent to the re-encryption process FO$^{\perp}$. The equivalence proof mainly relies on injectivity [14, 3] and rigidity [2] properties of underlying schemes.

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

Scheme	sec(c)	n	q	pk	ct	sk	$\log_2 \delta$	Gen	Encap	Decap
NTRU+576	115	576	3457	864	864	1728	-487	17	14	12
NTRU+768	161	768	3457	1152	1152	2304	-379	16	18	16
NTRU+864	188	864	3457	1296	1296	2592	-340	14	19	18
NTRU+1152	264	1152	3457	1728	1728	3456	-260	43	26	24
Kyber512	117	512	3329	800	768	1632	-139	26	35	26
Kyber768	181	768	3329	1184	1088	2400	-164	43	54	42
Kyber1024	253	1024	3329	1568	1568	3168	-174	59	78	63
ntru_hps_2048509	106	509	2048	699	699	935	$-\infty$	191	80	33
ntru_hrss_701	136	701	8192	1138	1138	1450	$-\infty$	251	58	51
ntru_hps_2048677	145	677	2048	930	930	1234	$-\infty$	298	109	48
ntru_hps_4096821	179	821	4096	1230	1230	1590	$-\infty$	407	130	62

sec(c) : classical security. n: polynomial degree of a ring. q: modulus. (pk, ct, sk): bytes. δ: worst-case (or perfect) correctness error. (Gen, Encap, Decap): K cycles with AVX2 optimization.

Table 2: Comparison between the finalist NTRU, Kyber and NTRU+

As a result, although the randomness-recoverable property seems to incur some additional decryption cost, it ends up making the decryption algorithm faster than in the original FO transform. Figure 1 presents the overview of security reductions from OW-CPA to IND-CCA.

Simple SOTP with More Convenient Sampling Distributions. As mentioned above, ACWC_2 is crucially based on an efficient construction of SOTP that takes m and $\mathsf{G}(\mathbf{r})$ as input 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 do this, we set ψ to be the centered binomial distribution ψ_1 that is easily obtained by subtracting two uniformly-random bits from each other. For a polynomial degree n and a hash function $\mathsf{G} : \{0,1\}^* \to \{0,1\}^{2n}$, m is chosen from the message space $\mathcal{M}' = \{0,1\}^n$ for 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 very easy to implement. We can also expect that, in a similar way to the case of ψ_1, SOTP is expanded to sample a centered binomial distribution module p (i.e., $\overline{\psi}_2$) by summing up and subtracting more number of uniformly-random bits.

NTT-Friendly Rings Over Cyclotomic Trinomials. 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 $n = 2^i 3^j$. [18] showed that, with appropriate parametrization 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, since the choice of a cyclotomic trinomial is moderate, it gives more flexibility to meet a certain desired level of security. Following these results, we choose four parameter sets for 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 2 presents comparison results between the finalist NTRU [4], Kyber [22] and NTRU+ in terms of security and efficiency. The (classical) security of NTRU+ is computed using the security analysis script of the Kyber[22], 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, Implementation results in Table 2 are estimated with AVX2 optimizations. We can see that NTRU+

outperforms the NTRU and Kyber at a similar security level.

1.2 Related Works

The first round NTRUEncrypt [23] submission to the NIST PQC standardization process was an NTRU-based encryption scheme combined with the so-called NAEP padding method [16]. Roughly speaking, NAEP is very 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 follows from 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, that is, $m - \mathsf{G}(\mathbf{hr})$ by bitwise operation, and then assigning them to coefficients of $\mathbf{m}$. Since m can be maliciously chosen by $\mathcal{A}$ in NTRUEncrypt, so can be the resulting $\mathbf{m}$, regardless of the hash value $\mathsf{G}(\mathbf{hr})$.

The finalist NTRU [4], which is a merger of two submissions NTRU-HRSS [21] and NTRU-HPS [23], was submitted as a key encapsulation mechanism (KEM) that provides four parameter sets for perfect correctness. To achieve chosen-ciphertext security, [4] relied on a variant of the SXY [20] conversion, which also avoids re-encryption during decapsulation. The SXY variant requires the rigidity [2] of an underlying scheme as in NTRU+, and uses the notion of deterministic public-key encryption (DPKE) where $(\mathbf{m}, \mathbf{r})$ are all recovered as a message in decryption. In NTRU construction, recovering $\mathbf{r}$ is conceptually the same as the existence of the randomness-recovery algorithm $\mathsf{Recover}^r$. 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.[18] proposed an efficient NTRU-based KEM, called NTTRU, by applying NTT to the ring defined by a cyclotomic trinomial $\mathbb{Z}_q[x]/\langle x^n - x^{n/2} + 1\rangle$. [18] showed that NTT can be applied to the ring $\mathbb{Z}_q[x]/\langle x^n - x^{n/2} + 1\rangle$, using the fact that $x^n - x^{n/2} + 1$ can be factored as $(x^{n/2} - \zeta) \times (x^{n/2} - (1 - \zeta))$ where ζ is the primitive sixth root of unity modulo q. NTTRU was based on the Dent [7] transformation without any encoding method, which resulted in about 2^{-13} worst-case correctness error even with 2^{-1230} average-case one. In order to overcome this large difference, NTTRU was modified to reduce the message space of an underlying scheme, while increasing the size of a ciphertext. This modification is later generalized to the generic ACWC_0 in [9].

In 2021, Duman et al. [9] proposed two generic transformations ACWC_0 and ACWC by which an average-case correctness error of an underlying scheme is almost equal to a worst-case one of a transformed scheme. In particular, ACWC introduced GOTP as an encoding method that prevents $\mathcal{A}$ from choosing $\mathbf{m}$ adversarially. ACWC_0 is simple but requires about 32 bytes of ciphertext expansion, whereas ACWC does not need to expand the ciphertext size. [9] analyzed security from ACWC_0 and ACWC in the classical and quantum random oracle model. As mentioned above, however, their NTRU instantiation from ACWC has a drawback that requires a message m to be chosen from a uniformly-random distribution over $\mathcal{M}' = \{-1, 0, 1\}^\lambda$. Indeed, it was an open problem [9] to construct a new transformation that permits a different, more easily sampled distribution of a message, while relying on the same security assumptions.

Recently, Fouque et al. [10] suggested a new NTRU-based KEM, called BAT, which can reduce the size of a ciphertext. In general, NTRU was considered hard to compress a ciphertext, because $\mathbf{m}$ is encoded in the least significant bits of a ciphertext. Instead of generating a ciphertext as a Ring-LWE instance, BAT encrypts a message m as $(\lfloor \mathbf{hr} \rceil, \mathsf{G}(\mathbf{r}) \oplus m)$ using a rounding operation $\lfloor \cdot \rceil$. BAT decrypts the ciphertext, using an NTRU trapdoor basis [19] as a secret key in a way that finds an NTRU-lattice point $\mathbf{hr}$ closest to $\lfloor \mathbf{hr} \rceil$. Even though BAT provides the compact size of a ciphertext, the proposed parameter sets for BAT are very limited to meet a desired level of security, basically because of a ring over a polynomial $x^n + 1$. Also, the key generation algorithm of BAT is much slower than other lattice-based competitive KEMs such as Kyber and NTRU+.

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 following three algorithms:

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

Correctness. We say that PKE has (worst-case) correctness error δ [14] 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 average-case correctness error δ relative to 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. [14, 3] We say that PKE has injectivity error μ 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

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

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 [12] of $\mathsf{Enc}(pk, m)$ by

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

Then, we say that PKE is γ-spread [12] if for every key pair $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$ and every message $m \in M$,

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

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 $\mathsf{Recover}^r$ such that for all $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$ and $m \in \mathcal{M}$ and $r \in \mathcal{R}$, we have that

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

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 $\mathsf{Recover}^r$ returns $\perp$ if $\mathsf{Recover}^r(pk, m', c) \notin \mathcal{R}$ or $\mathsf{Enc}(pk, m'; \mathsf{Recover}^r(pk, m', c)) \neq c$.

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

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

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 $\mathsf{Recover}^m$ returns $\perp$ if $\mathsf{Recover}^m(pk, r', c) \notin \mathcal{M}$ or $\mathsf{Enc}(pk, \mathsf{Recover}^m(pk, r', c); r') \neq c$.

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

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

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 a message space $\mathcal{M}$. Onewayness under chosen-plaintext attacks (OW-CPA) for a message distribution $\psi_\mathcal{M}$ is defined via the game OW-CPA defined in Figure 2 and the advantage function of an 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 a message space $\mathcal{M}$. Indistinguishability under chosen-plaintext attacks (IND-CPA) is defined via the game IND-CPA defined in Figure 2 and the advantage function of an 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	Game IND-CPA
1: $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$	1: $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$
2: $m \leftarrow \psi_\mathcal{M}$	2: $(m_0, m_1) \leftarrow \mathcal{A}_0(pk)$
3: $c^* \leftarrow \mathsf{Enc}(pk, m)$	3: $b \leftarrow \{0, 1\}$
4: $m' \leftarrow \mathcal{A}(pk, c^*)$	4: $c^* \leftarrow \mathsf{Enc}(pk, m_b)$
5: **return** $[\![m = m']\!]$	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 KEM = (Gen, Encap, Decap) with a key space $\mathcal{K}$ consists of following three algorithms:

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

Correctness. We say that KEM has correctness error δ if

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

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 KEM = (Gen, Encap, 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 described in Figure 3 and the advantage function of an adversary $\mathcal{A}$ is as follows:

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

<u>Game IND-CCA</u>

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** $[\![b = b']\!]$

<u>$\mathsf{Decap}(c \neq c^*)$</u>

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

Figure 3: Game IND-CCA for KEM

3 ACWC_2 Transformation

Let PKE be an encryption scheme with small average-case correctness error, and let G be a hash function modeled as a random oracle. We will now introduce our new ACWC transformation ACWC_2 by describing $\mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$ in Figure 4. Let $\mathsf{PKE}' = \mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$ be a resultant 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 an average-case one of PKE, and (2) PKE' is tightly $\mathsf{IND\text{-}CPA}$ secure if PKE is $\mathsf{OW\text{-}CPA}$ secure.

3.1 SOTP

First, we begin by defining a semi generalised one-time pad SOTP as follows:

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

1. Decoding: There exists an efficient inversion 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 $\mathsf{SOTP}(\mathsf{Inv}(y, u), u) = y$.

Compared to the generalized one-time pad (GOTP) defined in [9], SOTP does not need to have an additional *randomness-hiding* property, which requires that an output $y = \mathsf{SOTP}(x, u)$ follows the distribution $\psi_{\mathcal{Y}}$ and simultaneously does not leak any information about the used randomness u. The absence of such additional property allows us to design SOTP more flexibly and efficiently than GOTP. Instead, SOTP is required to be *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 $\mathsf{SOTP}(x, u) = y$.

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 a randomness $r \in \mathcal{R}$ are drawn from 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 the 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. 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.

Since $\mathsf{Recover}^r$ and Inv functions do not affect the correctness error of PKE', the factor that determines the success or failure of decryption is the result of $\mathsf{Dec}(sk, c)$ in Dec'. This means that, in the end, the correctness error of PKE' is determined by the selections of $M \in \mathcal{M}$ and $r \in \mathcal{R}$. We see that r is drawn according to the distribution $\psi_{\mathcal{R}}$ and M is an SOTP-encoded element in $\mathcal{M}$ following the distribution $\psi_{\mathcal{M}}$. We can here view SOTP as a sampling function using an internal randomness $\mathsf{G}(r)$, while hiding m. Eventually, both M and r are chosen according to their respective distributions initially intended. This is the same idea as in ACWC, and overall the proof strategy of Theorem 3.2 is essentially the same as that of [9] (Lemma 3.6 therein), except for slight modifications to message distribution.

$\underline{\mathsf{Gen}'(1^\lambda)}$

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

$\underline{\mathsf{Enc}'(pk, m \in \mathcal{M}'; r \leftarrow \psi_{\mathcal{R}})}$

1: $M := \mathsf{SOTP}(m, \mathsf{G}(r))$
2: $c := \mathsf{Enc}(pk, M; r)$
3: **return** c

$\underline{\mathsf{Dec}'(sk, c)}$

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

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

Theorem 3.2 (Average-Case to Worst-Case Correctness Error)**.** Let PKE be message recoverable and have 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 $\mathsf{G} : \mathcal{R} \to \psi_{\mathcal{U}}$ 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 choice of G and $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$, the worst-case correctness of PKE' is

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

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 choice of G, for a fixed key pair (pk, sk). For any fixed key pair and for any non-negative real t, we have that

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

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 $\mu(pk, sk) := \Pr_{M,r}[\mathsf{Dec}(sk, \mathsf{Enc}(pk, M; r)) \neq M]$. We can now use helper Lemma 3.3 below to argue that

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

To this end, we identify $g(m, r, u)$ in Lemma 3.3 as $g(m, r, u) = (\mathsf{SOTP}(m, u), r)$ and B as $\{(M, r) \in |\mathsf{Dec}(sk, \mathsf{Enc}(pk, M; r) \neq m\}$. 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.

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

Plugging Eq. (2) into Eq. (1) and taking the expectation yields

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

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

Lemma 3.3 (Variant of Lemma 3.7 from [9])**.** Let g be some function and B be some set such that

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

Let G be a random function mapping onto $\mathcal{U}$. Define $\|\psi_{\mathcal{R}}\| = \sqrt{\sum_r \psi_{\mathcal{R}}(r)^2}$. Then for all but e^{-c} fraction of random functions G, we have that $\forall m \in \mathcal{M}'$,

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

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 Eq. (3), we know that $\sum_r \psi_{\mathcal{R}}(r)p_r = \mu$. For each r, define a random variable X_r whose value is determined as follows: G chooses a random $u = \mathsf{G}(r)$ then checks whether $g(m,r,\mathsf{G}(r)) \in B$; if it is, then we set $X_r = 1$, and otherwise we set it to 0. Because G is a random function, the probability that $X_r = 1$ is exactly p_r.

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

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

Setting $t \ge \|\psi\|\cdot\sqrt{(c+\ln|\mathcal{M}'|)/2}$, we obtain that for a fixed m, Eq. (4) holds for all but a $e^{-c}\cdot|\mathcal{M}'|^{-1}$ fraction of random functions G. Applying the union bound gives us the claim in the lemma. □

Theorem 3.4 ($\mathsf{OW\text{-}CPA}$ of $\mathsf{PKE} \overset{\mathsf{ROM}}{\Longrightarrow}$ $\mathsf{IND\text{-}CPA}$ of $\mathsf{ACWC}_2[\mathsf{PKE},\mathsf{SOTP},\mathsf{G}]$)**.** Let PKE be a public-key encryption scheme with injectivity error μ and {randomness, message}-recoverable properties. For any adversary $\mathcal{A}$ against the $\mathsf{IND\text{-}CPA}$ 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 $\mathsf{OW\text{-}CPA}$ security of PKE with

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

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

$\underline{\mathcal{B}(pk, c^*)}$
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: **return** $M \leftarrow \psi_{\mathcal{M}}$

$\underline{\mathsf{G}(r)}$ // As simulated by $\mathcal{B}$
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: **return** u

$\underline{\mathcal{B}(pk, r, c^*)}$ // Find (r^*, M^*)
1: **if** $M := \mathsf{Recover}^m(pk, r, c^*) \in \mathcal{M}$
2: **return** M
3: **abort**

Figure 6: Adversary $\mathcal{B}$ for the proof of Theorem 3.4

$\underline{\text{Game } G_0}$
1: $\mathsf{G} \leftarrow (\mathcal{R} \to \{0,1\}^\lambda)$
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 Theorem 3.4

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

GAME G_0. G_0 (see Figure 5) is the the original IND-CPA game with $\mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$. In game G_0, $\mathcal{B}$ is given the challenge ciphertext $c^* := \mathsf{Enc}(pk, M^*; r^*)$ for some message M^* and randomness r^* unknown to $\mathcal{B}$. 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. G_1 is the same as G_0, except that $\mathcal{B}$ aborts when $\mathcal{A}$ queries two distinct r_1^* and r_2^* to G such that $\mathsf{Recover}^m(pk, r_1^*, c^*)$, $\mathsf{Recover}^m(pk, r_2^*, c^*) \in \mathcal{M}$. This case leads to breaking the injectivity of PKE. Thus, we have

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

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

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

Unless QUERY occurs, $\mathsf{G}(r^*)$ is a uniformly random value 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 SOTP, meaning that $\Pr[G_2^{\mathcal{A}} \Rightarrow 1] = 1/2$. On the other hand, i.e., if QUERY occurs, $\mathcal{B}$ can find $r^* \in \mathcal{L}_r$ such that $c^* := \mathsf{Enc}(pk, M^*; r^*)$, using the algorithm $\mathsf{Recover}^m$. Indeed, for each query r to G, $\mathcal{B}$ checks if $\mathsf{Recover}^m(pk, r, c^*) \in \mathcal{M}$. In the QUERY event, there exists $M^* := \mathsf{Recover}^m(pk, r^*, c^*) \in \mathcal{M}$ which can be the solution to its challenge ciphertext c^*. It follows that

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

which concludes the proof. □

Theorem 3.5 (Classical O2H, Theorem 3 from the eprint version of [1])**.** Let $S \subset \mathcal{R}$ be random. Let G, 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 arbitrary joint distribution.) Let $\mathcal{C}$ be a quantum oracle algorithm with query depth q_{G}, expecting input z. Let $\mathcal{D}$ be the algorithm which 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 3.6 (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 injectivity error μ and {randomness, message}-recoverable properties. For any quantum adversary $\mathcal{A}$ against the IND-CPA security of $\mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$ with 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 the theorem, we use games G_0 to G_7 defined in Figure 7 to 9, and Theorem 3.5. Before we apply the Theorem 3.5, we change the game G_0 to G_2. After that, we apply Theorem 3.5 to games G_2 and G_3. The detailed explanation of the security proof is given in the followings.

Game G_0

$\mathsf{G} \leftarrow (\mathcal{R} \rightarrow \{0,1\}^\lambda)$
$(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$
$(m_0, m_1) \leftarrow \mathcal{A}_0^{\mathsf{G}}(pk)$
$b \leftarrow \{0,1\}$
$r \leftarrow \psi_{\mathcal{R}}$
$M = \mathsf{SOTP}(m_b, \mathsf{G}(r))$
$c^* \leftarrow \mathsf{Enc}(pk, M; r)$
$b' \leftarrow \mathcal{A}_1^{\mathsf{G}}(pk, c^*)$
return $[\![b = b']\!]$

Figure 7: Game G_0 for the proof of Theorem 3.6

Games G_1-G_5

$\mathsf{G} \leftarrow (\mathcal{R} \to \{0,1\}^\lambda)$ // G_1
$r \leftarrow \psi_{\mathcal{R}}$
$u := \mathsf{G}(r)$ // G_1
$\mathsf{F} \leftarrow (\mathcal{R} \to \{0,1\}^\lambda)$ // G_2-G_5
$u \leftarrow \psi_{\mathcal{U}}$ // G_2-G_5
$\mathsf{G} := \mathsf{F}(r := u)$ // G_2-G_5
$w \leftarrow \mathcal{C}^{\mathsf{G}}(r,u)$ // G_1-G_2
$w \leftarrow \mathcal{C}^{\mathsf{F}}(r,u)$ // G_3
$T \leftarrow \mathcal{D}^{\mathsf{F}}(r,u)$ // G_4-G_5
return w // G_1-G_3
return $r \in T$ // G_4-G_5

$\underline{\mathcal{C}^{\mathsf{G}}(r,u)}$

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

$\underline{\mathcal{D}^{\mathsf{F}}(r,u)}$

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

Figure 8: Games G_1-G_5 for the proof of Theorem 3.6

GAME G_0. Game G_0 (see Figure 7) 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 game G_1 by moving part of the game G_0 inside an algorithm $\mathcal{C}^{\mathsf{G}}$. Also, we query $u := \mathsf{G}(r)$ before algorithm $\mathcal{C}^{\mathsf{G}}$ runs adversary $\mathcal{A}$. Since the changes are only conceptual,

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

GAME G_2. We change how G is defined in Game G_2. Instead of choosing G uniformly, We choose F and u uniformly, then set $\mathsf{G} := \mathsf{F}(r := u)$. Here, $\mathsf{G} = \mathsf{F}(r := u)$ is same function as F, except that it returns u on input r. Since the distribution of G and u are unchanged,

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

GAME G_3. We define game G_3 by providing function F to algorithm $\mathcal{C}$, instead of G. By applying Theorem 3.5 with $\mathcal{C}$, $S := \{r\}$, and $z := (r,u)$, we get:

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

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

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

GAME G_4 and G_5. We define game G_4 according to Theorem 3.5. Also, we define game G_5 by changing the way M is calculated. Instead of computing $M = \mathsf{SOTP}(m_b, u)$, we sample $M \leftarrow \psi_{\mathcal{M}}$. On the other hand,

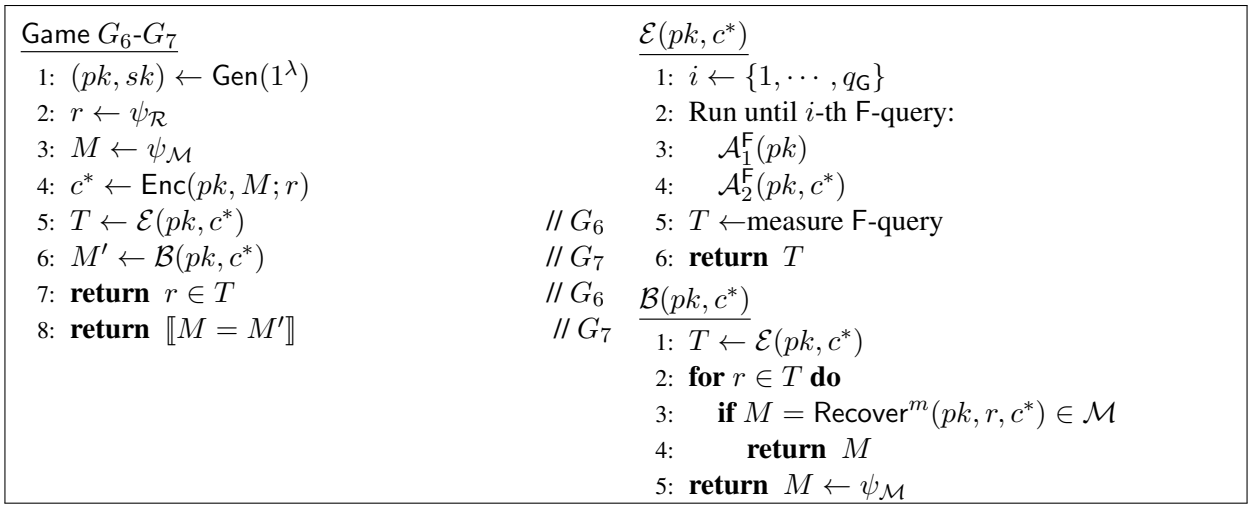

Figure 9: Games G_6-G_7 for the proof of Theorem 3.6

in game 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 distribution $\psi_{\mathcal{M}}$. Therefore,

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

GAME G_6. We define game G_6 by rearranging the game G_5 defined in Figure 9. Since the changes are only conceptual,

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

GAME G_7. Game G_7 is defined by algorithm $\mathcal{B}$ shown in Figure 9, moving from game G_6. G_7 is the same as G_6, except for the case when there are two distinct $r, r' \in T$ such that $\mathsf{Recover}^m(pk, r, c^*)$, $\mathsf{Recover}^m(pk, r', c^*) \in \mathcal{M}$. If this case happens, 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 now see that, in G_7, $\mathcal{B}$ wins if there exists $r \in T$ such that $m^* := \mathsf{Recover}^m(pk, r, c^*) \in \mathcal{M}$, as the solution of its challenge ciphertext c^*. Therefore, we have

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

Putting all (in)equalities and bounds together, we have

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

which concludes our proof. □

Lemma 3.7. 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 also a possible ciphertext with respect to PKE. Since PKE is γ-spread, we see that $\Pr_{r \leftarrow \psi_{\mathcal{R}}}[c = \mathsf{Enc}(pk, M; r)] \leq 2^{-\gamma}$, which gives $\Pr_{r \leftarrow \psi_{\mathcal{R}}}[c = \mathsf{Enc}'(pk, m; r)] \leq 2^{-\gamma}$. By averaging over (pk, sk) and $m \in \mathcal{M}'$, the proof of Lemma 3.7 is completed. □

4 Chosen-Ciphertext Secure KEM from $\mathbf{ACWC}_2$

4.1 FO Transform With Re-encryption

One can apply the Fujisaki-Okamoto transformation $\mathsf{FO}^{\perp}$ to the IND-CPA secure PKE' in Figure 4 in order to obtain an IND-CCA secure KEM. Figure 10 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', since Decap works correctly as long as Dec' is performed correctly. Regarding the IND-CCA security of KEM, we can make use of the previous results [14] and [8], which are stated in Theorem 4.1 and 4.2 below. By combining these results with Theorem 3.4 and 3.6, we can achieve the IND-CCA security of KEM in classical/quantum random oracle model, respectively. In case of quantum random oracle model (QROM), we need to further use the fact that IND-CPA implies OW-CPA generically.

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

$\underline{\mathsf{Encap}(pk)}$
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)

$\underline{\mathsf{Decap}(sk, c)}$
1: $m' := \mathsf{Dec}'(sk, c)$
- $M' = \mathsf{Dec}(sk, c)$
- $r' = \mathsf{Recover}^r(pk, M', c)$
- $m' = \mathsf{Inv}(M', \mathsf{G}(r'))$
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 10: $\mathsf{KEM} = \mathsf{FO}^{\perp}[\mathsf{PKE}', \mathsf{H}]$

Theorem 4.1 (IND-CPA of PKE' $\overset{\mathsf{ROM}}{\Longrightarrow}$ IND-CCA of KEM [14])**.** Let PKE' be a public key encryption scheme with message space $\mathcal{M}$. Let PKE' have (worst-case) correctness error δ and be (weakly) γ-spread. For any adversary $\mathcal{A}$, making at most q_{D} decapsulation, q_{H} hash queries, against the IND-CCA security of KEM, there exists an adversary $\mathcal{B}$ against the IND-CPA security of PKE' with

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

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

Theorem 4.2 (OW-CPA of PKE' $\overset{\mathsf{QROM}}{\Longrightarrow}$ IND-CCA of KEM [8])**.** 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, q_{H} (quantum) hash queries, against the IND-CCA security of KEM, there exists a quantum adversary $\mathcal{B}$ against the OW-CPA security of PKE' with

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

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 Decap algorithm to perform re-encryption to check if a ciphertext c is well-formed. Using m' as a 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. In this process, even if (m', r'') are the same as (m, r) used in Encap, this does not guarantee that $\mathsf{Enc}'(pk, m'; r'') = c$. In other words, there could exist so many other ciphertexts $\{c_i\}$ (including c as one of them), all of which are decrypted into the same m' and thus the same randomness r'' in Decap. In $\mathsf{FO}^{\perp}$ (and other FO transformations), there is still no way to find the same c (honestly) generated in Encap, other than 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 the information on sk, especially in lattice-based encryption schemes.

However, we demonstrate that $\mathsf{FO}^{\perp}$ based on ACWC_2 can eliminate such ciphertext comparison $c = \mathsf{Enc}'(pk, m'; r'')$ from Decap, and instead replace it with a simpler and much more efficient comparison $r' = r''$. We denote the new $\mathsf{FO}^{\perp}$ based on ACWC_2 as $\overline{\mathsf{FO}}^{\perp}$, which is shown in Figure 11. In $\overline{\mathsf{FO}}^{\perp}$, r' and r'' are values generated while performing Decap, where r' is the output of $\mathsf{Recover}^r(pk, M', c)$ and r'' is computed from $\mathsf{H}(m')$. Compared to $\mathsf{FO}^{\perp}$ in Figure 10, the only change is the boxed area from $c \neq \mathsf{Enc}'(pk, m'; r'')$ to $r' \neq r''$ and the remaining parts are all the same. Thus, 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 thus achieve the same level of IND-CCA security.

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

$\underline{\mathsf{Encap}(pk)}$
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)

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

Figure 11: $\mathsf{KEM} = \overline{\mathsf{FO}}^{\perp}[\mathsf{PKE}', \mathsf{H}]$

Lemma 4.3. Let PKE' and PKE be injective, and let PKE and SOTP be rigid (except for negligible rigidity errors). Then, $c = \mathsf{Enc}'(pk, m'; r'')$ in $\mathsf{FO}^{\perp}$ if and only if $r' = r''$ in $\overline{\mathsf{FO}}^{\perp}$.

Proof. Assume that $c = \mathsf{Enc}'(pk, m'; r'')$ holds in Decap of $\mathsf{FO}^{\perp}$. Because PKE' is injective, the pair of (m, r) used in Encap are the same as (m', r''). This is, the injectivity of PKE' guarantees that $m = m'$ and $r = r''$. In this case, the ciphertext c generated by Encap is expressed as $c = \mathsf{Enc}(pk, \mathsf{SOTP}(m', \mathsf{G}(r'')); r'')$.

Also, since PKE is rigid, for a ciphertext c given to Decap, the two equations $M' = \mathsf{Dec}(sk, c)$ and $r' = \mathsf{Recover}^r(pk, M', c)$ lead to $\mathsf{Enc}(pk, \mathsf{Dec}(sk, c); r') = c$. In addition, because of the rigidity of 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 $\mathsf{Enc}(pk, \mathsf{SOTP}(m', \mathsf{G}(r')); r') = c$.

Now we have two equations with respect to c generated by Enc. Since PKE is also injective, we see 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 Decap of $\overline{\mathsf{FO}}^{\perp}$. The rigidity of 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 $r' = \mathsf{Recover}^r(pk, M', c)$ lead to $\mathsf{Enc}(pk, \mathsf{Dec}(sk, c); r') = c$ and thus $\mathsf{Enc}(pk, \mathsf{Dec}(sk, c); r'') = c$. Since $\mathsf{Dec}(sk, c) = M' = \mathsf{SOTP}(m', \mathsf{G}(r''))$, we see that $\mathsf{Enc}(pk, \mathsf{SOTP}(m', \mathsf{G}(r'')); r'') = c$. Then, the left-hand side $\mathsf{Enc}(pk, \mathsf{SOTP}(m', \mathsf{G}(r'')); r'')$ can be expressed as $\mathsf{Enc}'(pk, m'; r'')$, which in turn shows $\mathsf{Enc}'(pk, m'; r'') = c$. □

5 GenNTRU[ψ_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 $\sum_{i=1}^{k}(b_i - b'_i)$.

In our NTRU construction hereafter, 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 chosen according to ψ_1.

5.1.2 NTT-Friendly Rings over Cyclotomic Trinomials

We use the polynomial ring $R_q := \mathbb{Z}_q[x]/\langle x^n - x^{n/2} + 1\rangle$, 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. Also, we use the notation '$\mathbf{h} \leftarrow R_q$' to show that a polynomial $\mathbf{h}$ is chosen uniformly at random from R_q. Later, to perform NTT over R_q, we will provide several parameter sets with respect to (n, q).

5.1.3 Other Notations

For a set $\{0,1\}^\ell$, we denote U^ℓ by the uniformly random distribution over the set $\{0,1\}^\ell$. Let $a \in \mathbb{Z}$ and $q \in \mathbb{Z}$ be a positive integer. We denote $x = a \bmod q$ the unique integer $x \in \{0, \cdots, q-1\}$ which satisfies $q|x-a$. For an odd integer q, we denote $y = a \bmod^{\pm} q$ the unique integer $y \in \{-(q-1)/2, \cdots, (q-1)/2\}$ which satisfies $q|x-a$.

5.2 Description of GenNTRU[ψ_1^n]

We define GenNTRU[ψ_1^n] relative to the distribution ψ_1^n over R_q. Since $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ should be message and randomness recoverable for our ACWC_2, Figure 12 includes two additional algorithms $\mathsf{Recover}^r$ and $\mathsf{Recover}^m$.

$\underline{\mathsf{Gen}(1^\lambda)}$
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** $(pk, sk) = (\mathbf{h}, \mathbf{f})$

$\underline{\mathsf{Enc}(\mathbf{h}, \mathbf{m} \leftarrow \psi_1^n; \mathbf{r} \leftarrow \psi_1^n)}$
1: **return** $\mathbf{c} = \mathbf{h}\mathbf{r} + \mathbf{m}$

$\underline{\mathsf{Dec}(\mathbf{f}, \mathbf{c})}$
1: **return** $\mathbf{m} = (\mathbf{c}\mathbf{f} \mod^{\pm} q) \mod^{\pm} 3$

$\underline{\mathsf{Recover}^r(\mathbf{h}, \mathbf{m}, \mathbf{c})}$
1: **return** $\mathbf{r} = (\mathbf{c} - \mathbf{m})h^{-1}$

$\underline{\mathsf{Recover}^m(\mathbf{h}, \mathbf{r}, \mathbf{c})}$
1: **return** $\mathbf{m} = \mathbf{c} - \mathbf{h}\mathbf{r}$

Figure 12: $\mathsf{GenNTRU}[\psi_1^n]$ with average-case correctness error

5.3 Cryptographic Assumptions

5.3.1 Security of GenNTRU

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 an 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 $(\mathbf{a}, \mathbf{b} = \mathbf{as} + \mathbf{e}) \in R_q \times R_q$ 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:

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

5.4 Security and Other Properties

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

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

Proof. We complete our proof through a sequence of games G_0, G_1. Let $\mathcal{A}$ be the adversary against the $\mathsf{OW\text{-}CPA}$ security experiment.
GAME G_0. In game G_0, we have the original $\mathsf{OW\text{-}CPA}$ game with $\mathsf{GenNTRU}[\psi_1^n]$. By definition, 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 game G_1, the public key $\mathbf{h}$ in the Gen is replaced by $\mathbf{h} \leftarrow R_q$. To distinguish G_1 from G_0 is equivalent to solving an $\mathsf{NTRU}_{n,q,\psi_1^n}$ problem. More precisely, there exist an adversary $\mathcal{B}$ with the same running time as that of $\mathcal{A}$ such that

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

Since $\mathbf{h} \leftarrow R_q$ is now changed to uniformly random polynomial from R_q, game 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 finishes the proof. □

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

Proof. For fixed message $\mathbf{m}$ and $\mathbf{c}$, there exist 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 $c = \mathsf{Enc}(\mathbf{h}, \mathbf{m}; \mathbf{r}_1) = \mathsf{Enc}(\mathbf{h}, \mathbf{m}; \mathbf{r}_2)$. By the assumption, $\mathbf{hr}_1 + \mathbf{m} = \mathbf{hr}_2 + \mathbf{m}$ holds. By subtracting $\mathbf{m}$ and multiplying $\mathbf{h}^{-1}$ to the both side of the equation, we can get $\mathbf{r} = \mathbf{r}'$. Therefore, there exist 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 exist only one $\mathbf{r}$ such that $\mathbf{c} = \mathsf{Enc}(\mathbf{h}, \mathbf{m}; \mathbf{r})$, $\Pr[\mathsf{Enc}(\mathbf{h}, \mathbf{m}; \mathbf{r}) = \mathbf{c}] = 2^{-n}$. Since it holds for any $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$ and $m \in \mathcal{M}$, $\mathsf{GenNTRU}[\psi_1^n]$ is n-spread. □

5.4.1 Average-Case Correctness Error

We analyze the average-case correctness error δ relative to the distribution $\psi_{\mathcal{M}} = \psi_{\mathcal{R}} = \psi_1^n$ by following the template given in [18]. We can expand $\mathbf{cf}$ in the decryption algorithm as follows:

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

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, $((\mathbf{cf})_i \mod^{\pm} q) \mod^{\pm} 3 = \mathbf{m}_i$ if the following inequality holds:

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

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

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

Then, assuming that each coefficients are independent from each other,

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

Since the coefficients of $\mathbf{m}$ have size at most 1,

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

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 $\epsilon_i' = \Pr\left[|\mathbf{gr} + \mathbf{m}\mathbf{f}'|_i \leq \frac{q-3}{6}\right]$. To do this, we need to analyze the distribution of $\mathbf{gr} + \mathbf{m}\mathbf{f}'$. By following the analysis in [18], we can check that for $i \in [n/2, n]$, the degree-i coefficient of $\mathbf{gr} + \mathbf{m}\mathbf{f}'$ is the sum of n independent random variables

$$c = ba + b'(a + a') \in \{0, \pm 1, \pm 2, \pm 3\}, \text{ where } a, b, a, b \leftarrow \psi_1. \tag{7}$$

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

$$c' = ba + b'a' \in \{0, \pm 1, \pm 2\} \text{ where } a, b, a', b' \leftarrow \psi_1. \tag{8}$$

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

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

where $\theta_j = \Pr[c = j]$ (whose distribution is shown in Table 3) and $\theta'_j = \Pr[c' = j]$ (whose 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 Eq. (6) together, we compute the average-case correctness error δ of $\mathsf{GenNTRU}[\psi_1^n]$.

± 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 $c' = ab + a'b'$

5.4.2 Rigidity and Injectivity

The rigidity of $\mathsf{GenNTRU}[\psi_1^n]$ is trivial from the algorithms Dec and $\mathsf{Recover}^r$ shown in Figure 12. 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 $\mathbf{m}_1 - \mathbf{m}_2$ have still small coefficients. For a lattice set

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

the short polynomials $\mathbf{r}_1 - \mathbf{r}_2$ and $\mathbf{m}_1 - \mathbf{m}_2$ become 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) over $\mathcal{L}_0^\perp$. It is known [9] 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 so is the injectivity of $\mathsf{GenNTRU}[\psi_1^n]$.

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 $\psi_{\mathcal{M}'} = U^n$, $\psi_{\mathcal{U}} = U^{2n}$, and $\psi_{\mathcal{M}} = \psi_1^n$. Figure 13 presents SOTP which is used for ACWC_2.

$\underline{\mathsf{SOTP}(x \in \{0,1\}^n, u \in \{0,1\}^{2n})}$	$\underline{\mathsf{Inv}(y \in \{-1,0,1\}^n, u \in \{0,1\}^{2n})}$
1: $u = (u_1, u_2) \in \{0,1\}^n \times \{0,1\}^n$	1: $u = (u_1, u_2) \in \{0,1\}^n \times \{0,1\}^n$
2: $y = (x \oplus u_1) - u_2 \in \{-1,0,1\}^n$	2: $x = (y + u_2) \oplus u_1 \in \{0,1\}^n$
3: **return** y	3: **return** x

Figure 13: SOTP

Message-Hiding and Rigidity Properties of SOTP. It is easily shown that SOTP is message-hiding because of the one-time pad property, especially for the 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 $\psi_{\mathcal{X}}$, and thus the resulting y follows ψ_1^n. In addition, the rigidity of SOTP is trivial, because $\mathsf{Inv}(y,u) = x$ implies $\mathsf{SOTP}(x,u) = y$.

6.2 CPA-NTRU+ (=PKE′)

We obtain $\mathsf{CPA\text{-}NTRU+} := \mathsf{ACWC}_2[\mathsf{GenNTRU}[\psi_1^n], \mathsf{SOTP}, \mathsf{G}]$ by applying ACWC_2 from Section 3 to $\mathsf{GenNTRU}[\psi_1^n]$. Because the underlying $\mathsf{GenNTRU}[\psi_1^n]$ provides message and randomness recoverable properties, Theorem 3.4 and 3.6 give us the IND-CPA security of the resulting CPA-NTRU+ in the classical and quantum random oracle model, respectively. Regarding the correctness error, Theorem 3.2 shows that CPA-NTRU+ has the worst-case correctness error that is almost close to the average-case correctness error of $\mathsf{GenNTRU}[\psi_1^n]$. For instance, in case where $(n,q) = (768, 3457)$, the worst-case correctness error becomes about 2^{-379}, based on the equation of Theorem 3.2 and Eq. (6).

Spreadness and Injectivity Properties of CPA-NTRU+. To achieve the IND-CCA security of the transformed KEM via $\overline{\mathsf{FO}}^{\perp}$, we need to show the spreadness and injectivity of CPA-NTRU+. The spreadness can be easily obtained by combining Lemma 3.7 with Lemma 5.4. Next, the injectivity of CPA-NTRU+ can also be proven under the assumption that the $\mathsf{NTRU}_{n,q,\psi_1^n}$ problem is infeasible, analogously to that of $\mathsf{GenNTRU}[\psi_1^n]$. More precisely, if there exist two pairs $(\mathbf{m}_1, \mathbf{r}_1)$ and $(\mathbf{m}_2, \mathbf{r}_2)$ such that $\mathsf{Enc}'(pk, \mathbf{m}_1; \mathbf{r}_1) = \mathsf{Enc}'(pk, \mathbf{m}_2; \mathbf{r}_2)$, this results in the equation $\mathbf{hr}_1 + \mathbf{m}_1 = \mathbf{hr}_2 + \mathbf{m}_2$, 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 that case, we still have two short polynomials $\mathbf{r}_1 - \mathbf{r}_2$ and $\mathbf{m}_1 - \mathbf{m}_2$ that can be a solution of approximate SVP over $\mathcal{L}_0^{\perp}$.

6.3 CCA-NTRU+ (=KEM)

Finally, we can achieve the IND-CCA secure KEM by applying $\overline{\mathsf{FO}}^{\perp}$ to the CPA-NTRU+. We denote such KEM by $\mathsf{CCA\text{-}NTRU+} := \overline{\mathsf{FO}}^{\perp}[\mathsf{CPA\text{-}NTRU+}, \mathsf{H}]$. Figure 15 presents the resultant CCA-NTRU+, which is the basis of our specification and implementation in the next section. Putting Theorem 4.1, Theorem 4.2 and Lemma 4.3 altogether, we can achieve the IND-CCA security of CCA-NTRU+. As for correctness error, CCA-NTRU+ preserves the worst-case correctness error of the underlying CPA-NTRU+.

$\underline{\mathsf{Gen}'(1^\lambda)}$

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 **then** restart
   - $(pk, sk) = (\mathbf{h} = 3\mathbf{g}\mathbf{f}^{-1}, \mathbf{f})$
2: **return** (pk, sk)

$\underline{\mathsf{Enc}'(pk, m \in \{0,1\}^n; \mathbf{r} \leftarrow \psi_1^n)}$

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** $\mathbf{c}$

$\underline{\mathsf{Dec}'(sk, \mathbf{c})}$

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

Figure 14: CPA-NTRU+

$\underline{\mathsf{Gen}(1^\lambda)}$

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
4: restart
5: **return** $(pk, sk) = (\mathbf{h} = 3\mathbf{g}\mathbf{f}^{-1}, \mathbf{f})$

$\underline{\mathsf{Encap}(pk)}$

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

$\underline{\mathsf{Decap}(sk, \mathbf{c})}$

1: $\mathbf{m} = (\mathbf{c}\mathbf{f} \mod^{\pm} q) \mod^{\pm} 3$
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 15: CCA-NTRU+

7 Algorithm Specification

7.1 Notation

Encoding and Decoding We define the function Encode_q in Algorithm 1, which compress the polynomial to $3n/2$ byte array. It assumes that each coefficient of polynomial are stored in 16-bit data type before it is compressed. The design concept of Encode_q to make it efficient when it is implemented with AVX2 instruction set. We define the function Decode_q as the inverse of Encode_q.

Algorithm 1 Encode_q

Require: Polynomial $\mathbf{f} \in R_q$
Ensure: Byte array $B = (b_0, \cdots, b_{3n/2-1})$

for i from 0 to $\lfloor n/64 \rfloor - 1$ **do**
  for j from 0 to 12 **do**
    $t_0 = f_{64j+i}$
    $t_1 = f_{64j+i+16}$
    $t_2 = f_{64j+i+32}$
    $t_3 = f_{64j+i+48}$
    $b_{96j+2i+0} = t_0$
    $b_{96j+2i+1} = (t_0 >> 8) + (t_1 << 4)$
    $b_{96j+2i+32} = t_1 >> 4$
    $b_{96j+2i+33} = t_2$
    $b_{96j+2i+64} = (t_2 >> 8) + (t_3 << 4)$
    $b_{96j+2i+65} = t_3 >> 4$
if n = 864
  for i from 0 to 7 **do**
    $t_0 = f_{832+i}$
    $t_1 = f_{832+i+8}$
    $t_2 = f_{832+i+16}$
    $t_3 = f_{832+i+24}$
    $b_{1248+2i+0} = t_0$
    $b_{1248+2i+1} = (t_0 >> 8) + (t_1 << 4)$
    $b_{1248+2i+16} = t_1 >> 4$
    $b_{1248+2i+17} = t_2$
    $b_{1248+2i+32} = t_1 >> 4$
    $b_{1248+2i+33} = t_2$
return $(b_0, \cdots, b_{3n/2-1})$

Sampling from a Binomial distribution NTRU+ uses centered binomial distribution with $\eta = 1$ to sample the coefficients of polynomials which is defined in Algorithm 3. We also define $\mathsf{BytesToBits}$ in Algorithm 2 to decide the order of sampled coefficients. $\mathsf{BytesToBits}$ help us to implement the CBD_1 and SOTP efficiently with the AVX2 instructions. We define $\mathsf{BitsToBytes}$ as the inverse of $\mathsf{BytesToBits}$ function.

Algorithm 2 BytesToBits

Require: Byte array $B = (b_0, b_1, \cdots, b_{n/4-1})$
Ensure: Polynomial $\mathbf{f} \in R_q$

$x = \lfloor n/256 \rfloor$
$y = n - 256x$
$(y_0, y_1, y_2, y_4, y_5, y_6, y_7, y_8) := bit\text{-}decompose(b)$ // $y = y_0 2^0 + \cdots y_8 2^8$
for i from 0 to $x - 1$ **do**
  for j from 0 to 7 **do**
    $t_1 = b_{32i+4j+3}|b_{32i+4j+2}|b_{32i+4j+1}|b_{32i+4j}$
    for k from 0 to 1 **do**
      for l from 0 to 16 **do**
        $f_{256i+16l+2j+k} = t_1 \& 1;$
        $t_1 >> 1;$
$c_1 = 256x$, $c_2 = 32x$
if $y_8 = 1$
  for j from 0 to 3 **do**
    $t_1 = b_{c_2+4j+3}|b_{c_2+4j+2}|b_{c_2+4j+1}|b_{c_2+4j}$
    for k from 0 to 1 **do**
      for l from 0 to 16 **do**
        $f_{c_1+8l+2j+k} = t_1 \& 1;$
        $t_1 >> 1;$
$c_1 = 256x + 128y_8$, $c_2 = 32x + 16y_8$
if $y_7 = 1$
  for j from 0 to 1 **do**
    $t_1 = b_{c_2+4j+3}|b_{c_2+4j+2}|b_{c_2+4j+1}|b_{c_2+4j}$
    for k from 0 to 1 **do**
      for l from 0 to 16 **do**
        $f_{c_1+4l+2j+k} = t_1 \& 1;$
        $t_1 >> 1;$
$c_1 = 256x + 128y_8 + 64y_7$, $c_2 = 32x + 16y_8 + 8y_7$
if $y_6 = 1$
  $t_1 = b_{c_2+3}|b_{c_2+2}|b_{c_2+1}|b_{c_2}$
  for k from 0 to 1 **do**
    for l from 0 to 16 **do**
      $f_{c_1+2l+k} = t_1 \& 1;$
      $t_1 >> 1;$
return $f_0 + f_1 x + f_2 x^2 + \cdots + f_{n-1} x^{n-1}$

Algorithm 3 $\mathsf{CBD}_1 : \mathcal{B}^{n/4} \to R_q$

Require: Byte array $B = (b_0, b_1, \cdots, b_{n/4-1})$
Ensure: Polynomial $y \in R_q$
1: $(\beta_0, \cdots, \beta_{2n-1}) := \mathsf{BytesToBits}(B)$
2: **for** i from 0 to $n-1$ **do**
3: $\quad f_i := \beta_i - \beta_{i+n}$
4: **return** $(f_0, \cdots, f_{n-1})$

Semi-generalized one time pad The function SOTP is identical to CBD_1 except that it computes exclusive or to the first half of the random bytes with the message before sampling centered binomial distribution. Therefore, the function SOTP defined in Algorithm 4 also uses the function BytesToBits as like in CBD_1. We define the function Inv in Algorithm 5 as the inverse of the function SOTP. It uses the function BitsToBytes to recover the bytes.

Algorithm 4 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: $(\beta_0, \cdots, \beta_{2n-1}) := \mathsf{BytesToBits}(B)$
2: $(m_0, \cdots, m_{n-1}) := \mathsf{BytesToBits}(m)$
3: **for** i from 0 to $n-1$ **do**
4: $\quad f_i := (m_i \oplus \beta_i) - \beta_{i+n}$
5: **return** $f_0 + f_1 x + f_2 x^2 + \cdots + f_{n-1} x^{n-1}$

Algorithm 5 Inv

Require: Polynomial $y \in R_q$
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: $(\beta_0, \cdots, \beta_{2n-1}) := \mathsf{BytesToBits}(B)$
2: **for** i from 0 to $n-1$ **do**
3: $\quad m_i := ((f_i + \beta_{i+n}) \& 1) \oplus \beta_i$
$m = \mathsf{BitsToBytes}((m_0, \cdots, m_{n-1}))$
4: **return** m

Symmetric Primitives NTRU+ uses extendable output function XOF and two different hash functions H and G as symmetric primitives. To instantiate XOF, we use AES256-CTR. To instantiate hash function G and H, we use SHA256, SHA512 and XOF as follows.

Number Theoretic Transform NTRU+ uses number theoretic transform to compute polynomial multiplications and polynomial inverses. We denote NTT the number theoretic transform function and NTT^{-1} the inverse number theoretic transform function. The detailed composition of number theoretic transform used in NTRU+ is described in section 9.1.

Algorithm 6 H

Require: Message Byte array $m = (m_0, m_1, \cdots, m_{n/8})$
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** $(b_0, \cdots b_{n/8+31})$

Algorithm 7 G

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

7.2 Specification of CCA-NTRU+

Algorithm 8 $\mathsf{Gen}(1^\lambda)$: 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** $\hat{\mathbf{f}}$, $\hat{\mathbf{g}}$ are not invertible in R_q
10: restart
11: $\hat{\mathbf{h}} = \hat{\mathbf{g}} \circ \hat{\mathbf{f}}^{-1}$
12: $pk := \mathsf{Encode}_q(\hat{\mathbf{h}})$
13: $sk := \mathsf{Encode}_q(\hat{\mathbf{f}}) || \mathsf{Encode}_q(\hat{\mathbf{h}}^{-1})$
14: **return** (pk, sk)

Algorithm 9 $\mathsf{Encap}(pk)$: encapsulation

Require: Public key $pk \in \mathcal{B}^{\lceil \log_2 q \rceil \cdot n/8}$
Ensure: Ciphertext $c \in \mathcal{B}^{\lceil \log_2 q \rceil \cdot n/8}$

$m \leftarrow \mathcal{B}^{n/8}$
$(K, \mathbf{r}) := \mathsf{H}(m)$
$\hat{\mathbf{h}} := \mathsf{Decode}_q(pk)$
$\hat{\mathbf{r}} = \mathsf{NTT}(\mathbf{r})$
$\mathbf{m} = \mathsf{SOTP}(m, \mathsf{H}(\hat{\mathbf{r}}))$
$\hat{\mathbf{m}} = \mathsf{NTT}(\mathbf{m})$
$\hat{\mathbf{c}} = \hat{\mathbf{h}} \circ \hat{\mathbf{r}} + \hat{\mathbf{m}}$
$c := \mathsf{Encode}_q(\hat{\mathbf{c}})$
return (c, K)

Algorithm 10 $\mathsf{Decap}(sk, c)$: decapsulation

Require: Secret key $sk \in \mathcal{B}^{\lceil \log_2 q \rceil \cdot n/4}$
Require: Ciphertext $c \in \mathcal{B}^{\lceil \log_2 q \rceil \cdot n/8}$
Ensure: Shared key $m \in \mathcal{B}^{32}$

$\hat{\mathbf{f}} = \mathsf{Decode}_q(sk)$
$\hat{\mathbf{c}} = \mathsf{Decode}_q(c)$
$\hat{\mathbf{h}}^{-1} = \mathsf{Decode}_q(sk + \lceil \log_2 q \rceil \cdot n/8)$
$\mathbf{m} = \mathsf{NTT}^{-1}(\hat{\mathbf{c}} \circ \hat{\mathbf{f}}) \mod^{\pm} 3$
$\hat{\mathbf{m}} = \mathsf{NTT}(\mathbf{m})$
$\hat{\mathbf{r}} = (\hat{\mathbf{c}} - \hat{\mathbf{m}}) \circ \hat{\mathbf{h}}^{-1}$ // $\mathsf{Recover}^r$
$m := \mathsf{Inv}(\mathbf{m}, \mathsf{G}(\mathsf{Encode}_q(\hat{\mathbf{r}})))$
$(K, \mathbf{r}) := \mathsf{H}(m)$
if $\mathbf{r} = \mathbf{r}'$
    return K
else
    return $\perp$

8 Parameter Sets and Security Analysis

8.1 Parameter Sets for NTRU+

We define four parameter sets for NTRU+, which we call NTRU+{576, 768, 864, 1152}. The parameters are listed in Table 5.

scheme	n	q	pk	ct	sk	sec(c)	sec(q)	$\log_2 \delta$
NTRU+576	576	3,457	864	864	1,728	115	104	-487
NTRU+768	768	3,457	1,152	1,152	2,304	164	148	-379
NTRU+864	864	3,457	1,296	1,296	2,592	188	171	-340
NTRU+1152	1,152	3,457	1,728	1,728	3,456	264	240	-260

n: polynomial degree of a ring. q: modulus. (pk, ct, sk): bytes. δ: worst-case correctness error. {sec(c), sec(q)}: classical and quantum security, respectively.

Table 5: Paramter sets for NTRU+

8.2 Concrete Security Strength

NTRU+ is constructed based on RLWE and NTRU problems. To analyze the concrete security strength of RLWE problem for NTRU+, we use the script of Kyber[22]. It can be found at `https://github.com/pq-crystals/security-estimates`. Table 6 shows classical and quantum core-SVP-hardness of the four parameter sets for NTRU+. The analysis result can also be found at `https://github.com/ntruplus/ntruplus/tree/main/scripts/security`. Since NTRU problem can be transformed into a unique SVP instance in the relevant NTRU lattice, the concrete security strength of the NTRU problem is expected to be similar to the case of RLWE problems.

	NTRU+576	NTRU+768	NTRU+864	NTRU+1152
NIST Security level	1	1	3	5

Core-SVP methodology (primal attack)

Lattice attack dim.	1054	1397	1573	2045
BKZ-blocksize	399	560	655	922
core-SVP classical hardness	116	163	191	269
core-SVP quantum hardness	105	148	173	244

Core-SVP methodology (dual attack)

Lattice attack dim.	1045	1370	1577	2009
BKZ-blocksize	395	553	645	905
core-SVP classical hardness	115	161	188	264
core-SVP quantum hardness	104	146	171	240

Table 6: Security strength and claimed security level for each parameter set

9 Performance Analysis

9.1 Implementation Consideration

We use NTT to implement a polynomial multiplication in a ring. To concretely realize NTT over the polynomial ring $\mathbb{Z}_q[x]/\langle x^n - x^{n/2} + 1\rangle$ with $n = 2^a 3^b$, we use three different types of NTT layers: Radix-2 NTT, Radix-2 NTT with cyclotomic trinomial, and Radix-3 NTT. Given a polynomial in a ring, we first factor the polynomial by adapting Radix-2 NTT layer with cyclotomic trinomial, which was first used in [18]. This can be viewed as an isomorphism between $\mathbb{Z}_q[x]/\langle x^n - x^{n/2} + 1\rangle$ and $\mathbb{Z}_q[x]/\langle x^{n/2} - \zeta\rangle \times \mathbb{Z}_q[x]/\langle x^{n/2} - (1-\zeta)\rangle$, where ζ is a primitive sixth root of unity modulus q. After that, we use Radix-3 NTT layers successively to factor each partitioned polynomial to reach a desired degree of a polynomial. At last, we use Radix-2 NTT layers until it reaches inertia degree 2 or 3 of polynomials. Note that we use Radix-3 NTT layers before Radix-2 NTT layers to minimize the size of the predefined tables required to multiply polynomials in NTT form.

n	q	Radix-2 with trinomial	Radix-3	Radix-2	inertia degree
576	3457	1	2	4	2
768	3457	1	1	4	2
864	3457	1	2	4	3
1152	3457	1	2	5	2

Table 7: Combinations of NTT layers

For the completeness, we describe the Radix-3 NTT layer used in our implementation. Radix-3 NTT layer is a ring isomorphism from $\mathbb{Z}_q[x]/\langle x^{3n} - \zeta^3\rangle$ to $\mathbb{Z}_q[x]/\langle x^n - \alpha\rangle \times \mathbb{Z}_q[x]/\langle x^n - \beta\rangle \times \mathbb{Z}_q[x]/\langle x^n - \gamma\rangle$ where $\alpha = \zeta$, $\beta = \zeta\omega$, $\gamma = \zeta\omega^2$ (ω is a primitive third root of unity modulus q). To transform $a(x) = a_0(x) + a_1(x)x^n + a_2(x)x^{2n} \in \mathbb{Z}_q[x]\langle x^{3n} - \zeta^3\rangle$($a_0(x)$, $a_1(x)$, and $a_2(x)$ are polynomials with maximum degree $n-1$) to $(\hat{a}_0(x), \hat{a}_1(x), \hat{a}_2(x)) \in \mathbb{Z}_q[x]/\langle x^n - \alpha\rangle \times \mathbb{Z}_q[x]/\langle x^n - \beta\rangle \times \mathbb{Z}_q[x]/\langle x^n - \gamma\rangle$, we need to compute following equations.

$$\hat{a}_0(x) = a_0(x) + a_1(x)\alpha + a_2(x)\alpha^2$$
$$\hat{a}_1(x) = a_0(x) + a_1(x)\beta + a_2(x)\beta^2$$
$$\hat{a}_2(x) = a_0(x) + a_1(x)\gamma + a_2(x)\gamma^2$$

Naively, we can compute above equations with $6n$ multiplications, $6n$ additions with 6 predefined values, α, α^2, β, β^2, γ, and γ^2. We can reduce the amount of computation to $4n$ multiplications, $5n$ additions, n subtractions with only 4 predefined values α, α^2, ω, and ω^2 as described in Algorithm 11. Note that ω, and ω^2 can be reused in the computation of other Radix-3 NTT layers.

Algorithm 11 Radix-3 NTT layer

Require: $a(x) = a_0(x) + a_1(x)x^n + a_2(x)x^{2n} \in \mathbb{Z}_q[x]/\langle x^{3n} - \zeta^3 \rangle$
Ensure: $(\hat{a}_0(x), \hat{a}_1(x), \hat{a}_2(x)) \in \mathbb{Z}_q[x]/\langle x^n - \alpha \rangle \times \mathbb{Z}_q[x]/\langle x^n - \beta \rangle \times \mathbb{Z}_q[x]/\langle x^n - \gamma \rangle$
1: $t_1(x) = a_1(x)\alpha$
2: $t_2(x) = a_2(x)\alpha^2$
3: $t_3(x) = t_1(x)\omega$ // $a_1(x)\beta$
4: $t_4(x) = t_2(x)\omega^2$ // $a_2(x)\beta^2$
5: $t_5(x) = t_1(x) + t_2(x)$ // $a_1(x)\alpha + a_2(x)\alpha^2$
6: $t_6(x) = t_3(x) + t_4(x)$ // $a_1(x)\beta + a_2(x)\beta^2$
7: $t_7(x) = t_5(x) + t_6(x)$ // $-a_1(x)\gamma - a_2(x)\gamma^2$
8: $\hat{a}_0(x) = a_0(x) + t_5(x)$ // $a_0(x) + a_1(x)\alpha + a_2(x)\alpha^2$
9: $\hat{a}_1(x) = a_0(x) + t_6(x)$ // $a_0(x) + a_1(x)\beta + a_2(x)\beta^2$
10: $\hat{a}_2(x) = a_0(x) - t_7(x)$ // $a_0(x) + a_1(x)\gamma + a_2(x)\gamma^2$
11: **return** $(\hat{a}_0(x), \hat{a}_1(x), \hat{a}_2(x))$

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

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

Naively, we can compute above equation with $6n$ multiplications, $6n$ additions with 6 predefined values, α^{-1}, α^{-2}, β^{-1}, β^{-2}, γ^{-1}, and γ^{-2}. We can reduce the over all computations to $4n$ multiplications, $5n$ additions, n subtractions with only 4 predefined values, α^{-1}, α^{-2}, ω, and ω^2 as described in Algorithm 12. Note that ω, and ω^2 can be reused in the computation of other Radix-3 NTT and Radix-3 Inverse NTT layers.

Algorithm 12 Radix-3 Inverse NTT layer

Require: $(\hat{a}_0(x), \hat{a}_1(x), \hat{a}_2(x)) \in \mathbb{Z}_q[x]/\langle x^n - \alpha \rangle \times \mathbb{Z}_q[x]/\langle x^n - \beta \rangle \times \mathbb{Z}_q[x]/\langle x^n - \gamma \rangle$
Ensure: $3a(x) = 3a_0(x) + 3a_1(x)x^n + 3a_2(x)x^{2n} \in \mathbb{Z}_q[x]/\langle x^{3n} - \zeta^3 \rangle$
1: $t_1(x) = \hat{a}_1(x) + \hat{a}_2(x)$
2: $t_2(x) = \hat{a}_1(x)\omega^2$ // $\hat{a}_1(x)\omega^{-1}$
3: $t_3(x) = \hat{a}_2(x)\omega$ // $\hat{a}_2(x)\omega^{-2}$
4: $t_4(x) = t_2(x) + t_3(x)$ // $\hat{a}_1(x)\omega^{-1} + a_2(x)\omega^{-2}$
5: $t_5(x) = t_1(x) + t_4(x)$ // $-a_1(x)\omega^{-2} - \hat{a}_2(x)\omega^{-4}$
6: $t_6(x) = \hat{a}_0(x) + t_4(x)$ // $\hat{a}_0(x) + \hat{a}_1(x)\omega^{-1} + a_2(x)\omega^{-2}$
7: $t_7(x) = \hat{a}_0(x) - t_5(x)$ // $\hat{a}_0(x) + a_1(x)\omega^{-2} + a_2(x)\omega^{-4}$
8: $3a_0(x) = \hat{a}_0(x) + t_1(x)$ // $\hat{a}_0(x) + \hat{a}_1(x) + \hat{a}_2(x)$
9: $3a_1(x) = t_6(x)\alpha^{-1}$ // $\hat{a}_0(x)\alpha^{-1} + \hat{a}_1(x)\beta^{-1} + \hat{a}_2(x)\gamma^{-1}$
10: $3a_2(x) = t_7(x)\alpha^{-2}$ // $a_0(x)\alpha^{-2} + a_1(x)\beta^{-2} + a_2(x)\gamma^{-2}$
11: **return** $3a(x) = 3a_0(x) + 3a_1(x)x^n + 3a_2(x)x^{2n}$

9.2 Implementation Results

All benchmarks were obtained on single core of an Intel Core i7-8700K (Coffee Lake) processor clocked at 3700 MHz. The benchmarking machine has 16 GB of RAM. Implementations were compiled with gcc version 9.4.0. The cycles listed below are the average of the cycle counts of 100,000 executions of the respective algorithms. Table 8 reports performance results of the reference and AVX2 implementation of NTRU+, along with the sizes of secret keys, public keys, and ciphertexts. The source code of NTRU+ is available for download on Github: `https://github.com/ntruplus/ntruplus`.

NTRU+576					
Size (Bytes)		**Cycles (ref)**		**Cycles (AVX2)**	
sk:	1,728	**gen:**	321,405	**gen:**	17,440
pk:	864	**encap:**	110,754	**encap:**	14,307
ct:	864	**decap:**	163,277	**decap:**	12,445
NTRU+768					
Size (Bytes)		**Cycles (ref)**		**Cycles (AVX2)**	
sk:	2,304	**gen:**	313,669	**gen:**	16,032
pk:	1,152	**encap:**	145,658	**encap:**	17,514
ct:	1,152	**decap:**	227,028	**decap:**	15,848
NTRU+864					
Size (Bytes)		**Cycles (ref)**		**Cycles (AVX2)**	
sk:	2,592	**gen:**	339,912	**gen:**	14,068
pk:	1,296	**encap:**	169,634	**encap:**	19,293
ct:	1,296	**decap:**	262,017	**decap:**	17,671
NTRU+1152					
Size (Bytes)		**Cycles (ref)**		**Cycles (AVX2)**	
sk:	3,456	**gen:**	905,131	**gen:**	42,993
pk:	1,728	**encap:**	230,448	**encap:**	25,592
ct:	1,728	**decap:**	348,076	**decap:**	24,063

Table 8: Implementation result of NTRU+

References

[1] 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.

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

[3] 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.

[4] 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/round-3-submissions.

[5] 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.

[6] 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/round-3-submissions.

[7] 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.

[8] 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.

[9] Julien Duman, Kathrin Hövelmanns, Eike Kiltz, Vadim Lyubashevsky, Gregor Seiler, and Dominique Unruh. A thorough treatment of highly-efficient NTRU instantiations. Cryptology ePrint Archive, Report 2021/1352, 2021. https://eprint.iacr.org/2021/1352.

[10] Pierre-Alain Fouque, Paul Kirchner, Thomas Pornin, and Yang Yu. BAT: small and fast KEM over NTRU lattices. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(2):240–265, 2022. https://tches.iacr.org/index.php/TCHES/article/view/9487.

[11] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554, Santa Barbara, CA, USA, August 15–19, 1999. Springer, Heidelberg, Germany.

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

[13] 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.

[14] 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.

[15] 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.

[16] 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`.

[17] Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. SWIFFT: A modest proposal for FFT hashing. In Kaisa Nyberg, editor, *Fast Software Encryption – FSE 2008*, volume 5086 of *Lecture Notes in Computer Science*, pages 54–72, Lausanne, Switzerland, February 10–13, 2008. Springer, Heidelberg, Germany.

[18] 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`.

[19] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions`.

[20] 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.

[21] John M. Schanck, Andreas Hulsing, Joost Rijneveld, and Peter Schwabe. NTRU-HRSS-KEM. Technical report, National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`.

[22] 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/round-3-submissions`.

[23] 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/round-1-submissions`.