NTRU+: Compact Construction of NTRU Using Simple Encoding Method

Authors	Jong Hwan Park (Sangmyung University) Jonghyun Kim (Korea University)
Contact	jhpark@smu.ac.kr
Homepage	https://www.ntruplus.org
Reference Code	https://github.com/ntruplus/reference Integrity Hash (SHA-256): 44fa5b5b 6edbb284 9322f41e 6b0e8088 74caa9f6 efcaa83b a72dce0d 47512615

January 30, 2026

NTRU+: Compact Construction of NTRU Using Simple Encoding Method

Jonghyun Kim* Jong Hwan Park†

January 30, 2026

Abstract

NTRU was the first practical public-key encryption scheme based on lattice problems over polynomial rings and has remained resilient against cryptanalysis for several decades. However, classical NTRU and its variants face several limitations, such as difficulty in achieving a negligible worst-case correctness error with a moderate modulus, complex message sampling such as fixed hamming weight sampling, and relatively slower performance compared to other lattice-based schemes.

In this work, we propose a new NTRU-based key encapsulation mechanism, called NTRU+, which addresses the aforementioned drawbacks. NTRU+ is constructed by sequentially applying two generic transformations, ACWC_2 and $\overline{\mathsf{FO}}^{\perp}$ (a variant of the Fujisaki–Okamoto transformation), where the former enables negligible worst-case correctness error and the latter enables chosen-ciphertext security without requiring re-encryption. Both ACWC_2 and $\overline{\mathsf{FO}}^{\perp}$ leverage a randomness-recovery algorithm unique to NTRU and a novel message-encoding method called the semi-generalized one-time pad (SOTP). In particular, SOTP supports messages sampled from natural bit-string spaces with arbitrary distributions. We provide three parameter sets for NTRU+ and present implementation results using NTT-friendly rings over cyclotomic trinomials.

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

1 Introduction

The NTRU encryption scheme [19] was introduced in 1998 by Hoffstein, Pipher, and Silverman as the first practical public-key encryption scheme based on lattice problems over polynomial rings. Its security relies on the NTRU problem [19], which has remained resilient against significant cryptanalytic attacks for over two decades. This longer history, compared to other lattice-based problems such as Ring-LWE and Module-LWE, has been regarded as an important factor in NTRU being selected as a finalist in the NIST PQC standardization process. Although the finalist NTRU [10] was not selected by NIST among the first four quantum-resistant cryptographic algorithms, it still offers several distinct advantages over other lattice-based schemes such as KYBER [34] and Saber [13]. Specifically, NTRU provides (1) a compact ciphertext structure consisting of a single polynomial and (2) potentially faster encryption and decryption without requiring coefficient sampling for the public key polynomial.

*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 the 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 $\mathbf{h} = p\mathbf{g}/(p\mathbf{f}' + 1) \in R_q$[1], where $\mathbf{g}$ and $\mathbf{f}'$ are sampled from a narrow distribution ψ, and p is a positive integer that is smaller than and co-prime to q (e.g., $p = 3$). The corresponding private key is $\mathbf{f} = p\mathbf{f}' + 1$. To encrypt a message m sampled from the message space $\mathcal{M}'$, one samples two polynomials $\mathbf{r}$ and $\mathbf{m}$, with coefficients drawn from the distribution ψ, and computes the ciphertext $\mathbf{c} = \mathbf{hr} + \mathbf{m}$ in R_q. An (efficient) encoding method may be used to encode $m \in \mathcal{M}'$ into $\mathbf{m}$ and $\mathbf{r} \in R_q$. Alternatively, one may directly sample $\mathbf{m}$ and $\mathbf{r}$ from ψ, where $\mathbf{m}$ is regarded as the message to be encrypted. To decrypt the ciphertext $\mathbf{c}$, one computes $\mathbf{cf}$ in R_q, recovers $\mathbf{m}$ by deriving the value $\mathbf{cf}$ modulo p, and (if necessary) decodes $\mathbf{m}$ to obtain the message m. The decryption of NTRU works correctly if all the coefficients of the polynomial $p(\mathbf{gr} + \mathbf{f}'\mathbf{m}) + \mathbf{m}$ are less than $q/2$. Otherwise, the decryption fails, and the probability of failure is referred to as the *correctness* (or *decryption*) *error*.

In the context of chosen-ciphertext attacks, NTRU, like other public-key encryption schemes, must guarantee a negligible worst-case correctness error. This requirement is essential to prevent the leakage of secret key information through adversarial decryption queries, as observed in attacks on lattice-based encryption schemes [12, 22]. Roughly speaking, the worst-case correctness error refers to the maximum probability of decryption failure taken over all possible messages. It reflects the possibility that an adversary $\mathcal{A}$ may *maliciously* craft messages and randomness rather than sampling them from their intended distributions. In the case of NTRU, a decryption failure on a ciphertext $\mathbf{c} = \mathbf{hr} + \mathbf{m}$ informs $\mathcal{A}$ that at least one coefficient of $p(\mathbf{gr} + \mathbf{f}'\mathbf{m}) + \mathbf{m}$ is greater than or equal to $q/2$. Therefore, if $\mathcal{A}$ has control over $\mathbf{r}$ and $\mathbf{m}$, even a single decryption failure may leak information about $\mathbf{g}$ and $\mathbf{f}'$.

When designing NTRU, two approaches can be used to achieve negligible worst-case correctness error. One is to draw $\mathbf{m}$ and $\mathbf{r}$ directly from the distribution ψ while setting the modulus q to be relatively large. Choosing a larger q guarantees, with high probability, that all coefficients of $p(\mathbf{gr} + \mathbf{f}'\mathbf{m}) + \mathbf{m}$ are less than $q/2$ for *nearly all* inputs $\mathbf{m}$ and $\mathbf{r}$, although this comes at the cost of increased key and ciphertext sizes. This approach is used by the third-round finalist NTRU [10], whose parameters achieve *perfect* correctness (i.e., the worst-case correctness error becomes zero for all possible $\mathbf{m}$ and $\mathbf{r}$). In contrast, the other approach [16] uses an encoding method in which a message $m \in \mathcal{M}'$ is used as a seed to sample $\mathbf{m}$ and $\mathbf{r}$ according to ψ. Under the Fujisaki–Okamoto (FO) transformation [17], decrypting a ciphertext $\mathbf{c}$ requires re-encrypting m by following the same sampling process used in encryption. Thus, any ill-formed ciphertext that violates the sampling rule will always fail to decrypt, implying that $\mathbf{m}$ and $\mathbf{r}$ must be *honestly* sampled by $\mathcal{A}$ according to ψ. Consequently, by preventing $\mathcal{A}$ from controlling $\mathbf{m}$ and $\mathbf{r}$, the NTRU scheme with the encoding method achieves a worst-case correctness error that is close to its average-case correctness error.

Based on the above observation, [16] proposed generic (average-case to worst-case) transformations[2] that make the average-case correctness error of an underlying scheme close to the worst-case error of the transformed scheme. One of their transformations (denoted by ACWC) is based on an encoding method called the generalized one-time pad (denoted by GOTP). Roughly speaking, ACWC works as follows: a message $m \in \mathcal{M}'$ is first used to sample $\mathbf{r}$ and $\mathbf{m}_1$ according to ψ, and then $\mathbf{m}_2 = \mathsf{GOTP}(m, \mathsf{G}(\mathbf{m}_1))$ is computed using a hash function G. Finally, $\mathbf{m}$ is constructed as $\mathbf{m}_1 \,\|\, \mathbf{m}_2$. If GOTP acts as a sampling function such that its output follows ψ, then $\mathbf{m}$ and $\mathbf{r}$ are generated from m according to ψ, which can be verified during decryption using the FO transformation. Specifically, for two inputs m and $\mathsf{G}(\mathbf{m}_1)$ that are sampled from $\{-1, 0, 1\}^\lambda$ for some integer λ, $\mathbf{m}_2 \in \{-1, 0, 1\}^\lambda$ is computed by doing the component-wise addition modulo 3 on the ternary strings m and $\mathsf{G}(\mathbf{m}_1)$. Thus, if $\mathsf{G}(\mathbf{m}_1)$ follows a uniformly random distribution over $\{-1, 0, 1\}^\lambda$, then m is hidden in $\mathbf{m}_2$ due to the one-time pad property.

[1] Alternatively, the public key can be generated as $\mathbf{h} = p\mathbf{g}/\mathbf{f}$, but we use $\mathbf{h} = p\mathbf{g}/(p\mathbf{f}' + 1)$ for more efficient decryption.

[2] They proposed two transformations, ACWC_0 and ACWC, but we focus on ACWC, which does not increase the ciphertext size.

Scheme	NTRU[10]	NTRU-B [16]	NTRU+
NTT-friendly	No	Yes	Yes
Correctness error	Perfect	Worst-case	Worst-case
$(\mathbf{m}, \mathbf{r})$-encoding	No	Yes	Yes
Message set	$\mathbf{m}, \mathbf{r} \leftarrow \{-1, 0, 1\}^n$	$m \leftarrow \{-1, 0, 1\}^\lambda$	$m \leftarrow \{0, 1\}^n$
Message distribution	Uniform/Fixed-weight	Uniform	Arbitrary
CCA transform	DPKE + SXY variant	ACWC + $\mathsf{FO}^\perp$	ACWC_2 + $\overline{\mathsf{FO}}^\perp$
Assumptions	NTRU, RLWE	NTRU, RLWE	NTRU, RLWE
Tight reduction	Yes	No	Yes

n: polynomial degree of the ring. λ: length of the message. DPKE: deterministic public-key encryption.
SXY variant: SXY transformation [33] described in the NTRU finalist.

Table 1: Comparison to previous NTRU constructions

However, an ACWC based on GOTP has two disadvantages in terms of security reduction and message distribution. First, [16] showed that ACWC converts a one-way CPA (OW-CPA) secure underlying scheme into a transformed one that remains OW-CPA secure, although the security reduction is loose[3] and causes an additional loss factor of q_{G}, the number of random oracle queries. Second, ACWC requires that even the message $m \in \mathcal{M}'$ follow a specific distribution because the security analysis of ACWC requires GOTP to have the additional randomness-hiding property, meaning that $\mathsf{G}(\mathbf{m}_1)$ should also be hidden from the output $\mathbf{m}_2$. Indeed, the NTRU instantiation from ACWC, called 'NTRU-B' [16], requires that m be chosen uniformly at random from $\mathcal{M}' = \{-1, 0, 1\}^\lambda$. Notably, it is difficult to generate exactly uniformly random samples from $\{-1, 0, 1\}$ in constant time due to rejection sampling. Therefore, it was an open problem [16] to construct a new transformation that permits a different, more easily sampled distribution of a message while relying on the same security assumptions.

1.1 Our Results

We propose a new practical NTRU construction called 'NTRU+' that addresses the two drawbacks of the previous ACWC. To achieve this, we introduce a new generic ACWC transformation, denoted as ACWC_2, which utilizes a simple encoding method. By using ACWC_2, NTRU+ achieves a worst-case correctness error close to the average-case error of the underlying NTRU. Additionally, NTRU+ requires the message m to be drawn from $\mathcal{M}' = \{0, 1\}^n$ (for a polynomial degree n), following an *arbitrary* distribution with high min-entropy, and is proven to be *tightly* secure under the same assumptions as NTRU-B, the NTRU and RLWE assumptions. To achieve chosen-ciphertext security, NTRU+ relies on a novel FO-equivalent transform without re-encryption, which makes the decryption algorithm of NTRU+ faster than in the ordinary FO transform. In terms of efficiency, we use the idea from [31] to apply the Number Theoretic Transform (NTT) to NTRU+ and therefore instantiate NTRU+ over a ring $R_q = \mathbb{Z}_q[x]/\langle f(x)\rangle$, where $f(x) = x^n - x^{n/2} + 1$ is a cyclotomic trinomial. By selecting appropriate (n, q) and ψ, we suggest three parameter sets for NTRU+ and provide the implementation results for NTRU+ in each parameter set. Table 1 lists the main differences between the previous NTRU constructions [10, 16] and NTRU+. In the following section, we describe our technique, focusing on these differences.

[3] [16] introduced the q-OW-CPA security notion, where an adversary outputs a set Q of size at most q and wins if the correct message corresponding to a challenged ciphertext is in Q. We believe that q-OW-CPA leads to a security loss of q.

	ACWC_0[16]	ACWC[16]	ACWC_2
Message encoding	No	GOTP	SOTP
Message distribution	Arbitrary	Uniform	Arbitrary
Ciphertext expansion	Yes	No	No
Transformation	OW-CPA $\rightarrow$ IND-CPA	OW-CPA $\rightarrow$ OW-CPA	OW-CPA $\rightarrow$ IND-CPA
Tight reduction	No	No	Yes
Underlying PKE	Any	Any	Injective + RR + AC-MR + VC-MR

{AC, VC}-MR: {arbitrary ciphertext, valid ciphertext} message-recoverable. RR: randomness-recoverable.

Table 2: Comparison to previous ACWC transformations

ACWC_2 Transformation with Tight Reduction. ACWC_2 is a new generic transformation that enables the aforementioned conversion from average-case to worst-case correctness error. To apply ACWC_2, the underlying scheme must satisfy injectivity, arbitrary-ciphertext and valid-ciphertext message-recoverability (AC-MR and VC-MR), and randomness-recoverability (RR) properties that are all inherent to NTRU.[4] Additionally, ACWC_2 introduces an encoding method called the semi-generalized one-time pad (denoted by SOTP). In contrast to the ACWC in [16], ACWC_2 equipped with $\mathsf{SOTP} = (\mathsf{Encode}, \mathsf{Decode})$ works as follows: first, a message $m \in \mathcal{M}'$ is used to sample $\mathbf{r}$ according to ψ, and then $\mathbf{m} = \mathsf{Encode}(m, \mathsf{G}(\mathbf{r}))$ is computed, where the coefficients follow ψ, using a hash function G. When decrypting a ciphertext $\mathbf{c} = \mathsf{Enc}(pk, \mathbf{m}; \mathbf{r})$ under a public key pk, $\mathbf{m}$ is recovered by the usual decryption algorithm, and using $\mathbf{m}$, $\mathbf{r}$ is also recovered by a randomness-recovery algorithm. Finally, applying Decode to $\mathbf{m}$ and $\mathsf{G}(\mathbf{r})$ yields the original message m.

The VC-MR property of the underlying scheme allows ACWC_2 to transform an OW-CPA secure scheme into an IND-CPA secure one without any significant security loss. The proof idea is straightforward: unless an IND-CPA adversary $\mathcal{A}$ queries the target randomness $\mathbf{r}$ to the (classical) random oracle G, $\mathcal{A}$ obtains no information about the challenge message m_b due to the message-hiding property of SOTP. However, for each query $\mathbf{r}_i$ made by $\mathcal{A}$ to G ($i = 1, \ldots, q_{\mathsf{G}}$), a reductionist can verify whether $\mathbf{r}_i$ corresponds to the OW-CPA challenge ciphertext by using the message-recovery algorithm. Consequently, the reductionist can identify the exact $\mathbf{r}_i$ among the q_{G} queries if $\mathcal{A}$ has queried it to G. In this security analysis, it is sufficient for SOTP to satisfy only the message-hiding property, making it simpler than GOTP, which must ensure both message-hiding and randomness-hiding.

Table 2 compares the previous ACWC transformations with our new ACWC_2. In addition to ACWC based on GOTP, [16] proposed another generic transformation, denoted by ACWC_0, which does not use any message-encoding method. In ACWC_0, a bit-string message m is encrypted as $(\mathsf{Enc}(pk, \mathbf{m}; \mathbf{r}), \mathsf{F}(\mathbf{m}) \oplus m)$ using a hash function F. This approach causes ciphertext expansion due to the additional term $\mathsf{F}(\mathbf{m}) \oplus m$, a redundancy that does not appear in either ACWC or ACWC_2. Like ACWC_2, ACWC_0 transforms an OW-CPA secure scheme into an IND-CPA secure one, but its security reduction is not as tight as that of ACWC_2. Furthermore, neither ACWC_0 nor ACWC_2 requires any specific message distribution, whereas ACWC requires $m \in \mathcal{M}'$ to be sampled uniformly from $\mathcal{M}'$. In terms of applicability, while ACWC_0 and ACWC are applicable to any OW-CPA secure scheme, ACWC_2 applies specifically to those that additionally satisfy injectivity, AC-MR, VC-MR, and RR.

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

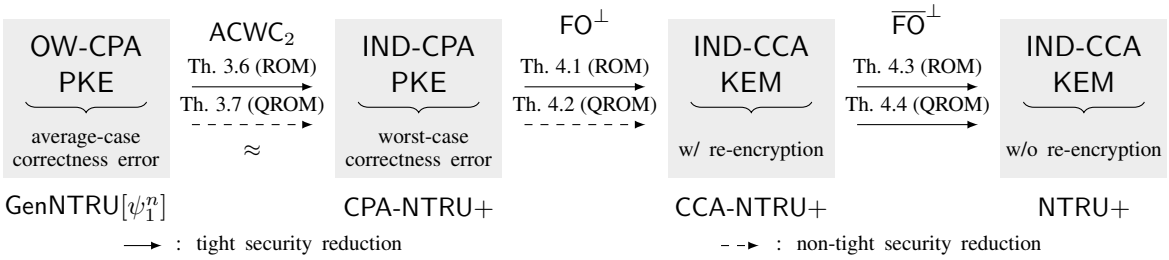

Figure 1: Overview of security reductions for KEM

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

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

NTT-Friendly Rings Over Cyclotomic Trinomials. $\mathsf{NTRU+}$ is instantiated over a polynomial ring $R_q = \mathbb{Z}_q[x]/\langle f(x)\rangle$, where $f(x) = x^n - x^{n/2} + 1$ is a cyclotomic trinomial of degree $n = 2^i 3^j$. [31] showed that, with appropriate parameterization of n and q, such a ring can also provide NTT operation essentially as fast as that over a ring $R_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$. Moreover, because the choice of a cyclotomic trinomial is moderate, it provides more flexibility to satisfy a certain level of security. Based on these results, we choose three parameter sets for $\mathsf{NTRU+}$, where the polynomial degree n of $f(x) = x^n - x^{n/2} + 1$ is set to be 768,

864, and 1152, and the modulus q is 3457 for all cases. Table 7 lists the comparison results between finalist NTRU [10], KYBER [34], and NTRU+ in terms of security and efficiency. To estimate the concrete security level of NTRU+, we use the Lattice estimator [1] for the RLWE problem and the NTRU estimator [10] for the NTRU problem, considering that all coefficients of each polynomial $\mathbf{f}'$, $\mathbf{g}$, $\mathbf{r}$, and $\mathbf{m}$ are drawn according to the centered binomial distribution ψ_1. The implementation results in Table 7 are estimated with reference and AVX2 optimizations. We can observe that NTRU+ outperforms NTRU at a similar security level.

1.2 Related Works

The first-round NTRUEncrypt [36] submission to the NIST PQC standardization process was an NTRU-based encryption scheme with the NAEP padding method [23]. Roughly speaking, NAEP is similar to our SOTP, but the difference is that it does not completely encode $\mathbf{m}$ to prevent an adversary $\mathcal{A}$ from choosing $\mathbf{m}$ maliciously. This is due to the fact that $\mathbf{m} := \mathsf{NAEP}(m, \mathsf{G}(\mathbf{hr}))$ is generated by subtracting two n-bit strings m and $\mathsf{G}(\mathbf{hr})$ from each other, i.e., $m - \mathsf{G}(\mathbf{hr})$ by bit-wise subtraction, and then assigning them to the coefficients of $\mathbf{m}$. Since m can be maliciously chosen by $\mathcal{A}$ in NTRUEncrypt, $\mathbf{m}$ can also be maliciously chosen, regardless of $\mathsf{G}(\mathbf{hr})$.

The finalist NTRU [10] was submitted as a key encapsulation mechanism (KEM) that provides four parameter sets for perfect correctness. To achieve chosen-ciphertext security, [10] relied on a variant of the SXY [33] conversion, which also avoids re-encryption during decapsulation. Similar to NTRU+, the SXY variant requires the rigidity [7] of an underlying scheme and uses the notion of deterministic public-key encryption (DPKE) where $(\mathbf{m}, \mathbf{r})$ are all recovered as a message during decryption. In the NTRU construction, the recovery of $\mathbf{r}$ is conceptually the same as the existence of the randomness-recovery algorithm RRec. Instead of removing re-encryption, the finalist NTRU needs to check whether $(\mathbf{m}, \mathbf{r})$ are selected correctly from predefined distributions.

In 2019, Lyubashevsky et al. [31] proposed an efficient NTRU-based KEM called NTTRU by applying NTT to the ring defined by a cyclotomic trinomial $\mathbb{Z}_q[x]/\langle x^n - x^{n/2} + 1\rangle$. NTTRU was based on the Dent [14] transformation without any encoding method, which resulted in an approximate worst-case correctness error of 2^{-13}, even with an average-case error of 2^{-1230}. To overcome this significant difference, NTTRU was modified to reduce the message space of the underlying scheme, while increasing the size of the ciphertext. This modification was later generalized to ACWC_0 in [16].

In 2021, Duman et al. [16] proposed two generic transformations, ACWC_0 and ACWC, which aim to make the average-case correctness error of an underlying scheme nearly equal to the worst-case error of the transformed scheme. Specifically, ACWC introduced GOTP as an encoding method to prevent $\mathcal{A}$ from adversarially choosing $\mathbf{m}$. While ACWC_0 is simple, it requires a ciphertext expansion of 32 bytes. On the other hand, ACWC does not require an expansion of the ciphertext size. The security of ACWC_0 and ACWC was analyzed in both the classical and quantum random oracle models [16]. However, their NTRU instantiation using ACWC has the drawback of requiring the message m to be chosen from a uniformly random distribution over $\mathcal{M}' = \{-1, 0, 1\}^\lambda$.

2 Preliminaries

2.1 Basic Notations

The set $\mathbb{Z}_q$ is defined as $\{-(q-1)/2, \ldots, (q-1)/2\}$, where q is a positive odd integer. Mapping an integer a from $\mathbb{Z}$ to $\mathbb{Z}_q$ uses the modulo operation, setting $x = a \bmod q$ as the unique integer in $\mathbb{Z}_q$ satisfying $q \mid (x-a)$. The polynomial ring R_q is defined as $\mathbb{Z}_q[x]/\langle f(x)\rangle$ with a polynomial $f(x)$. Cyclotomic trinomials $\Phi_{3n}(x) = x^n - x^{n/2} + 1$ where $n = 2^i \cdot 3^j$ for some positive integers i and j are used as $f(x)$ in our construction. Polynomials in R_q are denoted in non-italic bold as $\mathbf{a}$, with $\mathbf{a}_i$ as the i-th coefficient.

For sampling, $u \leftarrow X$ indicates that u is sampled uniformly at random from a set X, and $u \leftarrow D$ indicates that u is drawn according to a distribution D. The notation $u \leftarrow D^\ell$ forms a vector $u = (u_1, \ldots, u_\ell)$ with each u_i drawn independently from D. Especially, $\mathbf{a} \leftarrow D$ indicates that all coefficients of a polynomial $\mathbf{a}$ are drawn according to a distribution D. Sampling from the centered binomial distribution (CBD) ψ_k involves $2k$ bits that are independent and uniformly random, summing the first k bits and the second k bits separately, then outputting their difference.

2.2 Public-Key Encryption

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

- $\mathsf{Gen}(1^\lambda)$: The key generation algorithm Gen is a randomized algorithm that takes a security parameter 1^λ as input and outputs a pair of public/secret keys (pk, sk).
- $\mathsf{Enc}(pk, m; r)$: The encryption algorithm Enc is a randomized algorithm that takes a public key pk, a message $m \in \mathcal{M}$, and randomness $r \in \mathcal{R}$ as input and outputs a ciphertext $c \in \mathcal{C}$. We often write $\mathsf{Enc}(pk, m)$ to denote the encryption algorithm without explicitly mentioning the randomness.
- $\mathsf{Dec}(sk, c)$: The decryption algorithm Dec is a deterministic algorithm that takes a secret key sk and a ciphertext $c \in \mathcal{C}$ as input and outputs either a message $m \in \mathcal{M}$ or a special symbol $\perp \notin \mathcal{M}$ to indicate that c is not a valid ciphertext.

Correctness. We say that PKE has a (worst-case) correctness error δ [20] if

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

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

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

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

Injectivity. Injectivity of PKE is defined via the following GAME INJ, which is shown in Figure 2, and the relevant advantage of adversary $\mathcal{A}$ is

$$\mathsf{Adv}^{\mathsf{INJ}}_{\mathsf{PKE}}(\mathcal{A}) = \Pr[\mathsf{INJ}^{\mathcal{A}}_{\mathsf{PKE}} \Rightarrow 1].$$

Unlike the definition of injectivity in [8, 20], we define the injectivity in a computationally-secure sense.

GAME INJ

1: $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$
2: $(m, m', r, r') \leftarrow \mathcal{A}(pk)$
3: $c = \mathsf{Enc}(pk, m; r)$
4: $c' = \mathsf{Enc}(pk, m'; r')$
5: **return** $[\![(m, m', r, r') \in \mathcal{M}^2 \times \mathcal{R}^2 \wedge (m, r) \neq (m', r') \wedge c = c']\!]$

Figure 2: GAME INJ for PKE

Spreadness. PKE is γ-*spread* [20] if

$$\min_{m \in \mathcal{M}, (sk, pk)} \left(-\log \max_{c \in \mathcal{C}} \Pr_{r \leftarrow \psi_{\mathcal{R}}} [c = \mathsf{Enc}(pk, m; r)] \right) \geq \gamma,$$

where the minimum is taken over all key pairs that can be generated by Gen. This definition can be relaxed by considering an expectation over the choice of (pk, sk). PKE is *weakly* γ-*spread* [15] if

$$-\log \mathbb{E} \left[\max_{m \in \mathcal{M}, c \in \mathcal{C}} \Pr_{r \leftarrow \psi_{\mathcal{R}}} [c = \mathsf{Enc}(pk, m; r)] \right] \geq \gamma,$$

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

Randomness-Recoverability. PKE is defined as *randomness-recoverable* (RR) if there exists an algorithm RRec such that, for all key pairs $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$, and for any message $m \in \mathcal{M}$ and for any ciphertext $c \in \mathcal{C}$, the following condition holds:

$$\text{if } r = \mathsf{RRec}(pk, m, c) \in \mathcal{R}, \text{ then } c = \mathsf{Enc}(pk, m; r).$$

Arbitrary-Ciphertext Message-Recoverability. PKE is *arbitrary-ciphertext message-recoverable* (AC-MR) if there exists an algorithm MRec such that, for all key pairs $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$, and for any randomness $r \in \mathcal{R}$, and for anly ciphertexts $c \in \mathcal{C}$,

$$\text{if } m = \mathsf{MRec}(pk, r, c) \in \mathcal{M}, \text{ then } c = \mathsf{Enc}(pk, m; r).$$

Valid-Ciphertext Message-Recoverability. PKE is *valid-ciphertext message-recoverable* (VC-MR) if there exists an algorithm MRec such that, for all key pairs $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$, for any message $m \in \mathcal{M}$, for any randomness $r \in \mathcal{R}$, and for any ciphertexts $c \in \mathcal{C}$,

$$\text{if } c = \mathsf{Enc}(pk, m; r), \text{ then } \mathsf{MRec}(pk, r, c) = m.$$

Randomness-Uniqueness. PKE is defined as *randomness-unique* (RU) if for all key pairs $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$, and for any message $m \in \mathcal{M}$ and any randomness $r, r' \in \mathcal{R}$, the following condition holds:

$$\text{if } \mathsf{Enc}(pk, m; r) = \mathsf{Enc}(pk, m; r'), \text{ then } r = r'.$$

GAME OW-CPA

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

GAME IND-CPA

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

Figure 3: GAMES OW-CPA and IND-CPA for PKE

GAME IND-CCA

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

$\mathsf{Decap}(c \neq c^*)$

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

Figure 4: GAME IND-CCA for KEM

Definition 2.2 (OW-CPA security of PKE)**.** Let PKE = (Gen, Enc, Dec) be a public-key encryption scheme with message space $\mathcal{M}$. Onewayness under chosen-plaintext attacks (OW-CPA) for message distribution $\psi_{\mathcal{M}}$ is defined via the GAME OW-CPA, as shown in Figure 3, and the advantage function of adversary $\mathcal{A}$ is

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

Definition 2.3 (IND-CPA security of PKE)**.** Let PKE = (Gen, Enc, Dec) be a public-key encryption scheme with message space $\mathcal{M}$. Indistinguishability under chosen-plaintext attacks (IND-CPA) is defined via the GAME IND-CPA, as shown in Figure 3, and the advantage function of adversary $\mathcal{A}$ is

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

2.3 Key Encapsulation Mechanism

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

- $\mathsf{Gen}(1^\lambda)$: The key generation algorithm Gen is a randomized algorithm that takes a security parameter λ as input and outputs a pair of public key and secret key, (pk, sk).
- $\mathsf{Encap}(pk)$: The encapsulation algorithm Encap is a randomized algorithm that takes a public key pk as input, and outputs a ciphertext c and a key $K \in \mathcal{K}$.
- $\mathsf{Decap}(sk, c)$: The decryption algorithm Decap is a deterministic algorithm that takes a secret key sk and ciphertext c as input, and outputs either a key $K \in \mathcal{K}$ or a special symbol $\perp \notin \mathcal{K}$ to indicate that c is not a valid ciphertext.

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

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

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

Definition 2.5 (IND-CCA security of KEM)**.** Let $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Encap}, \mathsf{Decap})$ be a key encapsulation mechanism with a key space $\mathcal{K}$. Indistinguishability under chosen-ciphertext attacks (IND-CCA) is defined via the GAME $\mathsf{IND\text{-}CCA}$, as shown in Figure 4, and the advantage function of adversary $\mathcal{A}$ is as follows:

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

2.4 Complexity Assumptions

This section outlines complexity assumptions used in $\mathsf{NTRU}+$. Specifically, it introduces the NTRU and RLWE problems. Unlike the RLWE problem used in ElGamal-type schemes [2], RLWE here is defined in the computational sense.

Definition 2.6 (The NTRU problem [19])**.** Let ψ be a distribution over R_q. The NTRU problem $\mathsf{NTRU}_{n,q,\psi}$ is to distinguish $\mathbf{h} = \mathbf{g}(p\mathbf{f}' + 1)^{-1} \in R_q$ from $\mathbf{u} \in R_q$, where $\mathbf{f}', \mathbf{g} \leftarrow \psi$ and $\mathbf{u} \leftarrow R_q$. The advantage of adversary $\mathcal{A}$ in solving $\mathsf{NTRU}_{n,q,\psi}$ is defined as follows:

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

Definition 2.7 (The RLWE problem [30])**.** Let ψ be a distribution over R_q. The RLWE problem $\mathsf{RLWE}_{n,q,\psi}$ is to find $\mathbf{s}$ from $(\mathbf{a}, \mathbf{b} = \mathbf{as} + \mathbf{e}) \in R_q \times R_q$, where $\mathbf{a} \leftarrow R_q$, $\mathbf{s}, \mathbf{e} \leftarrow \psi$. The advantage of an adversary $\mathcal{A}$ in solving $\mathsf{RLWE}_{n,q,\psi}$ is defined as follows:

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

2.5 Auxiliary Lemmas for the Security Proofs

Lemma 2.8 (Fundamental lemma of game-playing [6, Lemma 1])**.** Let G and H be identical-until-bad games, meaning that both games maintain a boolean flag bad initially set to false and behave identically until bad is set to true. Then, for any adversary $\mathcal{A}$,

$$\left|\Pr[G^{\mathcal{A}} \Rightarrow 1] - \Pr[H^{\mathcal{A}} \Rightarrow 1]\right| \leq \Pr[G^{\mathcal{A}} \text{ sets } \mathsf{bad}].$$

Lemma 2.9 (Classical O2H, Theorem 3 from the eprint version of [3])**.** Let $S \subset \mathcal{R}$ be random. Let G and F be random functions satisfying $\forall r \notin S : \mathsf{G}(r) = \mathsf{F}(r)$. Let z be a random classical value (S, G, F, z may have an arbitrary joint distribution). Let $\mathcal{C}$ be a quantum oracle algorithm with query depth q_{G}, expecting input z. Let $\mathcal{D}$ be the algorithm that, on input z, samples a uniform i from $\{1, ..., q_{\mathsf{G}}\}$, runs $\mathcal{C}$ right before its i-th query to F, measures all query input registers, and outputs the set T of measurement outcomes. Then

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

Lemma 2.10 (Generic search problem [4, 25, 26])**.** Let $\gamma \in [0, 1]$. Let Z be a finite set. $\mathsf{N}_1 : Z \to \{0, 1\}$ is the following function: For each z, $\mathsf{N}_1(z) = 1$ with probability p_z ($p_z \leq \gamma$), and $\mathsf{N}_1(z) = 0$ else. Let N_2 be the function with $\forall z : \mathsf{N}_2(z) = 0$. If an oracle algorithm $\mathcal{A}$ makes at most q quantum queries to N_1 (or N_2), then

$$\left|\Pr[b = 1 : b \leftarrow \mathcal{A}^{\mathsf{N}_1}] - \Pr[b = 1 : b \leftarrow \mathcal{A}^{\mathsf{N}_2}]\right| \leq 2q\sqrt{\gamma}.$$

3 ACWC_2 Transformation

We introduce our new ACWC transformation, denoted ACWC_2, by presenting $\mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$ for a hash function G, as shown in Figure 5. Let $\mathsf{PKE}' = \mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$ be the resulting encryption scheme. By applying ACWC_2 to an underlying PKE, we show that (1) PKE' achieves a worst-case correctness error that is essentially as small as the average-case error of PKE, and (2) PKE' attains tight IND-CPA security provided that PKE is OW-CPA secure.

3.1 SOTP

Definition 3.1. A semi-generalized one-time pad $\mathsf{SOTP} = (\mathsf{Encode}, \mathsf{Decode})$, with a message space $\mathcal{X}$, a randomness space $\mathcal{U}$ (with corresponding distribution $\psi_{\mathcal{U}}$), and a code space $\mathcal{Y}$ (with corresponding distribution $\psi_{\mathcal{Y}}$), consists of the following two algorithms:

- $\mathsf{Encode}(x, u)$: The encoding algorithm Encode is a deterministic algorithm that takes a message $x \in \mathcal{X}$ and randomness $u \in \mathcal{U}$ as input, and outputs a code $y \in \mathcal{Y}$.
- $\mathsf{Decode}(y, u)$: The decoding algorithm Decode is a deterministic algorithm that takes a code $y \in \mathcal{Y}$ and randomness $u \in \mathcal{U}$ as input, and outputs a message $x \in \mathcal{X} \cup \{\perp\}$.

It also satisfies the following three properties:

1. Decoding: For all $x \in \mathcal{X}$ and $u \in \mathcal{U}$, $\mathsf{Decode}(\mathsf{Encode}(x, u), u) = x$.
2. Message-hiding: For all $x \in \mathcal{X}$, $\mathsf{Encode}(x, u)$ with $u \leftarrow \psi_{\mathcal{U}}$ follows the distribution $\psi_{\mathcal{Y}}$.
3. Rigid: For all $u \in \mathcal{U}$ and $y \in \mathcal{Y}$ such that $\mathsf{Decode}(y, u) \neq \perp$, we have $\mathsf{Encode}(\mathsf{Decode}(y, u), u) = y$.

In contrast to the GOTP defined in [16], SOTP does not need to satisfy an additional *randomness-hiding* property, which requires that the output $y = \mathsf{Encode}(x, u)$ follow the distribution $\psi_{\mathcal{Y}}$ while simultaneously revealing no information about the randomness u. The absence of this additional requirement allows SOTP to be designed more flexibly and efficiently than GOTP. Instead, SOTP is required to be *rigid*, meaning that for all $u \in \mathcal{U}$ and $y \in \mathcal{Y}$, if $x = \mathsf{Decode}(y, u) \neq \perp$, then $\mathsf{Encode}(x, u) = y$.

3.2 ACWC_2

Let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an underlying public-key encryption scheme with message space $\mathcal{M}$ and randomness space $\mathcal{R}$, where messages $M \in \mathcal{M}$ and randomness $r \in \mathcal{R}$ are drawn from the distributions $\psi_{\mathcal{M}}$ and $\psi_{\mathcal{R}}$, respectively. Similarly, let $\mathsf{PKE}' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ be the transformed encryption scheme with message space $\mathcal{M}'$ and randomness space $\mathcal{R}'$. Let $\mathsf{SOTP} = (\mathsf{Encode}, \mathsf{Decode})$ be a semi-generalized one-time pad, where $\mathsf{Encode} : \mathcal{M}' \times \mathcal{U} \to \mathcal{M}$ and $\mathsf{Decode} : \mathcal{M} \times \mathcal{U} \to \mathcal{M}'$ are defined with respect to distributions $\psi_{\mathcal{U}}$ and $\psi_{\mathcal{M}}$. Let $\mathsf{G} : \mathcal{R} \to \mathcal{U}$ be a hash function whose outputs are independently $\psi_{\mathcal{U}}$-distributed. Then $\mathsf{PKE}' = \mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$ is described in Figure 5.

Under the condition that $\mathsf{Dec}(sk, c)$ in Dec' yields the same M as in Enc, the deterministic functions RRec and Decode do not affect the correctness error of PKE'. Thus, the factor that determines the success or failure of $\mathsf{Dec}'(sk, c)$ is the result of $\mathsf{Dec}(sk, c)$ within Dec'. This implies that the correctness error of PKE is directly transferred to that of PKE', and is eventually determined by how the randomness $r \in \mathcal{R}$ and message $M \in \mathcal{M}$ are sampled in PKE'. We observe that r is drawn according to the distribution $\psi_{\mathcal{R}}$ and that M is

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

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

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

1: $r \leftarrow \psi_{\mathcal{R}}$ using the randomness R
2: $M := \mathsf{Encode}(m, \mathsf{G}(r))$
3: $c := \mathsf{Enc}(pk, M; r)$
4: **return** c

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

1: $M := \mathsf{Dec}(sk, c)$
2: $r := \mathsf{RRec}(pk, M, c)$
3: $m := \mathsf{Decode}(M, \mathsf{G}(r))$
4: **if** $r \notin \mathcal{R}$ or $m = \perp$, **return** $\perp$
5: **return** m

Figure 5: $\mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$

an SOTP-encoded element in $\mathcal{M}$. Because each output of G is independently $\psi_{\mathcal{U}}$-distributed, the message-hiding property of SOTP ensures that M follows the distribution $\psi_{\mathcal{M}}$ while hiding m. Consequently, both M and r are chosen according to their originally intended distributions.

However, since the choice of the random oracle G can affect the correctness error of PKE′, we need to incorporate this observation into the correctness analysis. Theorem 3.2 shows that, for all but a negligible fraction of random oracles G, the worst-case correctness of PKE′ (transformed by ACWC_2) is close to the average-case correctness of PKE. This mirrors the idea underlying ACWC, and the proof strategy of Theorem 3.2 is essentially the same as in [16] (Lemma 3.6 therein), except for slight modifications to the message distribution.

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

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

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

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

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

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

$$\begin{aligned}
\delta'(pk, sk) &= \mathbb{E}[\max_{m \in \mathcal{M}'} \Pr[\mathsf{Dec}'(sk, \mathsf{Enc}'(pk, m)) \neq m]] \\
&\leq t + \Pr_{\mathsf{G}}\left[\max_{m \in \mathcal{M}'} \Pr[\mathsf{Dec}'(sk, \mathsf{Enc}'(pk, m)) \neq m] \geq t\right] \qquad (1) \\
&\leq t + \Pr_{\mathsf{G}}\left[\max_{m \in \mathcal{M}'} \Pr_{r \leftarrow \psi_{\mathcal{R}}}[\mathsf{Dec}'(sk, \mathsf{Enc}(pk, M; r)) \neq m] \geq t\right] \\
&\leq t + \Pr_{\mathsf{G}}\left[\max_{m \in \mathcal{M}'} \Pr_{r \leftarrow \psi_{\mathcal{R}}}[\mathsf{Dec}(sk, \mathsf{Enc}(pk, M; r)) \neq M] \geq t\right], \qquad (2)
\end{aligned}$$

where $M = \mathsf{Encode}(m, \mathsf{G}(r))$. Note that Equation (1) holds by Lemma 3.4, and Equation (2) holds because

$$\Pr_{r \leftarrow \psi_{\mathcal{R}}}[\mathsf{Dec}'(sk, \mathsf{Enc}(pk, M; r)) \neq m] \leq \Pr_{r \leftarrow \psi_{\mathcal{R}}}[\mathsf{Dec}(sk, \mathsf{Enc}(pk, M; r)) \neq M].$$

For any fixed key pair and any positive c, let $t(pk, sk) := \mu(pk, sk) + \|\psi_{\mathcal{R}}\| \cdot \sqrt{(c + \ln|\mathcal{M}'|)/2}$, where $\mu(pk, sk) := \Pr_{M,r}[\mathsf{Dec}(sk, \mathsf{Enc}(pk, M; r)) \neq M]$. Then, we can use Lemma 3.5 to argue that

$$\Pr_{\mathsf{G}}\left[\max_{m \in \mathcal{M}'} \Pr_{r \leftarrow \psi_{\mathcal{R}}}[\mathsf{Dec}(sk, \mathsf{Enc}(pk, M; r)) \neq M] > t(pk, sk)\right] \leq e^{-c}. \tag{3}$$

To this end, we define

$$g(m, r, u) = (\mathsf{Encode}(m, u), r) \quad \text{and} \quad B = \{(M, r) \in \mathcal{M} \times \mathcal{R} \mid \mathsf{Dec}(sk, \mathsf{Enc}(pk, M; r)) \neq M\},$$

which will be used in Lemma 3.5. Note that $\Pr_{r \leftarrow \psi_{\mathcal{R}}, u \leftarrow \psi_{\mathcal{U}}}[g(m, r, u) \in B] = \mu(pk, sk)$ holds for all $m \in \mathcal{M}'$ by the message-hiding property of the SOTP. For all $m \in \mathcal{M}'$,

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

Combining Equation (3) with Equation (2) and taking the expectation yields

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

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

Corollary 3.3 (Average-Case to Worst-Case Correctness error)**.** Let PKE be RR and have a randomness space $\mathcal{R}$ relative to the distribution $\psi_{\mathcal{R}}$. Let $\mathsf{SOTP} = (\mathsf{Encode}, \mathsf{Decode})$, with $\mathsf{Encode} : \mathcal{M}' \times \mathcal{U} \to \mathcal{M}$ and $\mathsf{Decode} : \mathcal{M} \times \mathcal{U} \to \mathcal{M}'$, be a semi-generalized one-time pad (for distributions $\psi_{\mathcal{U}}$ and $\psi_{\mathcal{M}}$). Let $\mathsf{G} : \mathcal{R} \to \mathcal{U}$ be a random oracle whose outputs are distributed according to $\psi_{\mathcal{U}}$. If PKE is δ-average-case-correct, then

$$\mathbb{E}\left[\max_{m \in \mathcal{M}'} \Pr_{r \leftarrow \psi_{\mathcal{R}}}[\mathsf{Dec}(sk, \mathsf{Enc}(pk, M; r)) \neq M]\right] \leq \delta',$$

where $M = \mathsf{Encode}(m, \mathsf{G}(r))$, and

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

with $\|\psi_{\mathcal{R}}\| := \sqrt{\sum_r \psi_{\mathcal{R}}(r)^2}$, where the expectation is over G and $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$.

Proof. The proof is analogous to that of Theorem 3.2, and we omit the detailed argument. □

Lemma 3.4. Let X be a random variable and let f be a non-negative real-valued function with $f(X) \leq 1$. Then, for all $t \in \mathbb{R}^+$,

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

Proof. By applying the law of total probability and by partitioning the domain of x into the cases where $f(x) < t$ and $f(x) \geq t$, we obtain

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

The final equality follows from $\sum_{f(x)\geq t} \Pr[X = x] = \Pr[f(X) \geq t]$. □

Lemma 3.5 (Adapting Lemma 3.7 from [16])**.** Let g be a function, and let B be a set such that

$$\forall m \in \mathcal{M}', \quad \Pr_{r\leftarrow\psi_{\mathcal{R}},\, u\leftarrow\psi_{\mathcal{U}}}[g(m, r, u) \in B] \leq \mu \tag{4}$$

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

$$\Pr_{r\leftarrow\psi_{\mathcal{R}}}[g(m, r, \mathsf{G}(r)) \in B] \leq \mu + \|\psi_{\mathcal{R}}\| \cdot \sqrt{(c + \ln|\mathcal{M}'|)/2} \tag{5}$$

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

Proof. For fixed $m \in \mathcal{M}'$ and $r \in \mathcal{R}$, define $p_r := \Pr_{u\leftarrow\psi_{\mathcal{U}}}[g(m, r, u) \in B]$. From Equation (4), we know that $\sum_r \psi_{\mathcal{R}}(r)p_r \leq \mu$. For each r, define a random variable X_r whose value is determined as follows: G chooses a random $u = \mathsf{G}(r)$ and then checks whether $g(m, r, \mathsf{G}(r)) \in B$; if it does, then we set $X_r = 1$; otherwise we set it to zero. Because G is a random function, the probability that $X_r = 1$ is exactly p_r.

The probability of Equation (5) 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 $\mathbb{E}[\sum_r Y_r] = \mathbb{E}[\sum_r \psi_{\mathcal{R}}(r)X_r] = \sum_r \psi_{\mathcal{R}}(r)p_r \leq \mu$. By the Hoeffding bound, we have for all positive t,

$$\Pr[\sum_r Y_r > \mu + t] \leq \exp\left(\frac{-2t^2}{\sum \psi_{\mathcal{R}}(r)^2}\right) = \exp\left(\frac{-2t^2}{\|\psi_{\mathcal{R}}\|^2}\right). \tag{6}$$

By setting $t \geq \|\psi_{\mathcal{R}}\| \cdot \sqrt{(c + \ln|\mathcal{M}'|)/2}$, for a fixed m, Equation (5) holds for all but an $e^{-c} \cdot |\mathcal{M}'|^{-1}$ fraction of random functions G. Applying the union bound yields the claim in the lemma. □

GAMES G_0-G_2

1: $\mathsf{G} \leftarrow (\mathcal{R} \to \mathcal{U})$ //G_0
2: $\mathcal{L}_{\mathsf{G}}, \mathcal{L}_r := \emptyset$ //G_1-G_2
3: $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$
4: $(m_0, m_1) \leftarrow \mathcal{A}_0^{\mathsf{G}}(pk)$
5: $b \leftarrow \{0,1\}$
6: $r^* \leftarrow \psi_{\mathcal{R}}$
7: $M^* = \mathsf{Encode}(m_b, \mathsf{G}(r^*))$ //G_1
8: $M^* \leftarrow \psi_{\mathcal{M}}$ //G_2
9: $c^* \leftarrow \mathsf{Enc}(pk, M^*; r^*)$
10: $b' \leftarrow \mathcal{A}_1^{\mathsf{G}}(pk, c^*)$
11: **return** $[\![b = b']\!]$

$\mathsf{G}(r)$ //G_1-G_2

1: **if** $\exists\,(r, u) \in \mathcal{L}_{\mathsf{G}}$
2: **return** u
3: $u \leftarrow \psi_{\mathcal{U}}$
4: $\mathcal{L}_{\mathsf{G}} := \mathcal{L}_{\mathsf{G}} \cup \{(r, u)\}$
5: $\mathcal{L}_r := \mathcal{L}_r \cup \{r\}$
6: **return** u

Figure 6: GAMES G_0–G_2 of Theorem 3.6

3.2.1 Security Proof in the ROM

Theorem 3.6 (OW-CPA security of $\mathsf{PKE} \overset{\mathsf{ROM}}{\Longrightarrow}$ IND-CPA security of $\mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$)**.** Let PKE be a public-key encryption scheme with AC-MR, VC-MR, and RR properties. For any adversary $\mathcal{A}$ against the IND-CPA security of $\mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$, making at most q_{G} random oracle queries, there exists an adversary $\mathcal{B}$ against the injectivity of PKE and an adversary $\mathcal{C}$ against the OW-CPA security of PKE with

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

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

Proof. We prove the theorem by constructing adversaries $\mathcal{B}$ (Figure 7) and $\mathcal{C}$ (Figure 8) from $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, where $\mathcal{B}$ and $\mathcal{C}$ break the injectivity and the OW-CPA security of PKE, respectively, while $\mathcal{A}$ breaks the IND-CPA security of $\mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$.

GAME G_0. G_0 (see Figure 6) is the original IND-CPA game with $\mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$. By the definition of the IND-CPA game,

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

GAME G_1. G_1 is identical to G_0 except that the random oracle G is instantiated via lazy sampling rather than by sampling a full random function in advance. Concretely, we maintain a table $\mathcal{L}_{\mathsf{G}}$: upon a query r, if r has not yet been assigned a value, a fresh $u \leftarrow \psi_{\mathcal{U}}$ is sampled and (r, u) is added to $\mathcal{L}_{\mathsf{G}}$; otherwise the stored value is returned. Also, every input r queried by $\mathcal{A}$ is also recorded in the set $\mathcal{L}_r$. Since lazy sampling is equivalent to a uniform function, we have

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

GAME G_2. G_2 is identical to G_1 except that M^* is sampled directly from $\psi_{\mathcal{M}}$ rather than being computed as $M^* = \mathsf{Encode}(m_b, \mathsf{G}(r^*))$. Let BAD_1 be the event that $\mathcal{A}$ queries G at input r^* in G_2. If BAD_1 does not occur, then $\mathsf{G}(r^*)$ is uniform and independent of $\mathcal{A}$'s view in G_1, hence by the message-hiding of SOTP the

$\underline{\mathcal{B}(pk)}$

$\mathcal{L}_{\mathsf{G}}, \mathcal{L}_r := \emptyset$
$(m_0, m_1) \leftarrow \mathcal{A}_0^{\mathsf{G}}(pk)$
$b \leftarrow \{0,1\}$
$(M^*, r^*) \leftarrow \psi_{\mathcal{M}} \times \psi_{\mathcal{R}}$
$c^* \leftarrow \mathsf{Enc}(pk, M^*; r^*)$
$b' \leftarrow \mathcal{A}_1^{\mathsf{G}}(pk, c^*)$
for $r \in \mathcal{L}_r$ with $r \neq r^*$ **do**
  $M := \mathsf{MRec}(pk, r, c^*)$
  if $(M, r) \in \mathcal{M} \times \mathcal{R}$
    return (M, M^*, r, r^*)
$(M, r) \leftarrow \psi_{\mathcal{M}} \times \psi_{\mathcal{R}}$
return (M, r, M^*, r^*)

$\underline{\mathsf{G}(r)}$

if $\exists\, (r, u) \in \mathcal{L}_{\mathsf{G}}$
  return u
$u \leftarrow \psi_{\mathcal{U}}$
$\mathcal{L}_{\mathsf{G}} := \mathcal{L}_{\mathsf{G}} \cup \{(r, u)\}$
$\mathcal{L}_r := \mathcal{L}_r \cup \{r\}$
return u

Figure 7: Adversary $\mathcal{B}$ for the proof of Theorem 3.6

distribution of $M^* = \mathsf{Encode}(m_b, \mathsf{G}(r^*))$ equals $\psi_{\mathcal{M}}$. Therefore, by Lemma 2.8, unless BAD_1 occurs the distributions of G_1 and G_2 are identical, and

$$\left| \Pr[G_1^{\mathcal{A}} \Rightarrow 1] - \Pr[G_2^{\mathcal{A}} \Rightarrow 1] \right| \leq \Pr[\mathsf{BAD}_1 \text{ in } G_2^{\mathcal{A}}].$$

Moreover, in G_2 the pair (M^*, r^*) is sampled independently of b, hence

$$\Pr[G_2^{\mathcal{A}} \Rightarrow 1] = \tfrac{1}{2}.$$

Assume that BAD_1 occurs in G_2. Let BAD_2 be the event that $\mathcal{A}$ queries some $r \in \mathcal{L}_r$ with $r \neq r^*$ and $M := \mathsf{MRec}(pk, r, c^*) \in \mathcal{M}$. Then,

$$\begin{aligned} \Pr[\mathsf{BAD}_1 \text{ in } G_2^{\mathcal{A}}] &= \Pr[\mathsf{BAD}_1 \wedge \mathsf{BAD}_2 \text{ in } G_2^{\mathcal{A}}] + \Pr[\mathsf{BAD}_1 \wedge \neg\mathsf{BAD}_2 \text{ in } G_2^{\mathcal{A}}] \\ &\leq \Pr[\mathsf{BAD}_2 \text{ in } G_2^{\mathcal{A}}] + \Pr[\mathsf{BAD}_1 \wedge \neg\mathsf{BAD}_2 \text{ in } G_2^{\mathcal{A}}]. \end{aligned}$$

Assume that the event BAD_2 occurs in G_2. We construct an adversary $\mathcal{B}$ against GAME INJ for PKE as follows. Given pk from its challenger, $\mathcal{B}$ runs $\mathcal{A}_0^{\mathsf{G}}(pk)$, samples $(M^*, r^*) \leftarrow \psi_{\mathcal{M}} \times \psi_{\mathcal{R}}$, and generates $c^* \leftarrow \mathsf{Enc}(pk, M^*; r^*)$. It then runs $\mathcal{A}_1^{\mathsf{G}}(pk, c^*)$ while simulating the random oracle G. Since BAD_2 occurs, there exists some $r \in \mathcal{L}_r$ with $r \neq r^*$ such that $M := \mathsf{MRec}(pk, r, c^*) \in \mathcal{M}$. By the AC-MR property, (M, r) is a valid pre-image of c^*. Hence, $\mathcal{B}$ can identify such an $r \neq r^*$ in $\mathcal{L}_r$, compute $M := \mathsf{MRec}(pk, r, c^*)$, and output (M, r, M^*, r^*). Therefore, $\mathcal{B}$ breaks GAME INJ for PKE whenever BAD_2 occurs. Thus,

$$\Pr[\mathsf{BAD}_2 \text{ in } G_2] \;\leq\; \mathsf{Adv}^{\mathsf{INJ}}_{\mathsf{PKE}}(\mathcal{B}).$$

Assuming that the event $\mathsf{BAD}_1 \wedge \neg\mathsf{BAD}_2$ occurs in $G_2^{\mathcal{A}}$, we construct an adversary $\mathcal{C}$ against GAME OW-CPA for PKE as follows. The challenger provides $\mathcal{C}$ with (pk, c^*), where c^* is generated by sampling $(M^*, r^*) \leftarrow \psi_{\mathcal{M}} \times \psi_{\mathcal{R}}$ and computing $c^* \leftarrow \mathsf{Enc}(pk, M^*; r^*)$. Upon receiving (pk, c^*), the adversary $\mathcal{C}$ runs $b \leftarrow \mathcal{A}_0^{\mathsf{G}}(pk)$ and then invokes $\mathcal{A}_1^{\mathsf{G}}(pk, c^*)$, while simulating the random oracle G. Since we assumed that $\mathsf{BAD}_1 \wedge \neg\mathsf{BAD}_2$ occurs, we have $r^* \in \mathcal{L}_r$, and by the VC-MR property it follows that $M^* = \mathsf{MRec}(pk, r^*, c^*) \in \mathcal{M}$

$\underline{\mathcal{C}(pk, c^*)}$

$\mathcal{L}_{\mathsf{G}}, \mathcal{L}_r := \emptyset$
$(m_0, m_1) \leftarrow \mathcal{A}_0^{\mathsf{G}}(pk)$
$b' \leftarrow \mathcal{A}_1^{\mathsf{G}}(pk, c^*)$
for $r \in \mathcal{L}_r$ **do**
  $M := \mathsf{MRec}(pk, r, c^*)$
  if $(M, r) \in \mathcal{M} \times \mathcal{R}$
    return M
return $M \leftarrow \psi_{\mathcal{M}}$

$\underline{\mathsf{G}(r)}$

if $\exists\ (r, u) \in \mathcal{L}_{\mathsf{G}}$
  return u
$u \leftarrow \psi_{\mathcal{U}}$
$\mathcal{L}_{\mathsf{G}} := \mathcal{L}_{\mathsf{G}} \cup \{(r, u)\}$
$\mathcal{L}_r := \mathcal{L}_r \cup \{r\}$
return u

Figure 8: Adversary $\mathcal{C}$ for the proof of Theorem 3.6

and $c^* = \mathsf{Enc}(pk, M^*; r^*)$. Moreover, by the assumption $\neg\mathsf{BAD}_2$, there exists no $r \neq r^*$ such that $M := \mathsf{MRec}(pk, r, c^*) \in \mathcal{M}$. Thus, $\mathcal{C}$ can recover M^* uniquely. Hence,

$$\Pr[\mathsf{BAD}_1 \wedge \neg\mathsf{BAD}_2 \text{ in } G_2^{\mathcal{A}}] \leq \mathsf{Adv}^{\mathsf{OW\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{C}).$$

Putting everything together,

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{ACWC}_2[\mathsf{PKE},\mathsf{SOTP},\mathsf{G}]}(\mathcal{A}) &= \left|\Pr[G_0^{\mathcal{A}} \Rightarrow 1] - \tfrac{1}{2}\right| \\
&\leq \sum_{i=0}^{1} \left|\Pr[G_i^{\mathcal{A}} \Rightarrow 1] - \Pr[G_{i+1}^{\mathcal{A}} \Rightarrow 1]\right| + \left|\Pr[G_2^{\mathcal{A}} \Rightarrow 1] - \tfrac{1}{2}\right| \\
&\leq \Pr[\mathsf{BAD}_2 \text{ in } G_2] + \Pr[\mathsf{BAD}_1 \wedge \neg\mathsf{BAD}_2 \text{ in } G_2^{\mathcal{A}}] \\
&\leq \mathsf{Adv}^{\mathsf{INJ}}_{\mathsf{PKE}}(\mathcal{B}) + \mathsf{Adv}^{\mathsf{OW\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{C}).
\end{aligned}$$

□

3.2.2 Security Proof in the QROM

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

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

where the running time of $\mathcal{B}$ and $\mathcal{C}$ is bounded by $\mathbf{Time}(\mathcal{A}) + O(q_{\mathsf{G}})$.

Proof. To prove the theorem, we consider a sequence of games G_0 through G_7, defined in Figures 9, 10, and 11. We first analyze the transition from G_0 to G_2, and then apply Lemma 2.9 to bound the hop from G_2 to G_3. A detailed description of the security proof is given below.

GAME G_0

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

Figure 9: GAME G_0 for Theorem 3.7

GAME G_0. G_0 (see Figure 9) is the original IND-CPA game with $\mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$. By definition, we have

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

GAME G_1. We define G_1 in the same way as G_0, except that part of the challenger's logic is encapsulated into an algorithm $\mathcal{C}^{\mathsf{G}}$. In addition, before $\mathcal{C}^{\mathsf{G}}$ runs adversary $\mathcal{A}$, we sample $r^* \leftarrow \psi_{\mathcal{R}}$ and make a classical query $u := \mathsf{G}(r^*)$. As these changes are only conceptual, it follows that

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

GAME G_2. We define G_2 in the same way as G_1, except that we change the way G is defined. Instead of choosing G uniformly, we sample F and u uniformly and then set $\mathsf{G} := \mathsf{F}(r^* := u)$. In other words, G is identical to F except that it returns u on input r^*. Since the distribution of (G, u) remains the same, we have

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

GAME G_3. We define G_3 in the same way as G_2, except that algorithm $\mathcal{C}$ has oracle access to F instead of G. Recall that in G_2 the oracle G is defined as $\mathsf{F}(r^* := u)$, which coincides with F on all inputs except r^*, i.e., they differ only on the set $S := \{r^*\}$. This is exactly the setting of Lemma 2.9 with the set S and auxiliary input $z := (r^*, u)$. Hence, by applying the lemma with algorithm $\mathcal{C}$, we obtain

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

Moreover, since the random value u is only used in $\mathsf{Encode}(m_b, u)$, the message-hiding property of SOTP implies that M^* is independent of m_b. Hence,

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

GAME G_4 and G_5. We define G_4 in the same way as G_3, except that it is arranged according to Lemma 2.9. We then define G_5 in the same way as G_4, except that we change the way M^* is determined. Instead of computing $M^* = \mathsf{Encode}(m_b, u)$, we sample $M^* \leftarrow \psi_{\mathcal{M}}$. In G_4, however, since u is sampled from $\psi_{\mathcal{U}}$ and

GAMES G_1-G_5

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

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

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

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

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

Figure 10: GAMES G_1-G_5 for the proof of Theorem 3.7

used only for $\mathsf{Encode}(m_b, u)$, the message-hiding property of SOTP ensures that $M^* = \mathsf{Encode}(m_b, u)$ is distributed according to $\psi_{\mathcal{M}}$. Hence,

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

GAME G_6. We define G_6 in the same way as G_5, except that the challenger's procedure is rearranged as shown in Figure 11. Since this change is only conceptual, we have

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

Let BAD be the event that there exists $r \in T$ with $r \neq r^*$ and $M := \mathsf{MRec}(pk, r, c^*) \in \mathcal{M}$. Then,

$$\begin{aligned}\Pr[G_6^{\mathcal{E}} \Rightarrow 1] &= \Pr[G_6^{\mathcal{E}} \Rightarrow 1 \wedge \mathsf{BAD} \text{ in } G_6^{\mathcal{E}}] + \Pr[G_6^{\mathcal{E}} \Rightarrow 1 \wedge \neg\mathsf{BAD} \text{ in } G_6^{\mathcal{E}}] \\ &\leq \Pr[\mathsf{BAD} \text{ in } G_6^{\mathcal{E}}] + \Pr[G_6^{\mathcal{E}} \Rightarrow 1 \wedge \neg\mathsf{BAD} \text{ in } G_6^{\mathcal{E}}]\end{aligned}$$

GAME G_{7a}. We define G_{7a} by wrapping the generation of (M^*, r^*) and c^* into the adversary $\mathcal{B}$. Also, unlike G_6, after $\mathcal{E}$ outputs T, $\mathcal{B}$ searches for some $r \in T$ with $r \neq r^*$ such that $M = \mathsf{MRec}(pk, r, c^*) \in \mathcal{M}$, and outputs (M, M^*, r, r^*) if such an r exists. Otherwise, it samples random (M, r) and outputs (M, M^*, r, r^*). Suppose such an r exists. By AC-MR, the pair (M, r) is a valid pre-image of c^*. Since (M^*, r^*) is also a pre-image of c^* and $r \neq r^*$, the tuple (M, M^*, r, r^*) is a valid solution to the injectivity game. Therefore, we have

$$\Pr[\mathsf{BAD} \text{ in } G_6^{\mathcal{E}}] = \Pr[G_{7a}^{\mathcal{B}} \Rightarrow 1].$$

Moreover, by definition, G_{7a} is exactly the INJ game for PKE run with adversary $\mathcal{B}$; hence

$$\Pr[G_{7a}^{\mathcal{B}} \Rightarrow 1] = \mathsf{Adv}^{\mathsf{INJ}}_{\mathsf{PKE}}(\mathcal{B}).$$

GAME G_{7b}. We define G_{7b} in the same way as G_6, except that algorithm $\mathcal{C}$ explicitly outputs (M, r) and the game returns 1 if $(M^*, r^*) = (M, r)$. (see Figure 11). If there exist $r^* \in T$ and there does not exist

GAMES G_6

$(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$
$(M^*, r^*) \leftarrow \psi_{\mathcal{M}} \times \psi_{\mathcal{R}}$
$c^* \leftarrow \mathsf{Enc}(pk, M^*; r^*)$
$T \leftarrow \mathcal{E}(pk, c^*)$
return $[\![r^* \in T]\!]$

$\mathcal{E}(pk, c^*)$

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

GAMES G_{7a}

$(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$
$(M, M^*, r, r^*) \leftarrow \mathcal{B}(pk)$
$c = \mathsf{Enc}(pk, M; r)$
$c' = \mathsf{Enc}(pk, M^*; r^*)$
return $\left[\!\!\left[\begin{array}{l}(M, M^*, r, r^*) \in \mathcal{M}^2 \times \mathcal{R}^2 \\ \wedge\, (M, r) \neq (M^*, r^*) \wedge c = c'\end{array}\right]\!\!\right]$

$\mathcal{B}(pk)$

$(M^*, r^*) \leftarrow \psi_{\mathcal{M}} \times \psi_{\mathcal{R}}$
$c^* \leftarrow \mathsf{Enc}(pk, M^*; r^*)$
$T \leftarrow \mathcal{E}(pk, c^*)$
for $r \in T$ with $r \neq r^*$ **do**
  if $M = \mathsf{MRec}(pk, r, c^*) \in \mathcal{M}$
    return (M, M^*, r, r^*)
$(M, r) \leftarrow \psi_{\mathcal{M}} \times \psi_{\mathcal{R}}$
return (M, r, M^*, r^*)

GAMES G_{7b}

$(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$
$(M^*, r^*) \leftarrow \psi_{\mathcal{M}} \times \psi_{\mathcal{R}}$
$c^* \leftarrow \mathsf{Enc}(pk, M^*; r^*)$
$M \leftarrow \mathcal{C}(pk, c^*)$
return $[\![M^* = M]\!]$

$\mathcal{C}(pk, c^*)$

$T \leftarrow \mathcal{E}(pk, c^*)$
for $r \in T$ **do**
  if $M = \mathsf{MRec}(pk, r, c^*) \in \mathcal{M}$
    return (M, r)
return $(M, r) \leftarrow \psi_{\mathcal{M}} \times \psi_{\mathcal{R}}$

Figure 11: GAMES G_6-G_7 for the proof of Theorem 3.7

$r \in T$ with $r \neq r^*$ and $M = \mathsf{MRec}(pk, r, c^*) \in \mathcal{M}$, by VC-MR, we can recover $r^* = \mathsf{MRec}(pk, r^*, c^*)$. Therefore, we have

$$\Pr[G_6^{\mathcal{E}} \Rightarrow 1 \wedge \neg \mathsf{BAD} \text{ in } G_6^{\mathcal{E}}] = \Pr[G_{7b}^{\mathcal{C}} \Rightarrow 1].$$

By definition, G_{7b} is exactly the OW-CPA game for PKE run with adversary $\mathcal{C}$; hence

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

Combining all (in)equalities and bounds, we obtain

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

which concludes the proof. □

3.2.3 Spreadness of PKE'

Theorem 3.8. If PKE is (weakly) γ-spread, SOTP has the message hiding property, and G is modeled as a random oracle, then $\mathsf{PKE}' = \mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$ is (weakly) γ-spread.

Proof. For a fixed (pk, sk) and m, we consider the probability $\Pr_{R\leftarrow\mathcal{R}',\mathsf{G}}[c = \mathsf{Enc}'(pk, m; R)]$ for any ciphertext c. Since G is modeled as a random oracle, the probability is taken over the random choice of G. Given that r is sampled as $r \leftarrow \psi_{\mathcal{R}}$ using the randomness $R \leftarrow \mathcal{R}'$, the probability can be rewritten as

$$\Pr_{R\leftarrow\mathcal{R}',\mathsf{G}}[c = \mathsf{Enc}'(pk, m; R)] = \Pr_{r\leftarrow\psi_{\mathcal{R}},\mathsf{G}}[c = \mathsf{Enc}(pk, \mathsf{Encode}(m, \mathsf{G}(r)); r)].$$

By the law of total probability on possible $r \leftarrow \psi_{\mathcal{R}}$, we have:

$$\Pr_{r\leftarrow\psi_{\mathcal{R}},\mathsf{G}}[c = \mathsf{Enc}(pk, \mathsf{Encode}(m, \mathsf{G}(r)); r)] = \sum_{r_i\in\mathcal{R}} \Pr_{\mathsf{G}}[c = \mathsf{Enc}(pk, \mathsf{Encode}(m, \mathsf{G}(r_i)); r_i)] \Pr_{r\leftarrow\psi_{\mathcal{R}}}[r = r_i].$$

Since $\mathsf{G}(r_i)$ is $\psi_{\mathcal{U}}$-distributed, the message hiding property of SOTP ensures that the output $M = \mathsf{Encode}(m, \mathsf{G}(r_i))$ is $\psi_{\mathcal{M}}$-distributed over the random choice of G:

$$\begin{aligned}
&\sum_{r_i\in\mathcal{R}} \Pr_{\mathsf{G}}[c = \mathsf{Enc}(pk, \mathsf{Encode}(m, \mathsf{G}(r_i)); r_i)] \Pr_{r\leftarrow\psi_{\mathcal{R}}}[r = r_i] \\
&= \sum_{r_i\in\mathcal{R}} \Pr_{u\leftarrow\psi_{\mathcal{U}}}[c = \mathsf{Enc}(pk, \mathsf{Encode}(m, u); r_i)] \Pr_{r\leftarrow\psi_{\mathcal{R}}}[r = r_i] \\
&= \sum_{r_i\in\mathcal{R}} \Pr_{M\leftarrow\psi_{\mathcal{M}}}[c = \mathsf{Enc}(pk, M; r_i)] \Pr_{r\leftarrow\psi_{\mathcal{R}}}[r = r_i].
\end{aligned}$$

For the ease of analysis, we define an indicator function $\mathbf{I}(pk, M, r, c) = [\![c == \mathsf{Enc}(pk, M; r)]\!]$. Then,

$$\begin{aligned}
&\sum_{r_i\in\mathcal{R}} \Pr_{M\leftarrow\psi_{\mathcal{M}}}[c = \mathsf{Enc}(pk, M; r_i)] \Pr_{r\leftarrow\psi_{\mathcal{R}}}[r = r_i] \\
&= \sum_{r_i\in\mathcal{R}} \sum_{M_j\in\mathcal{M}} \mathbf{I}(pk, M_j, r_i, c) \Pr_{M\leftarrow\psi_{\mathcal{M}}}[M = M_j] \Pr_{r\leftarrow\psi_{\mathcal{R}}}[r = r_i] \\
&= \sum_{M_j\in\mathcal{M}} \sum_{r_i\in\mathcal{R}} \mathbf{I}(pk, M_j, r_i, c) \Pr_{r\leftarrow\psi_{\mathcal{R}}}[r = r_i] \Pr_{M\leftarrow\psi_{\mathcal{M}}}[M = M_j] \\
&= \sum_{M_j\in\mathcal{M}} \Pr_{r\leftarrow\psi_{\mathcal{R}}}[c = \mathsf{Enc}(pk, M_j; r)] \Pr_{M\leftarrow\psi_{\mathcal{M}}}[M = M_j].
\end{aligned}$$

Considering $\Pr_{r\leftarrow\psi_{\mathcal{R}}}[c = \mathsf{Enc}(pk, M_j; r)]$ as the γ-spreadness of PKE on any message M_j, the γ'-spreadness of PKE' is upper-bounded as follows:

$$\begin{aligned}
\Pr_{R\leftarrow\mathcal{R}',\mathsf{G}}[c = \mathsf{Enc}'(pk, m; R)] &= \sum_{M_j\in\mathcal{M}} \Pr_{r\leftarrow\psi_{\mathcal{R}}}[c = \mathsf{Enc}(pk, M_j; r)] \cdot \Pr_{M\leftarrow\psi_{\mathcal{M}}}[M = M_j] \\
&\le 2^{-\gamma} \cdot \sum_{M_j\in\mathcal{M}} \Pr_{M\leftarrow\psi_{\mathcal{M}}}[M = M_j] = 2^{-\gamma}.
\end{aligned}$$

By averaging over (pk, sk), the weak γ'-spreadness of PKE' is also obtained. □

4 IND-CCA Secure KEM from ACWC$_2$

4.1 FO Transform with Re-encryption

One can apply the Fujisaki-Okamoto transformation $\mathsf{FO}^{\perp}$ to the IND-CPA secure PKE', as shown in Figure 5, to obtain an IND-CCA secure KEM. Figure 12 shows the resultant $\mathsf{KEM} := \mathsf{FO}^{\perp}[\mathsf{PKE}', \mathsf{H}] = (\mathsf{Gen}, \mathsf{Encap}, \mathsf{Decap})$, where H is a hash function (modeled as a random oracle). Regarding the correctness error of KEM, KEM preserves the worst-case correctness error of PKE', as Decap works correctly as long as Dec' is performed correctly. Regarding the IND-CCA security of KEM, we can use the previous results [20] and [15], which are stated in Theorems 4.1 and 4.2, respectively. By combining these results with Theorems 3.6 and 3.7, we can achieve the IND-CCA security of KEM in the classical/quantum random oracle model. In the case of the quantum random oracle model (QROM), we need to further use the fact that IND-CPA security generically implies OW-CPA security.

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

1: $m \leftarrow \mathcal{M}$
2: $(R, K) := \mathsf{H}(m)$
3: $c := \mathsf{Enc}'(pk, m; R)$
- $r \leftarrow \psi_{\mathcal{R}}$ using the randomness R
- $M := \mathsf{Encode}(m, \mathsf{G}(r))$
- $c := \mathsf{Enc}(pk, M; r)$
4: **return** (K, c)

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

1: $m' := \mathsf{Dec}'(sk, c)$
- $M' = \mathsf{Dec}(sk, c)$
- $r' = \mathsf{RRec}(pk, M', c)$
- $m' = \mathsf{Decode}(M', \mathsf{G}(r'))$
- **if** $r' \notin \mathcal{R}$ or $m' = \perp$, **return** $\perp$
- **return** m'
2: $(R', K') := \mathsf{H}(m')$
3: **if** $m' = \perp$ or $c \neq \mathsf{Enc}'(pk, m'; R')$, **return** $\perp$
4: **else, return** K'

Figure 12: $\mathsf{KEM} = \mathsf{FO}^{\perp}[\mathsf{PKE}', \mathsf{H}]$

Theorem 4.1 (IND-CPA security of $\mathsf{PKE}' \overset{\mathsf{ROM}}{\Longrightarrow}$ IND-CCA security of KEM [20])**.** Let PKE' be a public-key encryption scheme with a message space $\mathcal{M}$. Let PKE' has (worst-case) correctness error δ and is (weakly) γ-spread. For any adversary $\mathcal{A}$ making at most q_{D} decapsulation and q_{H} hash queries, against the IND-CCA security of KEM, there exists an adversary $\mathcal{B}$ against the IND-CPA security of PKE' with

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

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

Theorem 4.2 (OW-CPA security of $\mathsf{PKE}' \overset{\mathsf{QROM}}{\Longrightarrow}$ IND-CCA security of KEM [15])**.** Let PKE' have (worst-case) correctness error δ and be (weakly) γ-spread. For any quantum adversary $\mathcal{A}$, making at most q_{D} decapsulation and q_{H} (quantum) hash queries against the IND-CCA security of KEM, there exists a quantum adversary $\mathcal{B}$ against the OW-CPA security of PKE' with

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

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

4.2 FO-Equivalent Transform Without Re-encryption

The aforementioned $\mathsf{FO}^{\perp}$ requires the Decap algorithm to perform re-encryption to check if ciphertext c is well-formed. Using m' as the result of $\mathsf{Dec}'(sk, c)$, a new randomness R' is obtained from $\mathsf{H}(m')$, and $\mathsf{Enc}'(pk, m'; R')$ is computed and compared with the (decrypted) ciphertext c. Even if m' is the same as m used in Encap, it does not guarantee that $\mathsf{Enc}'(pk, m'; R') = c$ without computing R' and performing re-encryption. In other words, there could exist many other ciphertexts $\{c_i\}$ (including c as one of them), all of which are decrypted into the same m' but generated with distinct randomness $\{R'\}$. In $\mathsf{FO}^{\perp}$ (and other FO transformations), there is still no way to find the same c (honestly) generated in Encap other than by comparing $\mathsf{Enc}'(pk, m'; R')$ and c. In the context of chosen-ciphertext attacks (using the inequality such as $c \neq \mathsf{Enc}'(pk, m'; R')$), 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 $\mathsf{FO}^{\perp}$ based on ACWC_2 can eliminate the need for ciphertext comparison $c = \mathsf{Enc}'(pk, m'; R')$ in Decap, and instead replace it with a simpler and more efficient comparison $r' = r''$. To do this, we first change Decap of Figure 12 into that of Figure 13, which are conceptually identical to each other. Rather, the change has the effect of preventing reaction attacks that can occur by returning distinct output errors of Decap. Next, we suggest the new $\mathsf{FO}^{\perp}$ conversion based on ACWC_2, denoted as $\overline{\mathsf{FO}}^{\perp}$, as shown in Figure 14. In $\overline{\mathsf{FO}}^{\perp}$, r' and r'' are values generated during the execution of Decap, where r' is the output of $\mathsf{RRec}(pk, M', c)$ and r'' is computed from the randomness R' of $\mathsf{H}(m')$. The only change compared to $\mathsf{FO}^{\perp}$ in Figure 13 is the boxed area, while the remaining parts remain the same. By proving that the two conditions $r' \notin \mathcal{R}$ and $c = \mathsf{Enc}'(pk, m'; R')$ are equivalent to the equality $r' = r''$ (where $r'' \leftarrow \psi_{\mathcal{R}}$ with the randomness R'), we can show that both $\mathsf{FO}^{\perp}$ and $\overline{\mathsf{FO}}^{\perp}$ work identically and thus achieve the same level of IND-CCA security.

$\underline{\mathsf{Decap}(sk, c)}$
1: $M' = \mathsf{Dec}(sk, c)$
2: $r' = \mathsf{RRec}(pk, M', c)$
3: $m' = \mathsf{Decode}(M', \mathsf{G}(r'))$
4: $(R', K') := \mathsf{H}(m')$
5: **if** $m' \neq \perp$ and $\boxed{r' \in \mathcal{R} \text{ and } c = \mathsf{Enc}'(pk, m'; R')}$
6: **return** K'
7: **else**
8: **return** $\perp$

Figure 13: Modified $\mathsf{KEM} = \mathsf{FO}^{\perp}[\mathsf{PKE}', \mathsf{H}]$

$\underline{\mathsf{Decap}(sk, c)}$
1: $M' = \mathsf{Dec}(sk, c)$
2: $r' = \mathsf{RRec}(pk, M', c)$
3: $m' = \mathsf{Decode}(M', \mathsf{G}(r'))$
4: $(R', K') := \mathsf{H}(m')$
5: $\boxed{r'' \leftarrow \psi_{\mathcal{R}} \text{ with the randomness } R'}$
6: **if** $m' \neq \perp$ and $\boxed{r' = r''}$
7: **return** K'
8: **else**
9: **return** $\perp$

Figure 14: $\mathsf{KEM}' = \overline{\mathsf{FO}}^{\perp}[\mathsf{PKE}', \mathsf{H}]$

4.2.1 Security Proof in the ROM

Theorem 4.3. Let KEM be a key encapsulation mechanism defined in Figure 12, and let KEM' be another mechanism defined in Figure 14, both constructed based on PKE. Assume that PKE is randomness recoverable and randomness unique, has an average-case correctness error δ, and ensures that outputs of Dec always belong to $\mathcal{M}$ and that SOTP is rigid. For any adversary $\mathcal{A}$ making at most q_{D} decapsulation and q_{H} hash queries against the IND-CCA security of KEM', there exists an adversary $\mathcal{B}$ against the IND-CCA security

of KEM with

$$\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}'}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}}(\mathcal{B}) + (q_{\mathsf{H}} + q_{\mathsf{D}})\delta',$$

where $\delta' = \delta + \|\psi_{\mathcal{R}}\| \cdot (1 + \sqrt{(\ln|\mathcal{M}'| - \ln\|\psi_{\mathcal{R}}\|)/2})$, and $\|\psi_{\mathcal{R}}\| := \sqrt{\sum_r \psi_{\mathcal{R}}(r)^2}$.

Proof. The security proof begins by analyzing hybrid games with a fixed key pair (pk, sk). A detailed explanation of the security proof is provided below.
GAME G_0. G_0 is the original IND-CCA game against KEM' with a fixed key pair (pk, sk).
GAME G_1. In contrast to G_0, the Decap oracle in G_1 is modified. Instead of returning K' when $m' \neq \perp$ and $r' = r''$, K' is now returned if $m' \neq \perp$, $r' \in \mathcal{R}$, and $c = c'$. Note that G_1 is the original IND-CCA game against KEM with a fixed key pair (pk, sk). For ease of analysis, for each sk and $m \in \mathcal{M}$, we define

$$\mathcal{R}'_{\text{bad}}(sk, m) := \Big\{ R \in \mathcal{R}' \ : \mathsf{Dec}(sk, \mathsf{Enc}(pk, M; r)) \neq M, \\ \text{where } r = \mathsf{Sample}(\mathcal{R}; R) \text{ and } M = \mathsf{Encode}(m, \mathsf{G}(r)) \Big\}.$$

and $\mathcal{R}'_{\text{good}}(sk, m) := \mathcal{R}' \setminus \mathcal{R}'_{\text{bad}}(sk, m)$. Additionally, we define $\delta(sk, m) := |\mathcal{R}'_{\text{bad}}(sk, m)|/|\mathcal{R}'|$ and $\delta_{sk} := \max_{m \in \mathcal{M}} \delta(sk, m)$.

Assuming $(R, K) = \mathsf{H}(m) \in \mathcal{R}'_{good}(sk, m) \times \mathcal{K}$ for all m that are queried to H, we now show that the changes in G_1 do not impact adversary $\mathcal{A}$, as the conditions actually imply each other.

Assume that $m' \neq \perp$ and $r' = r''$ hold for a ciphertext c in the Decap oracle. Given the rigidity of the SOTP, the condition $m' = \mathsf{Decode}(M', \mathsf{G}(r')) \neq \perp$ implies $M' = \mathsf{Encode}(m', \mathsf{G}(r'))$, and thus $M' = \mathsf{Encode}(m', \mathsf{G}(r''))$. Moreover, since $r' = r''$ and r'' is sampled from $\psi_{\mathcal{R}}$ using the randomness R', it follows that $r' \in \mathcal{R}$. Additionally, since $M' = \mathsf{Dec}(sk, c)$ is within $\mathcal{M}$ and $r' = \mathsf{RRec}(pk, M', c)$ is also in $\mathcal{R}$, the RR property of the PKE ensures that $c = \mathsf{Enc}(pk, M'; r') = \mathsf{Enc}(pk, \mathsf{Encode}(m', \mathsf{G}(r'')); r'') = c'$.

Conversely, assume that $m' \neq \perp$, $r' \in \mathcal{R}$, and $c = c'$ hold in the Decap oracle. Since $M' = \mathsf{Dec}(sk, c)$ is within $\mathcal{M}$ and $r' = \mathsf{RRec}(pk, M', c)$ is in $\mathcal{R}$, the RR property of the PKE ensures that $c = \mathsf{Enc}(pk, M'; r')$. Additionally, since $c = c' = \mathsf{Enc}(pk, \mathsf{Encode}(m', \mathsf{G}(r'')); r'')$ where r'' is sampled using R' from the pair $(R', K') = \mathsf{H}(m')$, by the definition of H, it follows that $M' = \mathsf{Dec}(sk, c) = \mathsf{Encode}(m', \mathsf{G}(r''))$. Consequently, this implies that $c = \mathsf{Enc}(pk, M'; r'')$. Since $c = \mathsf{Enc}(pk, M'; r') = \mathsf{Enc}(pk, M'; r'')$, the randomness uniqueness of PKE implies $r' = r''$.

GAMES G_0-G_1	$\underline{\mathsf{Decap}(sk, c)}$	
1: $\mathsf{G} \leftarrow (\mathcal{R} \to \mathcal{U})$	1: $M' = \mathsf{Dec}(sk, c)$	
2: $\mathsf{H} \leftarrow (\mathcal{M}' \to \mathcal{R}' \times \mathcal{K})$	2: $r' = \mathsf{RRec}(pk, M', c)$	
3: $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$	3: $m' = \mathsf{Decode}(M', \mathsf{G}(r'))$	
4: $(K_0, c^*) \leftarrow \mathsf{Encap}(pk)$	4: $(R', K') := \mathsf{H}(m')$	
5: $K_1 \leftarrow \mathcal{K}$	5: $r'' \leftarrow \psi_{\mathcal{R}}$ with the randomness R'	
6: $b \leftarrow \{0, 1\}$	6: $c' = \mathsf{Enc}(pk, \mathsf{Encode}(m', \mathsf{G}(r'')); r'')$	//G_1
7: $b' \leftarrow \mathcal{A}^{\mathsf{G},\mathsf{H},\mathsf{Decap}}(pk, c^*, K_b)$	7: **if** $m' \neq \perp$ and $r' = r''$	//G_0
8: **return** $[\![b = b']\!]$	8: **if** $m' \neq \perp$ and $r' \in \mathcal{R}$ and $c = c'$	//G_1
	9: **return** K'	
	10: **else, return** $\perp$	

Figure 15: GAMES G_0-G_1 for the proof of Theorem 4.3

Since these two conditions imply each other, assuming that $(R, K) = \mathsf{H}(m)$ belongs to $\mathcal{R}'_{good}(sk, m) \times \mathcal{K}$ for all m that are queried to H, by Lemma 2.8, the following holds:

$$\left|\Pr[G_0^{\mathcal{A}} \Rightarrow 1] - \Pr[G_1^{\mathcal{A}} \Rightarrow 1]\right| \leq (q_{\mathsf{H}} + q_{\mathsf{D}})\delta_{sk} \tag{7}$$

Therefore, by the triangular inequality, the following holds:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}',sk}(\mathcal{A}) &= \left|\Pr[G_0^{\mathcal{A}} \Rightarrow 1] - 1/2\right| \\ &\leq \left|\Pr[G_0^{\mathcal{A}} \Rightarrow 1] - \Pr[G_1^{\mathcal{A}} \Rightarrow 1]\right| + \left|\Pr[G_1^{\mathcal{A}} \Rightarrow 1] - 1/2\right| \\ &\leq \mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM},sk}(\mathcal{A}) + (q_{\mathsf{H}} + q_{\mathsf{D}})\delta_{sk}.\end{aligned}$$

By taking the expectation over $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$ and applying Corollary 3.3, we obtain the desired bound. □

4.2.2 Security Proof in the QROM

Theorem 4.4. Let KEM be the key encapsulation mechanism defined in Figure 12, and let KEM' be the mechanism defined in Figure 14, both constructed from the same public-key encryption scheme PKE. Assume that PKE is randomness recoverable and randomness unique, has average-case correctness error δ, and ensures that the outputs of Dec always lie in $\mathcal{M}$ and that SOTP is rigid. For any quantum adversary $\mathcal{A}$ making at most q_{D} decapsulation queries and q_{H} (quantum) hash queries against the IND-CCA security of KEM', there exists a quantum adversary $\mathcal{B}$ against the IND-CCA security of KEM with

$$\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}'}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}}(\mathcal{B}) + 4 \cdot (q_{\mathsf{H}} + q_{\mathsf{D}})\sqrt{\delta'},$$

where $\delta' = \delta + \|\psi_{\mathcal{R}}\| \cdot (1 + \sqrt{(\ln|\mathcal{M}'| - \ln\|\psi_{\mathcal{R}}\|)/2})$, and $\|\psi_{\mathcal{R}}\| := \sqrt{\sum_r \psi_{\mathcal{R}}(r)^2}$.

Proof. The security proof proceeds via a sequence of hybrid games, analyzed for a fixed key pair (pk, sk). A detailed explanation of the security proof is provided below.

GAME G_0. G_0 is the original $\mathsf{IND\text{-}CCA}$ game against KEM' with a fixed key pair (pk, sk).

GAME G_1. Unlike G_0, G_1 uses the function H' instead of H. The function H' takes a message m as input and selects randomness from the set $\mathcal{R}'_{\text{good}}$, which consists of all randomness that do not cause decryption errors when encrypting m under the public key pk. Specifically, for a fixed secret key sk and $m \in \mathcal{M}$, we define

$$\begin{aligned}\mathcal{R}'_{\text{bad}}(sk, m) := \Big\{ R \in \mathcal{R}' \ &: \mathsf{Dec}(sk, \mathsf{Enc}(pk, M; r)) \neq M, \\ &\text{where } r = \mathsf{Sample}(\mathcal{R}; R) \text{ and } M = \mathsf{Encode}(m, \mathsf{G}(r)) \Big\}.\end{aligned}$$

and $\mathcal{R}'_{\text{good}}(sk, m) := \mathcal{R}' \setminus \mathcal{R}'_{\text{bad}}(sk, m)$. The function H' is defined as a random function such that $\mathsf{H}'(m)$ is sampled uniformly from $\mathcal{R}'_{\text{good}}(sk, m) \times \mathcal{K}$. We denote by $\Omega_{\mathsf{H}'}$ the set of all possible choices of H'. Finally, we define $\delta(sk, m) := |\mathcal{R}'_{\text{bad}}(sk, m)|/|\mathcal{R}'|$ and $\delta_{sk} := \max_{m \in \mathcal{M}} \delta(sk, m)$.

Note that distinguishing between G_0 and G_1 is equivalent to distinguishing between H from H'. In particular, we construct an adversary $\mathcal{B}$ that distinguishes H from H'. This adversary uses the accessible oracle $\tilde{\mathsf{H}}$ (either H or H'), simulates the view of $\mathcal{A}$, and outputs the same results as in games G_0 and G_1. When $\tilde{\mathsf{H}} = \mathsf{H}$, $\mathcal{B}^{\tilde{\mathsf{H}}}(sk)$ perfectly simulates G_0, so $\Pr[1 \leftarrow \mathcal{B}^{\mathsf{H}}(sk)] = \Pr[G_0^{\mathcal{A}} \Rightarrow 1]$. Similarly, when $\tilde{\mathsf{H}} = \mathsf{H}'$, $\mathcal{B}^{\tilde{\mathsf{H}}}(sk)$ simulates G_1, yielding $\Pr[1 \leftarrow \mathcal{B}^{\mathsf{H}'}(sk)] = \Pr[G_1^{\mathcal{A}} \Rightarrow 1]$. Therefore,

$$|\Pr[G_0^{\mathcal{A}} \Rightarrow 1] - \Pr[G_1^{\mathcal{A}} \Rightarrow 1]| = |\Pr[1 \leftarrow \mathcal{B}^{\mathsf{H}}(sk)] - \Pr[1 \leftarrow \mathcal{B}^{\mathsf{H}'}(sk)]|. \tag{8}$$

GAMES G_0-G_3		$\underline{\mathsf{Decap}(sk, c)}$	
1: $\mathsf{G} \leftarrow (\mathcal{R} \to \mathcal{U})$		1: $M' = \mathsf{Dec}(sk, c)$	
2: $\mathsf{H} \leftarrow (\mathcal{M}' \to \mathcal{R}' \times \mathcal{K})$	//G_0, G_3	2: $r' = \mathsf{RRec}(pk, M', c)$	
3: $\mathsf{H}' \leftarrow \Omega_{\mathsf{H}'}$	//G_1-G_2	3: $m' = \mathsf{Decode}(M', \mathsf{G}(r'))$	
4: $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$		4: $(R', K') := \mathsf{H}(m')$	
5: $(K_0, c^*) \leftarrow \mathsf{Encap}(pk)$		5: $r'' \leftarrow \psi_{\mathcal{R}}$ with the randomness R'	
6: $K_1 \leftarrow \mathcal{K}$		6: $c' = \mathsf{Enc}(pk, \mathsf{Encode}(m', \mathsf{G}(r''))\,; r'')$	//G_2-G_3
7: $b \leftarrow \{0, 1\}$		7: **if** $m' \neq \perp$ and $r' = r''$	//G_0-G_1
8: $b' \leftarrow \mathcal{A}^{\mathsf{G},\mathsf{H},\mathsf{Decap}}(pk, c^*, K_b)$	//G_0, G_3	8: **if** $m' \neq \perp$ and $r' \in \mathcal{R}$ and $c = c'$	//G_2-G_3
9: $b' \leftarrow \mathcal{A}^{\mathsf{G},\mathsf{H}',\mathsf{Decap}}(pk, c^*, K_b)$	//G_1-G_2	9: **return** K'	
10: **return** $[\![b = b']\!]$		10: **else, return** $\perp$	

Figure 16: GAMES G_0-G_3 for the proof of Theorem 4.4

$\underline{\mathcal{C}^{\mathsf{N}}(sk)}$	$\underline{\tilde{\mathsf{H}}(m)}$
1: Select $2q_{\mathsf{H}}$-wise functions f_1 and f_2	1: **if** $\mathsf{N}(m) = 0$
2: $b \leftarrow \mathcal{B}^{\tilde{\mathsf{H}}}(sk)$	2: $R \leftarrow \mathcal{R}'_{\text{good}}(sk, m)$ with the randomness $f_1(m)$
3: **return** b	3: **else**
	4: $R \leftarrow \mathcal{R}'_{\text{bad}}(sk, m)$ with the randomness $f_1(m)$
	5: $K \leftarrow \mathcal{K}$ with randomness $f_2(m)$
	6: **return** (R, K)

Figure 17: $\mathcal{C}^{\mathsf{N}}(sk)$ for the proof of Theorem 4.4

Next, we demonstrate that any adversary $\mathcal{B}$ distinguishing H from H' can be converted into an adversary $\mathcal{C}$ distinguishing N_1 from N_2. Specifically, N_1 is a function where $\mathsf{N}_1(m)$ is sampled from the Bernoulli distribution $B_{\delta(sk,m)}$, meaning $\Pr[\mathsf{N}_1(m) = 1] = \delta(sk, m)$ and $\Pr[\mathsf{N}_1(m) = 0] = 1 - \delta(sk, m)$. In contrast, N_2 is a constant function that always outputs 0. For any adversary $\mathcal{B}^{\mathsf{H}}(sk)$, we construct an adversary $\mathcal{C}^{\mathsf{N}}(sk)$ as described in Figure 17. Importantly, $\mathcal{C}$ simulates $\tilde{\mathsf{H}} = \mathsf{H}$ when $\mathsf{N} = \mathsf{N}_1$ and $\tilde{\mathsf{H}} = \mathsf{H}'$ when $\mathsf{N} = \mathsf{N}_2$. Thus, $\Pr[1 \leftarrow \mathcal{C}^{\mathsf{N}_1}] = \Pr[1 \leftarrow \mathcal{B}^{\mathsf{H}}]$ and $\Pr[1 \leftarrow \mathcal{C}^{\mathsf{N}_2}] = \Pr[1 \leftarrow \mathcal{B}^{\mathsf{H}'}]$. Therefore, by Lemma 2.10

$$\left|\Pr[1 \leftarrow \mathcal{B}^{\mathsf{H}}(sk)] - \Pr[1 \leftarrow \mathcal{B}^{\mathsf{H}'}(sk)]\right| \tag{9}$$

$$= \left|\Pr[1 \leftarrow \mathcal{C}^{\mathsf{N}_1}(sk)] - \Pr[1 \leftarrow \mathcal{C}^{\mathsf{N}_2}(sk)]\right| \leq 2 \cdot (q_{\mathsf{H}} + q_{\mathsf{D}})\sqrt{\delta_{sk}}. \tag{10}$$

Therefore, by combining Equations (8)–(10),

$$\left|\Pr[G_0^{\mathcal{A}} \Rightarrow 1] - \Pr[G_1^{\mathcal{A}} \Rightarrow 1]\right| \leq 2 \cdot (q_{\mathsf{H}} + q_{\mathsf{D}})\sqrt{\delta_{sk}}. \tag{11}$$

GAME G_2. In contrast to G_1, the Decap oracle in G_2 is modified. Instead of returning K' when $m' \neq \perp$ and $r' = r''$, K' is now returned if $m' \neq \perp$, $r' \in \mathcal{R}$, and $c = c'$. We can show that this modification does not affect the adversary $\mathcal{A}$ by proving that these two conditions actually imply each other.

Assume that $m' \neq \perp$ and $r' = r''$ hold for a ciphertext c in the Decap oracle. Given the rigidity of the SOTP, the condition $m' = \mathsf{Decode}(M', \mathsf{G}(r')) \neq \perp$ implies $M' = \mathsf{Encode}(m', \mathsf{G}(r'))$, and thus $M' = \mathsf{Encode}(m', \mathsf{G}(r''))$. Moreover, since $r' = r''$ holds and r'' is sampled from $\psi_{\mathcal{R}}$ using the randomness R', it follows that $r' \in \mathcal{R}$. Additionally, since $M' = \mathsf{Dec}(sk, c)$ is within $\mathcal{M}$ and $r' = \mathsf{RRec}(pk, M', c)$ is also in $\mathcal{R}$, the RR property of the PKE ensures that $c = \mathsf{Enc}(pk, M'; r') = \mathsf{Enc}(pk, \mathsf{Encode}(m', \mathsf{G}(r'')); r'') = c'$.

Conversely, assume that $m' \neq \perp$, $r' \in \mathcal{R}$, and $c = c'$ hold in the Decap oracle. Since $M' = \mathsf{Dec}(sk, c)$ is within $\mathcal{M}$ and $r' = \mathsf{RRec}(pk, M', c)$ is in $\mathcal{R}$, the RR property of the PKE ensures that $c = \mathsf{Enc}(pk, M'; r')$. Additionally, since $c = c' = \mathsf{Enc}(pk, \mathsf{Encode}(m', \mathsf{G}(r'')); r'')$ where r'' is sampled using R' from the pair $(R', K') = \mathsf{H}(m')$, by the definition of H, it follows that $M' = \mathsf{Dec}(sk, c) = \mathsf{Encode}(m', \mathsf{G}(r''))$. Consequently, this implies that $c = \mathsf{Enc}(pk, M'; r'')$. Since $c = \mathsf{Enc}(pk, M'; r') = \mathsf{Enc}(pk, M'; r'')$, the randomness uniqueness of PKE implies $r' = r''$.

Since these two conditions imply each other, the following holds:

$$\left|\Pr[G_1^{\mathcal{A}} \Rightarrow 1] - \Pr[G_2^{\mathcal{A}} \Rightarrow 1]\right| = 0. \tag{12}$$

GAME G_3. Unlike G_2, G_3 uses the function H instead of H'. Note that G_3 is the original $\mathsf{IND\text{-}CCA}$ game against KEM with a fixed key pair (pk, sk). By the similar analysis between G_0 and G_1, the following holds:

$$\left|\Pr[G_2^{\mathcal{A}} \Rightarrow 1] - \Pr[G_3^{\mathcal{A}} \Rightarrow 1]\right| \leq 2 \cdot (q_{\mathsf{H}} + q_{\mathsf{D}})\sqrt{\delta_{sk}}. \tag{13}$$

By combining Equations (11)-(13) with the triangle inequality, the following holds:

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}',sk}(\mathcal{A}) &= \left|\Pr[G_0^{\mathcal{A}} \Rightarrow 1] - 1/2\right| \\
&\leq \left|\Pr[G_0^{\mathcal{A}} \Rightarrow 1] - \Pr[G_1^{\mathcal{A}} \Rightarrow 1]\right| + \left|\Pr[G_1^{\mathcal{A}} \Rightarrow 1] - \Pr[G_2^{\mathcal{A}} \Rightarrow 1]\right| \\
&\quad + \left|\Pr[G_2^{\mathcal{A}} \Rightarrow 1] - \Pr[G_3^{\mathcal{A}} \Rightarrow 1]\right| + \left|\Pr[G_3^{\mathcal{A}} \Rightarrow 1] - 1/2\right| \\
&\leq \mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM},sk}(\mathcal{A}) + 4 \cdot (q_{\mathsf{H}} + q_{\mathsf{D}})\sqrt{\delta_{sk}}.
\end{aligned}$$

By taking the expectation over $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$ and applying corollary 3.3 yields the required bound of the theorem. □

5 NTRU+

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

Figure 18 defines $\mathsf{GenNTRU}[\psi_1^n]$ relative to the distribution ψ_1^n over R_q. Since $\mathsf{GenNTRU}[\psi_1^n]$ must satisfy both MR and RR for our ACWC_2 transformation, Figure 18 also includes two auxiliary algorithms, RRec and MRec. We observe that $\mathsf{RRec}(\mathbf{h}, \mathbf{m}, \mathbf{c})$ is required during ACWC_2, because $\mathbf{r}$ must be recovered from a ciphertext $\mathbf{c}$ once the corresponding message $\mathbf{m}$ is obtained. The RR property ensures that this randomness-recovery process works well, because for a ciphertext $\mathbf{c} = \mathsf{Enc}(\mathbf{h}, \mathbf{m}, \mathbf{r}) = \mathbf{hr} + \mathbf{m}$, we have $\mathsf{RRec}(\mathbf{h}, \mathbf{m}, \mathbf{c}) = (\mathbf{c} - \mathbf{m})\mathbf{h}^{-1} = \mathbf{r} \in \mathcal{R}$. On the other hand, $\mathsf{MRec}(\mathbf{h}, \mathbf{r}, \mathbf{c})$ is used only in the IND-CPA security proof of the ACWC_2-transformed scheme. The security analysis requires that for a challenge ciphertext $\mathbf{c}^* = \mathsf{Enc}(\mathbf{h}, \mathbf{m}^*, \mathbf{r}^*) = \mathbf{hr}^* + \mathbf{m}^*$, the algorithm $\mathsf{MRec}(\mathbf{h}, \mathbf{r}^*, \mathbf{c}^*)$ returns the corresponding message $\mathbf{m}^*$ if the queried randomness $\mathbf{r}^*$ was used for $\mathbf{c}^*$. The MR property guarantees that once $\mathbf{r}^*$ is given, $\mathsf{MRec}(\mathbf{h}, \mathbf{r}^*, \mathbf{c}^*) = \mathbf{c}^* - \mathbf{hr}^* = \mathbf{m}^* \in \mathcal{M}$.

$\underline{\mathsf{Gen}(1^\lambda)}$
1: **repeat**
2: $\quad \mathbf{f}' \leftarrow \psi_1^n$
3: $\quad \mathbf{f} = 3\mathbf{f}' + 1$
4: **until** $\mathbf{f}$ is invertible in R_q
5: **repeat**
6: $\quad \mathbf{g} \leftarrow \psi_1^n$
7: **until** $\mathbf{g}$ is invertible in R_q
8: $\mathbf{h} = 3\mathbf{g}\mathbf{f}^{-1}$
9: **return** $(pk, sk) = (\mathbf{h}, \mathbf{f})$

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

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

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

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

Figure 18: $\mathsf{GenNTRU}[\psi_1^n]$ with average-case correctness error

5.1.1 Security Proofs

Theorem 5.1 (OW-CPA security of $\mathsf{GenNTRU}[\psi_1^n]$)**.** For any adversary $\mathcal{A}$, there exist adversaries $\mathcal{B}$ and $\mathcal{C}$ such that

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

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

GAME G_0. In G_0, we have the original OW-CPA game with $\mathsf{GenNTRU}[\psi_1^n]$. By the definition of the advantage function of the adversary $\mathcal{A}$ against the OW-CPA game, we have

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

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

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

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

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

Combining all the probabilities completes the proof. □

5.1.2 Average-Case Correctness Error

We analyze the average-case correctness error δ relative to the distribution $\psi_{\mathcal{M}} = \psi_{\mathcal{R}} = \psi_1^n$ using the template provided in [31]. We can expand $\mathbf{cf}$ in the decryption algorithm as follows:

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

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

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

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

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

Assuming that each coefficient is independent, we have

$$\Pr\left[\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) \neq m\right] = 1 - \prod_{i=0}^{n-1} \epsilon_i. \tag{14}$$

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

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

Therefore,

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

Now, we analyze $\epsilon_i' = \Pr\left[|\mathbf{gr} + \mathbf{mf}'|_i \leq \frac{q-3}{6}\right]$. To achieve this, we analyze the distribution of $\mathbf{gr}+\mathbf{mf}'$. Following the analysis in [31], we observe that for $i \in [n/2, n]$, the degree-i coefficient of $\mathbf{gr} + \mathbf{mf}'$ is the sum of n independent random variables:

$$c = ba + b'(a + a') \in \{0, \pm 1, \pm 2, \pm 3\}, \text{ where } a, b, a', b' \leftarrow \psi_1. \tag{15}$$

± 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 $c' = ab + a'b'$

Additionally, for $i \in [0, n/2 - 1]$, the degree-i coefficient of $\mathbf{gr} + \mathbf{mf}'$ is the sum of $n - 2i$ random variables c (as in Equation (15)), and $2i$ independent random variables c' of the form:

$$c' = ba + b'a' \in \{0, \pm 1, \pm 2\} \text{ where } a, b, a', b' \leftarrow \psi_1. \tag{16}$$

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

$$\rho_i(X) = \begin{cases} \sum_{j=-3n}^{3n} \rho_{i,j} X^j = \left(\sum_{j=-3}^{3} \theta_j X^j \right)^n \text{ for } i = [n/2, n-1], \\ \sum_{j=-(3n-2i)}^{3n-2i} \rho_{i,j} X^j = \left(\sum_{j=-3}^{3} \theta_j X^j \right)^{n-2i} \left(\sum_{j=-2}^{2} \theta'_j X^j \right)^{2i} \text{ for } i = [0, n/2 - 1], \end{cases} \tag{17}$$

where $\theta_j = \Pr[c = j]$ (see Table 3) and $\theta'_j = \Pr[c' = j]$ (see 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:

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

using the symmetry $\rho_{i,j} = \rho_{i,-j}$. Substituting ϵ'_i into Equation (14) yields the average-case correctness error δ of $\mathsf{GenNTRU}[\psi_1^n]$.

5.1.3 Injectivity

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

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

$(\mathbf{r}_1 - \mathbf{r}_2, \mathbf{m}_1 - \mathbf{m}_2)$ becomes an approximate shortest vector in $\mathcal{L}_0^{\perp}$. Thus, if the injectivity is broken against $\mathsf{GenNTRU}[\psi_1^n]$, we can solve the approximate shortest vector problem (SVP) (of length at most $2\sqrt{n}$) over $\mathcal{L}_0^{\perp}$. It is well-known [16] that the approximate SVP over $\mathcal{L}_0^{\perp}$ is at least as hard as the $\mathsf{NTRU}_{n,q,\psi_1^n}$ problem (defined above). Hence, if the $\mathsf{NTRU}_{n,q,\psi_1^n}$ assumption holds, then the injectivity of $\mathsf{GenNTRU}[\psi_1^n]$ also holds.

5.1.4 Spreadness

Lemma 5.2 (Spreadness). $\mathsf{GenNTRU}[\psi_1^n]$ is n-spread.

Proof. For a fixed message $\mathbf{m}$ and ciphertext $\mathbf{c}$, there exists at most one $\mathbf{r}$ such that $\mathbf{c} = \mathsf{Enc}(\mathbf{h}, \mathbf{m}; \mathbf{r})$. Suppose there exist $\mathbf{r}_1$ and $\mathbf{r}_2$ such that $c = \mathsf{Enc}(\mathbf{h}, \mathbf{m}; \mathbf{r}_1) = \mathsf{Enc}(\mathbf{h}, \mathbf{m}; \mathbf{r}_2)$. Based on this assumption,

$\mathbf{hr}_1 + \mathbf{m} = \mathbf{hr}_2 + \mathbf{m}$ holds. By subtracting $\mathbf{m}$ and multiplying $\mathbf{h}^{-1}$ on both sides of the equation, we obtain $\mathbf{r} = \mathbf{r}'$. Therefore, there exists at most one $\mathbf{r}$ such that $\mathbf{c} = \mathsf{Enc}(\mathbf{h}, \mathbf{m}; \mathbf{r})$.

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

5.1.5 Randomness-Recoverability

Lemma 5.3. $\mathsf{GenNTRU}[\psi_1^n]$ is randomness recoverable.

Proof. Suppose $\mathbf{r} = \mathsf{RRec}(\mathbf{h}, \mathbf{m}, \mathbf{c}) = (\mathbf{c} - \mathbf{m})\mathbf{h}^{-1} \in \mathcal{R} = \{-1, 0, 1\}$ for $\mathbf{m} \in \mathcal{M} = \{-1, 0, 1\}$ and $\mathbf{c} \in \mathcal{C} = R_q$. Then, multiplying $\mathbf{h}$ and then adding $\mathbf{m}$ to both sides of $\mathbf{r} = (\mathbf{c} - \mathbf{m})\mathbf{h}^{-1}$ leads to $\mathbf{c} = \mathbf{hr} + \mathbf{m} = \mathsf{Enc}(\mathbf{h}, \mathbf{m}; \mathbf{r})$. □

5.1.6 Message-Recoverability

Lemma 5.4. $\mathsf{GenNTRU}[\psi_1^n]$ is AC-MR.

Proof. Suppose $\mathbf{m} = \mathsf{MRec}(\mathbf{h}, \mathbf{r}, \mathbf{c}) = \mathbf{c} - \mathbf{hr} \in \mathcal{M} = \{-1, 0, 1\}^n$ for $\mathbf{r} \in \mathcal{R} = \{-1, 0, 1\}^n$ and $\mathbf{c} \in \mathcal{C} = R_q$. Then, adding $\mathbf{hr}$ to both sides of $\mathbf{m} = \mathbf{c} - \mathbf{hr}$ leads to $\mathbf{c} = \mathbf{hr} + \mathbf{m} = \mathsf{Enc}(\mathbf{h}, \mathbf{m}; \mathbf{r})$. □

Lemma 5.5. $\mathsf{GenNTRU}[\psi_1^n]$ is VC-MR.

Proof. Suppose that $\mathbf{c} = \mathsf{Enc}(\mathbf{h}, \mathbf{m}; \mathbf{r}) = \mathbf{hr} + \mathbf{m}$ with $\mathbf{m} \in \mathcal{M} = \{-1, 0, 1\}^n$ and $\mathbf{r} \in \mathcal{R} = \{-1, 0, 1\}^n$. When we apply MRec to $(\mathbf{h}, \mathbf{r}, \mathbf{c})$, we obtain $\mathsf{MRec}(\mathbf{h}, \mathbf{r}, \mathbf{c}) = \mathbf{c} - \mathbf{hr} = (\mathbf{hr} + \mathbf{m}) - \mathbf{hr} = \mathbf{m}$. Therefore, whenever $\mathbf{c}$ is a valid ciphertext of $\mathbf{m}$ under randomness $\mathbf{r}$, the algorithm MRec recovers $\mathbf{m}$ exactly, as required by VC-MR. Hence $\mathsf{GenNTRU}[\psi_1^n]$ is VC-MR. □

5.1.7 Randomness-Uniqueness

Lemma 5.6. $\mathsf{GenNTRU}[\psi_1^n]$ is randomness unique.

Proof. Suppose $\mathbf{c} = \mathsf{Enc}(\mathbf{h}, \mathbf{m}; \mathbf{r}) = \mathbf{hr} + \mathbf{m}$ and $\mathbf{c}' = \mathsf{Enc}(\mathbf{h}, \mathbf{m}; \mathbf{r}') = \mathbf{hr}' + \mathbf{m}$ satisfy $\mathbf{c} = \mathbf{c}'$. Then, $\mathbf{hr} + \mathbf{m} = \mathbf{hr}' + \mathbf{m}$. Since $\mathbf{h}$ is invertible, we can conclude that $\mathbf{r} = \mathbf{r}'$ by subtracting $\mathbf{m}$ and then multiplying both sides of the equation by $\mathbf{h}$. □

5.2 CPA-NTRU+ (=PKE $'$)

5.2.1 Instantiation of SOTP

We introduce an instantiation of $\mathsf{SOTP} = (\mathsf{Encode}, \mathsf{Decode})$, where $\mathsf{Encode} : \mathcal{M}' \times \mathcal{U} \to \mathcal{M}$ and $\mathsf{Decode} : \mathcal{M} \times \mathcal{U} \to \mathcal{M}'$, with $\mathcal{M}' = \{0, 1\}^n$, $\mathcal{U} = \{0, 1\}^{2n}$, and $\mathcal{M} = \{-1, 0, 1\}^n$, along with distributions $\psi_{\mathcal{U}} = U^{2n}$ and $\psi_{\mathcal{M}} = \psi_1^n$ as shown in Figure 19, which is used for ACWC_2. We note that, following [28], the values of $y + u_2$ generated by Decode should be checked to determine whether they are 0 or 1.

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, we can easily check that SOTP is perfectly rigid as long as $y + u_2 \in \{0, 1\}^n$.

$\underline{\mathsf{Encode}(x \in \mathcal{M}', u \leftarrow U^{2n})}$

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

$\underline{\mathsf{Decode}(y \in \mathcal{M}, u \in U^{2n})}$

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

Figure 19: SOTP instantiation for NTRU+

5.2.2 CPA-NTRU+ (=PKE $'$)

We obtain CPA-NTRU+ := ACWC_2 [GenNTRU[ψ_1^n],SOTP, G] by applying ACWC_2 from Section 3 to GenNTRU[ψ_1^n]. Because the underlying GenNTRU[ψ_1^n] provides injectivity, AC-MR, VC-MR, and RR properties, Theorems 3.6 and 3.7 provide us with 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 the worst-case correctness error of CPA-NTRU+ and the average-case correctness error of GenNTRU[ψ_1^n] differ by the amount of $\Delta = \|\psi_{\mathcal{R}}\| \cdot (1 + \sqrt{(\ln|\mathcal{M}'| - \ln\|\psi_{\mathcal{R}}\|)/2})$, where $\psi_{\mathcal{R}}$ and $\mathcal{M}'$ are specified by ψ_1^n and $\{0,1\}^n$, respectively. For instance, when $n = 768$, we obtain about $\Delta = 2^{-1083}$.

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

1: $(pk, sk) := \mathsf{GenNTRU}[\psi_1^n].\mathsf{Gen}(1^\lambda)$
  - **repeat**
    - $\mathbf{f}', \mathbf{g} \leftarrow \psi_1^n$
    - $\mathbf{f} = 3\mathbf{f}' + 1$
  - **until** $\mathbf{f}$ is invertible in R_q
  - **repeat**
    - $\mathbf{g} \leftarrow \psi_1^n$
  - **until** $\mathbf{g}$ is invertible in R_q
  - $(pk, sk) = (\mathbf{h} = 3\mathbf{g}\mathbf{f}^{-1} \bmod q, \mathbf{f})$
2: **return** (pk, sk)

$\underline{\mathsf{Enc}'(pk, m \in \{0,1\}^n; R \leftarrow \{0,1\}^{2n})}$

1: $\mathbf{r} \leftarrow \psi_1^n$ using the randomness R
2: $\mathbf{m} = \mathsf{Encode}(m, \mathsf{G}(\mathbf{r}))$
3: $\mathbf{c} = \mathsf{GenNTRU}[\psi_1^n].\mathsf{Enc}(pk, \mathbf{m}; \mathbf{r})$
  - $\mathbf{c} = \mathbf{h}\mathbf{r} + \mathbf{m}$
4: **return** $\mathbf{c}$

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

1: $\mathbf{m} = \mathsf{GenNTRU}[\psi_1^n].\mathsf{Dec}(sk, \mathbf{c})$
  - $\mathbf{m} = (\mathbf{c}\mathbf{f} \bmod q) \bmod 3$
2: $\mathbf{r} = \mathsf{RRec}(pk, \mathbf{c}, \mathbf{m})$
  - $\mathbf{r} = (\mathbf{c} - \mathbf{m})\mathbf{h}^{-1}$
3: $m = \mathsf{Decode}(\mathbf{m}, \mathsf{G}(\mathbf{r}))$
4: **if** $m = \perp$ or $\mathbf{r} \notin \{-1,0,1\}^n$, **return** $\perp$
5: **return** m

Figure 20: CPA-NTRU+

Spreadness Properties of CPA-NTRU+. To achieve IND-CCA security of the KEM and PKE via $\overline{\mathsf{FO}}^\perp$ and $\overline{\mathsf{FO}}^\perp_{\mathsf{PKE}}$, we need to show the spreadness of CPA-NTRU+. The spreadness can be easily obtained by combining Lemma 3.8 with Lemma 5.2.

5.3 NTRU+

Finally, we achieve IND-CCA secure KEM by applying $\overline{\mathsf{FO}}^\perp$ to CPA-NTRU+. We denote such KEM by NTRU+ := $\overline{\mathsf{FO}}^\perp$[CPA-NTRU+, H]. Figure 21 shows the resultant NTRU+, which is the basis of our implementation in the next section. By combining Theorems 4.1, 4.2, and Theorem 4.3, we can achieve

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

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

repeat
  $\mathbf{f}', \mathbf{g} \leftarrow \psi_1^n$
  $\mathbf{f} = 3\mathbf{f}' + 1$
until $\mathbf{f}$ is invertible in R_q
repeat
  $\mathbf{g} \leftarrow \psi_1^n$
until $\mathbf{g}$ is invertible in R_q
return $(pk, sk) = (\mathbf{h} = 3\mathbf{g}\mathbf{f}^{-1}, \mathbf{f})$

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

$m \leftarrow \{0,1\}^n$
$(R, K) = \mathsf{H}(m)$
$\mathbf{r} \leftarrow \psi_1^n$ using the randomness R
$\mathbf{m} = \mathsf{Encode}(m, \mathsf{G}(\mathbf{r}))$
$\mathbf{c} = \mathbf{h}\mathbf{r} + \mathbf{m}$
return $(\mathbf{c}, K)$

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

$\mathbf{m} = (\mathbf{c}\mathbf{f} \bmod q) \bmod 3$
$\mathbf{r} = (\mathbf{c} - \mathbf{m})\mathbf{h}^{-1}$
$m = \mathsf{Decode}(\mathbf{m}, \mathsf{G}(\mathbf{r}))$
$(R', K) = \mathsf{H}(m)$
$\mathbf{r}' \leftarrow \psi_1^n$ using the randomness R'
if $m = \perp$ or $\mathbf{r} \neq \mathbf{r}'$
  return $\perp$
else
  return K

Figure 21: NTRU+

6 Algorithm Specification

6.1 Auxiliary Functions

Bit Ordering Functions. To convert between byte arrays and bit arrays, we introduce the BytesToBits and BitsToBytes functions in Algorithms 1 and 2. These functions form an inverse pair, so applying one after the other recovers the original input. They are used later in Algorithms 3, 4, and 5.

Algorithm 1: BytesToBits

Require: Byte array $B = (b_0, b_1, \cdots, b_{n/8-1}) \in \mathcal{B}^{n/8}$
Ensure: Bit array $f = (f_0, \cdots, f_{n-1}) \in \{0,1\}^n$
1: **for** i from 0 to $n/8 - 1$ **do**
2: $\quad t = b_i$
3: $\quad$ **for** j from 0 to 7 **do**
4: $\quad\quad f_{8i+j} = t \,\&\, 1$
5: $\quad\quad t = t \gg 1$
6: **return** $f = (f_0, \cdots, f_{n-1})$

Algorithm 2: BitsToBytes

Require: Bit array $f = (f_0, \ldots, f_{n-1}) \in \{0,1\}^n$
Ensure: Byte array $B = (b_0, b_1, \ldots, b_{n/8-1}) \in \mathcal{B}^{n/8}$
1: **for** i from 0 to $n/8 - 1$ **do**
2: $\quad b_i = 0$
3: $\quad$ **for** j from 0 to 7 **do**
4: $\quad\quad b_i = b_i + f_{8i+j} \times 2^j$
5: **return** $(b_0, \ldots, b_{n/8-1})$

Sampling from a Binomial Distribution. To sample the coefficients of a polynomial from the centered binomial distribution with $\eta = 1$, we introduce the CBD_1 function in Algorithm 3. To determine the bit ordering of the input bytes, this function uses the BytesToBits function defined in Algorithm 1.

Algorithm 3: CBD_1

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

Semi-generalized One-Time Pad (SOTP). We define $\mathsf{SOTP} = (\mathsf{Encode}, \mathsf{Decode})$ in Algorithms 4 and 5, respectively. The Encode function is identical to CBD_1, except that it XORs the first half of the random bytes with the message before applying the centered binomial sampling. Accordingly, Encode also uses the BytesToBits (Algorithm 1), as in CBD_1. The corresponding Decode function, given in Algorithm 5, acts as the inverse of Encode and uses BitsToBytes (Algorithm 2) to recover the original byte array.

Algorithm 4: Encode

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

Algorithm 5: Decode

Require: Polynomial $\mathbf{f} \in R_q$
Require: Byte array $B = (b_0, b_1, \cdots, b_{n/4-1})$
Ensure: Message Byte array $m = (m_0, m_1, \cdots, m_{31})$
1: $(\beta_0, \cdots, \beta_{n-1}) := \mathsf{BytesToBits}((b_0, \cdots, b_{n/8-1}))$
2: $(\beta_n, \cdots, \beta_{2n-1}) := \mathsf{BytesToBits}((b_{n/8}, \cdots, b_{n/4-1}))$
3: **for** i from 0 to $n-1$ **do**
4: $\quad$ **if** $f_i + \beta_{i+n} \notin \{0, 1\}$, **return** $\perp$; //Refer to line 8 in Algorithm 13
5: $\quad m_i := ((f_i + \beta_{i+n}) \,\&\, 1) \oplus \beta_i$
$m = \mathsf{BitsToBytes}((m_0, \cdots, m_{n-1}))$
6: **return** m

Encoding and Decoding. To convert a polynomial in R_q to and from its $3n/2$-byte representation, we introduce the Encode_q and Decode_q functions in Algorithms 6 and 7. The Encode_q function assumes that each coefficient of the input polynomial lies in $\{0, \ldots, q-1\}$ before packing it into the byte array. The Decode_q function performs the inverse transformation by recovering the coefficients from the byte array. Together, the two functions form an inverse pair, so applying one after the other recovers the original polynomial.

Algorithm 6: Encode_q

Require: Polynomial $\mathbf{f} = (f_0, \ldots, f_{n-1}) \in R_q$, with each $f_i \in \{0, \ldots, q-1\}$
Ensure: Byte array $B = (b_0, \ldots, b_{3n/2-1})$
1: **for** i from 0 to $n/2 - 1$ **do**
2: $\quad t_0 = f_{2i}$
3: $\quad t_1 = f_{2i+1}$
4: $\quad b_{3i} = t_0$
5: $\quad b_{3i+1} = (t_0 \gg 8) + (t_1 \ll 4)$
6: $\quad b_{3i+2} = t_1 \gg 4$
7: **return** B

Algorithm 7: Decode_q

Require: Byte array $B = (b_0, \ldots, b_{3n/2-1})$
Ensure: Polynomial $\mathbf{f} = (f_0, \ldots, f_{n-1}) \in R_q$, with each $f_i \in \{0, \ldots, q-1\}$

1: **for** i from 0 to $n/2 - 1$ **do**
2: $\quad t_0 = b_{3i}$
3: $\quad t_1 = b_{3i+1}$
4: $\quad t_2 = b_{3i+2}$
5: $\quad f_{2i} = t_0 \mid ((t_1 \,\&\, \text{0x0f}) \ll 8)$
6: $\quad f_{2i+1} = ((t_1 \gg 4) \,\&\, \text{0x0f}) \mid (t_2 \ll 4)$
7: **return** $\mathbf{f}$

Symmetric Primitives. The scheme uses three distinct hash functions, denoted by F, G, and H. Each function is instantiated with SHAKE-256, as shown in Algorithms 8, 9, and 10. We also use SHAKE-256 as an extendable-output function (XOF) when sampling the polynomials $\mathbf{f}'$ and $\mathbf{g}'$ in Algorithm 11.

Algorithm 8: F

Require: Byte array $m = (m_0, m_1, \ldots, m_{3n/2-1})$
Ensure: Byte array $B = (b_0, b_1, \ldots, b_{31})$

1: $(b_0, \ldots, b_{31}) := \mathsf{SHAKE\text{-}256}(\text{0x00} \,\|\, m)$
2: **return** $(b_0, \ldots, b_{31})$

Algorithm 9: G

Require: Byte array $m = (m_0, m_1, \ldots, m_{n/8-1})$
Ensure: Byte array $B = (b_0, b_1, \ldots, b_{n/4-1})$

1: $(b_0, \ldots, b_{n/4-1}) := \mathsf{SHAKE\text{-}256}(\text{0x01} \,\|\, m,\ n/4)$
2: **return** $(b_0, \ldots, b_{n/4-1})$

Algorithm 10: H

Require: Byte array $m = (m_0, m_1, \ldots, m_{n/8+31})$
Ensure: Byte array $B = (b_0, b_1, \ldots, b_{n/4+31})$

1: $(b_0, \ldots, b_{n/4+31}) := \mathsf{SHAKE\text{-}256}(\text{0x02} \,\|\, m,\ n/4 + 32)$
2: **return** $(b_0, \ldots, b_{n/4+31})$

6.2 Number Theoretic Transform

Polynomial Rings and Number Theoretic Transform Throughout this specification, we consider the quotient rings $R = \mathbb{Z}[x]/\langle \Phi_{3n}(x) \rangle$ and $R_q = \mathbb{Z}_q[x]/\langle \Phi_{3n}(x) \rangle$, where $\Phi_{3n}(x) = x^n - x^{n/2} + 1$ denotes the $3n$-th cyclotomic polynomial of degree $n = 2^{a+1}3^b$ for $a, b \in \mathbb{N} \cup \{0\}$. To optimize polynomial multiplication, the Number Theoretic Transform (NTT) is used to establish a ring isomorphism based on the Generalized Chinese Remainder Theorem (GCRT):

$$R_q \cong \prod_{i=0}^{n/d-1} \mathbb{Z}_q[x]/\langle x^d - \zeta^{\texttt{index}[i]} \rangle, \quad d \in \{3, 4\}.$$

Here, ζ is a primitive ℓ-th root of unity modulo q with $\ell = 3n/d$, where the values (d, ζ, ℓ) are defined in Table 5, and the array `index` is defined in Figure 22. This decomposition ensures that high-degree multiplication is reduced to n/d independent multiplications in smaller component rings. Under this decomposition, the forward transform NTT can be written as

$$\begin{aligned} \hat{f} = \mathsf{NTT}(f) &= \left(f \bmod (x^d - \zeta^{\texttt{index}[0]}), \ldots, f \bmod (x^d - \zeta^{\texttt{index}[n/d-1]}) \right) \\ &= \underbrace{(\hat{f}_0, \ldots, \hat{f}_{d-1})}_{\mathbb{Z}_q[x]/\langle x^d - \zeta^{\texttt{index}[0]} \rangle} \| \cdots \| \underbrace{(\hat{f}_{n-d}, \ldots, \hat{f}_{n-1})}_{\mathbb{Z}_q[x]/\langle x^d - \zeta^{\texttt{index}[n/d-1]} \rangle}, \end{aligned}$$

and the inverse transform NTT^{-1} is defined as its inverse.

Concretely, NTT is performed through three different sequential stages of butterfly layers, with iterations determined by the parameter set in Table 5:

1. **Initial Radix-2 Layer:** This layer decomposes the initial cyclotomic ring R_q into two sub-rings:

   $$R_q = \mathbb{Z}_q[x]/\langle x^n - x^{n/2} + 1 \rangle \cong \mathbb{Z}_q[x]/\langle x^{n/2} - \zeta^{\ell/6} \rangle \times \mathbb{Z}_q[x]/\langle x^{n/2} - \zeta^{5\ell/6} \rangle.$$

   This factorization provides the initial split for the recursive transform [31].

2. **Standard Radix-3 Layer:** This layer decomposes the quotient ring as

   $$\mathbb{Z}_q[x]/\langle x^{3m} - \alpha^3 \rangle \cong \prod_{j=0}^{2} \mathbb{Z}_q[x]/\langle x^m - \alpha\omega^j \rangle,$$

   where $\omega = \zeta^{\ell/3}$ is a primitive third root of unity (see [18] or Appendix B).

3. **Standard Radix-2 Layer:** This layer decomposes the quotient ring as

   $$\mathbb{Z}_q[x]/\langle x^{2m} - \psi^2 \rangle \cong \mathbb{Z}_q[x]/\langle x^m - \psi \rangle \times \mathbb{Z}_q[x]/\langle x^m + \psi \rangle.$$

   This decomposition can be equivalently written as $\mathbb{Z}_q[x]/\langle x^m - \psi \rangle \times \mathbb{Z}_q[x]/\langle x^m - \psi\zeta^{\ell/2} \rangle$ because $\zeta^{\ell/2} \equiv -1 \pmod{q}$. Since $\ell/2$ is even, this bisection is well-defined provided that ψ is a quadratic residue. For the chosen parameter sets, this bisection continues until $m = d \in \{3, 4\}$. Note that the Radix-3 Layer is executed prior to the Radix-2 Layer to avoid the need for an additional precomputation table for the base multiplication.

By combining these sequential layers, the initial ring is decomposed into a collection of smaller component rings, ordered according to the `index` array. To make this construction explicit, we describe how the `index` array is generated by following the sequence of butterfly layers defined above. Starting from the initial root values, the corresponding layer operations are applied recursively according to the prescribed schedule in Table 5, with each layer expanding all current leaf nodes before proceeding to the next layer.

- **Initial Radix-2 Layer**: Initializes two independent trees with the root values $\{\ell/6, 5\ell/6\}$.
- **Standard Radix-3 Layer**: For each application of the radix-3 layer, every current leaf node with value ψ is expanded into three ordered child nodes $\{\psi/3, (\psi+\ell)/3, (\psi+2\ell)/3\}$.
- **Standard Radix-2 Layer**: For each application of the radix-2 layer, every current leaf node with value ψ is expanded into two ordered child nodes $\{\psi/2, (\psi+\ell)/2\}$.

Multiplication in the NTT Domain. After transforming a polynomial in R_q into its components in the product rings, multiplication is performed independently in each ring $\mathbb{Z}_q[x]/\langle x^d-\zeta\rangle$. Let $a(x) = \sum_{j=0}^{d-1} a_j x^j$ and $b(x) = \sum_{j=0}^{d-1} b_j x^j$ be polynomials in this ring.

For $d = 2$, the product $c(x) = a(x)b(x)$ is

$$c(x) = a(x)b(x) = (a_0b_0 + a_1b_1\zeta) + (a_0b_1 + a_1b_0)x,$$

which can be written in matrix form as

$$\begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} a_0 & a_1\zeta \\ a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}.$$

For $d = 3$, the product $c(x) = a(x)b(x)$ becomes:

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

which corresponds to the matrix representation

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} a_0 & a_2\zeta & a_1\zeta \\ a_1 & a_0 & a_2\zeta \\ a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}.$$

For $d = 4$, the product $c(x) = a(x)b(x)$ is

$$\begin{aligned} c(x) = a(x)b(x) =& (a_0b_0 + (a_1b_3 + a_2b_2 + a_3b_1)\zeta) + (a_0b_1 + a_1b_0 + (a_2b_3 + a_3b_2)\zeta)x \\ &+ (a_0b_2 + a_1b_1 + a_2b_0 + a_3b_3\zeta)x^2 + (a_0b_3 + a_1b_2 + a_2b_1 + a_3b_0)x^3, \end{aligned}$$

with the corresponding matrix form

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3\zeta & a_2\zeta & a_1\zeta \\ a_1 & a_0 & a_3\zeta & a_2\zeta \\ a_2 & a_1 & a_0 & a_3\zeta \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}.$$

Inversion in the NTT Domain. In the NTT domain, inversion is performed independently in each component ring $\mathbb{Z}_q[x]/\langle x^d - \zeta\rangle$, analogous to multiplication. Let $a(x) = \sum_{j=0}^{d-1} a_j x^j$ and write its inverse as $b(x) = \sum_{j=0}^{d-1} b_j x^j$.

For $d = 2$, the inverse $b(x)$ is obtained from

$$\begin{bmatrix} b_0 \\ b_1 \end{bmatrix} = \begin{bmatrix} a_0 & a_1\zeta \\ a_1 & a_0 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{a_0^2 - a_1^2\zeta} \begin{bmatrix} a_0 & -a_1\zeta \\ -a_1 & a_0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{a_0^2 - a_1^2\zeta} \begin{bmatrix} a_0 \\ -a_1 \end{bmatrix}.$$

For $d = 3$, we similarly compute

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} a_0 & a_2\zeta & a_1\zeta \\ a_1 & a_0 & a_2\zeta \\ a_2 & a_1 & a_0 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = d^{-1} \begin{bmatrix} a_0' \\ a_1' \\ a_2' \end{bmatrix},$$

where $a_0' = a_0^2 - \zeta a_1 a_2$, $a_1' = \zeta a_2^2 - a_0 a_1$, $a_2' = a_1^2 - a_0 a_2$, and $d = a_0 a_0' + \zeta(a_1 a_2' + a_2 a_1')$.

For $d = 4$, direct matrix inversion is less efficient. Following [35], we reduce inversion in $\mathbb{Z}_q[x]/\langle x^4 - \zeta\rangle$ to inversion in $\mathbb{Z}_q[z]/\langle z^2 - \zeta\rangle$, where $z = x^2$. We rewrite

$$a(x) = \tilde{a}_0(z) + \tilde{a}_1(z)x, \quad \text{where } \tilde{a}_0(z) = a_0 + a_2 z, \tilde{a}_1(z) = a_1 + a_3 z.$$

For $b(x) = \tilde{b}_0(z) + \tilde{b}_1(z)x$, the product becomes

$$\begin{aligned} c(x) = a(x)b(x) &= (\tilde{a}_0(z) + \tilde{a}_1(z)x) \cdot (\tilde{b}_0(z) + \tilde{b}_1(z)x) \\ &= \tilde{a}_0(z)\tilde{b}_0(z) + (\tilde{a}_0(z)\tilde{b}_1(z) + \tilde{a}_1(z)\tilde{b}_0(z))x + \tilde{a}_1(z)\tilde{b}_1(z)x^2 \\ &= (\tilde{a}_0(z)\tilde{b}_0(z) + \tilde{a}_1(z)\tilde{b}_1(z)z) + (\tilde{a}_0(z)\tilde{b}_1(z) + \tilde{a}_1(z)\tilde{b}_0(z))x, \end{aligned}$$

which corresponds to the matrix product

$$\begin{bmatrix} \tilde{c}_0(z) \\ \tilde{c}_1(z) \end{bmatrix} = \begin{bmatrix} \tilde{a}_0(z) & \tilde{a}_1(z)z \\ \tilde{a}_1(z) & \tilde{a}_0(z) \end{bmatrix} \begin{bmatrix} \tilde{b}_0(z) \\ \tilde{b}_1(z) \end{bmatrix}.$$

To find the inverse $b(x) = \tilde{b}_0(z) + \tilde{b}_1(z)x$, we use:

$$\begin{aligned} \begin{bmatrix} \tilde{b}_0(z) \\ \tilde{b}_1(z) \end{bmatrix} &= \begin{bmatrix} \tilde{a}_0(z) & \tilde{a}_1(z)z \\ \tilde{a}_1(z) & \tilde{a}_0(z) \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\tilde{a}_0^2(z) - \tilde{a}_1^2(z)z} \begin{bmatrix} \tilde{a}_0(z) & -\tilde{a}_1(z)z \\ -\tilde{a}_1(z) & \tilde{a}_0(z) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ &= \frac{1}{\tilde{a}_0^2(z) - \tilde{a}_1^2(z)z} \begin{bmatrix} \tilde{a}_0(z) \\ -\tilde{a}_1(z) \end{bmatrix} \in \mathbb{Z}_q[z]/\langle z^2 - \zeta\rangle^{1\times 2}. \end{aligned}$$

The inverse of $\tilde{a}_0^2(z) - \tilde{a}_1^2(z)z$ in $\mathbb{Z}_q[z]/\langle z^2 - \zeta\rangle$ can be computed using the $d = 2$ case, and once the inversion is completed in that ring, the final result is obtained by substituting $z = x^2$.

In all cases, the multiplicative inverse modulo q must be computed. To reduce the risk of side-channel leakage, we use Fermat's Little Theorem instead of the extended Euclidean algorithm: if $a \neq 0$ and is coprime with q, then $a^{q-1} \equiv 1 \pmod{q}$, and thus the inverse of a can be obtained as $a^{q-2} \bmod q$. Note that $a = 0$ is never invertible in $\mathbb{Z}_q$, and such values are excluded by construction.

n	q	Initial Radix-2	Standard Radix-3	Standard Radix-2	d	ζ	ℓ
768	3457	1	1	5	4	22	576
864	3457	1	2	4	3	9	864
1152	3457	1	2	4	4	9	864

Table 5: Layer configurations for the NTT

- NTRU+768

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

- NTRU+864 and NTRU+1152

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

Figure 22: Index for the NTT

6.3 Specification of NTRU+

6.3.1 NTRU+

We specify the key encapsulation mechanism NTRU+. In contrast to the construction in Section 5.3, this version instantiates a variant of $\overline{\mathsf{FO}}^{\perp}$ in which the public key is hashed into the key-derivation input, following the public-key–dependent FO transformation first introduced in KYBER [34] to enhance robustness in multi-user settings. Algorithms 11, 12, and 13 specify the key generation, encapsulation, and decapsulation procedures of NTRU+.

Algorithm 11: $\mathsf{Gen}(1^\lambda)$: key generation

Ensure: Public key $pk \in \mathcal{B}^{\lceil \log_2 q \rceil \cdot n/8}$
Ensure: Secret key $sk \in \mathcal{B}^{\lceil \log_2 q \rceil \cdot n/4+32}$
repeat
  $d \leftarrow \mathcal{B}^{32}$
  $f := \mathsf{SHAKE\text{-}256}(d, n/4)$
  $\mathbf{f}' := \mathsf{CBD}_1(f)$
  $\mathbf{f} := 3\mathbf{f}' + 1$
  $\hat{\mathbf{f}} := \mathsf{NTT}(\mathbf{f})$
until $\mathbf{f}$ is invertible in R_q
repeat
  $d \leftarrow \mathcal{B}^{32}$
  $g := \mathsf{SHAKE\text{-}256}(d, n/4)$
  $\mathbf{g}' := \mathsf{CBD}_1(g)$
  $\mathbf{g} := 3\mathbf{g}'$
  $\hat{\mathbf{g}} := \mathsf{NTT}(\mathbf{g})$
until $\mathbf{g}$ is invertible in R_q
$\hat{\mathbf{h}} := \hat{\mathbf{g}} \circ \hat{\mathbf{f}}^{-1}$
$pk := \mathsf{Encode}_q(\hat{\mathbf{h}})$
$sk := \mathsf{Encode}_q(\hat{\mathbf{f}}) \,\|\, \mathsf{Encode}_q(\hat{\mathbf{h}}^{-1}) \,\|\, \mathsf{F}(pk)$
return $(pk,\ sk)$

Algorithm 12: $\mathsf{Encap}(pk)$: encapsulation

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

Algorithm 13: $\mathsf{Decap}(sk, c)$: decapsulation

Require: Secret key $sk \in \mathcal{B}^{\lceil \log_2 q \rceil \cdot n/4+32}$
Require: Ciphertext $c \in \mathcal{B}^{\lceil \log_2 q \rceil \cdot n/8}$
Ensure: Shared key $K \in \mathcal{B}^{32}$
1: Parse $sk = (sk_1, sk_2, sk_3) \in \mathcal{B}^{\lceil \log_2 q \rceil \cdot n/8} \times \mathcal{B}^{\lceil \log_2 q \rceil \cdot n/8} \times \mathcal{B}^{32}$
2: $\hat{\mathbf{f}} := \mathsf{Decode}_q(sk_1)$
3: $\hat{\mathbf{c}} := \mathsf{Decode}_q(c)$
4: $\mathbf{m} := \mathsf{NTT}^{-1}(\hat{\mathbf{c}} \circ \hat{\mathbf{f}}) \bmod {}^{\pm}3$
5: $\hat{\mathbf{m}} := \mathsf{NTT}(\mathbf{m})$
6: $\hat{\mathbf{h}}^{-1} := \mathsf{Decode}_q(sk_2)$
7: $\hat{\mathbf{r}} := (\hat{\mathbf{c}} - \hat{\mathbf{m}}) \circ \hat{\mathbf{h}}^{-1}$ //randomness recovery (RRec)
8: $m' := \mathsf{Decode}(\mathbf{m}, \mathsf{G}(\mathsf{Encode}_q(\hat{\mathbf{r}})))$ //Check $m' = \perp$ in line 12
9: $(K, r') := \mathsf{H}(m', sk_3)$
10: $\mathbf{r}' := \mathsf{CBD}_1(r')$
11: $\hat{\mathbf{r}}' := \mathsf{NTT}(\mathbf{r}')$
12: **if** $m' = \perp$ or $\hat{\mathbf{r}} \neq \hat{\mathbf{r}}'$, **return** $\perp$ //Check if $m' = \perp$ or $\mathbf{r}' \notin R_q$
13: **else**, **return** K

7 Parameters and Security Analysis

We define three parameter sets for NTRU+, which are listed in Table 7. We denote them by NTRU+{768, 864, and 1152}, respectively, depending on the degree $n \in \{768, 864, 1152\}$ of the polynomial $x^n - x^{n/2} + 1$. In all parameter sets the modulus is fixed to $q = 3457$, and the coefficients of $\mathbf{m}$ and $\mathbf{r}$ are sampled according to the distribution ψ_1^n (i.e., $\psi_{\mathcal{R}} = \psi_{\mathcal{M}} = \psi_1^n$). For each tuple $(n, q, \psi_1^n, \mathcal{M}' = \{0,1\}^n)$, the worst-case correctness error δ' is calculated as $\delta' = \delta + \Delta$, where δ is the average-case correctness error of $\mathsf{GenNTRU}[\psi_1^n]$ and $\Delta = \|\psi_{\mathcal{R}}\| \cdot \left(1 + \sqrt{(\ln|\mathcal{M}'| - \ln\|\psi_{\mathcal{R}}\|)/2}\right)$ as given in Theorem 3.2. Since Δ is negligible for all parameter sets, the worst-case correctness error of NTRU+ is essentially identical to the average-case correctness error of the corresponding $\mathsf{GenNTRU}[\psi_1^n]$.

To estimate the concrete security level of NTRU+, we analyze the hardness of the $\mathsf{RLWE}_{n,q,\psi_1^n}$ and $\mathsf{NTRU}_{n,q,\psi_1^n}$ problems for each parameter set. For the RLWE problem, we employ the Lattice estimator [1] [5], which evaluates the best-known lattice attacks using the BKZ reduction algorithm [11], including the primal [2] and dual [29] attacks. For the NTRU problem, we use the NTRU estimator provided by the finalist NTRU submission [10], which incorporates both the primal attack and Howgrave–Graham's hybrid attack [21] over the NTRU lattice. The primal attack over the NTRU lattice is essentially the same as the attack using the BKZ algorithm, and Howgrave-Graham's hybrid attack is also based on the BKZ algorithm combined with Odlyzko's Meet-in-the-Middle (MitM) attack [24] on a (reduced) sub-lattice. As a result, the concrete security level of the NTRU problem closely matches that of the RLWE problem. Table 6 summarizes the resulting security levels for each NTRU+ parameter set. For the BKZ cost model, we use $2^{0.292\beta}$ [5] in the classical setting and $2^{0.257\beta}$ [9] in the quantum setting.

Recently, Lee *et al.* [27] proposed a combinatorial attack that improves upon May's Meet-LWE attack [32] and analyzed the concrete security level of NTRU+. Their analysis shows that the security of NTRU+ against this combinatorial attack does not fall below the levels predicted by the above Lattice and NTRU estimators.

Scheme	classical		quantum	
	LWE	NTRU	LWE	NTRU
NTRU+768	156	164	139	144
NTRU+864	179	189	160	166
NTRU+1152	248	266	222	233

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

[5] https://github.com/malb/lattice-estimator/tree/352ddaf4a288a0543f5d9eb588d2f89c7acec463

8 Performance Analysis

All benchmarks were obtained on a single core of an Intel Core i7-8700K (Coffee Lake) processor running at 3.7 GHz, with the benchmarking machine equipped with 64 GB of RAM. Implementations were compiled using gcc version 11.4.0. Table 7 lists the execution time of our optimized C and AVX2 implementations of NTRU+, as well as those of NTRU[6], and KYBER[7], together with the estimated security level and the sizes of the secret key, public key, and ciphertext. Execution time represent the average cycle counts over 100,000 runs of each algorithm. The source code for NTRU+ is available at https://github.com/ntruplus/ntruplus.

When comparing NTRU and NTRU+, Table 7 shows that the two schemes achieve similar bandwidth (public key plus ciphertext) at comparable security levels. For example, NTRU+864 at the 179-bit security level requires 2,592 bytes of bandwidth, while ntruhps4096821 at the 171-bit security level requires 2,460 bytes. In terms of secret-key stroage, however, NTRU+ requires almost twice the cost of NTRU, because it stores $(\mathbf{f}, \mathbf{h}^{-1}, \mathsf{F}(pk))$ as the secret key, whereas NTRU stores only $\mathbf{f}$. On the other hand, with respect to execution time, NTRU+ outperforms NTRU, largely due to the use of NTT-friendly rings.

When comparing KYBER and NTRU+, the bandwidth of NTRU+ is slightly larger than that of KYBER at comparable security levels. This is because KYBER uses a rounding technique to reduce the ciphertext size. In terms of efficiency, Table 7 shows that, at similar security levels, the key generation, encapsulation, and decapsulation of NTRU+ are all faster than those of KYBER in both the optimized C and AVX2 implementations.

[6] We use the code submitted to the NIST PQC Standardization Round 3.

[7] https://github.com/pq-crystals/kyber/tree/4768bd37c02f9c40a46cb49d4d1f4d5e612bb882

Table 7: Comparison between the finalist NTRU, KYBER and NTRU+

Scheme	security level		n	q	pk	ct	sk	$\log_2 \delta'$	optimized C			AVX2		
	classical	quantum							Gen	Encap	Decap	Gen	Encap	Decap
NTRU+768	156	139	768	3457	1152	1152	2336	-379	99	75	78	27	31	19
NTRU+864	179	160	864	3457	1296	1296	2624	-340	111	87	90	29	36	24
NTRU+1152	248	222	1152	3457	1728	1728	3488	-260	176	118	126	44	45	30
KYBER512	115	103	512	3329	800	768	1632	-139	135	159	203	28	29	30
KYBER768	174	155	768	3329	1184	1088	2400	-164	232	258	318	45	44	46
KYBER1024	241	215	1024	3329	1568	1568	3168	-174	334	388	463	62	62	66
ntruhps2048509	104	93	509	2048	699	699	935	$-\infty$	9296	624	1657	177	45	40
ntruhrss701	132	117	701	8192	1138	1138	1450	$-\infty$	17414	1060	3119	270	40	61
ntruhps2048677	142	127	677	2048	930	930	1234	$-\infty$	16148	1071	2900	286	62	58
ntruhps4096821	171	152	821	4096	1230	1230	1590	$-\infty$	21879	2003	4261	400	72	73

n: polynomial degree of the ring. q: modulus. (pk, ct, sk): bytes. δ': worst-case (or perfect) correctness error.
(Gen, Encap, Decap): K cycles of optimized C or AVX2 implementations.

References

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

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

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

[4] Andris Ambainis, Ansis Rosmanis, and Dominique Unruh. Quantum attacks on classical proof systems: The hardness of quantum rewinding. In *55th Annual Symposium on Foundations of Computer Science*, pages 474–483, Philadelphia, PA, USA, October 18–21, 2014. IEEE Computer Society Press.

[5] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In Robert Krauthgamer, editor, *27th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 10–24, Arlington, VA, USA, January 10–12, 2016. ACM-SIAM.

[6] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426, St. Petersburg, Russia, May 28 – June 1, 2006. Springer Berlin Heidelberg, Germany.

[7] Daniel J. Bernstein and Edoardo Persichetti. Towards KEM unification. Cryptology ePrint Archive, Report 2018/526, 2018.

[8] 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, Cham, Switzerland.

[9] André Chailloux and Johanna Loyer. Lattice sieving via quantum random walks. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021, Part IV*, volume 13093 of *Lecture Notes in Computer Science*, pages 63–91, Singapore, December 6–10, 2021. Springer, Cham, Switzerland.

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

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

[12] 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, Cham, Switzerland.

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

[14] 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 Berlin Heidelberg, Germany.

[15] 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, Cham, Switzerland.

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

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

[18] Chenar Abdulla Hassan and Oğuz Yayla. Radix-3 NTT-based polynomial multiplication for lattice-based cryptography. Cryptology ePrint Archive, Report 2022/726, 2022.

[19] 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, June 1998.

[20] 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, Cham, Switzerland.

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

[22] 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 Berlin Heidelberg, Germany.

[23] 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.

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

[25] Andreas Hülsing, Joost Rijneveld, and Fang Song. Mitigating multi-target attacks in hash-based signatures. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016: 19th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 9614 of *Lecture Notes in Computer Science*, pages 387–416, Taipei, Taiwan, March 6–9, 2016. Springer Berlin Heidelberg, Germany.

[26] Haodong Jiang, Zhenfeng Zhang, Long Chen, Hong Wang, and Zhi Ma. IND-CCA-secure key encapsulation mechanism in the quantum random oracle model, revisited. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 96–125, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Cham, Switzerland.

[27] Eunmin Lee, Joohee Lee, and Yuntao Wang. Improved meet-LWE attack via ternary trees. Cryptology ePrint Archive, Report 2024/824, 2024.

[28] Joohee Lee, Minju Lee, Hansol Ryu, and Jaehui Park. A novel CCA attack for NTRU+ KEM. Cryptology ePrint Archive, Report 2023/1188, 2023.

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

[30] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23, French Riviera, May 30 – June 3, 2010. Springer Berlin Heidelberg, Germany.

[31] Vadim Lyubashevsky and Gregor Seiler. NTTRU: Truly fast NTRU using NTT. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(3):180–201, 2019.

[32] Alexander May. How to meet ternary LWE keys. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part II*, volume 12826 of *Lecture Notes in Computer Science*, pages 701–731, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland.

[33] 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, Cham, Switzerland.

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

[35] Jiang Zhang, Dengguo Feng, and Di Yan. NEV: Faster and smaller NTRU encryption using vector decoding. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023, Part VII*, volume 14444 of *Lecture Notes in Computer Science*, pages 157–189, Guangzhou, China, December 4–8, 2023. Springer, Singapore, Singapore.

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

A Factoring the Trinomial

For a better understanding of applying the NTT, we describe the recursive factorization of the polynomial $x^{1152} - x^{576} + 1$ in $\mathbb{Z}_q[x]$. Considering the initial Radix-2 NTT layer, the polynomial is factored as

$$x^{1152} - x^{576} + 1 = (x^{576} - \zeta^{144})(x^{576} - \zeta^{720}),$$

where ζ is a primitive $\ell = 864$-th root of unity, and $\zeta^{\ell/6} = \zeta^{144}$ is a primitive sixth root of unity modulo q.

Considering the Radix-3 NTT layer, these factors are further decomposed into lower-degree polynomials. For instance, the factorization of $x^{576} - \zeta^{144}$ yields

$$\begin{aligned} x^{576} - \zeta^{144} &= (x^{192} - \zeta^{48})(x^{192} - \zeta^{48}\omega)(x^{192} - \zeta^{48}\omega^2) \\ &= (x^{192} - \zeta^{48})(x^{192} - \zeta^{336})(x^{192} - \zeta^{624}), \end{aligned}$$

where $\omega = \zeta^{288} = \zeta^{\ell/3}$ is a primitive third root of unity modulo q.

Considering the subsequent Radix-2 NTT layers, these factors are further bisected. For example,

$$x^{192} - \zeta^{48} = (x^{96} - \zeta^{24})(x^{96} + \zeta^{24}) = (x^{96} - \zeta^{24})(x^{96} - \zeta^{456}),$$

where ζ^{432} is a primitive second root ($\zeta^{\ell/2} \equiv -1 \pmod{q}$). This recursive factorization continues until the polynomials reach the terminal degree $d = 4$, defining the final component rings of the NTT domain.

B Radix-3 NTT layer

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

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

Naively, these equations might appear to require $2n$ multiplications and $2n$ additions, relying on six predefined values: α, α^2, β, β^2, γ, and γ^2. Nevertheless, following the techniques in [18], this cost can be reduced to n multiplications, n additions, and $4n/3$ subtractions, while using only three predefined values α, α^2, and ω as shown in Algorithm 14.

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

Algorithm 14: Radix-3 NTT layer

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

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

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

Naively, these equations might appear to require $2n$ multiplications and $2n$ additions, relying on six predefined values: α^{-1}, α^{-2}, β^{-1}, β^{-2}, γ^{-1}, and γ^{-2}. Nevertheless, by following the techniques in [18], we can significantly reduce this computational load to n multiplications, n additions, and $4n/3$ subtractions, while using only three predefined values: α^{-1}, α^{-2}, and ω, as described in Algorithm 15.

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

Algorithm 15: Radix-3 Inverse NTT layer

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

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

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