

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

Jonghyun Kim[†]

Jong Hwan Park[‡]

December 20, 2022

Abstract

NTRU was the first practical public key encryption scheme constructed on a lattice over a polynomial-based ring and has been considered secure against significant cryptanalytic attacks in the past few decades. Despite such a long history, NTRU and its variants suffer from several drawbacks, such as the difficulty of achieving the 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 propose a new NTRU-based key encapsulation mechanism (KEM), called NTRU+, which overcomes nearly all existing drawbacks. NTRU+ is constructed based on two new generic transformations: $ACWC_2$ and \overline{FO}^\perp (a variant of the Fujisaki-Okamoto transform). $ACWC_2$ is used to easily achieve worst-case correctness error, and \overline{FO}^\perp is used to achieve chosen-ciphertext security without re-encryption. $ACWC_2$ and \overline{FO}^\perp are all defined using a randomness-recovery algorithm and an encoding method. In particular, our simple encoding method, semi-generalized one-time pad (SOTP), allows us to sample a message from a natural bit-string space with an arbitrary distribution. We provide four parameter sets for NTRU+ and 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 NTRU is crucially based on the NTRU problem [13], which has withstood significant cryptanalytic attacks over the past few decades. Such a longer history than other lattice-based problems (such as ring/module-LWE) has been considered 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, it has several distinct advantages over other lattice-based competitive schemes such as Kyber [22] and Saber [6]. Indeed, the advantages of NTRU include (1) the compact structure of a ciphertext consisting of a single polynomial, and (2) (possibly) faster encryption and decryption without the need to sample the coefficients of a public key polynomial.

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

In the context of chosen-ciphertext attacks, the NTRU (as ordinary public key encryption schemes) must guarantee extremely negligible worst-case correctness error, because otherwise information about the private key may be leaked by adversarial decryption queries (e.g., attacks [15, 5] against lattice-based encryption schemes). Roughly speaking, worst-case correctness error means the probability that decryption fails for any ciphertext that can be generated with all possible messages and randomnesses in their respective spaces. The worst-case correctness error considers that an adversary, \mathcal{A} , can *maliciously* choose messages and randomnesses, without sampling normally according to their original distributions (if possible). In the case of NTRU, the failure to decrypt a specific ciphertext $\mathbf{c} = \mathbf{hr} + \mathbf{m}$ provides \mathcal{A} the information that one of the coefficients of $p(\mathbf{g}\mathbf{r} + \mathbf{f}'\mathbf{m}) + \mathbf{m}$ is larger than or equal to $q/2$. If \mathcal{A} has control over the choice of \mathbf{r} and \mathbf{m} , even one such decryption failure may open a path to associated decryption queries to obtain more information about secret polynomials \mathbf{g} and \mathbf{f} .

When designing the NTRU, two approaches can be used 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 a high probability that all coefficients of $p(\mathbf{g}\mathbf{r} + \mathbf{f}'\mathbf{m}) + \mathbf{m}$ are less than $q/2$ for *nearly all possible* \mathbf{m} and \mathbf{r} in their spaces, although it causes inefficiency in terms of public key and ciphertext sizes. Indeed, this approach has been used by the third-round finalist NTRU [4], wherein all recommended parameters provide *perfect* correctness error (i.e., the worst-case correctness error becomes zero for all possible \mathbf{m} and \mathbf{r}). By contrast, the other approach [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 re-encrypting m by following the same sampling process as encryption. Thus, an ill-formed ciphertext that does not follow the sampling rule will always fail to be successfully decrypted, implying that \mathbf{m} and \mathbf{r} should be *honestly* sampled by \mathcal{A} according to ψ . Consequently, by disallowing \mathcal{A} to have control over \mathbf{m} and \mathbf{r} , the NTRU with an encoding method has worst-case correctness error that is close to an average-case error.

Based on the aforementioned observation, [9] proposed generic (average-case to worst-case) transformations² that make the average-case correctness error of an underlying scheme nearly close to worst-case error of a transformed scheme. One of their transformations (denoted by ACWC) is based on an encoding method called 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 to ψ , and $\mathbf{m}_2 = \text{GOTP}(m, \mathbf{G}(\mathbf{m}_1))$ using a hash function \mathbf{G} . If the GOTP acts as a sampling function

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

²They proposed two transformations called ACWC₀ 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 + FO [⊥]	ACWC ₂ + FO [⊥]
Assumptions	NTRU, RLWE	NTRU, RLWE	NTRU, RLWE
Tight reduction	Yes	No	Yes

n : polynomial degree of the ring. λ : length of the message. DPKE: deterministic public key encryption. SXY variant: SXY transformation [20] described in the finalist NTRU.

Table 1: Comparison to previous NTRU constructions

wherein output follows ψ , \mathbf{m} and \mathbf{r} are created from m following ψ , which can be verified in decryption using the FO transform. However, an ACWC based on the GOTP has two disadvantages in terms of security reduction and message distribution. First, [9] showed that ACWC converts a one-way CPA (OW-CPA) secure underlying scheme into a transformed scheme that is also OW-CPA secure, although their security reduction is loose³ by causing a security loss factor of q_G , the number of random oracle queries. Second, ACWC requires m to be sampled from 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 . Notably, it is difficult to generate exactly uniform numbers from $\{-1, 0, 1\}$ in a constant time, because of rejection sampling.

1.1 Our Results

We propose a new practical NTRU construction, called ‘NTRU+’, which overcomes the two drawbacks of the previous ACWC. To achieve our goal, we provide a new generic ACWC transformation (denoted by ACWC₂) that works with a simple encoding method. Using ACWC₂, NTRU+ achieves worst-case correctness error that is close to an average-case error of the 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 the decryption algorithm of NTRU+ faster than in the ordinary FO transform. In terms of efficiency, we use the idea of [18] to apply Number Theoretic Transform(NTT) [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 provide the implementation results for NTRU+ in each parameter set. Table 1 lists the main differences between previous NTRU constructions [4, 9] and NTRU+. In the following section, we describe our technique, focusing on these differences.

ACWC₂ Transformation with Tight Reduction. ACWC₂ is a new generic transformation that allows for the aforementioned average-case to worst-case correctness error conversion. However, to apply ACWC₂,

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

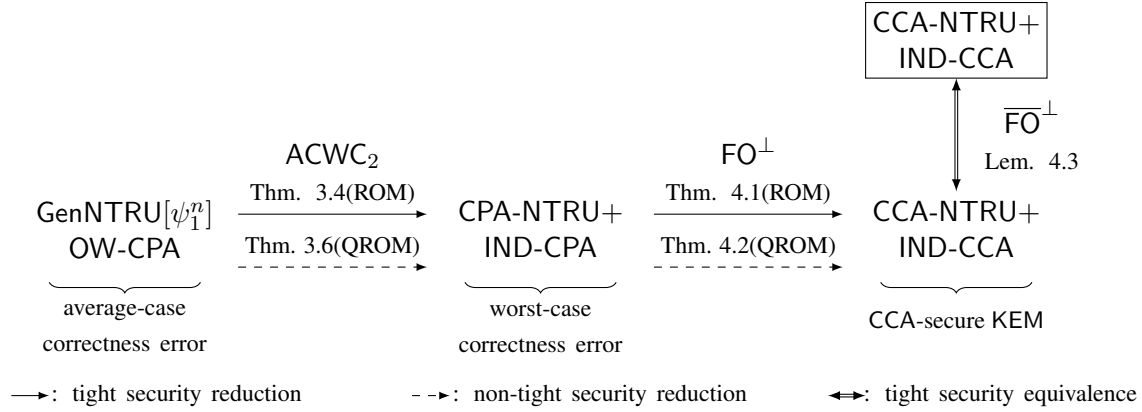


Figure 1: Overview of security reductions

an underlying scheme is required to have randomness-recoverable (RR) and message-recoverable (MR) properties, which are typical of NTRU.⁴ Additionally, $ACWC_2$ involves an encoding method, called semi-generalized one-time pad (denoted by SOTP). In contrast to the GOTP in [9], SOTP works as follows: first, a message $m \in \mathcal{M}'$ is used to sample \mathbf{r} based on ψ , and then $\mathbf{m} = \text{SOTP}(m, G(\mathbf{r}))$ whose coefficients follow ψ , using a hash function G . In decrypting a ciphertext $\mathbf{c} = \text{Enc}(pk, \mathbf{m}; \mathbf{r})$ 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 $G(\mathbf{r})$ and \mathbf{m} gives m .

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

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, which is IND-CPA secure. As with other FO-transformed schemes, the resulting scheme from $ACWC_2$ and FO^\perp is still required to perform re-encryption in the decryption process to check if (1) (\mathbf{m}, \mathbf{r}) are correctly generated from m and (2) a (decrypted) ciphertext \mathbf{c} is correctly encrypted from (\mathbf{m}, \mathbf{r}) . However, by using the RR property of the underlying scheme, we further remove the re-encryption process from FO^\perp . Instead, the more advanced transform (denoted by \overline{FO}^\perp) simply checks whether \mathbf{r} from the randomness-recovery algorithm is the same as the (new) randomness \mathbf{r}' created from m . We show that \overline{FO}^\perp is functionally identical to FO^\perp by proving that the randomness-checking process in \overline{FO}^\perp is equivalent to the re-encryption process FO^\perp . The equivalence proof relies mainly on the injectivity [14, 3] and rigidity [2] properties of the underlying schemes.

⁴In the decryption of NTRU with $pk = \mathbf{h}$, given $(pk, \mathbf{c}, \mathbf{m})$, \mathbf{r} is recovered as $\mathbf{r} = (\mathbf{c} - \mathbf{m})\mathbf{h}^{-1}$. 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 the 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 RR property seems to incur some additional decryption cost, it ends up making the decryption algorithm faster than the original FO transform. Figure 1 presents an overview of security reductions from OW-CPA to IND-CCA.

Simple SOTP with More Convenient Sampling Distributions. As mentioned previously, ACWC₂ is based on an efficient construction of SOTP that takes m and $G(\mathbf{r})$ as inputs and outputs $\mathbf{m} = \text{SOTP}(m, G(\mathbf{r}))$. In particular, computing $\mathbf{m} = \text{SOTP}(m, G(\mathbf{r}))$ requires that each coefficient of \mathbf{m} should follow ψ , while preserving the message-hiding property. To achieve this, we set ψ as the centered binomial distribution (CBD) ψ_k with $k = 1$, which is easily obtained by subtracting two uniformly random bits from each other. For a polynomial degree n and hash function $G : \{0, 1\}^* \rightarrow \{0, 1\}^{2n}$, m is chosen from the message space $\mathcal{M}' = \{0, 1\}^n$ for an arbitrary distribution (with high min-entropy) and $G(\mathbf{r}) = y_1 || y_2 \in \{0, 1\}^n \times \{0, 1\}^n$. SOTP then computes $\tilde{m} = (m \oplus y_1) - y_2$ by bitwise subtraction and assigns each subtraction value of \tilde{m} to the coefficient of \mathbf{m} . By the one-time pad property, it is easily shown that $m \oplus y_1$ becomes uniformly random in $\{0, 1\}^n$ (and thus message-hiding) and each coefficient of \mathbf{m} follows ψ_1 . Since \mathbf{r} is also sampled from a hash value of m according to ψ_1 , all sampling distributions in NTRU+ are easy to implement. We can also expect that, similar to the case of ψ_1 , the SOTP is expanded to sample a centered binomial distribution reduced modulo 3 (i.e., $\bar{\psi}_2$) by summing up and subtracting more 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, because the choice of a cyclotomic trinomial is moderate, it provides more flexibility to satisfy a certain level of security. Based on these results, we choose four parameter sets for 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 lists the comparison results between 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 Kyber[22], considering that all coefficients of each polynomial f' , \mathbf{g} , \mathbf{r} , and \mathbf{m} are drawn according to the centered binomial distribution ψ_1 , and the implementation results in Table 2 are estimated with AVX2 optimizations. We can observe 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 NAEP padding method [16]. Roughly speaking, NAEP is similar to our SOTP, but the difference is that it does not completely encode \mathbf{m} to prevent an adversary \mathcal{A} from choosing \mathbf{m} maliciously. This follows from the fact that $\mathbf{m} := \text{NAEP}(m, G(\mathbf{hr}))$ is generated by subtracting two n -bit strings m and $G(\mathbf{hr})$ from each other, that is, $m - G(\mathbf{hr})$ by bitwise subtraction, and then assigning them to the coefficients of \mathbf{m} . Since m can be maliciously chosen by \mathcal{A} in NTRUEncrypt, so can be the resulting \mathbf{m} , regardless of the hash value $G(\mathbf{hr})$.

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. Similar to NTRU+, the SXY variant requires the rigidity [2] of an underlying scheme and uses the notion of deterministic public key encryption (DPKE) where (\mathbf{m}, \mathbf{r}) are all recovered as a message during decryption. In the NTRU construction, recovering \mathbf{r} is conceptually the same as the existence of the randomness-recovery algorithm 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 errors. 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 ACWC₀ in [9].

In 2021, Duman et al. [9] proposed two generic transformations ACWC₀ and ACWC by which an average-case correctness error of an underlying scheme is almost equal to 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. ACWC₀ is simple but requires about 32 bytes of ciphertext expansion, whereas ACWC does not need to expand the ciphertext size. [9] analyzed security from ACWC₀ 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 $(\lfloor \mathbf{hr} \rfloor, G(\mathbf{r}) \oplus m)$ using a rounding operation $\lfloor \cdot \rfloor$. 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 $\lfloor \mathbf{hr} \rfloor$. 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 $\text{PKE} = (\text{Gen}, \text{Enc}, \text{Dec})$ with a message space \mathcal{M} and a randomness space \mathcal{R} consists of the following three algorithms:

- $\text{Gen}(1^\lambda)$: 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) .
- $\text{Enc}(pk, m)$: 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 $\text{Enc}(pk, m; r)$ with the used randomness $r \in \mathcal{R}$.
- $\text{Dec}(sk, c)$: 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

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

where the expectation is taken over $(pk, sk) \leftarrow \text{Gen}(1^\lambda)$, 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

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

where the expectation is taken over $(pk, sk) \leftarrow \text{Gen}(1^\lambda)$, 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 $(pk, sk) \leftarrow \text{Gen}(1^\lambda)$ and $m, m' \in \mathcal{M}$ and $r, r' \in \mathcal{R}$, we have that

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

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

Spreadness. For $(pk, sk) \leftarrow \text{Gen}(1^\lambda)$ and $m \in \mathcal{M}$, we define the min-entropy [12] of $\text{Enc}(pk, m)$ by

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

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

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

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

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

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

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

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

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

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

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

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

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

2.2 Security

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

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

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

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

Game OW-CPA	Game IND-CPA
1: $(pk, sk) \leftarrow \text{Gen}(1^\lambda)$	1: $(pk, sk) \leftarrow \text{Gen}(1^\lambda)$
2: $m \leftarrow \psi_{\mathcal{M}}$	2: $(m_0, m_1) \leftarrow \mathcal{A}_0(pk)$
3: $c^* \leftarrow \text{Enc}(pk, m)$	3: $b \leftarrow \{0, 1\}$
4: $m' \leftarrow \mathcal{A}(pk, c^*)$	4: $c^* \leftarrow \text{Enc}(pk, m_b)$
5: return $\llbracket m = m' \rrbracket$	5: $b' \leftarrow \mathcal{A}_1(pk, c^*)$
	6: return $\llbracket b = b' \rrbracket$

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 $\text{KEM} = (\text{Gen}, \text{Encap}, \text{Decap})$ with a key space \mathcal{K} consists of the following three algorithms:

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

Correctness. We say that KEM has correctness error δ if

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

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

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

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

Game IND-CCA	<u>Decap($c \neq c^*$)</u>
1: $(pk, sk) \leftarrow \text{Gen}(1^\lambda)$	1: return $\text{Decap}(sk, c)$
2: $(K_0, c^*) \leftarrow \text{Encap}(pk)$	
3: $K_1 \leftarrow \mathcal{K}$	
4: $b \leftarrow \{0, 1\}$	
5: $b' \leftarrow \mathcal{A}^{\text{Decap}}(pk, c^*, K_b)$	
6: return $\llbracket b = b' \rrbracket$	

Figure 3: Game IND-CCA for KEM

3 ACWC₂ Transformation

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

3.1 SOTP

First, we define an semi-generalized one-time pad SOTP as follows:

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

1. Decoding: There exists an efficient inversion algorithm Inv such that for all $x \in \mathcal{X}, u \in \mathcal{U}, \text{Inv}(\text{SOTP}(x, u), u) = x$.
2. Message-hiding: For all $x \in \mathcal{X}$, the random variable $\text{SOTP}(x, u)$, 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 $\text{SOTP}(\text{Inv}(y, u), u) = y$.

In contrast to the GOTP defined in [9], SOTP does not need to have an additional *randomness-hiding* property, which requires that the output $y = \text{SOTP}(x, u)$ follows the distribution $\psi_{\mathcal{Y}}$ and simultaneously does not leak any information about randomness u . The absence of such additional property allows us to design the SOTP more flexibly and efficiently than the GOTP. By contrast, 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 , $\text{Inv}(y, u) = x$ implies that $\text{SOTP}(x, u) = y$.

3.2 ACWC₂

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

<u>Gen'(1^λ)</u>	
1: $(pk, sk) := \text{Gen}(1^\lambda)$	
2: return (pk, sk)	
<u>Enc'(pk, m ∈ M'; r ← ψ_R)</u>	<u>Dec'(sk, c)</u>
1: $M := \text{SOTP}(m, G(r))$	1: $M := \text{Dec}(sk, c)$
2: $c := \text{Enc}(pk, M; r)$	2: $r := \text{Recover}^r(pk, M, c)$
3: return c	3: $m := \text{Inv}(M, G(r))$
	4: return m

Figure 4: ACWC₂[PKE, SOTP, G]

As the 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 $\text{Dec}(sk, c)$ in Dec' . This means that the correctness error of PKE' is determined by the selections of $M \in \mathcal{M}$ and $r \in \mathcal{R}$. We can 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 $\psi_{\mathcal{M}}$. Here, we can consider the SOTP as a sampling function using an internal randomness $G(r)$ while hiding m . Eventually, both M and r are chosen according to their respective initially intended distributions. This is the same idea as in ACWC and the proof strategy of Theorem 3.2 is essentially the same as that of [9] (Lemma 3.6 therein), except for slight modifications to the message distribution.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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 fractions except the e^{-c} of random functions G , we have that $\forall m \in \mathcal{M}'$,

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

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

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

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

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

Theorem 3.4 (OW-CPA of PKE $\xrightarrow{\text{ROM}}$ IND-CPA of ACWC₂[PKE, SOTP, G]). Let PKE be a public key encryption scheme with RR and MR properties, and injectivity error μ . For any adversary \mathcal{A} against the IND-CPA security of ACWC₂[PKE, SOTP, G], making at most q_G random oracle queries, there exists an adversary \mathcal{B} against the OW-CPA security of PKE with

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

where the running time of \mathcal{B} is about $\text{Time}(\mathcal{A}) + O(q_G)$.

$\mathcal{B}(pk, c^*)$ 1: $\mathcal{L}_G, \mathcal{L}_r := \emptyset$ 2: $b \leftarrow \{0, 1\}$ 3: $(m_0, m_1) \leftarrow \mathcal{A}_0^G(pk)$ 4: $b' \leftarrow \mathcal{A}_1^G(pk, c^*)$ 5: return $M \leftarrow \psi_{\mathcal{M}}$	$\mathcal{G}(r)$ // As simulated by \mathcal{B} 1: if $\exists (r, u) \in \mathcal{L}_G$ 2: return u 3: else 4: $u \leftarrow \psi_{\mathcal{U}}$ 5: $\mathcal{L}_G := \mathcal{L}_G \cup \{(r, u)\}$ 6: return u $\mathcal{B}(pk, r, c^*)$ // Find (r^*, M^*) 1: if $M := \text{Recover}^m(pk, r, c^*) \in \mathcal{M}$ 2: return M 3: abort
---	--

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

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

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 $\text{ACWC}_2[\text{PKE}, \text{SOTP}, G]$.

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

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

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 $\text{Recover}^m(pk, r_1^*, c^*)$ and $\text{Recover}^m(pk, r_2^*, c^*) \in \mathcal{M}$. This leads to breaking the injectivity of the PKE. Thus, we have

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

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

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

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

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

which concludes the proof. \square

Theorem 3.5 (Classical O2H, Theorem 3 from the eprint version of [1]). Let $S \subset \mathcal{R}$ be random. Let G and F be random functions satisfying $\forall r \notin S : G(r) = F(r)$. Let z be a random classical value (S, G, F, z may have an arbitrary joint distribution). Let \mathcal{C} be a quantum oracle algorithm with query depth q_G , expecting input z . Let \mathcal{D} be the algorithm that, on input z samples a uniform i from $\{1, \dots, q_G\}$, 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

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

Theorem 3.6 (OW-CPA of PKE $\xrightarrow{\text{QROM}}$ IND-CPA of $\text{ACWC}_2[\text{PKE}, \text{SOTP}, G]$). Let PKE be a public key encryption scheme with RR and MR properties, and injectivity error μ . For any quantum adversary \mathcal{A} against the IND-CPA security of $\text{ACWC}_2[\text{PKE}, \text{SOTP}, G]$ with a query depth at most q_G , there exists a quantum adversary \mathcal{B} against the OW-CPA security of PKE with

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

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

Proof. To prove this theorem, we use a sequence of games G_0 to G_7 defined in Figures 7 to 9, and Theorem 3.5. Before applying Theorem 3.5, we change G_0 to G_2 . Subsequently, we apply Theorem 3.5 to G_2 and G_3 . A detailed explanation of the security proof is provided in the followings.

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

Figure 7: Game G_0 for the proof of Theorem 3.6

GAME G_0 . G_0 (see Figure 7) is the original IND-CPA game with $\text{ACWC}_2[\text{PKE}, \text{SOTP}, G]$. By definition, we have

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

Games G_1 - G_5		$\mathcal{C}^G(r, u)$	
1: $G \leftarrow (\mathcal{R} \rightarrow \{0, 1\}^\lambda)$	// G_1	1: $(pk, sk) \leftarrow \text{Gen}(1^\lambda)$	
2: $r \leftarrow \psi_{\mathcal{R}}$		2: $(m_0, m_1) \leftarrow \mathcal{A}_0^G(pk)$	
3: $u := G(r)$	// G_1	3: $b \leftarrow \{0, 1\}$	// G_1 - G_4
4: $F \leftarrow (\mathcal{R} \rightarrow \{0, 1\}^\lambda)$	// G_2 - G_5	4: $M = \text{SOTP}(m_b, u)$	// G_1 - G_4
5: $u \leftarrow \psi_{\mathcal{U}}$	// G_2 - G_5	5: $M \leftarrow \psi_{\mathcal{M}}$	// G_5
6: $G := F(r := u)$	// G_2 - G_5	6: $c^* \leftarrow \text{Enc}(pk, M; r)$	
7: $w \leftarrow \mathcal{C}^G(r, u)$	// G_1 - G_2	7: $b' \leftarrow \mathcal{A}_1^G(pk, c^*)$	
8: $w \leftarrow \mathcal{C}^F(r, u)$	// G_3	8: return $\llbracket b = b' \rrbracket$	
9: $T \leftarrow \mathcal{D}^F(r, u)$	// G_4 - G_5	$\mathcal{D}^F(r, u)$	
10: return w	// G_1 - G_3	1: $i \leftarrow \{1, \dots, q_G\}$	
11: return $r \in T$	// G_4 - G_5	2: Run $\mathcal{C}^F(r, u)$ till i -th query	
		3: $T \leftarrow$ measure F-query	
		4: return T	

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

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

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

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

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

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

$$|\Pr[G_2^{\mathcal{A}} \Rightarrow 1] - \Pr[G_3^{\mathcal{A}} \Rightarrow 1]| \leq 2q_G \sqrt{\Pr[G_4 \Rightarrow 1]}.$$

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

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

GAME G_4 and G_5 . We define G_4 according to Theorem 3.5. In addition, we define G_5 by changing the way M is calculated. Instead of computing $M = \text{SOTP}(m_b, u)$, we sample $M \leftarrow \psi_{\mathcal{M}}$. By contrast, in G_4 , since u is sampled from $\psi_{\mathcal{U}}$ and used only for computing $\text{SOTP}(m_b, u)$, the message-hiding property of SOTP shows that $M = \text{SOTP}(m_b, u)$ follows the distribution $\psi_{\mathcal{M}}$. Therefore,

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

GAME G_6 . We define G_6 by rearranging G_5 , as shown in Figure 9. As the changes are only conceptual,

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

Game G_6 - G_7	$\mathcal{E}(pk, c^*)$
1: $(pk, sk) \leftarrow \text{Gen}(1^\lambda)$	1: $i \leftarrow \{1, \dots, q_G\}$
2: $r \leftarrow \psi_{\mathcal{R}}$	2: Run until i-th F-query:
3: $M \leftarrow \psi_{\mathcal{M}}$	3: $\mathcal{A}_1^F(pk)$
4: $c^* \leftarrow \text{Enc}(pk, M; r)$	4: $\mathcal{A}_2^F(pk, c^*)$
5: $T \leftarrow \mathcal{E}(pk, c^*)$	// G_6 5: $T \leftarrow$ measure F-query
6: $M' \leftarrow \mathcal{B}(pk, c^*)$	// G_7 6: return T
7: return $r \in T$	// G_6 $\mathcal{B}(pk, c^*)$
8: return $\llbracket M = M' \rrbracket$	// G_7 1: $T \leftarrow \mathcal{E}(pk, c^*)$
	2: for $r \in T$ do
	3: if $M = \text{Recover}^m(pk, r, c^*) \in \mathcal{M}$
	4: return M
	5: return $M \leftarrow \psi_{\mathcal{M}}$

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

GAME G_7 . G_7 is defined by Algorithm \mathcal{B} , as shown in Figure 9, moving from G_6 . G_7 is the same as G_6 , except for the case in which there are two distinct $r, r' \in T$ such that $\text{Recover}^m(pk, r, c^*)$, $\text{Recover}^m(pk, r', c^*) \in \mathcal{M}$. If this occurs, the injectivity of PKE is broken. Thus, we have

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

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

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

Combining all (in)equalities and bounds, we have

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

which concludes the proof. \square

Lemma 3.7. If PKE is γ -spread, then so is $\text{PKE}' = \text{ACWC}_2[\text{PKE}, \text{SOTP}, \text{G}]$.

Proof. For a fixed key pair (pk, sk) and a fixed m (with respect to PKE'), we consider the probability that $\Pr_{r \leftarrow \psi_{\mathcal{R}}}[c = \text{Enc}'(pk, m; r)]$ for every possible ciphertext c . Whenever $r \leftarrow \psi_{\mathcal{R}}$, the equation $c = \text{Enc}'(pk, m; r)$ is equivalently transformed into $c = \text{Enc}(pk, M; r)$, where $M = \text{SOTP}(m, \text{G}(r))$ is a message and c is a possible ciphertext with respect to PKE. Since PKE is γ -spread, we observe that $\Pr_{r \leftarrow \psi_{\mathcal{R}}}[c = \text{Enc}(pk, M; r)] \leq 2^{-\gamma}$, which yields $\Pr_{r \leftarrow \psi_{\mathcal{R}}}[c = \text{Enc}'(pk, m; r)] \leq 2^{-\gamma}$. By averaging over (pk, sk) and $m \in \mathcal{M}'$, the proof of Lemma 3.7 is completed. \square

4 Chosen-Ciphertext Secure KEM from ACWC_2

4.1 FO Transform with Re-encryption

One can apply the Fujisaki-Okamoto transformation FO^\perp to the IND-CPA secure PKE' , shown in Figure 4, to obtain an IND-CCA secure KEM. Figure 10 shows the resultant KEM $:= \text{FO}^\perp[\text{PKE}', \text{H}] =$

(Gen, Encap, Decap), where H is a hash function (modeled as a random oracle). Regarding the correctness error of KEM, KEM preserves the worst-case correctness error of PKE', as Decap works correctly as long as Dec' is performed correctly. Regarding the IND-CCA security of KEM, we can use the previous results [14] and [8], which are stated in Theorems 4.1 and 4.2, respectively. By combining these results with Theorems 3.4 and 3.6, we can achieve the IND-CCA security of KEM in the classical/quantum random oracle model. In the case of the quantum random oracle model (QROM), we need to further use the fact that IND-CPA generically implies OW-CPA.

<p><u>Gen(1^λ)</u></p> <p>1: $(pk, sk) := \text{Gen}'(1^\lambda)$</p> <p>2: return (pk, sk)</p> <p><u>Encap(pk)</u></p> <p>1: $m \leftarrow \mathcal{M}$</p> <p>2: $(r, K) := \text{H}(m)$</p> <p>3: $c := \text{Enc}'(pk, m; r)$</p> <p style="padding-left: 20px;">- $M := \text{SOTP}(m, G(r))$</p> <p style="padding-left: 20px;">- $c := \text{Enc}(pk, M; r)$</p> <p>4: return (K, c)</p>	<p><u>Decap(sk, c)</u></p> <p>1: $m' := \text{Dec}'(sk, c)$</p> <p style="padding-left: 20px;">- $M' = \text{Dec}(sk, c)$</p> <p style="padding-left: 20px;">- $r' = \text{Recover}^r(pk, M', c)$</p> <p style="padding-left: 20px;">- $m' = \text{Inv}(M', G(r'))$</p> <p>2: $(r'', K') := \text{H}(m')$</p> <p>3: if $m' = \perp$ or $c \neq \text{Enc}'(pk, m'; r'')$</p> <p>4: return \perp</p> <p>5: else</p> <p>6: return K'</p>
---	--

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

Theorem 4.1 (IND-CPA of PKE' $\xrightarrow{\text{QROM}}$ 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

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

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

Theorem 4.2 (OW-CPA of PKE' $\xrightarrow{\text{QROM}}$ 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

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

where $q := 2(q_H + q_D)$ and $\text{Time}(\mathcal{B}) \approx \text{Time}(\mathcal{A}) + O(q_H \cdot q_D \cdot \text{Time}(\text{Enc}) + q^2)$.

4.2 FO-Equivalent Transform Without Re-encryption

The aforementioned FO^\perp requires the Decap algorithm to perform re-encryption to check if ciphertext c is well-formed. Using m' as a result of $\text{Dec}'(sk, c)$, a new randomness r'' is obtained from $\text{H}(m')$, and $\text{Enc}'(pk, m'; r'')$ is computed and compared with the (decrypted) ciphertext c . In this process, even if (m', r'') is the same as (m, r) used in Encap, it does not guarantee that $\text{Enc}'(pk, m'; r'') = c$. In other words, there could exist many other ciphertexts $\{c_i\}$ (including c as one of them), all of which are decrypted into

the same m' and thus have the same randomness r'' in Decap. In FO^\perp (and other FO transformations), there is still no way to find the same c (honestly) generated in Encap, other than by comparing $\text{Enc}'(pk, m'; r'')$ and c . In the context of chosen-ciphertext attacks, it is well known that decapsulation queries using $\{c_i\}$ can leak information on sk , particularly in lattice-based encryption schemes.

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

<u>Gen(1^λ)</u>	<u>Decap(sk, c)</u>
1: $(pk, sk) := \text{Gen}'(1^\lambda)$	1: $m' := \text{Dec}'(sk, c)$
2: return (pk, sk)	- $M' = \text{Dec}(sk, c)$
<u>Encap(pk)</u>	- $r' = \text{Recover}^r(pk, M', c)$
1: $m \leftarrow \mathcal{M}$	- $m' = \text{Inv}(M', G(r'))$
2: $(r, K) := \text{H}(m)$	2: $(r'', K') := \text{H}(m')$
3: $c := \text{Enc}'(pk, m; r)$	3: if $m' = \perp$ or $r' \neq r''$
- $M := \text{SOTP}(m, G(r))$	4: return \perp
- $c := \text{Enc}(pk, M; r)$	5: else
4: return (K, c)	6: return K'

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

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

Proof. Assume that $c = \text{Enc}'(pk, m'; r'')$ holds in the Decap of FO^\perp . Because PKE' is injective, the pair (m, r) used in Encap is the same as (m', r'') . Therefore, ciphertext c generated by Encap is expressed as $c = \text{Enc}(pk, \text{SOTP}(m', G(r'')); r'')$.

Furthermore, because PKE is rigid, for a ciphertext c given to Decap, the two equations $M' = \text{Dec}(sk, c)$ and $r' = \text{Recover}^r(pk, M', c)$ lead to $\text{Enc}(pk, \text{Dec}(sk, c); r') = c$. In addition, because of the rigidity of the SOTP , the equation $m' = \text{Inv}(M', G(r'))$ implies $M' = \text{SOTP}(m', G(r'))$. Thus, using $\text{Dec}(sk, c) = M' = \text{SOTP}(m', G(r'))$, we can express the ciphertext c in Decap as $\text{Enc}(pk, \text{SOTP}(m', G(r'))); r') = c$.

We now have two equations with respect to c generated by Enc. Because PKE is also injective, we observe that $\text{SOTP}(m', G(r')) = \text{SOTP}(m', G(r''))$, and $r' = r''$, as required.

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

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

5.1 Notations

5.1.1 Centered Binomial Distribution ψ_k

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

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

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

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 . In addition, we use the notation ' $\mathbf{h} \leftarrow R_q$ ' to show that polynomial \mathbf{h} is chosen uniformly at random from R_q . Subsequently, to perform NTT over R_q , we provide several parameter sets with respect to (n, q) .

5.1.3 Other Notations

Let U be a uniformly random distribution over $\{0, 1\}$. We denote U^ℓ by the uniformly random distribution over the set $\{0, 1\}^\ell$. We use the notation ' $u \leftarrow U^\ell$ ' to represent that each bit of u is drawn according to distribution U . Let $a \in \mathbb{Z}$ and $q \in \mathbb{Z}$ be positive integers. We denote $x = a \bmod q$ the unique integer $x \in \{0, \dots, q-1\}$, which satisfies $q|x - a$. For an odd integer q , we denote $y = a \bmod^{\pm} q$ the unique integer $y \in \{-(q-1)/2, \dots, (q-1)/2\}$, which satisfies $q|x - a$.

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

We define GenNTRU $[\psi_1^n]$ relative to the distribution ψ_1^n over R_q . Since PKE = (Gen, Enc, Dec) should be MR and RR for our ACWC₂, Figure 12 shows two additional algorithms Recover^r and Recover^m.

<p><u>Gen(1^λ)</u></p> <ol style="list-style-type: none"> 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 $(pk, sk) = (\mathbf{h}, \mathbf{f})$ 	<p><u>Enc($\mathbf{h}, \mathbf{m} \leftarrow \psi_1^n; \mathbf{r} \leftarrow \psi_1^n$)</u></p> <ol style="list-style-type: none"> 1: return $\mathbf{c} = \mathbf{h}\mathbf{r} + \mathbf{m}$ <p><u>Dec(\mathbf{f}, \mathbf{c})</u></p> <ol style="list-style-type: none"> 1: return $\mathbf{m} = (\mathbf{c}\mathbf{f} \bmod^{\pm} q) \bmod^{\pm} 3$ <p><u>Recover^r($\mathbf{h}, \mathbf{m}, \mathbf{c}$)</u></p> <ol style="list-style-type: none"> 1: return $\mathbf{r} = (\mathbf{c} - \mathbf{m})\mathbf{h}^{-1}$ <p><u>Recover^m($\mathbf{h}, \mathbf{r}, \mathbf{c}$)</u></p> <ol style="list-style-type: none"> 1: return $\mathbf{m} = \mathbf{c} - \mathbf{h}\mathbf{r}$
---	---

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

5.3 Security and Other Properties

5.3.1 Cryptographic Assumptions

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

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

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

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

5.3.2 Security Proofs

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

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

Proof. We complete our proof through a sequence of games G_0 to G_1 . Let \mathcal{A} be the adversary against the OW-CPA security experiment.

GAME G_0 . In G_0 , we have the original OW-CPA game with $\text{GenNTRU}[\psi_1^n]$. By definition, we have that

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

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

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

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

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

Combining all the probabilities completes the proof. \square

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

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

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

5.3.3 Average-Case Correctness Error

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

$$\mathbf{c}\mathbf{f} = (\mathbf{h}\mathbf{r} + \mathbf{m})\mathbf{f} = (3\mathbf{g}\mathbf{f}^{-1}\mathbf{r} + \mathbf{m})(3\mathbf{f}' + 1) = 3(\mathbf{g}\mathbf{r} + \mathbf{m}\mathbf{f}') + \mathbf{m}.$$

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

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

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

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

Then, assuming that each coefficient is independent,

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

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

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

Therefore,

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

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

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

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

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

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

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

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

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

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

± 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 $c' = ab + a'b'$

5.3.4 Rigidity and Injectivity

The rigidity of $\text{GenNTRU}[\psi_1^n]$ is trivial from the algorithms Dec and Recover^r shown in Figure 12. The injectivity of $\text{GenNTRU}[\psi_1^n]$ 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 $\text{Enc}(\mathbf{h}, \mathbf{m}_1; \mathbf{r}_1) = \text{Enc}(\mathbf{h}, \mathbf{m}_2; \mathbf{r}_2)$, the equality indicates that $(\mathbf{r}_1 - \mathbf{r}_2)\mathbf{h} + (\mathbf{m}_1 - \mathbf{m}_2) = \mathbf{0}$, where $\mathbf{r}_1 - \mathbf{r}_2$ and $\mathbf{m}_1 - \mathbf{m}_2$ still have small coefficients of length, at most $2\sqrt{n}$. For a lattice set

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

$(\mathbf{r}_1 - \mathbf{r}_2, \mathbf{m}_1 - \mathbf{m}_2)$ becomes an approximate shortest vector in \mathcal{L}_0^\perp . Thus, if the injectivity is broken against $\text{GenNTRU}[\psi_1^n]$, we can solve the approximate shortest vector problem (SVP) (of length at most $2\sqrt{n}$) over \mathcal{L}_0^\perp . It is well-known [9] that the approximate SVP over \mathcal{L}_0^\perp is at least as hard as the $\text{NTRU}_{n,q,\psi_1^n}$ problem (defined above). Hence, if the $\text{NTRU}_{n,q,\psi_1^n}$ assumption holds, then injectivity of $\text{GenNTRU}[\psi_1^n]$ also holds.

6 NTRU+

6.1 Instantiation of SOTP

We introduce $\text{SOTP} : \mathcal{M}' \times \mathcal{U} \rightarrow \mathcal{M}$, where $\mathcal{M}' = \{0, 1\}^n$, $\mathcal{U} = \{0, 1\}^{2n}$, and $\mathcal{M} = \{-1, 0, 1\}^n$ relative to distributions $\psi_{\mathcal{U}} = U^{2n}$ and $\psi_{\mathcal{M}} = \psi_1^n$. Figure 13 shows the SOTP used for ACWC_2 .

$\text{SOTP}(x \in \mathcal{M}', u \leftarrow U^{2n})$ <ol style="list-style-type: none"> 1: $u = (u_1, u_2) \in \{0, 1\}^n \times \{0, 1\}^n$ 2: $y = (x \oplus u_1) - u_2 \in \{-1, 0, 1\}^n$ 3: return y 	$\text{Inv}(y \in \mathcal{M}, u \in U^{2n})$ <ol style="list-style-type: none"> 1: $u = (u_1, u_2) \in \{0, 1\}^n \times \{0, 1\}^n$ 2: $x = (y + u_2) \oplus u_1 \in \{0, 1\}^n$ 3: return x
--	---

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, particularly for part $x \oplus u_1$. That is, unless u_1 is known, the message $x \in \mathcal{M}'$ is unconditionally hidden from $y \in \mathcal{M}$. Similarly, $x \oplus u_1$ becomes uniformly random over $\{0, 1\}^n$, regardless of the message distribution $\psi_{\mathcal{M}'}$, and thus the resulting y follows ψ_1^n . In addition, the rigidity of the SOTP is trivial because $\text{Inv}(y, u) = x$ implies that $\text{SOTP}(x, u) = y$.

6.2 CPA-NTRU+ (=PKE')

We obtain CPA-NTRU+ := ACWC₂[GenNTRU[ψ_1^n], SOTP, G] by applying ACWC₂ from Section 3 to GenNTRU[ψ_1^n]. Because the underlying GenNTRU[ψ_1^n] provides MR and RR properties, Theorems 3.4 and 3.6 provide us the IND-CPA security of the resulting CPA-NTRU+ in the classical and quantum random oracle models, respectively. Regarding the correctness error, Theorem 3.2 shows that CPA-NTRU+ has worst-case correctness error that is close to the average-case correctness error of GenNTRU[ψ_1^n]. For instance, in the case where $(n, q) = (768, 3457)$, the worst-case correctness error becomes about 2^{-379} , based on the equation of Theorem 3.2 and Equation (6).

$\text{Gen}'(1^\lambda)$ <ol style="list-style-type: none"> 1: $(pk, sk) := \text{GenNTRU}[\psi_1^n].\text{Gen}(1^\lambda)$ <ul style="list-style-type: none"> - $\mathbf{f}', \mathbf{g} \leftarrow \psi_1^n$ - $\mathbf{f} = 3\mathbf{f}' + 1$ - 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) 	
$\text{Enc}'(pk, m \in \{0, 1\}^n; \mathbf{r} \leftarrow \psi_1^n)$ <ol style="list-style-type: none"> 1: $\mathbf{m} = \text{SOTP}(m, \mathbf{G}(\mathbf{r}))$ 2: $\mathbf{c} = \text{GenNTRU}[\psi_1^n].\text{Enc}(pk, \mathbf{m}; \mathbf{r})$ <ul style="list-style-type: none"> - $\mathbf{c} = \mathbf{h}\mathbf{r} + \mathbf{m}$ 3: return \mathbf{c} 	$\text{Dec}'(sk, \mathbf{c})$ <ol style="list-style-type: none"> 1: $\mathbf{m} = \text{GenNTRU}[\psi_1^n].\text{Dec}(sk, \mathbf{c})$ <ul style="list-style-type: none"> - $\mathbf{m} = (\mathbf{c}\mathbf{f} \bmod^{\pm} q) \bmod^{\pm} 3$ 2: $\mathbf{r} = \text{Recover}^r(pk, \mathbf{c}, \mathbf{m})$ <ul style="list-style-type: none"> - $\mathbf{r} = (\mathbf{c} - \mathbf{m})\mathbf{h}^{-1}$ 3: $m = \text{Inv}(\mathbf{m}, \mathbf{G}(\mathbf{r}))$ 4: return m

Figure 14: CPA-NTRU+

Spreadness and Injectivity Properties of CPA-NTRU+. To achieve IND-CCA security of the transformed KEM via $\overline{\text{FO}}^\perp$, 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 NTRU _{n, q, ψ_1^n} problem is infeasible, analogous to GenNTRU[ψ_1^n].

Specifically, if there exist two distinct pairs (m_1, \mathbf{r}_1) and (m_2, \mathbf{r}_2) such that $\text{Enc}'(pk, m_1; \mathbf{r}_1) = \text{Enc}'(pk, m_2; \mathbf{r}_2)$, this results in the equation $\mathbf{h}\mathbf{r}_1 + \mathbf{m}_1 = \mathbf{h}\mathbf{r}_2 + \mathbf{m}_2$, where $\mathbf{m}_1 = \text{SOTP}(m_1, \mathbf{G}(\mathbf{r}_1))$ and $\mathbf{m}_2 = \text{SOTP}(m_2, \mathbf{G}(\mathbf{r}_2))$. In this case, we have two short polynomials: $\mathbf{r}_1 - \mathbf{r}_2$ and $\mathbf{m}_1 - \mathbf{m}_2$, which can be a solution of approximate SVP (of length at most $2\sqrt{n}$) over \mathcal{L}_0^\perp .

6.3 CCA-NTRU+ (=KEM)

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

<p><u>Gen(1^λ)</u></p> <ol style="list-style-type: none"> 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 $(pk, sk) = (\mathbf{h} = 3\mathbf{g}\mathbf{f}'^{-1}, \mathbf{f})$ <p><u>Encap(pk)</u></p> <ol style="list-style-type: none"> 1: $m \leftarrow \{0, 1\}^n$ 2: $(\mathbf{r}, K) = \text{H}(m)$ 3: $\mathbf{m} = \text{SOTP}(m, \mathbf{G}(\mathbf{r}))$ 4: $\mathbf{c} = \mathbf{h}\mathbf{r} + \mathbf{m}$ 5: return (\mathbf{c}, K) 	<p><u>Decap(sk, \mathbf{c})</u></p> <ol style="list-style-type: none"> 1: $\mathbf{m} = (\mathbf{c}\mathbf{f} \bmod \pm q) \bmod \pm 3$ 2: $\mathbf{r} = (\mathbf{c} - \mathbf{m})\mathbf{h}^{-1}$ 3: $m = \text{Inv}(\mathbf{m}, \mathbf{G}(\mathbf{r}))$ 4: $(\mathbf{r}', K) = \text{H}(m)$ 5: if $\mathbf{r} = \mathbf{r}'$ 6: return K 7: else 8: return \perp
---	---

Figure 15: CCA-NTRU+

7 Parameter Sets and Security Analysis

7.1 Parameter Sets for NTRU+

We define four parameter sets for NTRU+, NTRU+{576, 768, 864, and 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 the ring. q : modulus. (pk, ct, sk) : bytes. $\{\text{sec}(c), \text{sec}(q)\}$: classical and quantum security, respectively. δ : worst-case correctness error.

Table 5: Paramter sets for NTRU+

	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

7.2 Concrete Security Strength

NTRU+ is constructed based on the RLWE and NTRU problems. To analyze the concrete security strength of the RLWE problem for NTRU+, we use the script of Kyber[22]. This can be found at <https://github.com/pq-crystals/security-estimates>. Table 6 lists the classical and quantum core-SVP-hardness of the four parameter sets for NTRU+. The analysis results can be found at <https://github.com/ntruplus/ntruplus/tree/main/scripts/security>. As the 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 that of RLWE problems.

8 Performance Analysis

8.1 Implementation of NTT

We use NTT to implement polynomial multiplication in a ring. To achieve 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 to cyclotomic trinomial, and Radix-3 NTT. Given a polynomial in a ring, we first factor the polynomial by adapting the Radix-2 NTT to cyclotomic trinomial layer, 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 . Subsequently, we successively use Radix-3 NTT layers to factor each partitioned polynomial to reach the desired degree of a polynomial. Finally, we use Radix-2 NTT layers until it reaches inertia degree two or three of the polynomials. We use Radix-3 NTT layers before the Radix-2 NTT layers to minimize the size of the predefined tables required to multiply polynomials in the NTT form. Appendix A describes the details of Radix-3 NTT layer.

n	q	Radix-2 to cyclotomic 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

8.2 Implementation Results

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

NTRU+576					
Size (Bytes)		Cycles (ref)		Cycles (AVX2)	
sk:	1,728	gen:	329,794	gen:	17,123
pk:	864	encap:	126,378	encap:	13,920
ct:	864	decap:	179,431	decap:	12,102
NTRU+768					
Size (Bytes)		Cycles (ref)		Cycles (AVX2)	
sk:	2,304	gen:	319,640	gen:	15,662
pk:	1,152	encap:	162,650	encap:	17,451
ct:	1,152	decap:	235,869	decap:	15,800
NTRU+864					
Size (Bytes)		Cycles (ref)		Cycles (AVX2)	
sk:	2,592	gen:	349,226	gen:	13,784
pk:	1,296	encap:	191,331	encap:	18,964
ct:	1,296	decap:	277,870	decap:	17,329
NTRU+1152					
Size (Bytes)		Cycles (ref)		Cycles (AVX2)	
sk:	3,456	gen:	916,994	gen:	42,206
pk:	1,728	encap:	252,244	encap:	24,853
ct:	1,728	decap:	362,996	decap:	23,418

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 Hülsing, 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 Hülsing, 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, Tancrede 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>.

A Radix-3 NTT Layer

For more clarity, we describe the Radix-3 NTT layer used in our implementation. The 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 $\alpha = \zeta$, $\beta = \zeta\omega$, and $\gamma = \zeta\omega^2$ (ω is a primitive third root of unity modulus q). To transform $a(x) = a_0(x) + a_1(x)x^n + a_2(x)x^{2n} \in \mathbb{Z}_q[x]/\langle x^{3n} - \zeta^3 \rangle$ (where $a_0(x)$, $a_1(x)$, 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 the following equations.

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

Naively, we can compute the aforementioned equations with $6n$ multiplications and $6n$ additions with six predefined values: α , α^2 , β , β^2 , γ , and γ^2 . We can reduce the amount of computation to $4n$ multiplications, $5n$ additions, and n subtractions with only four predefined values, α , α^2 , ω , and ω^2 , as described in Algorithm 1. Notably, ω and ω^2 can be reused in the computation of the other Radix-3 NTT layers.

Algorithm 1 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 aforementioned Radix-3 NTT layer, we need to compute the 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 the aforementioned equations with $6n$ multiplications and $6n$ additions with six predefined values: α^{-1} , α^{-2} , β^{-1} , β^{-2} , γ^{-1} , and γ^{-2} . We can reduce all the computations to $4n$ multiplications, $5n$ additions, and n subtractions with only four predefined values, α^{-1} , α^{-2} , ω , and ω^2 , as described in Algorithm 2. Notably, ω and ω^2 can be reused in the computation of other Radix-3 Inverse NTT layers.

Algorithm 2 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: $t_4(x) = t_2(x) + t_3(x)$ // $\hat{a}_1(x)\omega^{-1} + \hat{a}_2(x)\omega^{-2}$
5: $t_5(x) = t_1(x) + t_4(x)$ // $-a_1(x)\omega^{-2} - \hat{a}_2(x)\omega^{-4}$
6: $t_6(x) = \hat{a}_0(x) + t_4(x)$ // $\hat{a}_0(x) + \hat{a}_1(x)\omega^{-1} + \hat{a}_2(x)\omega^{-2}$
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** $3a(x) = 3a_0(x) + 3a_1(x)x^n + 3a_2(x)x^{2n}$
