NTRU+

Algorithm Specifications And Supporting Documentation

(version 2.2.1)

Changelog

Version 1.1

In terms of the specification, the primary changes are as follows:

1. Modifying the Inv function of SOTP to defend against Lee's attacks
   In June 2023, Joohee Lee announced a chosen-ciphertext attack against $\mathsf{NTRU+KEM}$, which occurred due to the absence of the bit-checking process in the Inv function. Version 1.1 of $\mathsf{NTRU+KEM}$ addressed this issue by adding the bit-checking process and providing a more clarified definition of SOTP.

2. Modifying the Encap and Decap algorithms to consider multi-target attacks
   In Version 1.0, $\mathsf{NTRU+KEM}$ did not consider the multi-target attacks. To achieve the multi-target security in Version 1.1, we have adopted the well-known technique to add the hash value $\mathsf{F}(pk)$ of the public key pk into the hashing such as $(r, K) = \mathsf{H}(m, \mathsf{F}(pk))$ when applying the Fujisaki-Okamoto transform. Accordingly, we also have changed the secret key into $sk = (f, h^{-1}, \mathsf{F}(pk))$, which increases the secret key size by 32 bytes in all sets of parameters.

3. Modifying the NTT structure for $\mathsf{NTRU+KEM576}$ and $\mathsf{NTRU+KEM1152}$
   The ring structures for $\mathsf{NTRU+KEM576}$ and $\mathsf{NTRU+KEM1152}$ can be factored all the way down to $\prod_{i=0}^{n} \mathbb{Z}q[x]/\langle x - \zeta_i \rangle$. When applying NTT for $\prod_{i=0}^{n} \mathbb{Z}_q[x]/\langle x - \zeta_i \rangle$, n modular inversions are required during key generation to compute f^{-1}. To reduce the number of modular inversions by $n/2$, we have factored the rings into $\prod_{i=0}^{n/2} \mathbb{Z}_q[x]/\langle x^2 - \zeta_i \rangle$ in Version 1.0. However, in Version 1.1, we have further reduced the n modular inversions by $n/3$ by applying NTT for $\prod_{i=0}^{n/3} \mathbb{Z}_q[x]/\langle x^3 - \zeta_i \rangle$.

4. Clarification regarding randomness-polynomial sampling from binary bit-strings
   In Encap of Version 1.0, the coefficients of the randomness-polynomial $\mathbf{r}$ were described as if they were composed of bit strings. In Version 1.1, we clarified this mistake by defining $\mathbf{r} := \mathsf{CBD}_1(r)$.

Next, in terms of our implementation, the changes are as follows:

1. Modifying the Inv algorithm of SOTP to defend against Lee's attacks

2. Modifying the Encap and Decap algorithms to consider multi-target attacks

3. Modifying the NTT structure for $\mathsf{NTRU+KEM576}$ and $\mathsf{NTRU+KEM1152}$
   This allows for improving the key generation timings and reducing the size of pre-computation tables.

4. Modifying the Radix-3 NTT implementation
   Implementing Radix-3 NTT naively requires $2n$ multiplications per layer. In Version 1.0, we reduced this to $4n/3$ multiplications, but by adapting the recent result (https://eprint.iacr.org/2022/726.pdf), we can further reduce the number of multiplications from $4n/3$ to n.

5. Removing the dependencies on OpenSSL and AVX in Reference implementation
   The initial implementation of $\mathsf{NTRU+KEM}$ was mainly based on the code of NTTRU (that are found in 'https://github.com/gregorseiler/NTTRU'), which uses AVX assembly codes for the implementation of AES-256-CTR. Also, the initial implementation used the 'rng.c' provided by NIST, which also has OpenSSL dependencies. To remove those dependencies, we have referred to the code of CRYSTALS-Kyber (https://github.com/pq-crystals/kyber).

6. Reducing the size of the pre-computation table in Reference implementation
   In Version 1.0, performing NTT and Inverse NTT operations required two separate pre-computation tables. The revised implementation have changed to use a single table by adapting the code of CRYSTALS-Kyber, along with our additional manipulation to support the Radix-3 NTT layer.

Version 2.0

In terms of the specification, the primary changes are as follows:

1. In Version 1.1, we adapted countermeasures against the attack proposed by Joohee Lee. However, some ambiguity remained in the proof of Lemma 4.3. In Version 2.0, we addressed these issues by making the following modifications:
   (a) Redefined the definition of injectivity and rigidity of PKE in Section 2.2, along with revising the analysis of injectivity and rigidity for $\mathsf{GenNTRU}[\psi_1^n]$ in Section 6.1.4.
   (b) Redefined the definition of rigidity for SOTP in Section 3.1, and revised the analysis of rigidity for the instantiation of SOTP used in $\mathsf{CPA\text{-}NTRU+}$ in Section 6.2.1.
   (c) Slightly modified the definition of the ACWC_2 transformation in Section 3.2.
   (d) Updated Theorems 3.5 and 3.6 to reflect the redefined definition of injectivity.
   (e) Modified Section 4.2 (and Lemma 4.3) to address the comments made by Joohee Lee.

2. We propose a new NTRU-based $\mathsf{IND\text{-}CCA}$ secure PKE called '$\mathsf{NTRU+PKE}$'.
   $\mathsf{NTRU+PKE}$ is constructed by applying a variant of $\mathsf{FO}^{\perp}_{\mathsf{PKE}}$, called $\overline{\mathsf{FO}}^{\perp}_{\mathsf{KEM}}$, to $\mathsf{CPA\text{-}NTRU+}$. Here, $\mathsf{FO}^{\perp}_{\mathsf{PKE}}$ refers to the transformation proposed in [17], which converts $\mathsf{IND\text{-}CPA}$ secure PKE into $\mathsf{IND\text{-}CCA}$ secure PKE. To avoid confusion, we rename the previous $\mathsf{NTRU+}$ to $\mathsf{NTRU+KEM}$.

3. To provide the theoretical background for $\mathsf{NTRU+PKE}$, we include the following:
   (a) We analyze the security of $\mathsf{FO}^{\perp}_{\mathsf{PKE}}$ in ROM and QROM, by taking into account correctness errors that were not clearly addressed in the analysis of [17]. It can be found in Theorem 5.1 and 5.2.
   (b) We analyze the equivalence between $\mathsf{FO}^{\perp}_{\mathsf{PKE}}$ and $\overline{\mathsf{FO}}^{\perp}_{\mathsf{PKE}}$ in Lemma 5.3, similar to Lemma 4.3.

4. We correct some errors in Appendix B, which is necessary for reusing the predefined table in order to compute the Inverse NTT.

Version 2.1

Following Professor D. J. Bernstein's comments on the implementation of NTRU+ (https://groups.google.com/g/kpqc-bulletin/c/exrFyRPhFJ8), we investigated and identified errors in the AVX2 implementation of NTRU+. The following changes were made:

1. **Changes in** $\mathsf{NTRU+}\{\mathsf{KEM},\mathsf{PKE}\}\mathsf{864}$
   Memory access violations were discovered and corrected in the 'poly_add', 'poly_sub', and 'poly_triple' functions.

2. **Changes in** $\mathsf{NTRU+}\{\mathsf{KEM},\mathsf{PKE}\}\mathsf{1152}$
   An error in the 'poly_sotp' function was found, where 'vmovdqa' was applied to non-aligned memory. This was corrected by replacing 'vmovdqa' with 'vmovdqu'.

3. **Other Adjustments**
   To address warnings regarding the End of File (EOF) encountered during clang compilation, necessary adjustments were made throughout the codebase.

Version 2.2

The primary changes to the specification are as follows:

1. **Definition of Hash Function**
   Following comments by Dr. Seongkwang Kim indicating that AES256CTR is not suitable for instantiating the random oracle model (https://groups.google.com/g/kpqc-bulletin/c/C-mtPvzo3QA/m/vuQ0sis6AgAJ), we revised how we instantiate the hash functions G, HKEM, and HPKE in the specification by replacing AES256CTR with SHAKE256.

2. **Definition of SOTP**
   To reduce confusion in the definition of SOTP, we changed the notation. Previously, the function for encoding messages was named SOTP, and the function for recovering messages was named Inv. However, the encoding function has now been renamed to Encode. SOTP is defined as including both functions, Encode and Inv, and is expressed as $\mathsf{SOTP} = (\mathsf{Encode}, \mathsf{Inv})$.

3. **Changes in the Key Generation**
   In the key generation process, f and g were originally sampled together from the same random seed until both were invertible. To enhance efficiency, we separated the sampling of the invertible polynomials f and g: first, we sample f until it is invertible, then we sample g until it is invertible. This sequential sampling minimizes unnecessary rejections. Additionally, f and g are now generated using separate random seeds.

4. **Changes in the NTT Structures**
   To improve the efficiency of key generation, we reduced the number of modular inverse operations, which are the most computationally intensive part of the key generation process, by modifying the way the NTT is applied. As mentioned in the changelog of Version 1.1, the ring structures of $\mathsf{NTRU+}\{\mathsf{KEM}, \mathsf{PKE}\}\{576, 1152\}$ can be factored as $\prod_{i=0}^{n-1} \mathbb{Z}_q[x]/\langle x - \zeta_i \rangle$. Additionally, the ring structure of $\mathsf{NTRU+}\{\mathsf{KEM}, \mathsf{PKE}\}768$ can be factored as $\prod_{i=0}^{n/2-1} \mathbb{Z}_q[x]/\langle x^2 - \zeta_i \rangle$. To further reduce the number of modular inversions for the parameter sets $\mathsf{NTRU+}\{\mathsf{KEM}, \mathsf{PKE}\}\{576, 768, 1152\}$, we modified the application of the NTT to factor the ring as $\prod_{i=0}^{n/4} \mathbb{Z}_q[x]/\langle x^4 - \zeta_i \rangle$.

5. **Spreadness of** $\mathsf{PKE}' = \mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$
   We re-analyzed the spreadness of the $\mathsf{PKE}' = \mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$ in Section 3.2. In PKE', $\mathsf{SOTP} = (\mathsf{Encode}, \mathsf{Inv})$ is used as $\mathsf{Encode}(m, \mathsf{G}(r))$ with a hash function G. In the underlying PKE, a ciphertext is generated as $c = \mathsf{Enc}(pk, \mathsf{Encode}(m, \mathsf{G}(r)); r)$. To analyze γ-spreadness, the message m must be fixed for each randomness r (honestly chosen from $\mathcal{R}$). However, when using SOTP, the encoded message $\mathsf{Encode}(m, \mathsf{G}(r))$ also changes as r changes. In the previous analysis, we did not consider this point, so we revise the proof of γ-spreadness.

6. **Parameter Adjustment in** $\mathsf{NTRU+PKE}$
   To conservatively set the parameters, we modified the maximum message length supported by $\mathsf{NTRU+PKE}$ to 32 bytes for all parameter sets $\mathsf{NTRU+PKE}\{576, 768, 864, 1152\}$.

7. **Revisions to the Definition of PKE**
   In response to comments by Prof. Sven Schäge through private email communication, we updated the definition of PKE, rigidity of PKE. Also, we included the definition of weakly spreadness in [15], which is weaker version of spreadness defined in [21]. Based on the changed definitions, we revised the lemmas 4.3 and 5.3.

8. **Revisions to Lemma**
   In response to comments by Prof. Joohee Lee presented at the 8th KpqC workshop, we updated the proofs of Lemma 4.3. Specifically, Prof. Joohee Lee noted that the precondition for applying the rigidity of PKE in Lemma 4.3 was not fully satisfied.

Changes to the implementation are as follows:

1. **Source Code for the Hash Functions**
   We replaced the source code of SHA256 and additionally used the source code for SHAKE256, adapted from https://github.com/kpqc-cryptocraft/KpqClean_ver2.

2. **Changes in the Key Generation**
   To improve the efficiency of key generation, we adopted an early abort approach when checking the invertibility of a polynomial. When checking the invertibility of a polynomial, we need to verify that it is invertible in all rings $\mathbb{Z}_q[x]/\langle x^d - \zeta_i \rangle$. For efficiency, we abort as soon as we find the first ring in which the polynomial is not invertible. One may wonder whether this type of early abort could leak information about the randomness used to sample the polynomial. However, since the rejected polynomial is not reused as part of the secret key, we believe this approach is secure, provided that the underlying `randombytes` function is forward-secure.

3. **Changes in Ring Multiplication and Inversion**
   We implemented ring operations in $\prod_{i=0}^{n/4-1} \mathbb{Z}_q[x]/\langle x^4 - w \rangle$, which are required to realize the newly proposed NTT structure in Version 2.2 for the parameter sets $\mathsf{NTRU+}\{\mathsf{KEM}, \mathsf{PKE}\}\{576, 768, 1152\}$. We referred to the ideas presented in [38] to implement the inversion in $\mathbb{Z}_q[x]/\langle x^4 - w \rangle$.

   Additionally, to improve the efficiency of key generation, we adopted lazy Montgomery reduction [34] in the implementation of ring operations (multiplication and inversion) in $\prod_{i=0}^{n/d-1} \mathbb{Z}_q[x]/\langle x^d - w \rangle$ for $d = 3, 4$. During multiplication and inversion, we need to compute the sum of several products of polynomial coefficients. To reduce the number of Montgomery and Barrett reductions, we applied Montgomery reduction after accumulating the 32-bit data.

   Lastly, to enhance the efficiency of the modular inversion using Fermat's Little Theorem, $a^{-1} \equiv a^{q-2} \pmod{q}$, we leveraged the binary structure of $q - 2 = 3455 = 110101111111_{(2)}$, inspired by the fast modular inversion in Curve25519 [5]. While the standard square-and-multiply approach requires 20 `fqmul` operations, we reduced this number to 16 by reusing intermediate values.

Version 2.2.1

In this version, we corrected typographical errors in Table 5, specifically regarding the number of Radix-2 NTT layers required. Additionally, the following changes were made in the implementation:

1. **Adjustment in Lazy Barrett Reduction**
   While the code functioned correctly, there was an issue with the placement of Barrett reduction in the

inverse NTT implementation in worst-case scenarios. To resolve this, we adjusted the placement of Barrett reduction across all relevant parts of the code. In the optimized implementation, we further minimized the use of Barrett reduction by unrolling loops in the inverse NTT. Note that, prior to Version 2.2.1, the reference and optimized implementations were identical. However, starting with Version 2.2.1, we have decided to manage them separately.

2. **Replacing Barrett Reduction with Montgomery Reduction**
   In both the reference and optimized implementations, Barrett reduction was applied at the end of the NTT layers. In the optimized implementation, we replaced the Barrett reduction with Montgomery reduction in the final NTT layer to improve performance.

3. **Updates to consts.c for AVX2**
   We revised the values in the 'consts.c' file for $\mathsf{NTRU}{+}\{\mathsf{KEM}, \mathsf{PKE}\}864$, which were previously set in the range 0 to $q-1$. These values have now been updated to the range $-(q-1)/2$ to $(q-1)/2$.

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

Jonghyun Kim† Jong Hwan Park‡

October 10, 2024

Abstract

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

In this work, we propose two new NTRU-based primitives: a key encapsulation mechanism (KEM) called '$\mathsf{NTRU+KEM}$' and a public key encryption (PKE) called '$\mathsf{NTRU+PKE}$'. These new primitives overcome nearly all the above-mentioned drawbacks. They are constructed based on two new generic transformations: ACWC_2 and $\overline{\mathsf{FO}}^{\perp}$. ACWC_2 is used to easily achieve worst-case correctness error, and $\overline{\mathsf{FO}}^{\perp}$ (a variant of the Fujisaki-Okamoto transform) is used to achieve chosen-ciphertext security without performing re-encryption. Both ACWC_2 and $\overline{\mathsf{FO}}^{\perp}$ are defined using a randomness-recovery algorithm (that is unique to NTRU) and a novel message-encoding method. In particular, our encoding method, called the semi-generalized one-time pad (SOTP), allows us to use a message sampled from a natural bit-string space with an arbitrary distribution. We provide four parameter sets for $\mathsf{NTRU}+\{\mathsf{KEM},\mathsf{PKE}\}$ 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 [20] was introduced in 1998 by Hoffstein, Pipher, and Silverman as the first practical public key encryption scheme using lattices over polynomial rings. The hardness of NTRU is crucially based on the NTRU problem [20], which has withstood significant cryptanalytic attacks over the past few decades. This longer history, compared to other lattice-based problems (such as ring/module-LWE), has been considered an important factor in selecting NTRU as a finalist in the NIST PQC standardization process. While the finalist NTRU [10] has not been chosen by NIST as one of the first four quantum-resistant cryptographic algorithms, it still has several distinct advantages over other lattice-based competitive schemes such as KYBER [33] and Saber [13]. Specifically, the advantages of NTRU include: (1) the compact structure of a ciphertext consisting of a single polynomial, and (2) (possibly) faster encryption and decryption without the need to sample the coefficients of a public key polynomial.

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

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

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

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

In the context of chosen-ciphertext attacks, NTRU, like other ordinary public key encryption schemes, must guarantee an extremely negligible worst-case correctness error. This is essential to prevent the leakage of information about the private key through adversarial decryption queries, such as attacks against lattice-based encryption schemes [12, 23]. Roughly speaking, the worst-case correctness error refers to the probability that decryption fails for any ciphertext that can be generated with all possible messages and randomness in their respective spaces. The worst-case correctness error considers that an adversary, $\mathcal{A}$, can *maliciously* choose messages and randomness without sampling normally according to their original distributions (if possible). In the case of NTRU, the failure to decrypt a specific ciphertext $\mathbf{c} = \mathbf{hr} + \mathbf{m}$ provides $\mathcal{A}$ with the information that one of the coefficients of $p(\mathbf{gr} + \mathbf{f}'\mathbf{m}) + \mathbf{m}$ is larger than or equal to $q/2$. If $\mathcal{A}$ has control over the choice of $\mathbf{r}$ and $\mathbf{m}$, even one such decryption failure may open a path to associated decryption queries to obtain more information about secret polynomials $\mathbf{g}$ and $\mathbf{f}$.

When designing NTRU, two approaches can be used to achieve worst-case correctness error. One approach is to draw $\mathbf{m}$ and $\mathbf{r}$ directly from ψ, while setting the modulus q to be relatively large. The larger q guarantees a high probability that all coefficients of $p(\mathbf{gr}+\mathbf{f}'\mathbf{m})+\mathbf{m}$ are less than $q/2$ for *nearly all possible* $\mathbf{m}$ and $\mathbf{r}$ in their spaces, although it causes inefficiency in terms of public key and ciphertext sizes. Indeed, this approach has been used by the third-round finalist NTRU [10], wherein all recommended parameters provide *perfect* correctness error (i.e., the worst-case correctness error becomes zero for all possible $\mathbf{m}$ and $\mathbf{r}$). By contrast, the other approach [16] is to use an encoding method by which a message $m \in \mathcal{M}'$ is used as a randomness to sample $\mathbf{m}$ and $\mathbf{r}$ according to ψ. Under the Fujisaki-Okamoto (FO) transform [18], decrypting a ciphertext $\mathbf{c}$ requires re-encrypting m by following the same sampling process as encryption. Thus, an ill-formed ciphertext that does not follow the sampling rule will always fail to be successfully decrypted, implying that $\mathbf{m}$ and $\mathbf{r}$ should be *honestly* sampled by $\mathcal{A}$ according to ψ. Consequently, by disallowing $\mathcal{A}$ to have control over $\mathbf{m}$ and $\mathbf{r}$, the NTRU with an encoding method has a worst-case correctness error that is close to an average-case error.

Based on the aforementioned observation, [16] proposed generic (average-case to worst-case) transformations[2] that make the average-case correctness error of an underlying scheme nearly close to the worst-case error of a transformed scheme. One of their transformations (denoted by ACWC) is based on an encoding method called the generalized one-time pad (denoted by GOTP). Roughly speaking, GOTP works as follows: a message $m \in \mathcal{M}'$ is first used to sample $\mathbf{r}$ and $\mathbf{m}_1$ according to ψ, and $\mathbf{m}_2 = \mathsf{GOTP}(m, \mathsf{G}(\mathbf{m}_1))$ using a hash function G, and then $\mathbf{m}$ is constructed as $\mathbf{m}_1||\mathbf{m}_2$. If the GOTP acts as a sampling function

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

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

Scheme	NTRU[10]	NTRU-B [16]	NTRU+KEM
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 + $\mathsf{FO}^{\perp}_{\mathsf{KEM}}$	ACWC_2 + $\overline{\mathsf{FO}}^{\perp}_{\mathsf{KEM}}$
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 [32] described in the NTRU finalist.

Table 1: Comparison to previous NTRU constructions

wherein the output follows ψ, $\mathbf{m}$ and $\mathbf{r}$ are created from m following ψ, which can be verified in decryption using the FO transform. Specifically, for two inputs m and $\mathsf{G}(\mathbf{m}_1)$ that are sampled from $\{-1,0,1\}^\lambda$ for some integer λ, $\mathbf{m}_2 \in \{-1,0,1\}^\lambda$ is computed by doing the component-wise exclusive-or modulo 3 of two ternary strings m and $\mathsf{G}(\mathbf{m}_1)$. Thus, if $\mathsf{G}(\mathbf{m}_1)$ follows a uniformly random distribution ψ over $\{-1,0,1\}^\lambda$, m is hidden from $\mathbf{m}_2$ because of the one-time pad property.

However, an ACWC based on the GOTP has two disadvantages in terms of security reduction and message distribution. First, [16] showed that ACWC converts a one-way CPA (OW-CPA) secure underlying scheme into a transformed one that is still OW-CPA secure, besides the fact that their security reduction is loose[3] by causing a security loss factor of q_{G}, the number of random oracle queries. Second, ACWC forces even a message $m \in \mathcal{M}'$ to follow a specific distribution because their security analysis of ACWC requires GOTP to have the additional randomness-hiding property, meaning that $\mathsf{G}(\mathbf{m}_1)$ should also be hidden from the output $\mathbf{m}_2$. Indeed, the NTRU instantiation from ACWC, called 'NTRU-B' [16], requires that m should be chosen uniformly at random from $\mathcal{M}' = \{-1,0,1\}^\lambda$. Notably, it is difficult to generate exactly uniformly random numbers from $\{-1,0,1\}$ in constant time due to rejection sampling. Therefore, it was an open problem [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+KEM' 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+KEM achieves a worst-case correctness error close to the average-case error of the underlying NTRU. Additionally, NTRU+KEM 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+KEM relies on a novel FO-equivalent transform without re-encryption, which makes the decryption algorithm of NTRU+KEM faster than in the ordinary FO transform. In terms of efficiency, we use the idea from [30] to

[3][16] introduced a new security notion, q-OW-CPA, which states that an adversary outputs a set Q with a maximum size of q and wins if the correct message corresponding to a challenged ciphertext belongs to Q. We believe that q-OW-CPA causes a security loss of q.

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

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

Table 2: Comparison to previous ACWC transformations

apply the Number Theoretic Transform (NTT) to NTRU+KEM and therefore instantiate NTRU+KEM over a ring $R_q = \mathbb{Z}_q[x]/\langle f(x)\rangle$, where $f(x) = x^n - x^{n/2} + 1$ is a cyclotomic trinomial. By selecting appropriate (n, q) and ψ, we suggest four parameter sets for NTRU+KEM and provide the implementation results for NTRU+KEM in each parameter set. Table 1 lists the main differences between the previous NTRU constructions [10, 16] and NTRU+KEM. In the following section, we describe our technique, focusing on these differences.

ACWC$_2$ Transformation with Tight Reduction. ACWC$_2$ is a new generic transformation that allows for the aforementioned average-case to worst-case correctness error conversion. However, to apply ACWC$_2$, the underlying scheme is required to have injectivity, randomness-recoverable (RR), and message-recoverable (MR) properties, which are typical of NTRU.[4] Additionally, ACWC$_2$ involves an encoding method called semi-generalized one-time pad (denoted by SOTP). In contrast to the GOTP in [16], $\mathsf{SOTP} = (\mathsf{Encode}, \mathsf{Inv})$ works in a generic sense as follows: first, a message $m \in \mathcal{M}'$ is used to sample $\mathbf{r}$ based on ψ, and then $\mathbf{m} = \mathsf{Encode}(m, \mathsf{G}(\mathbf{r}))$ 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 a normal decryption algorithm, and using $\mathbf{m}$, $\mathbf{r}$ is also recovered by a randomness-recovery algorithm. Finally, an inverse of Encode called Inv with $\mathsf{G}(\mathbf{r})$ and $\mathbf{m}$ yields m.

The MR property of an underlying scheme allows us to show that, without causing any security loss, ACWC$_2$ transforms an OW-CPA secure scheme into a chosen-plaintext (IND-CPA) secure scheme. The proof idea is simple: unless an IND-CPA adversary $\mathcal{A}$ queries $\mathbf{r}$ to a (classical) random oracle G, $\mathcal{A}$ does not obtain any information on m_b (that $\mathcal{A}$ submits) for $b \in \{0, 1\}$ because of the basic message-hiding property of SOTP. However, whenever $\mathcal{A}$ queries $\mathbf{r}_i$ to G for $i = 1, \cdots, q_{\mathsf{G}}$, a reductionist can check whether each $\mathbf{r}_i$ is the randomness used for its OW-CPA challenge ciphertext using a message-recovery algorithm. Therefore, the reductionist can find the exact $\mathbf{r}_i$ among the q_{G} number of queries if $\mathcal{A}$ queries $\mathbf{r}_i$ (with respect to its IND-CPA challenge ciphertext) to G. In this security analysis, it is sufficient for SOTP to have the message-hiding property, which makes SOTP simpler than GOTP because GOTP must have both message-hiding and randomness-hiding properties.

Table 2 presents a detailed comparison between previous ACWC transformations and our new ACWC$_2$. Unlike the previous ACWC based on GOTP, [16] proposed another generic ACWC transformation (denoted by ACWC$_0$) without using any message-encoding method. In ACWC$_0$, a (bit-string) message m is encrypted with a ciphertext $\mathbf{c} = (\mathsf{Enc}(pk, \mathbf{m}; \mathbf{r}), \mathsf{F}(\mathbf{m}) \oplus m)$ using a hash function F, which causes the ciphertext expansion of $\mathsf{F}(\mathbf{m}) \oplus m$, whereas such a ciphertext redundancy does not occur in ACWC and ACWC$_2$. Like

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

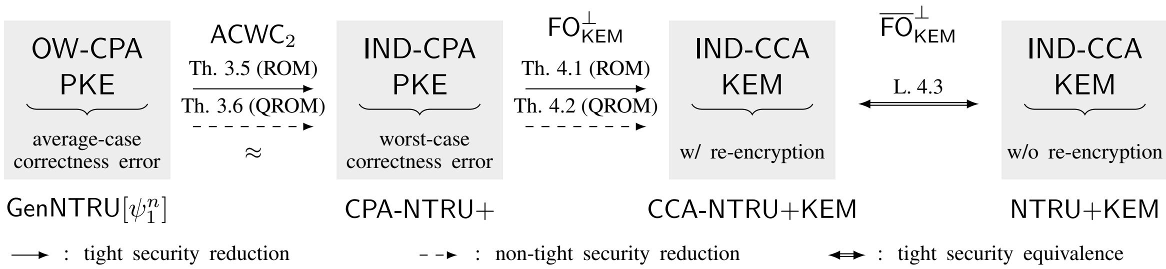

Figure 1: Overview of security reductions for KEM

ACWC$_2$, ACWC$_0$ transforms any OW-CPA secure scheme into an IND-CPA secure one, but their security reduction is not tight as in ACWC. ACWC$_0$ and ACWC$_2$ requires no specific message distribution, whereas ACWC requires $m \in \mathcal{M}'$ to be sampled according to a uniformly random distribution from $\mathcal{M}'$. ACWC$_0$ and ACWC work for any OW-CPA secure scheme, but ACWC$_2$ works for any OW-CPA secure scheme satisfying injectivity, MR, and RR properties.

FO-Equivalent Transform without Re-encryption. To achieve chosen-ciphertext (IND-CCA) security, we apply the generic transform $\mathsf{FO}^{\perp}_{\mathsf{KEM}}$ to the ACWC$_2$-derived scheme, which is IND-CPA secure. As with other FO-transformed schemes, the resulting scheme from ACWC$_2$ and $\mathsf{FO}^{\perp}_{\mathsf{KEM}}$ 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 $\mathsf{FO}^{\perp}_{\mathsf{KEM}}$. Instead, the more advanced transform (denoted by $\overline{\mathsf{FO}}^{\perp}_{\mathsf{KEM}}$) 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 $\overline{\mathsf{FO}}^{\perp}_{\mathsf{KEM}}$ is functionally identical to $\mathsf{FO}^{\perp}_{\mathsf{KEM}}$ by proving that the randomness-checking process in $\overline{\mathsf{FO}}^{\perp}_{\mathsf{KEM}}$ is equivalent to the re-encryption process $\mathsf{FO}^{\perp}_{\mathsf{KEM}}$. The equivalence proof relies mainly on the injectivity [7, 21] and rigidity [6] properties of the underlying schemes. As a result, although the RR property seems to incur some additional decryption cost, it ends up making the decryption algorithm faster than the original FO transform. Figure 1 presents an overview of security reductions from OW-CPA to IND-CCA.

Simple SOTP Instantiation with More Convenient Sampling Distributions. As mentioned previously, ACWC$_2$ is based on an efficient construction of $\mathsf{SOTP} = (\mathsf{Encode}, \mathsf{Inv})$ 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 $\mathsf{G} : \{0,1\}^* \rightarrow \{0,1\}^{2n}$, m is chosen from the message space $\mathcal{M}' = \{0,1\}^n$ for an arbitrary distribution (with high min-entropy) and $\mathsf{G}(\mathbf{r}) = y_1 || y_2 \in \{0,1\}^n \times \{0,1\}^n$. SOTP then computes $\tilde{m} = (m \oplus y_1) - y_2$ by bitwise subtraction and assigns each subtraction value of $\tilde{m}$ to the coefficient of $\mathbf{m}$. By the one-time pad property, it is easily shown that $m \oplus y_1$ becomes uniformly random in $\{0,1\}^n$ (and thus message-hiding) and each coefficient of $\mathbf{m}$ follows ψ_1. Since $\mathbf{r}$ is also sampled from a hash value of m according to ψ_1, all sampling distributions in NTRU+KEM 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. $\mathsf{NTRU{+}KEM}$ 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 $n = 2^i 3^j$. [30] showed that, with appropriate parameterization of n and q, such a ring can also provide NTT operation essentially as fast as that over a ring $R_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$. Moreover, because the choice of a cyclotomic trinomial is moderate, it provides more flexibility to satisfy a certain level of security. Based on these results, we choose four parameter sets for $\mathsf{NTRU{+}KEM}$, where the polynomial degree n of $f(x) = x^n - x^{n/2} + 1$ is set to be 576, 768, 864, and 1152, and the modulus q is 3457 for all cases. Table 7 lists the comparison results between finalist NTRU [10], KYBER, KYBER-90s [33], and NTRU+ in terms of security and efficiency. To estimate the concrete security level of $\mathsf{NTRU{+}KEM}$, 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 $\mathsf{NTRU{+}KEM}$ outperforms NTRU at a similar security level.

1.2 Related Works

The first-round NTRUEncrypt [39] submission to the NIST PQC standardization process was an NTRU-based encryption scheme with the NAEP padding method [24]. Roughly speaking, NAEP is similar to our SOTP, but the difference is that it does not completely encode $\mathbf{m}$ to prevent an adversary $\mathcal{A}$ from choosing $\mathbf{m}$ maliciously. This is due to the fact that $\mathbf{m} := \mathsf{NAEP}(m, \mathsf{G}(\mathbf{hr}))$ is generated by subtracting two n-bit strings m and $\mathsf{G}(\mathbf{hr})$ from each other, i.e., $m - \mathsf{G}(\mathbf{hr})$ by bitwise subtraction, and then assigning them to the coefficients of $\mathbf{m}$. Since m can be maliciously chosen by $\mathcal{A}$ in NTRUEncrypt, $\mathbf{m}$ can also be maliciously chosen, regardless of $\mathsf{G}(\mathbf{hr})$.

The finalist NTRU [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 [32] conversion, which also avoids re-encryption during decapsulation. Similar to $\mathsf{NTRU{+}KEM}$, the SXY variant requires the rigidity [6] 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. [30] 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 requires an expansion of the ciphertext size. The security of ACWC_0 and ACWC was analyzed in both the classical and quantum random oracle models [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 $\mathbf{a}$ is 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 [21] 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

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

Unlike the definition of injectivity in [7, 21], we define the injectivity in a computationally-secure sense.

$\underline{\textsc{Game}\ \mathsf{INJ}}$

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

Figure 2: GAME INJ for PKE

Spreadness. PKE is γ-*spread* [21] 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 is an algorithm RRec such that for all $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$, $m \in \mathcal{M}$, and $r \in \mathcal{R}$,

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

where the probability is taken over $c \leftarrow \mathsf{Enc}(pk, m; r)$ and $\mathsf{Pre}^m(pk, c)$ defined as $\{m \in \mathcal{M} | \ \exists r \in \mathcal{R} : \mathsf{Enc}(pk, m; r) = c\}$.

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

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

where the probability is calculated over $c \leftarrow \mathsf{Enc}(pk, m; r)$ and $\mathsf{Pre}^r(pk, c)$ defined as $\{r \in \mathcal{R} | \exists\, m \in \mathcal{M} : \mathsf{Enc}(pk, m; r) = c\}$.

Rigidity. PKE is said to be *rigid* if, for all key pairs $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$ and for any ciphertext $c \in \mathcal{C}$, the following holds:

$$\text{If } m' = \mathsf{Dec}(sk, c) \in \mathcal{M} \text{ and } r' = \mathsf{RRec}(pk, m', c) \in \mathcal{R}, \text{ then } \mathsf{Enc}(pk, m'; r') = c.$$

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

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

Definition 2.4 (IND-CCA Security of PKE)**.** Let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme with message space $\mathcal{M}$. Indistinguishability under chosen ciphertext attacks (IND-CCA) is defined via the GAME IND-CCA, as shown in Figure 3, and the advantage function of adversary $\mathcal{A}$ is

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

<u>GAME OW-CPA</u>

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

<u>GAME IND-CPA</u>

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

<u>GAME IND-CCA</u>

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

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

return $\mathsf{Dec}(sk, c)$

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

2.3 Key Encapsulation Mechanism

Definition 2.5 (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 2.6 (IND-CCA Security of KEM)**.** Let KEM = (Gen, Encap, Decap) be a key encapsulation mechanism with a key space $\mathcal{K}$. Indistinguishability under chosen-ciphertext attacks (IND-CCA) is defined via the GAME IND-CCA, as shown in Figure 4, and the advantage function of adversary $\mathcal{A}$ is as follows:

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

Game IND-CCA	$\mathsf{Decap}(c \neq c^*)$
1: $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$	1: **return** $\mathsf{Decap}(sk, c)$
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** $[\![b = b']\!]$	

Figure 4: GAME IND-CCA for KEM

2.4 Complexity Assumptions

This section outlines complexity assumptions used in NTRU+$\{\mathsf{KEM}, \mathsf{PKE}\}$. 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 2.7 (The NTRU problem [20])**.** 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 2.8 (The RLWE problem [29])**.** 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 Proof Tools for QROM

Unlike the traditional ROM, the QROM must handle outputs for superpositioned inputs, making it challenging to directly apply ROM proof techniques like adaptive programming and security proofs using hash tables [8]. This section introduces essential proof tools for QROM security analysis that circumvent these constraints: the O2H lemma [37] and the extractable random oracle simulator [15].

2.5.1 One-way to Hiding

The O2H lemma, first introduced by D. Unruh [37], serves as a important proof tool for the QROM. This lemma quantifies the advantage of a quantum adversary in distinguishing between two scenarios: one that uses random oracle outputs for specific inputs and another that uses truly random values. The fundamental idea is that the probability of an adversary successfully measuring the specific input, for which the hash function output has been replaced with a truly random value, bounds the advantage between these two scenarios. In the ROM, the corresponding concept is the difference lemma proposed by Victor Shoup [35], which similarly analyzes the differences between two games but is applicable in a classical context. This subsection outlines the variations of the O2H lemma used in the security proofs presented in this work.

Lemma 2.9 (Adaptive O2H, Lemma 14 of [36])**.** Let $\mathsf{H} : \{0,1\}^* \to \{0,1\}^n$ be a random oracle. Consider an oracle algorithm $\mathcal{A}_1$ that uses the final state of $\mathcal{A}_0$ and makes at most q_1 queries to H. Let $\mathcal{C}_1$ be an oracle algorithm that on input (j, B, x) does the following: run $\mathsf{A}_1^{\mathsf{H}}(x, B)$ until (just before) the j-th query, measure the argument of the query in the computational basis, output the measurement outcome. (When $\mathcal{A}$ makes less than j queries, $\mathcal{C}_1$ outputs $\perp \notin \{0,1\}^*$.) Let

$$\begin{aligned}
P_{\mathcal{A}}^1 &:= \Pr[b' = 1 : \mathsf{H} \leftarrow (\{0,1\}^* \to \{0,1\}^n), m \leftarrow \mathcal{A}_0^{\mathsf{H}}(), x \leftarrow \{0,1\}^\ell, \\
&\qquad\qquad\qquad\qquad b' \leftarrow \mathcal{A}_1^{\mathsf{H}}(x, \mathsf{H}(x\|m))], \\
P_{\mathcal{A}}^2 &:= \Pr[b' = 1 : \mathsf{H} \leftarrow (\{0,1\}^* \to \{0,1\}^n), m \leftarrow \mathcal{A}_0^{\mathsf{H}}(), x \leftarrow \{0,1\}^\ell, \\
&\qquad\qquad\qquad\qquad B \leftarrow \{0,1\}^n, b' \leftarrow \mathcal{A}_1^{\mathsf{H}}(x, B)], \\
P_{\mathcal{C}} &:= \Pr[x = x' \wedge m = m' : \mathsf{H} \leftarrow (\{0,1\}^* \to \{0,1\}^n), m \leftarrow \mathcal{A}_0^{\mathsf{H}}(), x \leftarrow \{0,1\}^\ell, \\
&\qquad\qquad\qquad\qquad B \leftarrow \{0,1\}^n, j \leftarrow \{1, ..., q_1\}, x'||m' \leftarrow \mathcal{C}_1^{\mathsf{H}}(j, B, x)].
\end{aligned}$$

Then $\left|P_{\mathcal{A}}^1 - P_{\mathcal{A}}^2\right| \leq 2q_1\sqrt{P_{\mathcal{C}}} + q_0 2^{-\ell/2+2}$.

Lemma 2.10 (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

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

2.5.2 Extractable RO-Simulator $\mathcal{S}$

The extractable random oracle simulator, proposed by J. Don et al. [15], is another important proof tool for security proofs in QROM. It addresses challenges in retrieving hash inputs from superpositioned queries. This random oracle simulator is indistinguishable from a real random oracle and can extract queried inputs under specific conditions, thereby enabling security proofs in the QROM settings.

Definition 2.11. For a function $f : \mathcal{X} \times \{0,1\}^n \to \mathcal{T}$, define

$$\Gamma(f) := \max_{x,t} |\{y \mid f(x,y) = t\}| \text{ and } \Gamma'(f) := \max_{x \neq x', y'} \left|\{y \mid f(x,y) = f(x',y')\}\right|.$$

Theorem 2.12 (Theorem 4.3 of [15])**.** The extractable RO-simulator $\mathcal{S}$ constructed above, with interfaces $\mathcal{S}.RO$ and $\mathcal{S}.E$, satisfies the following properties.

1. If $\mathcal{S}.E$ is unused, $\mathcal{S}$ is perfectly indistinguishable from the random oracle RO.
2. (a) Any two subsequent independent queries to $\mathcal{S}.RO$ commute. In particular, two subsequent classical $\mathcal{S}.RO$-queries with the same input x give identical responses.
   (b) Any two subsequent independent queries to $\mathcal{S}.E$ commute. In particular, two subsequent classical $\mathcal{S}.E$-queries with the same input t give identical responses.
   (c) Any two subsequent independent queries to $\mathcal{S}.E$ and $\mathcal{S}.RO$ $8\sqrt{2\Gamma(f)/2^n}$-almost-commute.
3. (a) Any classical query $\mathcal{S}.RO(x)$ is idempotent.
   (b) Any classical query $\mathcal{S}.E(t)$ is idempotent.
4. (a) If $\hat{x} = \mathcal{S}.E(t)$ and $\hat{h} = \mathcal{S}.RO(\hat{x})$ are two subsequent classical queries then

$$\Pr[f(\hat{x},\hat{h}) \neq t \wedge \hat{x} \neq \emptyset] \leq Pr[f(\hat{x},\hat{h}) \neq t | \hat{x} \neq \emptyset] \leq 2 \cdot 2^{-n}\Gamma(f).$$

   (b) If $h = \mathcal{S}.RO(x)$ and $\hat{x} = \mathcal{S}.E(f(x,h))$ are two subsequent classical queries such that no prior query to $\mathcal{S}.E$ has been made, then

$$\Pr[\hat{x} = \emptyset] \leq 2 \cdot 2^{-n}.$$

Furthermore, the total runtime of $\mathcal{S}$, when implemented using the sparse representation of the compressed oracle, is bounded as

$$T_{\mathcal{S}} = O(q_{RO} \cdot q_E \cdot \text{Time}[f] + q_{RO}^2),$$

where q_E and q_{RO} are the number of queries to $\mathcal{S}.E$ and $\mathcal{S}.RO$, respectively.

Theorem 2.13 (Proposition 4.4. of [15])**.** Let $R' \subseteq \mathcal{X} \times \mathcal{T}$ be a relation. Consider a query algorithm $\mathcal{A}$ that makes q queries to the $\mathcal{S}.RO$ interface of $\mathcal{S}$ but no query to $\mathcal{S}.E$, outputting some $\mathbf{t} \in T^\ell$. For each i, let $\hat{x}_i$ then be obtained by making an additional query to $\mathcal{S}.E$ on input t_i. Then

$$\Pr_{\mathbf{t} \leftarrow \mathcal{A}^{\mathcal{S}.RO}, \hat{x}_i \leftarrow \mathcal{S}.E(t_i)}[\exists i : (\hat{x}_i, t_i) \in R'] \leq 128 \cdot q^2 \Gamma_R / 2^n,$$

where $R \subseteq \mathcal{X} \times \mathcal{Y}$ is the relation $(x,y) \in R \Leftrightarrow (x, f(x,y)) \in R'$ and

$$\Gamma_R := \max_{x \in \mathcal{X}} |\{y \in \{0,1\}^n | (x,y) \in R\}|.$$

3 $\mathbf{ACWC_2}$ Transformation

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

3.1 SOTP

Definition 3.1. A semi-generalized one-time pad $\mathsf{SOTP} = (\mathsf{Encode}, \mathsf{Inv})$ with a message space $\mathcal{X}$, a random 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 random $u \in \mathcal{U}$ as input, and outputs a code $y \in \mathcal{Y}$.
- $\mathsf{Inv}(y, u)$: The decoding algorithm Inv is a deterministic algorithm that takes a code $y \in \mathcal{Y}$ and random $u \in \mathcal{U}$ as input, and outputs a message $x \in \mathcal{X} \cup \{\perp\}$.

It also follows three properties as follows:

1. Decoding: For all $x \in \mathcal{X}$, $u \in \mathcal{U}$, $\mathsf{Inv}(\mathsf{Encode}(x, u), u) = x$.
2. Message-hiding: For all $x \in \mathcal{X}$, the random variable $\mathsf{Encode}(x, u)$, for $u \leftarrow \psi_{\mathcal{U}}$, has the same distribution as $\psi_{\mathcal{Y}}$.
3. Rigid: For all $u \in \mathcal{U}$, $y \in \mathcal{Y}$ with $\mathsf{Inv}(y, u) \neq \perp$, $\mathsf{Encode}(\mathsf{Inv}(y, u), u) = y$.

In contrast to the GOTP defined in [16], SOTP does not need to have an additional *randomness-hiding* property, which requires that the output $y = \mathsf{Encode}(x, u)$ follows the distribution $\psi_{\mathcal{Y}}$ and simultaneously does not leak any information about the randomness u. The absence of such an additional property allows us to design SOTP more flexibly and efficiently than GOTP. Instead, SOTP is required to be *rigid*, which means that for all $u \in \mathcal{U}$ and $y \in \mathcal{Y}$, $x = \mathsf{Inv}(y, u) \neq \perp$ implies that $\mathsf{Encode}(x, u) = y$.

3.2 $\mathbf{ACWC_2}$

Let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an underlying public key encryption scheme with message space $\mathcal{M}$ and randomness space $\mathcal{R}$, where a message $M \in \mathcal{M}$ and randomness $r \in \mathcal{R}$ are drawn from the distributions $\psi_{\mathcal{M}}$ and $\psi_{\mathcal{R}}$, respectively. Similarly, let $\mathsf{PKE}' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ be a transformed encryption scheme with message space $\mathcal{M}'$ and randomness space $\mathcal{R}'$. Let $\mathsf{SOTP} = (\mathsf{Encode}, \mathsf{Inv})$ with $\mathsf{Encode} : \mathcal{M}' \times \mathcal{U} \rightarrow \mathcal{M}$ and $\mathsf{Inv} : \mathcal{M} \times \mathcal{U} \rightarrow \mathcal{M}'$ be a semi-generalized one-time pad for distributions $\psi_{\mathcal{U}}$ and $\psi_{\mathcal{M}}$, and let $\mathsf{G} : \mathcal{R} \rightarrow \mathcal{U}$ be a hash function such that every output is 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 RRec and Inv functions do not affect the correctness error of PKE'. Thus, the factor that determines the success or failure of $\mathsf{Dec}'(\mathsf{sk}, \mathsf{c})$ is the result of $\mathsf{Dec}(sk, c)$ in Dec'. This means that the correctness error of PKE is straightforwardly transferred to that of PKE', and eventually determined by how randomness $r \in \mathcal{R}$ and message $M \in \mathcal{M}$ are sampled in PKE'. We see that r is drawn according to the distribution $\psi_{\mathcal{R}}$ and M

$\underline{\mathsf{Gen}'(1^\lambda)}$
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: $m := \mathsf{Inv}(M, \mathsf{G}(r)))$
4: **if** $r \notin \mathcal{R}$ or $m = \perp$, **return** $\perp$
5: **return** m

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

is an SOTP-encoded element in $\mathcal{M}$. Because every output of G is independently $\psi_{\mathcal{U}}$-distributed, we can expect that the message-hiding property of SOTP makes M follow the distribution $\psi_{\mathcal{M}}$ while hiding m. Eventually, both M and r are chosen according to their respective initially-intended distributions.

However, since the choice of the random oracle G can affect the correctness error of PKE$'$, we need to include this observation in the analysis of the correctness error. Theorem 3.2 shows that for all but a negligible fraction of random oracles G, the worst-case correctness of PKE$'$ (transformed by ACWC_2) is close to the average-case correctness of PKE. This is the same idea as in ACWC, and the proof strategy of Theorem 3.2 is essentially the same as that of [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 $\mathsf{SOTP} = (\mathsf{Encode}, \mathsf{Inv})$ with $\mathsf{SOTP} : \mathcal{M}' \times \mathcal{U} \to \mathcal{M}$ and $\mathsf{SOTP} : \mathcal{M} \times \mathcal{U} \to \mathcal{M}'$ be a semi-generalized one-time pad (for distributions $\psi_{\mathcal{U}}, \psi_{\mathcal{M}}$), and let $\mathsf{G} : \mathcal{R} \to \psi_{\mathcal{U}}$ be a random oracle. If PKE is δ-average-case-correct, then $\mathsf{PKE}' := \mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$ is δ'-worst-case-correct for

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

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

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

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

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

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

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

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

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

To this end, we define $g(m, r, u)$ and B as $g(m, r, u) = (\mathsf{Encode}(m, u), r)$ and $B = \{(M, r) \in |\mathsf{Dec}(sk, \mathsf{Enc}(pk, M; r)) \neq M\}$, which will be used in Lemma 3.4. Note that $\Pr_{r \leftarrow \psi_{\mathcal{R}}, u \leftarrow \psi_{\mathcal{U}}}[g(m, r, u) \in B] = \mu(pk, sk)$ holds for all $m \in \mathcal{M}'$ by the message-hiding property of the SOTP. For all $m \in \mathcal{M}'$,

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

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

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

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

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

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

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

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

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

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

Lemma 3.4 (Adapting Lemma 3.7 from [16])**.** Let g be a function, and B be some set such that

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

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

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

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

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

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

$$\Pr[\sum_r Y_r > \mu + t] \le \exp\left(\frac{-2t^2}{\sum \psi_{\mathcal{R}}(r)^2}\right) = \exp\left(\frac{-2t^2}{\|\psi_{\mathcal{R}}\|^2}\right). \tag{4}$$

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

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

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

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

Proof. We show that there exists an algorithm $\mathcal{B}$ (see Figure 7) which breaks the $\mathsf{OW\text{-}CPA}$ security of PKE using an algorithm $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ that breaks the $\mathsf{IND\text{-}CPA}$ security of $\mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$.

GAME G_0. G_0 (see Figure 6) is the original $\mathsf{IND\text{-}CPA}$ game with $\mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$. In G_0, $\mathcal{A}$ is given the challenge ciphertext $c^* := \mathsf{Enc}(pk, M^*; r^*)$ for some unknown message M^* and randomness r^*. By the definition of the $\mathsf{IND\text{-}CPA}$ game, we have

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

Game G_0

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

Figure 6: GAME G_0 of Theorems 3.5 and 3.6

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

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

$\mathsf{G}(r)$

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

Figure 7: Adversary $\mathcal{B}$ for the proof of Theorem 3.5

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

$$\left|\Pr[G_1^{\mathcal{A}} \Rightarrow 1] - \Pr[G_0^{\mathcal{A}} \Rightarrow 1]\right| \leq \mathsf{Adv}_{\mathsf{PKE}}^{\mathsf{INJ}}(\mathcal{C}).$$

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

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

Unless QUERY occurs, $\mathsf{G}(r^*)$ is a uniformly random value that is independent of $\mathcal{A}$'s view. In this case, $M^* := \mathsf{Encode}(m_b, \mathsf{G}(r^*))$ does not leak any information about m_b by the message-hiding property of the SOTP, meaning that $\Pr[G_2^{\mathcal{A}} \Rightarrow 1] = 1/2$. By contrast, if QUERY occurs, $\mathcal{B}$ (defined in Figure 7) can find $r^* \in \mathcal{L}_r$ such that $c^* := \mathsf{Enc}(pk, M^*; r^*)$, using the algorithm MRec. Indeed, for each query r to G, $\mathcal{B}$ checks whether $\mathsf{MRec}(pk, r, c^*) \in \mathcal{M}$. In the QUERY event, there exists $M^* := \mathsf{MRec}(pk, r^*, c^*) \in \mathcal{M}$ which can be the solution to its challenge ciphertext c^*. It follows that

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

which concludes the proof. □

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

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

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

Proof. To prove this theorem, we use a sequence of games G_0 to G_7 defined in Figures 6, 8, and 9, and Lemma 2.10. Before applying Lemma 2.10, we change G_0 to G_2. Subsequently, we apply Lemma 2.10 to G_2 and G_3. A detailed explanation of the security proof is provided in the following.
GAME G_0. G_0 (see Figure 6) is the original IND-CPA game with ACWC$_2$[PKE, SOTP, G]. By definition, we have

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

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

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

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

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

<u>Games G_1-G_5</u>

$\mathsf{G} \leftarrow (\mathcal{R} \to \mathcal{U})$ // G_1
$r \leftarrow \mathcal{R}$
$u := \mathsf{G}(r)$ // 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

<u>$\mathcal{C}^{\mathsf{G}}(r, u)$</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 $[\![b = b']\!]$

<u>$\mathcal{D}^{\mathsf{F}}(r, u)$</u>

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

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

Game G_6-G_7

$(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$
$r \leftarrow \psi_{\mathcal{R}}$
$M \leftarrow \psi_{\mathcal{M}}$
$c^* \leftarrow \mathsf{Enc}(pk, M; r)$
$T \leftarrow \mathcal{E}(pk, c^*)$ // G_6
$M' \leftarrow \mathcal{B}(pk, c^*)$ // G_7
return $r \in T$ // G_6
return $[\![M = M']\!]$ // G_7

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

$i \leftarrow \{1, \cdots, q_{\mathsf{G}}\}$
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

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

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

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

GAME G_3. We define G_3 by providing function F to algorithm $\mathcal{C}$ instead of G. By applying Lemma 2.10 with $\mathcal{C}$, $S := \{r\}$, and $z := (r, u)$, we obtain the following:

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

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

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

GAME G_4 and G_5. We define G_4 according to Lemma 2.10. In addition, we define G_5 by changing the way M is calculated. Instead of computing $M = \mathsf{Encode}(m_b, u)$, we sample $M \leftarrow \psi_{\mathcal{M}}$. By contrast, in G_4, since u is sampled from $\psi_{\mathcal{U}}$ and used only for computing $\mathsf{Encode}(m_b, u)$, the message-hiding property of SOTP shows that $M = \mathsf{Encode}(m_b, u)$ follows the distribution $\psi_{\mathcal{M}}$. Therefore,

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

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

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

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

$$\left|\Pr[G_6^{\mathcal{A}} \Rightarrow 1] - \Pr[G_7^{\mathcal{A}} \Rightarrow 1]\right| \le \mathsf{Adv}_{\mathsf{PKE}}^{\mathsf{INJ}}(\mathcal{C}).$$

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

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

Combining all (in)equalities and bounds, we have

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

which concludes the proof. □

Theorem 3.7. If PKE is (weakly) γ-spread, SOTP has the message hiding property, and G is modeled as a random oracle, $\mathsf{PKE}' = \mathsf{ACWC}_2[\mathsf{PKE},\mathsf{SOTP},\mathsf{G}]$ is (weakly) γ'-spread with

$$\gamma' = \gamma - \log_2(|\mathcal{M}| \cdot \max_{M\in\mathcal{M}} \psi_{\mathcal{M}}(M)),$$

where $\mathcal{M}$ is the message space of PKE and $\psi_{\mathcal{M}}(M)$ is the probability that $M \in \mathcal{M}$ is sampled from the distribution $\psi_{\mathcal{M}}$.

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

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

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

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

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:

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

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

4 IND-CCA Secure KEM from ACWC_2

4.1 FO Transform with Re-encryption

One can apply the Fujisaki-Okamoto transformation $\mathsf{FO}^{\perp}_{\mathsf{KEM}}$ to the IND-CPA secure PKE', as shown in Figure 5, to obtain an IND-CCA secure KEM. Figure 10 shows the resultant $\mathsf{KEM} := \mathsf{FO}^{\perp}_{\mathsf{KEM}}[\mathsf{PKE}', \mathsf{H}] = (\mathsf{Gen}, \mathsf{Encap}, \mathsf{Decap})$, 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 [21] and [15], which are stated in Theorems 4.1 and 4.2, respectively. By combining these results with Theorems 3.5 and 3.6, we can achieve the IND-CCA security of KEM in the classical/quantum random oracle model. In the case of the quantum random oracle model (QROM), we need to further use the fact that IND-CPA generically implies OW-CPA.

$\underline{\mathsf{Encap}(pk)}$	$\underline{\mathsf{Decap}(sk, c)}$
1: $m \leftarrow \mathcal{M}$	1: $m' := \mathsf{Dec}'(sk, c)$
2: $(R, K) := \mathsf{H}(m)$	- $M' = \mathsf{Dec}(sk, c)$
3: $c := \mathsf{Enc}'(pk, m; R)$	- $r' = \mathsf{RRec}(pk, M', c)$
- $r \leftarrow \psi_{\mathcal{R}}$ using the randomness R	- $m' = \mathsf{Inv}(M', \mathsf{G}(r'))$
- $M := \mathsf{Encode}(m, \mathsf{G}(r))$	- **if** $r' \notin \mathcal{R}$ or $m' = \perp$, **return** $\perp$
- $c := \mathsf{Enc}(pk, M; r)$	- **return** m'
4: **return** (K, c)	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 10: $\mathsf{KEM} = \mathsf{FO}^{\perp}_{\mathsf{KEM}}[\mathsf{PKE}', \mathsf{H}]$

Theorem 4.1 (IND-CPA of $\mathsf{PKE}' \overset{\mathsf{ROM}}{\Longrightarrow}$ IND-CCA of KEM [21])**.** 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 of $\mathsf{PKE}' \overset{\mathsf{QROM}}{\Longrightarrow}$ IND-CCA 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

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

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 $\mathsf{FO}_{\mathsf{KEM}}^{\perp}$ 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 $\mathsf{FO}_{\mathsf{KEM}}^{\perp}$ (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 $\mathsf{FO}_{\mathsf{KEM}}^{\perp}$ 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 10 into that of Figure 11, 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 $\mathsf{FO}_{\mathsf{KEM}}^{\perp}$ conversion based on ACWC_2, denoted as $\overline{\mathsf{FO}}_{\mathsf{KEM}}^{\perp}$, as shown in Figure 12. In $\overline{\mathsf{FO}}_{\mathsf{KEM}}^{\perp}$, 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 $\mathsf{FO}_{\mathsf{KEM}}^{\perp}$ in Figure 11 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 $\mathsf{FO}_{\mathsf{KEM}}^{\perp}$ and $\overline{\mathsf{FO}}_{\mathsf{KEM}}^{\perp}$ work identically and thus achieve the same level of IND-CCA security.

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

$M' = \mathsf{Dec}(sk, c)$
$r' = \mathsf{RRec}(pk, M', c)$
$m' = \mathsf{Inv}(M', \mathsf{G}(r'))$
$(R', K') := \mathsf{H}(m')$
if $m' = \perp$ or $\boxed{r' \notin \mathcal{R} \text{ or } c \neq \mathsf{Enc}'(pk, m'; R')}$
  return $\perp$
else
  return K'

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

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

$M' = \mathsf{Dec}(sk, c)$
$r' = \mathsf{RRec}(pk, M', c)$
$m' = \mathsf{Inv}(M', \mathsf{G}(r'))$
$(R', K') := \mathsf{H}(m')$
$\boxed{r'' \leftarrow \psi_{\mathcal{R}} \text{ with the randomness } R'}$
if $m' = \perp$ or $\boxed{r' \neq r''}$
  return $\perp$
else
  return K'

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

Lemma 4.3. Assume that the output of Dec in PKE always belongs to $\mathcal{M}$, PKE is injective in the injectivity game of Figure 2, and PKE and SOTP are rigid. Then, $r' \in \mathcal{R}$ and $c = \mathsf{Enc}'(pk, \tilde{m}'; R')$ in $\mathsf{FO}^{\perp}_{\mathsf{KEM}}$ holds if and only if $r' = r''$ in $\overline{\mathsf{FO}}^{\perp}_{\mathsf{KEM}}$ holds.

Proof. Assume that $m' \neq \perp$, $r' \in \mathcal{R}$, and $c = \mathsf{Enc}'(pk, m'; R')$ holds in the Decap of $\mathsf{FO}^{\perp}_{\mathsf{KEM}}$. By the definition of Enc', we have $c = \mathsf{Enc}(pk, \mathsf{Encode}(m', \mathsf{G}(r'')); r'')$, where $r'' \leftarrow \psi_{\mathcal{R}}$ is sampled using the randomness R'. Furthermore, since $M' = \mathsf{Dec}(sk, c) \in \mathcal{M}$ and $r' = \mathsf{RRec}(pk, M', c) \in \mathcal{R}$, the rigidity of the PKE leads to the equality $c = \mathsf{Enc}(pk, M'; r')$. Because PKE is injective, these two equations with respect to c imply that $r' = r''$.

Conversely, assume that $m' \neq \perp$ and $r' = r''$ holds for a ciphertext c in the Decap of $\overline{\mathsf{FO}}^{\perp}_{\mathsf{KEM}}$. By the rigidity of the SOTP, $m' = \mathsf{Inv}(M', \mathsf{G}(r')) \neq \perp$ implies $M' = \mathsf{Encode}(m', \mathsf{G}(r'))$, thus $M' = \mathsf{Encode}(m', \mathsf{G}(r''))$. Also, since $r'' \leftarrow \psi_{\mathcal{R}}$ is sampled using the randomness R' and $r' = r''$, it follows that $r' \in \mathcal{R}$. Since $M' = \mathsf{Dec}(sk, c) \in \mathcal{M}$ and $r' = \mathsf{RRec}(pk, M', c) \in \mathcal{R}$, by the rigidity of the PKE, $c = \mathsf{Enc}(pk, \mathsf{Dec}(sk, c); r') = \mathsf{Enc}(pk, \mathsf{Encode}(m', \mathsf{G}(r'')); r'') = \mathsf{Enc}'(pk, m'; R')$ holds. $\square$

5 IND-CCA Secure PKE from ACWC_2

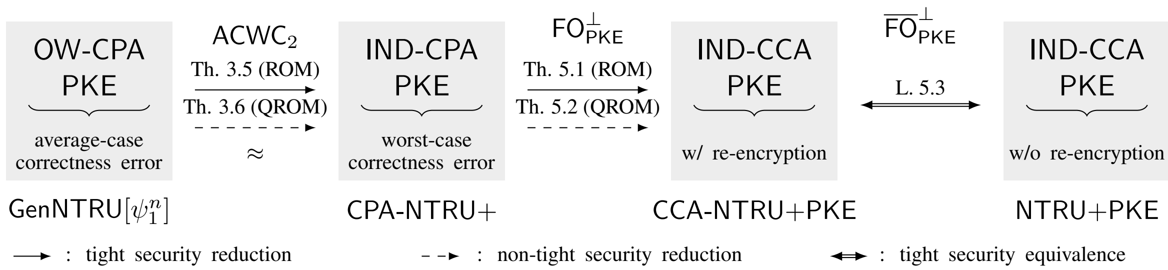

Figure 13: Overview of security reductions for PKE

5.1 FO Transform with Re-encryption

If the message space $\mathcal{M}'$ of an $\mathsf{IND\text{-}CPA}$ secure PKE' is sufficiently large, we can apply the another well-known Fujisaki-Okamoto transformation $\mathsf{FO}^{\perp}_{\mathsf{PKE}}$ [17] to the $\mathsf{IND\text{-}CPA}$ secure PKE' to obtain an $\mathsf{IND\text{-}CCA}$ secure PKE''. For the simplicity's sake, let $\mathcal{M}' = \{0,1\}^{\ell_m + \ell_r}$ for some integers ℓ_m and ℓ_r. The idea behind the $\mathsf{FO}^{\perp}_{\mathsf{PKE}}$ is to concatenate an arbitrary message $m \in \{0,1\}^{\ell_m}$ and a random bit-string $r \in \{0,1\}^{\ell_r}$ and set a new message $\tilde{m} := m||r \in \{0,1\}^{\ell_m + \ell_r}$ for the $\mathsf{IND\text{-}CPA}$ secure PKE'. During the decryption of PKE'', the message m is recovered by taking $[\tilde{m}]_{\ell_m}$, the most significant bits of length ℓ_m from $\tilde{m}$. Figure 14 shows the resultant $\mathsf{IND\text{-}CCA}$ secure $\mathsf{PKE}'' := \mathsf{FO}^{\perp}_{\mathsf{PKE}}[\mathsf{PKE}', \mathsf{H}] = (\mathsf{Gen}'', \mathsf{Enc}'', \mathsf{Dec}'')$, where H is a hash function (modeled as a random oracle).

As in the previous KEM, PKE'' preserves the worst-case correctness error of PKE', since Dec'' works correctly as long as Dec' is performed correctly. Regarding the $\mathsf{IND\text{-}CCA}$ security of PKE'', Figure 13 shows the overview of security reductions for PKE. Based on the $\mathsf{IND\text{-}CPA}$ security of PKE', we prove that PKE'' is $\mathsf{IND\text{-}CCA}$-secure in the random oracle model by adapting and modifying the previous security proof of [17]. Next, we prove that PKE'' is also $\mathsf{IND\text{-}CCA}$-secure in the quantum random oracle model by using the adaptive O2H lemma [36] and the extractable RO (random oracle)-simulator [15]. Later, as in $\overline{\mathsf{FO}}^{\perp}_{\mathsf{KEM}}$, an

$\underline{\mathsf{Enc}''(pk, m \in \{0,1\}^{\ell_m})}$

$r \leftarrow \{0,1\}^{\ell_r}$
$\tilde{m} = m || r \in \{0,1\}^{\ell_m + \ell_r}$
$R := \mathsf{H}(\tilde{m})$
$c := \mathsf{Enc}'(pk, \tilde{m}; R)$
- $r \leftarrow \psi_{\mathcal{R}}$ using the randomness R
- $M := \mathsf{Encode}(\tilde{m}, \mathsf{G}(r))$
- $c := \mathsf{Enc}(pk, M; r)$

return c

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

$\tilde{m}' = \mathsf{Dec}'(sk, c)$
- $M' = \mathsf{Dec}(sk, c)$
- $r' = \mathsf{RRec}(pk, M', c)$
- $\tilde{m}' = \mathsf{Inv}(M', \mathsf{G}(r'))$
- **if** $r' \notin \mathcal{R}$ or $\tilde{m}' = \perp$, **return** $\perp$
- **return** $\tilde{m}'$

$R' := \mathsf{H}(\tilde{m}')$
if $\tilde{m}' = \perp$ **or** $c \neq \mathsf{Enc}'(pk, \tilde{m}'; R')$
    return $\perp$
else
    return $[\tilde{m}']_{\ell_m}$

Figure 14: $\mathsf{FO}^{\perp}_{\mathsf{PKE}}[\mathsf{PKE}', \mathsf{H}] = (\mathsf{Gen}'', \mathsf{Enc}'', \mathsf{Dec}'')$

analogous transform $\overline{\mathsf{FO}}^{\perp}_{\mathsf{PKE}}$ for public-key encryption will convert PKE'' into more efficient PKE scheme that does not need to do re-encryption during decryption.

5.2 Security Proof in the ROM

Theorem 5.1 (IND-CPA of $\mathsf{PKE}' \overset{\text{ROM}}{\Longrightarrow}$ IND-CCA of PKE'')**.** Let PKE' be a public-key encryption scheme with worst-case correctness error δ and weakly γ-spreadness. For any classical adversary $\mathcal{A}$ against the IND-CCA security of PKE'', making at most q_{D} queries to the decryption oracle Dec'' and at most q_{H} queries to $\mathsf{H} : \mathcal{M} \to \mathcal{R}$, there exists a classical adversary $\mathcal{B}$ against the IND-CPA security of PKE' such that

$$\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{PKE}''}(\mathcal{A}) \leq 2 \cdot \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}'}(\mathcal{B}) + (q_{\mathsf{H}} + q_{\mathsf{D}}) \cdot (2^{-\gamma} + \delta) + q_{\mathsf{H}} \cdot 2^{-\ell_r}.$$

Proof. For the security proof, we analyze hybrid games G_0 to G_5, defined in Figures 15 and 16, with a fixed key pair (pk, sk). To do this, we define $\delta_{sk} := \max_{m \in \mathcal{M}} \Pr_{r \leftarrow \psi_R}[\mathsf{Dec}'(sk, \mathsf{Enc}'(pk, m; r)) \neq m]$ as the maximum probability of a decryption error and $\gamma_{sk} := -\log \max_{m \in \mathcal{M}, c \in \mathcal{C}} \Pr_{r \leftarrow \psi_R}[c = \mathsf{Enc}'(pk, m; r)]$ as the negative logarithm of the maximum probability of any ciphertext for the fixed key pair (pk, sk), ensuring $\mathbb{E}[\delta_{sk}] \leq \delta$ and $\mathbb{E}[2^{-\gamma_{sk}}] \leq 2^{-\gamma}$, with expectations taken over $(pk, sk) \leftarrow \mathsf{Gen}'(1^\lambda)$. A detailed explanation of the security proof is provided below.

GAME G_0. G_0 is the $\mathsf{IND\text{-}CCA}$ game against PKE'' with a fixed key pair (pk, sk) (see Figure 15). Here, we define the advantage of an adversary $\mathcal{A}$ in the $\mathsf{IND\text{-}CCA}$ game against PKE'' for a fixed key pair (pk, sk) as:

$$\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{PKE}'', sk}(\mathcal{A}) = \left| \Pr[G_0^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2} \right|.$$

GAME G_1. G_1 is defined by modifying the Dec'' oracle, as shown in Figure 15. In G_1, the Dec'' oracle is altered to first compute $\tilde{m}' = \mathsf{Dec}'(sk, c)$ and return $[\tilde{m}']_{\ell_m}$ if there exists $(\tilde{m}, \tilde{r}) \in \mathcal{L}_{\mathsf{H}}$ such that $\mathsf{Enc}'(pk, \tilde{m}; \tilde{r}) = c$ and $\tilde{m} = \tilde{m}'$. The Dec'' oracle in G_0 differs from that in G_1 if $\mathsf{H}(\tilde{m})$ has not been queried, which occurs with probability $\cdot 2^{-\gamma_{sk}}$. By the union bound:

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

GAMES G_0-G_2

$(pk, sk) \leftarrow \mathsf{Gen}''(1^\lambda)$
$(m_0, m_1) \leftarrow \mathcal{A}_0^{\mathsf{H},\mathsf{Dec}''}(pk)$
$b \leftarrow \{0,1\}$
$r \leftarrow \{0,1\}^{\ell_r}$
$\tilde{m} = m_b || r \in \{0,1\}^{n=\ell_m+\ell_r}$
$\tilde{r} = \mathsf{H}(\tilde{m})$
$c^* = \mathsf{Enc}'(pk, \tilde{m}; \tilde{r})$
$b' \leftarrow \mathcal{A}_1^{\mathsf{H},\mathsf{Dec}''}(pk, c^*)$
return $[\![b = b']\!]$

GAME G_3

$(pk, sk) \leftarrow \mathsf{Gen}''(1^\lambda)$
$(m_0, m_1) \leftarrow \mathcal{A}_0^{\mathsf{H},\mathsf{Dec}''}(pk)$
$(r_0, r_1) \leftarrow \{0,1\}^{\ell_r} \times \{0,1\}^{\ell_r}$
$b \leftarrow \{0,1\}$
$\tilde{m}_b = m_b || r_b \in \{0,1\}^{n=\ell_m+\ell_r}$
$\tilde{r} = \mathsf{H}(\tilde{m}_b)$
$c^* := \mathsf{Enc}'(pk, \tilde{m}_b; \tilde{r})$
$b' \leftarrow \mathcal{A}_1^{\mathsf{H},\mathsf{Dec}''}(pk, c)$
return $[\![b = b']\!]$

$\mathsf{H}(\tilde{m})$

if $\exists \tilde{r}$ such that $(\tilde{m}, \tilde{r}) \in \mathcal{L}_\mathsf{H}$
  return $\tilde{r}$
$\tilde{r} \leftarrow \mathcal{R}$
$\mathcal{L}_\mathsf{H} := \mathcal{L}_\mathsf{H} \cup \{(\tilde{m}, \tilde{r})\}$
return $\tilde{r}$

$\mathsf{Dec}''(c \neq c^*)$ //G_0

$\tilde{m}' = \mathsf{Dec}'(sk, c)$
if $\tilde{m}' = \perp$ or
  $c \neq \mathsf{Enc}'(pk, \tilde{m}'; \mathsf{H}(\tilde{m}'))$
  return $\perp$
else, return $[\tilde{m}']_{\ell_m}$

$\mathsf{Dec}''(c \neq c^*)$ //G_1-G_3

$\tilde{m}' = \mathsf{Dec}'(sk, c)$
if $\exists (\tilde{m}, \tilde{r}) \in \mathcal{L}_\mathsf{H}$ such that
  $c = \mathsf{Enc}'(pk, \tilde{m}; \tilde{r})$ //G_1-G_3
  and $\tilde{m} = \tilde{m}'$ //G_1
  return $[\tilde{m}]_{\ell_m}$
else, return $\perp$

Figure 15: GAMES G_0-G_3 for the proof of Theorem 5.1

GAME G_2. G_2 is defined by modifying the Dec'' oracle, as shown in Figure 15. In G_2, Dec'' no longer checks whether $\tilde{m} = \tilde{m}'$, where $\tilde{m}' = \mathsf{Dec}'(sk, c)$. Instead, it returns $\tilde{m}$ directly if there exists $(\tilde{m}, \tilde{r}) \in \mathcal{L}_\mathsf{H}$ such that $\mathsf{Enc}'(pk, \tilde{m}; \tilde{r}) = c$. Since the Dec'' oracle in G_1 is identical to that of G_2 if there are no hash queries to H that lead to a correctness error, by the union bound, the following holds:

$$\left|\Pr[G_1^\mathcal{A} \Rightarrow 1] - \Pr[G_2^\mathcal{A} \Rightarrow 1]\right| \leq (q_\mathsf{H} + q_\mathsf{D}) \cdot \delta_{sk}.$$

Note that the Dec'' oracle in G_2 no longer requires the secret key.

GAME G_3. G_3 is defined by replacing $\tilde{m}$ by $\tilde{m}_b$, as shown in Figure 15. Since this change is only conceptual, the following holds:

$$\Pr[G_2^\mathcal{A} \Rightarrow 1] = \Pr[G_3^\mathcal{A} \Rightarrow 1].$$

GAME G_4. G_4 is defined by moving part of the game into an adversary $\mathcal{C}^\mathsf{H} = (\mathcal{C}_0^\mathsf{H}, \mathcal{C}_1^\mathsf{H})$, defined in Figure 16. Since the change is only conceptual, the following holds:

$$\Pr[G_3^\mathcal{A} \Rightarrow 1] = \Pr[G_4^\mathcal{A} \Rightarrow 1].$$

GAME G_5. G_5 is defined by changing how $\tilde{r}^*$ is chosen. In G_5, instead of generating $\tilde{r}^*$ using H, $\tilde{r}^*$ is chosen randomly from $\mathcal{R}$, which will not be noticed by $\mathcal{A}$ as long as $\mathcal{A}$ does not query $\tilde{r}$ to H. Let QUERY be an event that $\mathcal{A}$ queries H on $\tilde{m}_b$. Due to the difference lemma [35], the following holds:

$$\left|\Pr[G_4^\mathcal{A} \Rightarrow 1] - \Pr[G_5^\mathcal{A} \Rightarrow 1]\right| \leq \Pr[\mathsf{QUERY}].$$

$\underline{\textsc{Games}\ G_4\text{-}G_5}$

$(pk, sk) \leftarrow \mathsf{Gen}''(1^\lambda)$
$(\tilde{m}_0, \tilde{m}_1) \leftarrow \mathcal{C}_0^{\mathsf{H}}(pk)$
$b \leftarrow \{0,1\}$
$\tilde{r}^* = \mathsf{H}(\tilde{m}_b)$ $\quad //G_4$
$\tilde{r}^* \leftarrow \mathcal{R}$ $\quad //G_5$
$c^* := \mathsf{Enc}'(pk, \tilde{m}_b; \tilde{r}^*)$
$b' \leftarrow \mathcal{C}_1^{\mathsf{H}}(pk, c^*)$
return $[\![b = b']\!]$

$\underline{\mathsf{H}(\tilde{m})}$

if $\exists \tilde{r}$ such that $(\tilde{m}, \tilde{r}) \in \mathcal{L}_{\mathsf{H}}$
  return $\tilde{r}$
else, $\tilde{r} \leftarrow \mathcal{R}$
$\mathcal{L}_{\mathsf{H}} := \mathcal{L}_{\mathsf{H}} \cup \{(\tilde{m}, \tilde{r})\}$
return $\tilde{r}$

$\underline{\mathsf{Dec}''(c \neq c^*)}$

if $\exists (\tilde{m}, \tilde{r}) \in \mathcal{L}_{\mathsf{H}}$ such that $c = \mathsf{Enc}'(pk, \tilde{m}; \tilde{r})$
  return $[\tilde{m}]_{\ell_m}$
else, return $\perp$

$\underline{\mathcal{C}_0^{\mathsf{H}}(pk)}$

$(m_0, m_1) \leftarrow \mathcal{A}_0^{\mathsf{H},\mathsf{Dec}''}(pk)$
$(r_0, r_1) \leftarrow \{0,1\}^{\ell_r} \times \{0,1\}^{\ell_r}$
return $(\tilde{m}_0, \tilde{m}_1) = (m_0 || r_0, m_1 || r_1)$

$\underline{\mathcal{C}_1^{\mathsf{H}}(pk)}$

$b' \leftarrow \mathcal{A}_1^{\mathsf{H},\mathsf{Dec}''}(pk, c^*)$
return b'

Figure 16: Games G_4-G_5 of Theorem 5.1

Also, since the adversary $\mathcal{C}$ in G_5 is playing the original $\mathsf{IND\text{-}CPA}$ game against PKE', the following holds

$$\left|\Pr[G_5^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right| = \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}',sk}(\mathcal{C}).$$

Now, construct an adversary $\mathcal{D}^{\mathcal{H}} = (\mathcal{D}_0^{\mathcal{H}}, \mathcal{D}_1^{\mathcal{H}})$ in Figure 17 that solves the $\mathsf{IND\text{-}CPA}$ game with PKE' when the event QUERY occurs. Since r_{1-b} is completely hidden from the adversary $\mathcal{A}$, the probability that $\mathcal{A}$ ever queries $\tilde{m}_{1-b} = (m_{1-b} || r_{1-b})$ to H can be bounded to $q_{\mathsf{H}} \cdot 2^{-\ell_r}$. Therefore, the following holds:

$$\Pr[\mathsf{QUERY}] \leq \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}',sk}(\mathcal{D}) + q_{\mathsf{H}} \cdot 2^{-l_r}.$$

Combining the intermediate results and folding $\mathcal{C}$ and $\mathcal{D}$ into one single adversary $\mathcal{B}$ against $\mathsf{IND\text{-}CPA}$ with PKE', and then taking the expectation over $(pk, sk) \leftarrow \mathsf{Gen}'(1^\lambda)$ yields the required bound of the theorem. □

$\underline{\mathcal{D}_0^{\mathsf{H}}(pk)}$

$\mathcal{L}_{\mathsf{H}}, \mathcal{L}_{\tilde{m}} := \emptyset$
$(\tilde{m}_0, \tilde{m}_1) \leftarrow \mathcal{C}_0^{\mathsf{H}}(pk)$
return $(\tilde{m}_0, \tilde{m}_1)$

$\underline{\mathcal{D}_1^{\mathsf{H}}(pk, c^*)}$

$\mathcal{C}_1^{\mathsf{H}}(pk, c^*)$
if $\tilde{m}_0 \in \mathcal{L}_{\tilde{m}}$**, return** $b' = 0$
else, return $b' = 1$

$\underline{\mathsf{H}(\tilde{m})}$

if $\exists \tilde{r}$ such that $(\tilde{m}, \tilde{r}) \in \mathcal{L}_{\mathsf{H}}$
  return $\tilde{r}$
$\tilde{r} \leftarrow \mathcal{R}$
$\mathcal{L}_{\mathsf{H}} := \mathcal{L}_{\mathsf{H}} \cup \{(\tilde{m}, \tilde{r})\}$
$\mathcal{L}_{\tilde{m}} := \mathcal{L}_{\tilde{m}} \cup \{\tilde{m}\}$
return $\tilde{r}$

Figure 17: The adversary $\mathcal{D}$ in Theorem 5.1

5.3 Security Proof in the QROM

Theorem 5.2 (IND-CPA of $\mathsf{PKE}' \overset{\text{QROM}}{\Longrightarrow}$ IND-CCA of PKE'')**.** Let PKE' be a public-key encryption scheme with a worst-case correctness error δ that satisfies weak γ-spreadness. For any quantum adversary $\mathcal{A}$ against the IND-CCA security of PKE'', making at most q_{D} queries to the decryption oracle Dec'' and at most q_{H} queries to $\mathsf{H} : \mathcal{M} \to \mathcal{R}$, there exist a quantum adversary $\mathcal{B}$ against the IND-CPA security of PKE' such that

$$\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{PKE}''}(\mathcal{A}) \le (2q_{\mathsf{H}} + 2q_{\mathsf{D}} + 1)\sqrt{2\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}'}(\mathcal{B}) + \varepsilon} + (q_{\mathsf{H}} + q_{\mathsf{D}}) \cdot 2^{-\ell_r/2+2}$$

where $\varepsilon = 128(q_{\mathsf{H}} + q_{\mathsf{D}})^2\delta + q_{\mathsf{D}} \cdot (q_{\mathsf{H}} + q_{\mathsf{D}}) \cdot 2^{(-\gamma+9)/2} + q_{\mathsf{D}} \cdot 2^{-\ell_r+1}$.

The proof strategy for Theorem 5.2 closely follows Theorem 6.1 in [15], with a key distinction in the application of the O2H lemma. While [15] used Lemma 2.10 (Theorem 3 of [3]) to prove the IND-CCA security of the KEM, an adaptive version of the O2H lemma, as outlined in Lemma 2.9, is used to prove the IND-CCA security of PKE''.

Proof. The security proof begins by analyzing hybrid games with a fixed key pair (pk, sk). To do this, we define $\delta_{sk} := \max_{m\in\mathcal{M}} \Pr_{r\leftarrow\psi_R}[\mathsf{Dec}'(sk, \mathsf{Enc}'(pk, m; r)) \neq m]$ as the maximum probability of a decryption error and $\gamma_{sk} := -\log \max_{m\in\mathcal{M},c\in\mathcal{C}} \Pr_{r\leftarrow\psi_R}[c = \mathsf{Enc}'(pk, m; r)]$ as the negative logarithm of the maximum probability of any ciphertext for the fixed key pair (pk, sk), ensuring $\mathbb{E}[\delta_{sk}] \le \delta$ and $\mathbb{E}[2^{-\gamma_{sk}}] \le 2^{-\gamma}$, with expectations taken over $(pk, sk) \leftarrow \mathsf{Gen}'(1^\lambda)$. A detailed explanation of the security proof is provided below.

GAME G_0. G_0 is the original IND-CCA game against PKE'' with the fixed key pair (pk, sk). Here, define the advantage of adversary $\mathcal{A}$ in the IND-CCA game against PKE'' for a fixed key pair (pk, sk) as:

$$\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{PKE}'',sk}(\mathcal{A}) = \left|\Pr[G_0^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right|.$$

GAME G_1. G_1 is defined by moving parts of the game into a set of algorithms $\mathcal{C}^{\mathsf{H}} = (\mathcal{C}_0^{\mathsf{H}}, \mathcal{C}_1^{\mathsf{H}})$, as shown in Figure 18. Since this change is only conceptual, it holds that:

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

GAMES G_2 AND G_3. G_2 and G_3 are defined by applying Lemma 2.9 to G_1 and $\mathcal{C}^{\mathsf{H}}$ (see Figure 18). Note that G_2 and G_3 generate $\tilde{r} \leftarrow \mathcal{R}$ instead of $\tilde{r} = \mathsf{H}(\tilde{m})$. As a result, it holds that:

$$\left|\Pr[G_1^{\mathcal{A}} \Rightarrow 1] - \Pr[G_2^{\mathcal{A}} \Rightarrow 1]\right| \le 2 \cdot (q_{\mathsf{H}} + q_{\mathsf{D}})\sqrt{\Pr[G_3 \Rightarrow 1]} + (q_{\mathsf{H}} + q_{\mathsf{D}}) \cdot 2^{-\ell_r/2+2}.$$

Combining the analyses of G_0 to G_3, the following inequality holds:

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{PKE}',sk}(\mathcal{A}) &= \left|\Pr[G_0^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right| = \left|\Pr[G_1^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right| \\
&\le \left|\Pr[G_1^{\mathcal{A}} \Rightarrow 1] - \Pr[G_2^{\mathcal{A}} \Rightarrow 1]\right| + \left|\Pr[G_2^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right| \\
&\le 2 \cdot (q_{\mathsf{H}} + q_{\mathsf{D}})\sqrt{\Pr[G_3 \Rightarrow 1]} + (q_{\mathsf{H}} + q_{\mathsf{D}}) \cdot 2^{-\ell_r/2+2} + \left|\Pr[G_2^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right|. \qquad (5)
\end{aligned}$$

GAME G_0

$\mathsf{H} \leftarrow (\mathcal{M} \rightarrow \mathcal{R})$
$(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$
$(m_0, m_1) \leftarrow \mathcal{A}_0^{\mathsf{H},\mathsf{Dec}''}(pk)$
$b \leftarrow \{0,1\}$
$r \leftarrow \{0,1\}^{\ell_r}$
$\tilde{m} = m_b || r \in \{0,1\}^{n=\ell_m+\ell_r}$
$\tilde{r} = \mathsf{H}(\tilde{m})$
$c^* = \mathsf{Enc}'(pk, \tilde{m}; \tilde{r})$
$b' \leftarrow \mathcal{A}_1^{\mathsf{H},\mathsf{Dec}''}(pk, c^*)$
return $[\![b = b']\!]$

GAMES G_1-G_3

$\mathsf{H} \leftarrow (\mathcal{M} \rightarrow \mathcal{R})$
$m_b \leftarrow \mathcal{C}_0^{\mathsf{H}}()$
$r \leftarrow \{0,1\}^{\ell_r}$
$\tilde{m} = m_b || r$
$\tilde{r} := \mathsf{H}(\tilde{m})$ //G_1
$\tilde{r} \leftarrow \mathcal{R}$ //G_2-G_3
$b' \leftarrow \mathcal{C}_1^{\mathsf{H}}(r, \tilde{r})$ //G_1-G_2
$\tilde{m}' \leftarrow \mathcal{D}^{\mathsf{H}}(r, \tilde{r})$ //G_3
return $[\![b = b']\!]$ //G_1-G_2
return $[\![\tilde{m}_b = \tilde{m}']\!]$ //G_3

$\mathsf{Dec}''(c \neq c^*)$

$\tilde{m}' = \mathsf{Dec}'(sk, c)$
$\tilde{r}' = \mathsf{H}(\tilde{m}')$
if $c \neq \mathsf{Enc}'(pk, \tilde{m}'; \tilde{r}')$
  return $\perp$
else, return $[\![\tilde{m}']\!]_{\ell_m}$

$\mathcal{C}_0^{\mathsf{H}}()$

$(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$
$(m_0, m_1) \leftarrow \mathcal{A}_0^{\mathsf{H},\mathsf{Dec}''}(pk)$
$b \leftarrow \{0,1\}$
return m_b

$\mathcal{C}_1^{\mathsf{H}}(r, \tilde{r})$

$c^* \leftarrow \mathsf{Enc}'(pk, \tilde{m}; \tilde{r})$
$b' \leftarrow \mathcal{A}_1^{\mathsf{H},\mathsf{Dec}''}(pk, c^*)$
return b'

$\mathcal{D}^{\mathsf{H}}(r, \tilde{r})$

$i \leftarrow \{1, \cdots, q_{\mathsf{H}}\}$
Run $\mathcal{C}_1^{\mathsf{H}}(r, \tilde{r})$ till i-th H-query
$\tilde{m}' \leftarrow$ measure i-th H-query
return $\tilde{m}'$

Figure 18: GAMES G_0-G_3 for the proof of Theorem 5.2

GAME $G_{2.1}$. $G_{2.1}$ is defined by modifying G_2, moving parts of the set of algorithms $\mathcal{C}^{\mathsf{H}} = (\mathcal{C}_0^{\mathsf{H}}, \mathcal{C}_1^{\mathsf{H}})$ into the game, as shown in Figure 19. Since this change is only conceptual, it holds that:

$$\Pr[G_2^{\mathcal{A}} \Rightarrow 1] = \Pr[G_{2.1}^{\mathcal{A}} \Rightarrow 1].$$

GAME $G_{2.2}$. $G_{2.2}$ is defined by modifying the generation of $\tilde{m}$, as shown in Figure 19. Since this change is only conceptual, the following holds:

$$\Pr[G_{2.1}^{\mathcal{A}} \Rightarrow 1] = \Pr[G_{2.2}^{\mathcal{A}} \Rightarrow 1].$$

GAME $G_{2.3}$. $G_{2.3}$ is defined by moving parts of the game into a set of algorithms $\mathcal{E}^{\mathsf{H},\mathsf{Dec}''} = (\mathcal{E}_0^{\mathsf{H},\mathsf{Dec}''}, \mathcal{E}_1^{\mathsf{H},\mathsf{Dec}''})$, as shown in Figure 19. Since this change is conceptual, it holds that:

$$\Pr[G_{2.2}^{\mathcal{A}} \Rightarrow 1] = \Pr[G_{2.3}^{\mathcal{A}} \Rightarrow 1].$$

GAME $G_{2.4}$. $G_{2.4}$ is defined by replacing the random oracle H with the extractable RO-simulator $\mathcal{S}$ for the relation $R_t := \{(x, y) \mid f(x, y) = t\}$, where $f(x, y) = \mathsf{Enc}'(pk, x; y)$ from Theorem 2.12, as shown in Figure 19. Furthermore, at the end of the game, the extractor interface $\mathcal{S}.E$ is invoked to compute

GAMES $G_{2.1}$-$G_{2.2}$

$\mathsf{H} \leftarrow (\mathcal{M} \to \mathcal{R})$
$(pk, sk) \leftarrow \mathsf{Gen}'(1^\lambda)$
$(m_0, m_1) \leftarrow \mathcal{A}_0^{\mathsf{H},\mathsf{Dec}''}(pk)$
$(r_0, r_1) \leftarrow \{0,1\}^{\ell_r} \times \{0,1\}^{\ell_r}$ //$G_{2.2}$
$b \leftarrow \{0,1\}$
$r \leftarrow \{0,1\}^{\ell_r}$ //$G_{2.1}$
$\tilde{m} = m_b || r$ //$G_{2.1}$
$\tilde{m} = m_b || r_b$ //$G_{2.2}$
$\tilde{r} \leftarrow \mathcal{R}$
$c^* \leftarrow \mathsf{Enc}'(pk, \tilde{m}; \tilde{r})$
$b' \leftarrow \mathcal{A}_1^{\mathsf{H},\mathsf{Dec}''}(pk, c^*)$
return $[\![b = b']\!]$

GAMES $G_{2.3}$-$G_{2.7}$

$\mathsf{H} \leftarrow (\mathcal{M} \to \mathcal{R})$ //$G_{2.3}$
$\mathsf{H} = \mathcal{S}.RO$ //$G_{2.4}$-$G_{2.7}$
$(pk, sk) \leftarrow \mathsf{Gen}'(1^\lambda)$
$(\tilde{m}_0, \tilde{m}_1) \leftarrow \mathcal{E}_0^{\mathsf{H},\mathsf{Dec}''}(pk)$
$b \leftarrow \{0,1\}$
$\tilde{r} \leftarrow \mathcal{R}$
$c^* \leftarrow \mathsf{Enc}'(pk, \tilde{m}_b; \tilde{r})$
$b' \leftarrow \mathcal{E}_1^{\mathsf{H},\mathsf{Dec}''}(pk, c^*)$
return $[\![b = b']\!]$
while $i \in I$ **do** //$G_{2.4}$
  $\hat{m}_i \leftarrow \mathcal{S}.E(c_i)$ //$G_{2.4}$

$\mathsf{Dec}''(c \neq c^*)$

$\tilde{m}' = \mathsf{Dec}'(sk, c)$ //$G_{2.1}$-$G_{2.6}$
$\tilde{r}' = \mathsf{H}(\tilde{m}')$ //$G_{2.1}$-$G_{2.6}$
if $c \neq \mathsf{Enc}'(pk, \tilde{m}'; \tilde{r}')$ //$G_{2.1}$-$G_{2.5}$
  return $\perp$ //$G_{2.1}$-$G_{2.5}$
else, return $[\![\tilde{m}']\!]_{\ell_m}$ //$G_{2.1}$-$G_{2.5}$
$\hat{m}' \leftarrow \mathcal{S}.E(c)$ //$G_{2.5}$-$G_{2.7}$
if $\hat{m}' = \perp$, **return** $\perp$ //$G_{2.6}$-$G_{2.7}$
else, return $[\![\hat{m}']\!]_{\ell_m}$ //$G_{2.6}$-$G_{2.7}$

$\mathcal{E}_0^{\mathsf{H},\mathsf{Dec}''}(pk)$

$(m_0, m_1) \leftarrow \mathcal{A}_0^{\mathsf{H},\mathsf{Dec}''}(pk)$
$(r_0, r_1) \leftarrow \{0,1\}^{\ell_r} \times \{0,1\}^{\ell_r}$
return $(\tilde{m}_0, \tilde{m}_1) = (m_0 || r_0, m_1 || r_1)$

$\mathcal{E}_1^{\mathsf{H},\mathsf{Dec}''}(pk, c^*)$

$b' \leftarrow \mathcal{A}_1^{\mathsf{H},\mathsf{Dec}''}(pk, c^*)$
return b'

Figure 19: GAMES $G_{2.1}$-$G_{2.7}$ for the proof of Theorem 5.2

$\hat{m}_i := \mathcal{S}.E(c_i)$ for each c_i that $\mathcal{A}$ queried to Dec'' during its run. According to the first statement of Theorem 2.12,

$$\Pr[G_{2.3}^{\mathcal{A}} \Rightarrow 1] = \Pr[G_{2.4}^{\mathcal{A}} \Rightarrow 1].$$

Furthermore, applying Theorem 2.13 for $R' := \{(m, c) : \mathsf{Dec}'(sk, c) \neq m\}$, the event

$$P^\dagger := [\forall i : \hat{m}_i = \tilde{m}_i' := \mathsf{Dec}'(sk, c_i) \vee \hat{m}_i = \emptyset]$$

holds except with probability $\varepsilon_{1,sk} := 128(q_{\mathsf{H}} + q_{\mathsf{D}})^2 \mathit{\Gamma}_R / |\mathcal{R}| = 128(q_{\mathsf{H}} + q_{\mathsf{D}})^2 \delta_{sk}$. Thus,

$$\left| \Pr[G_{2.4}^{\mathcal{A}} \Rightarrow 1] - \Pr[G_{2.4}^{\mathcal{A}} \Rightarrow 1 \wedge P^\dagger] \right| \leq \varepsilon_{sk}^1.$$

GAME $G_{2.5}$. $G_{2.5}$ is defined by moving each query $\mathcal{S}.E(c_i)$ to the end of the $\mathsf{Dec}''(c_i)$ oracle. Since $\mathcal{S}.RO(m)$ and $\mathcal{S}.E(c_i)$ now form consecutive classical queries, it follows from the contraposition of 4.(b) of Theorem 2.12 that, except with probability $2 \cdot 2^{-\ell_r}$, $\hat{m}_i = \emptyset$ implies $\mathsf{Enc}'(pk, m_i; \mathcal{S}.RO(m_i)) \neq c_i$. Applying the union bound, $P^\dagger$ implies

$$P := [\forall i : \hat{m}_i = m_i \vee (\hat{m}_i = \emptyset \wedge \mathsf{Enc}'(pk, m_i; \mathcal{S}.RO(m_i)) \neq c_i)]$$

except with probability $q_{\mathsf{D}} \cdot 2 \cdot 2^{-\ell_r}$. Furthermore, by 2.(c) of Theorem 2.12, each swap of a $\mathcal{S}.RO$ with a $\mathcal{S}.E$ query affects the final probability by at most $8\sqrt{2\Gamma(f)/|\mathcal{R}|} = 8\sqrt{2 \cdot 2^{-\gamma_{sk}}}$. Thus,

$$\left|\Pr[G_{2.4}^{\mathcal{A}} \Rightarrow 1 \wedge P^\dagger] - \Pr[G_{2.5}^{\mathcal{A}} \Rightarrow 1 \wedge P]\right| \leq \varepsilon_{2,sk}$$

with $\varepsilon_{2,sk} = 2q_{\mathsf{D}} \cdot ((q_{\mathsf{H}} + q_{\mathsf{D}}) \cdot 4\sqrt{2 \cdot 2^{-\gamma_{sk}}} + 2^{-\ell_r})$.
GAME $G_{2.6}$. In $G_{2.6}$, the decryption oracle Dec'' uses $\hat{m}_i'$ instead of $\tilde{m}_i'$ to response to the queries. However, Dec'' still queries $\mathcal{S}.RO(\tilde{m}_i')$, maintaining the interaction pattern between Dec'' and $\mathcal{S}.RO$ as in $G_{2.5}$.

Note that if the event

$$P_i := [\hat{m}_i' = m_i \vee (\hat{m}_i = \emptyset \wedge \mathsf{Enc}'(pk, m_i; \mathcal{S}.RO(m_i)) \neq c_i)]$$

holds for a given i, then the above change will not affect the response of Dec'' and thus will not affect the probability for P_{i+1} to hold as well. Therefore, by mathematical induction, the following holds:

$$\Pr[G_{2.5}^{\mathcal{A}} \Rightarrow 1 \wedge P] = \Pr[G_{2.6}^{\mathcal{A}} \Rightarrow 1 \wedge P].$$

GAME $G_{2.7}$. In $G_{2.7}$, all $\tilde{r}' = \mathsf{H}(\tilde{m}')$ queries in Dec'' are dropped or, equivalently, moved to the very end of the game execution. Invoking 2.(c) of Theorem 2.12 once again, the following holds:

$$\left|\Pr[G_{2.6}^{\mathcal{A}} \Rightarrow 1 \wedge P] - \Pr[G_{2.7}^{\mathcal{A}} \Rightarrow 1 \wedge P]\right| \leq \varepsilon_{3,sk}.$$

with $\varepsilon_{3,sk} = q_{\mathsf{D}} \cdot (q_{\mathsf{D}} + q_{\mathsf{H}}) \cdot 8\sqrt{2 \cdot 2^{-\gamma_{sk}}}$. Also, note that $G_{2.7}$ works without knowledge of the secret key sk and thus constitutes a $\mathsf{IND\text{-}CPA}$ attacker $\mathcal{E}$ against PKE for a fixed key pair (pk, sk). Therefore,

$$\left|\Pr[G_{2.7}^{\mathcal{A}} \Rightarrow 1 \wedge P] - \frac{1}{2}\right| \leq \mathsf{Adv}_{\mathsf{PKE},sk}^{\mathsf{IND\text{-}CPA}}(\mathcal{E}),$$

where $\mathsf{Adv}_{\mathsf{PKE},sk}^{\mathsf{IND\text{-}CPA}}(\mathcal{E})$ is the advantage of the adversary $\mathcal{E}$ in the $\mathsf{IND\text{-}CPA}$ game against PKE for a fixed key pair (pk, sk). Combining the analyses from G_2 to $G_{2.7}$ so far, the following holds:

$$\begin{aligned}
\left|\Pr[G_2^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right| &= \left|\Pr[G_{2.4}^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right| \\
&\leq \left|\Pr[G_{2.4}^{\mathcal{A}} \Rightarrow 1] - \Pr[G_{2.4}^{\mathcal{A}} \Rightarrow 1 \wedge P^\dagger]\right| + \left|\Pr[G_{2.4}^{\mathcal{A}} \Rightarrow 1 \wedge P] - \frac{1}{2}\right| \\
&\leq \left|\Pr[G_{2.4}^{\mathcal{A}} \Rightarrow 1 \wedge P^\dagger] - \frac{1}{2}\right| + \varepsilon_{1,sk} \\
&\leq \left|\Pr[G_{2.4}^{\mathcal{A}} \Rightarrow 1 \wedge P^\dagger] - \Pr[G_{2.5}^{\mathcal{A}} \Rightarrow 1 \wedge P]\right| + \left|\Pr[G_{2.5}^{\mathcal{A}} \Rightarrow 1 \wedge P] - \frac{1}{2}\right| + \varepsilon_{1,sk} \\
&\leq \left|\Pr[G_{2.5}^{\mathcal{A}} \Rightarrow 1 \wedge P] - \frac{1}{2}\right| + \varepsilon_{1,sk} + \varepsilon_{2,sk}
\end{aligned}$$

$$
\begin{aligned}
&= \left|\Pr[G_{2.6}^{\mathcal{A}} \Rightarrow 1 \wedge P] - \frac{1}{2}\right| + \varepsilon_{1,sk} + \varepsilon_{2,sk} \\
&\leq \left|\Pr[G_{2.6}^{\mathcal{A}} \Rightarrow 1 \wedge P] - \Pr[G_{2.7}^{\mathcal{A}} \Rightarrow 1 \wedge P]\right| + \left|\Pr[G_{2.7}^{\mathcal{A}} \Rightarrow 1 \wedge P] - \frac{1}{2}\right| + \varepsilon_{1,sk} + \varepsilon_{2,sk} \\
&\leq \left|\Pr[G_{2.7}^{\mathcal{A}} \Rightarrow 1 \wedge P] - \frac{1}{2}\right| + \varepsilon_{1,sk} + \varepsilon_{2,sk} + \varepsilon_{3,sk} \\
&\leq \mathsf{Adv}_{\mathsf{PKE},sk}^{\mathsf{IND\text{-}CPA}}(\mathcal{E}) + \varepsilon_{sk},
\end{aligned} \tag{6}
$$

where $\varepsilon_{sk} = \varepsilon_{1,sk} + \varepsilon_{2,sk} + \varepsilon_{3,sk}$.

GAME $G_{3.1}$. $G_{3.1}$ is defined by modifying G_3, moving parts of the set of algorithms $\mathcal{C}^{\mathsf{H}} = (\mathcal{C}_0^{\mathsf{H}}, \mathcal{C}_1^{\mathsf{H}})$ to the game and the algorithm $\mathcal{F}_1^{\mathsf{H},\mathsf{Dec}''}$, as shown in Figure 20. Since this change is only conceptual, the following holds:

$$\Pr[G_3^{\mathcal{A}} \Rightarrow 1] = \Pr[G_{3.1}^{\mathcal{A}} \Rightarrow 1].$$

GAME $G_{3.2}$. $G_{3.2}$ is defined by modifying the generation of $\tilde{m}_b$, as shown in Figure 20. Since this change is only conceptual, the following holds:

$$\Pr[G_{3.1}^{\mathcal{A}} \Rightarrow 1] = \Pr[G_{3.2}^{\mathcal{A}} \Rightarrow 1].$$

GAME $G_{3.3}$. $G_{3.3}$ is defined by moving parts of the game into the algorithm $\mathcal{F}_0^{\mathsf{H},\mathsf{Dec}''}$, as defined in Figure 20. Since this change is only conceptual, the following holds:

$$\Pr[G_{3.2}^{\mathcal{A}} \Rightarrow 1] = \Pr[G_{3.3}^{\mathcal{A}} \Rightarrow 1].$$

GAME $G_{3.4}$. $G_{3.4}$ is defined by replacing the random oracle H with the extractable RO-simulator $\mathcal{S}$ for the relation $R_t := \{(x, y) \mid f(x, y) = t\}$, where $f(x, y) = \mathsf{Enc}'(pk, x; y)$ from Theorem 2.12, as shown in Figure 20. Furthermore, at the end of the game, the extractor interface $\mathcal{S}.E$ is invoked to compute $\hat{m}_i := \mathcal{S}.E(c_i)$ for each c_i that $\mathcal{A}$ queried to Dec'' during its run. According to the first statement of Theorem 2.12,

$$\Pr[G_{3.3}^{\mathcal{A}} \Rightarrow 1] = \Pr[G_{3.4}^{\mathcal{A}} \Rightarrow 1].$$

Furthermore, applying Theorem 2.13 for $R' := \{(m, c) : \mathsf{Dec}'(sk, c) \neq m\}$, the event

$$P^{\dagger} := [\forall i : \hat{m}_i = \tilde{m}_i' := \mathsf{Dec}'(sk, c_i) \vee \hat{m}_i = \emptyset]$$

holds except with probability $\varepsilon_{1,sk} := 128(q_{\mathsf{H}} + q_{\mathsf{D}})^2 \delta_{sk}$. Thus,

$$\left|\Pr[G_{3.4}^{\mathcal{A}} \Rightarrow 1] - \Pr[G_{3.4}^{\mathcal{A}} \Rightarrow 1 \wedge P^{\dagger}]\right| \leq \varepsilon_{1,sk}.$$

GAME $G_{3.5}$. $G_{3.5}$ is defined by moving each query $\mathcal{S}.E(c_i)$ to the end of the $\mathsf{Dec}''(c_i)$ oracle. Since $\mathcal{S}.RO(m)$ and $\mathcal{S}.E(c_i)$ now form consecutive classical queries, it follows from the contraposition of 4.(b) of Theorem 2.12 that, except with probability $2 \cdot 2^{-\ell_r}$, $\hat{m}_i = \emptyset$ implies $\mathsf{Enc}'(pk, m_i; \mathcal{S}.RO(m_i)) \neq c_i$. Applying the union bound, $P^{\dagger}$ implies

$$P := [\forall i : \hat{m}_i = m_i \vee (\hat{m}_i = \emptyset \wedge \mathsf{Enc}'(pk, m_i; \mathcal{S}.RO(m_i)) \neq c_i)]$$

<u>GAMES $G_{3.1}$-$G_{3.8}$</u>

$\mathsf{H} \leftarrow (\mathcal{M} \rightarrow \mathcal{R})$ //$G_{3.1}$-$G_{3.3}$
$\mathsf{H} = \mathcal{S}.RO$ //$G_{3.4}$-$G_{3.8}$
$(pk, sk) \leftarrow \mathsf{Gen}'(1^\lambda)$
$(m_0, m_1) \leftarrow \mathcal{A}_0^{\mathsf{H},\mathsf{Dec}''}(pk)$ //$G_{3.1}$-$G_{3.2}$
$(r_0, r_1) \leftarrow \{0,1\}^{\ell_r} \times \{0,1\}^{\ell_r}$ //$G_{3.2}$
$(\tilde{m}_0, \tilde{m}_1) \leftarrow \mathcal{F}_0^{\mathsf{H},\mathsf{Dec}''}(pk)$ //$G_{3.3}$-$G_{3.8}$
$b \leftarrow \{0,1\}$
$r_b \leftarrow \{0,1\}^{\ell_r}$ //$G_{3.1}$
$\tilde{m}_b = m_b || r_b$ //$G_{3.1}$-$G_{3.2}$
$\tilde{r} \leftarrow \mathcal{R}$
$c^* \leftarrow \mathsf{Enc}'(pk, \tilde{m}_b; \tilde{r})$
$\tilde{m}' \leftarrow \mathcal{F}_1^{\mathsf{H},\mathsf{Dec}''}(pk, c^*)$ //$G_{3.1}$-$G_{3.7}$
$b' \leftarrow \mathcal{G}_1^{\mathsf{H}}(pk, c^*)$ //$G_{3.8}$
return $[\![\tilde{m}_b = \tilde{m}']\!]$ //$G_{3.1}$-$G_{3.7}$
return $[\![b = b']\!]$ //$G_{3.8}$
while $i \in I$ **do** //$G_{3.4}$
    $\hat{m}_i \leftarrow \mathcal{S}.E(c_i)$ //$G_{3.4}$

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

$\tilde{m}' = \mathsf{Dec}'(sk, c)$ //$G_{3.1}$-$G_{3.6}$
$\tilde{r}' = \mathsf{H}(\tilde{m}')$ //$G_{3.1}$-$G_{3.6}$
if $c \neq \mathsf{Enc}'(pk, \tilde{m}'; \tilde{r}')$ //$G_{3.1}$-$G_{3.5}$
    return $\perp$ //$G_{3.1}$-$G_{3.5}$
else, return $[\![\tilde{m}']\!]_{\ell_m}$ //$G_{3.1}$-$G_{3.5}$
$\hat{m}' \leftarrow \mathcal{S}.E(c)$ //$G_{3.5}$-$G_{3.8}$
if $\hat{m}' = \perp$, **return** $\perp$ //$G_{3.6}$-$G_{3.8}$
else, return $[\![\hat{m}']\!]_{\ell_m}$ //$G_{3.6}$-$G_{3.8}$

<u>$\mathcal{F}_0^{\mathsf{H},\mathsf{Dec}''}(pk)$</u>

$(m_0, m_1) \leftarrow \mathcal{A}_0^{\mathsf{H},\mathsf{Dec}''}(pk)$
$(r_0, r_1) \leftarrow \{0,1\}^{\ell_r} \times \{0,1\}^{\ell_r}$
return $(m_0||r_0, m_1||r_1)$

<u>$\mathcal{F}_1^{\mathsf{H},\mathsf{Dec}''}(pk, c^*)$</u>

$i \leftarrow \{1, \cdots, q_{\mathsf{H}}\}$
Run $\mathcal{A}_1^{\mathsf{H},\mathsf{Dec}''}(r, \tilde{r})$ till i-th H-query
$\tilde{m}' \leftarrow$ measure i-th H-query
return $\tilde{m}'$

<u>$\mathcal{G}_1^{\mathsf{H}}(pk, c^*)$</u>

$\tilde{m}' \leftarrow \mathcal{F}_1^{\mathsf{H}}(pk, c^*)$
if $\tilde{m}_0 = \tilde{m}'$, **return** 0
else if $\tilde{m}_1 = \tilde{m}'$, **return** 1
else, return $b' \leftarrow \{0,1\}$

Figure 20: GAMES $G_{3.1}$-$G_{3.8}$ for the proof of Theorem 5.2

except with probability $q_{\mathsf{D}} \cdot 2 \cdot 2^{-\ell_r}$. Furthermore, by 2.(c) of Theorem 2.12, each swap of $\mathcal{S}.RO$ with $\mathcal{S}.E$ affects the final probability by at most $8\sqrt{2\Gamma(f)/|\mathcal{R}|} = 8\sqrt{2 \cdot 2^{-\gamma_{sk}}}$. Thus,

$$\left|\Pr[G_{3.4}^{\mathcal{A}} \Rightarrow 1 \wedge P^\dagger] - \Pr[G_{3.5}^{\mathcal{A}} \Rightarrow 1 \wedge P]\right| \leq \varepsilon_{2,sk}$$

with $\varepsilon_{2,sk} = 2q_{\mathsf{D}} \cdot ((q_{\mathsf{H}} + q_{\mathsf{D}}) \cdot 4\sqrt{2 \cdot 2^{-\gamma_{sk}}} + 2^{-\ell_r})$.

GAME $G_{3.6}$. In $G_{3.6}$, the Dec'' oracle uses $\hat{m}_i'$ instead of $\tilde{m}_i'$ to respond to the queries, but still queries $\mathcal{S}.RO(\tilde{m}_i')$, maintaining the interaction pattern from $G_{3.5}$.

Note that if the event

$$P_i := [\hat{m}_i' = m_i \vee (\hat{m}_i = \emptyset \wedge \mathsf{Enc}'(pk, m_i; \mathcal{S}.RO(m_i)) \neq c_i)]$$

holds for a given i, then the above change will not affect the response of Dec'' and thus will not affect the probability for P_{i+1} to hold as well. Thus, by mathematical induction,

$$\Pr[G_{3.5}^{\mathcal{A}} \Rightarrow 1 \wedge P] = \Pr[G_{3.6}^{\mathcal{A}} \Rightarrow 1 \wedge P].$$

GAME $G_{3.7}$. In $G_{3.7}$, all $\tilde{r}' = \mathsf{H}(\tilde{m}')$ queries in Dec'' are dropped or, equivalently, moved to the very end of the game execution. Invoking 2.(c) of Theorem 2.12, it holds that:

$$\left|\Pr[G_{3.6}^{\mathcal{A}} \Rightarrow 1 \wedge P] - \Pr[G_{3.7}^{\mathcal{A}} \Rightarrow 1 \wedge P]\right| \leq \varepsilon_{3,sk},$$

where $\varepsilon_{3,sk} = q_{\mathsf{D}} \cdot (q_{\mathsf{D}} + q_{\mathsf{H}}) \cdot 8\sqrt{2 \cdot 2^{-\gamma_{sk}}}$. Note that $G_{3.7}$ works without the secret key sk.

GAME $G_{3.8}$. $G_{3.8}$ is defined by constructing the adversary $\mathcal{G} = (\mathcal{F}_0, \mathcal{G}_1)$ from the adversary $\mathcal{F} = (\mathcal{F}_0, \mathcal{F}_1)$, as shown in Figure 20. The adversary $\mathcal{G}$ is now playing an IND-CPA game with PKE for a fixed key pair (pk, sk). Similar to the analysis in $G_{2.7}$, it holds that:

$$\left|\Pr[G_{3.8}^{\mathcal{A}} \Rightarrow 1 \wedge P] - \frac{1}{2}\right| = \mathsf{Adv}_{\mathsf{PKE},sk}^{\mathsf{IND\text{-}CPA}}(\mathcal{G}).$$

Also, since $G_{3.8} \Rightarrow 1$ holds if $G_{3.7} \Rightarrow 1$ hold, the following holds:

$$\begin{aligned}\Pr[G_{3.8} \Rightarrow 1 \wedge P] &= \Pr[G_{3.7} \Rightarrow 1 \wedge P] + \frac{1}{2}(1 - \Pr[G_{3.7} \Rightarrow 1 \wedge P]) \\ &= \frac{1}{2}\Pr[G_{3.7} \Rightarrow 1 \wedge P] + \frac{1}{2}.\end{aligned}$$

The above equality can be simplified as follows:

$$\Pr[G_{3.7} \Rightarrow 1 \wedge P] = 2\Pr[G_{3.8} \Rightarrow 1 \wedge P] - 1 \leq 2\mathsf{Adv}_{\mathsf{PKE},sk}^{\mathsf{IND\text{-}CPA}}(\mathcal{G}).$$

Combining the analyses from G_3 to $G_{3.8}$ so far, the following inequality holds:

$$\begin{aligned}\Pr[G_3^{\mathcal{A}} \Rightarrow 1] &= \Pr[G_{3.1}^{\mathcal{A}} \Rightarrow 1] = \Pr[G_{3.2}^{\mathcal{A}} \Rightarrow 1] = \Pr[G_{3.3}^{\mathcal{A}} \Rightarrow 1] = \Pr[G_{3.4}^{\mathcal{A}} \Rightarrow 1] \\ &\leq \Pr[G_{3.4}^{\mathcal{A}} \Rightarrow 1 \wedge P^{\dagger}] + \varepsilon_{1,sk} \\ &\leq \Pr[G_{3.5}^{\mathcal{A}} \Rightarrow 1 \wedge P] + \varepsilon_{2,sk} + \varepsilon_{1,sk} = \Pr[G_{3.6}^{\mathcal{A}} \Rightarrow 1 \wedge P] + \varepsilon_{2,sk} + \varepsilon_{1,sk} \\ &\leq \Pr[G_{3.7}^{\mathcal{A}} \Rightarrow 1 \wedge P] + \varepsilon_{3,sk} + \varepsilon_{2,sk} + \varepsilon_{1,sk} \\ &= 2\mathsf{Adv}_{\mathsf{PKE}}^{\mathsf{IND\text{-}CPA}}(\mathcal{G}) + \varepsilon_{sk}. \end{aligned} \tag{7}$$

The claimed bound is obtained by combining inequalities (5), (6), and (7) as follows and then taking the expectation over $(pk, sk) \leftarrow \mathsf{Gen}'(1^{\lambda})$:

$$\begin{aligned}\mathsf{Adv}_{\mathsf{PKE}',sk}^{\mathsf{IND\text{-}CPA}}(\mathcal{A}) &\leq 2 \cdot (q_{\mathsf{H}} + q_{\mathsf{D}})\sqrt{\Pr[G_3 \Rightarrow 1]} + (q_{\mathsf{H}} + q_{\mathsf{D}}) \cdot 2^{-\ell_r/2+2} + \left|\Pr[G_2^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right| \\ &\leq 2 \cdot (q_{\mathsf{H}} + q_{\mathsf{D}})\sqrt{2\mathsf{Adv}_{\mathsf{PKE},sk}^{\mathsf{IND\text{-}CPA}}(\mathcal{G}) + \varepsilon_{sk}} + (q_{\mathsf{H}} + q_{\mathsf{D}}) \cdot 2^{-\ell_r/2+2} + \mathsf{Adv}_{\mathsf{PKE},sk}^{\mathsf{IND\text{-}CPA}}(\mathcal{E}) + \varepsilon_{sk} \\ &\leq (2q_{\mathsf{H}} + 2q_{\mathsf{D}} + 1)\sqrt{2\mathsf{Adv}_{\mathsf{PKE},sk}^{\mathsf{IND\text{-}CPA}}(\mathcal{G}) + \varepsilon_{sk}} + (q_{\mathsf{H}} + q_{\mathsf{D}}) \cdot 2^{-\ell_r/2+2}.\end{aligned}$$

□

5.4 FO-Equivalent Transform Without Re-encryption

As in the case of $\overline{\mathsf{FO}}_{\mathsf{KEM}}^{\perp}$, we can show that $\mathsf{FO}_{\mathsf{PKE}}^{\perp}$ based on ACWC_2 can be identically converted into more efficient transform $\overline{\mathsf{FO}}_{\mathsf{PKE}}^{\perp}$ (shown in Figure 22), where the ciphertext comparison $c = \mathsf{Enc}'(pk, \tilde{m}'; R')$ in Dec'' is replaced with a simpler comparison of $r' = r''$. To do this, we first change Dec'' of Figure 14 into that of Figure 21, which are conceptually identical to each other. Next, we show that Dec'' of Figure 21 works equivalently to that of Figure 22 by proving the Lemma 5.3. As a result, the resulting schemes $\mathsf{FO}_{\mathsf{PKE}}^{\perp}[\mathsf{PKE}', \mathsf{H}]$ and $\overline{\mathsf{FO}}_{\mathsf{PKE}}^{\perp}[\mathsf{PKE}', \mathsf{H}]$ operates identically.

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

$M' = \mathsf{Dec}(sk, c)$
$r' = \mathsf{RRec}(pk, M', c)$
$\tilde{m}' = \mathsf{Inv}(M', \mathsf{G}(r'))$
$R' := \mathsf{H}(\tilde{m}')$
if $\tilde{m}' = \perp$ or $\boxed{r' \notin \mathcal{R} \text{ or } c \neq \mathsf{Enc}'(pk, \tilde{m}'; R')}$
  return $\perp$
else, return $[\tilde{m}']_{\ell_m}$

Figure 21: Modified $\mathsf{PKE}'' = \mathsf{FO}^{\perp}_{\mathsf{PKE}}[\mathsf{PKE}', \mathsf{H}]$

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

$M' = \mathsf{Dec}(sk, c)$
$r' = \mathsf{RRec}(pk, M', c)$
$\tilde{m}' = \mathsf{Inv}(M', \mathsf{G}(r'))$
$R' := \mathsf{H}(\tilde{m}')$
$\boxed{r'' \leftarrow \psi_{\mathcal{R}} \text{ with the randomness } R'}$
if $\tilde{m}' = \perp$ or $\boxed{r' \neq r''}$
  return $\perp$
else, return $[\tilde{m}']_{\ell_m}$

Figure 22: $\mathsf{PKE}'' = \overline{\mathsf{FO}}^{\perp}_{\mathsf{PKE}}[\mathsf{PKE}', \mathsf{H}]$

Lemma 5.3. Assume that the output of Dec in PKE always belongs to $\mathcal{M}$, PKE is injective in the injectivity game of Figure 2, and PKE and SOTP are rigid. Then, $r' \in \mathcal{R}$ and $c = \mathsf{Enc}'(pk, \tilde{m}'; R')$ in $\mathsf{FO}^{\perp}_{\mathsf{PKE}}$ holds if and only if $r' = r''$ in $\overline{\mathsf{FO}}^{\perp}_{\mathsf{PKE}}$ holds.

Proof. The proof is exactly the same as that of Lemma 4.3, except that $\tilde{m}$ is used instead of m. □

6 NTRU+

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

Figure 23 defines $\mathsf{GenNTRU}[\psi_1^n]$ relative to the distribution ψ_1^n over R_q. Since $\mathsf{GenNTRU}[\psi_1^n]$ should be MR and RR for our ACWC_2, Figure 23 shows two additional algorithms RRec and MRec. We notice that $\mathsf{RRec}(\mathbf{h}, \mathbf{m}, \mathbf{c})$ is necessary for performing ACWC_2 where $\mathbf{r}$ should be recovered from $\mathbf{c}$ once $\mathbf{m}$ is obtained. The RR property guarantees that such a randomness-recovery process works well, because for a ciphertext $\mathbf{c} = \mathsf{Enc}(\mathbf{h}, \mathbf{m}, \mathbf{r}) = \mathbf{hr} + \mathbf{m}$ we see that $\mathsf{RRec}(\mathbf{h}, \mathbf{m}, \mathbf{c}) = (\mathbf{c} - \mathbf{m})\mathbf{h}^{-1} = \mathbf{r} \in \mathcal{R}$. On the other hand, $\mathsf{MRec}(\mathbf{h}, \mathbf{r}, \mathbf{c})$ is only used for proving IND-CPA security of the ACWC_2-transformed scheme. The security analysis requires that for a challenge ciphertext $\mathbf{c}^* = \mathsf{Enc}(\mathbf{h}, \mathbf{m}^*, \mathbf{r}^*) = \mathbf{hr}^* + \mathbf{m}^*$ the algorithm $\mathsf{MRec}(\mathbf{h}, \mathbf{r}^*, \mathbf{c}^*)$ returns the corresponding message $\mathbf{m}^*$ if a queried $\mathbf{r}^*$ was used for $\mathbf{c}^*$. The MR property guarantees that once $\mathbf{r}^*$ is given, $\mathsf{MRec}(\mathbf{h}, \mathbf{r}^*, \mathbf{c}^*) = \mathbf{c}^* - \mathbf{hr}^* = \mathbf{m}^* \in \mathcal{M}$.

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

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

$\underline{\mathsf{Enc}(\mathbf{h}, \mathbf{m} \leftarrow \psi_1^n; \mathbf{r} \leftarrow \psi_1^n)}$

return $\mathbf{c} = \mathbf{hr} + \mathbf{m}$

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

return $\mathbf{m} = (\mathbf{cf} \bmod q) \bmod 3$

$\underline{\mathsf{RRec}(\mathbf{h}, \mathbf{m}, \mathbf{c})}$

return $\mathbf{r} = (\mathbf{c} - \mathbf{m})\mathbf{h}^{-1}$

$\underline{\mathsf{MRec}(\mathbf{h}, \mathbf{r}, \mathbf{c})}$

return $\mathbf{m} = \mathbf{c} - \mathbf{hr}$

Figure 23: $\mathsf{GenNTRU}[\psi_1^n]$ with average-case correctness error

6.1.1 Security Proofs

Theorem 6.1 (OW-CPA security of $\mathsf{GenNTRU}[\psi_1^n]$)**.** For any adversary $\mathcal{A}$, there exist adversaries $\mathcal{B}$ and $\mathcal{C}$ such that

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

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

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

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

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

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

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

Combining all the probabilities completes the proof. □

6.1.2 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 [30]. We can expand $\mathbf{cf}$ in the decryption algorithm as follows:

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

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

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

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

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

Then, assuming that each coefficient is independent,

$$\Pr\left[\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) \ne m\right] = 1 - \prod_{i=0}^{n-1} \epsilon_i. \tag{8}$$

±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 $c' = ab + a'b'$

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

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

Therefore,

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

Now, we analyze $\epsilon_i' = \Pr\left[|\mathbf{gr}+\mathbf{mf}'|_i \le \frac{q-3}{6}\right]$. To achieve this, we need to analyze the distribution of $\mathbf{gr}+\mathbf{mf}'$. By following the analysis in [30], we can check that for $i \in [n/2, n]$, the degree-i coefficient of $\mathbf{gr}+\mathbf{mf}'$ is the sum of n independent random variables:

$$c = ba + b'(a+a') \in \{0, \pm1, \pm2, \pm3\}, \text{ where } a, b, a, b \leftarrow \psi_1. \tag{9}$$

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 (9)), and $2i$ independent random variables c' of the form:

$$c' = ba + b'a' \in \{0, \pm1, \pm2\} \text{ where } a, b, a', b' \leftarrow \psi_1. \tag{10}$$

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

$$\rho_i(X) = \begin{cases} \sum_{j=-3n}^{3n} \rho_{i,j} X^j = \left(\sum_{j=-3}^{3} \theta_j X^j\right)^n \text{ for } i = [n/2, n-1], \\ \sum_{j=-(3n-2i)}^{3n-2i} \rho_{i,j} X^j = \left(\sum_{j=-3}^{3} \theta_j X^j\right)^{n-2i} \left(\sum_{j=-2}^{2} \theta_j' X^j\right)^{2i} \text{ for } i = [0, n/2-1], \end{cases} \tag{11}$$

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

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

where we used the symmetry $\rho_{i,j} = \rho_{i,-j}$. Putting ϵ_i' into Equation (8), we compute the average-case correctness error δ of $\mathsf{GenNTRU}[\psi_1^n]$.

6.1.3 Spreadness

Lemma 6.2 (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, $\mathbf{h}\mathbf{r}_1 + \mathbf{m} = \mathbf{h}\mathbf{r}_2 + \mathbf{m}$ holds. By subtracting $\mathbf{m}$ and multiplying $\mathbf{h}^{-1}$ on both sides of the equation, we obtain $\mathbf{r} = \mathbf{r}'$. Therefore, there exists at most one $\mathbf{r}$ such that $\mathbf{c} = \mathsf{Enc}(\mathbf{h}, \mathbf{m}; \mathbf{r})$.

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

6.1.4 Injectivity and rigidity

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

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

$(\mathbf{r}_1 - \mathbf{r}_2, \mathbf{m}_1 - \mathbf{m}_2)$ 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.

We can also easily check the rigidity of $\mathsf{GenNTRU}[\psi_1^n]$ as follows. For any $\mathbf{c} \in \mathcal{C} = R_q$ satisfying the two conditions $\mathbf{m}' = \mathsf{Dec}(\mathbf{f}, \mathbf{c}) \in \mathcal{M} = \{-1, 0, 1\}^n$ and $\mathbf{r}' = \mathsf{RRec}(\mathbf{h}, \mathbf{m}, \mathbf{c}) \in \mathcal{R} = \{-1, 0, 1\}^n$, the definition of RRec implies $\mathbf{r}' = (\mathbf{c} - \mathbf{m}')\mathbf{h}^{-1}$. Equivalently, the equality implies that $\mathbf{c} = \mathbf{h}\mathbf{r}' + \mathbf{m}' = \mathsf{Enc}(\mathbf{h}, \mathbf{m}'; \mathbf{r}')$ holds.

6.2 CPA-NTRU+ (=PKE ′)

6.2.1 Instantiation of SOTP

We introduce an instantiation of $\mathsf{SOTP} = (\mathsf{Encode}, \mathsf{Inv})$, where $\mathsf{Encode} : \mathcal{M}' \times \mathcal{U} \to \mathcal{M}$ and $\mathsf{Inv} : \mathcal{M} \times \mathcal{U} \to \mathcal{M}'$, 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 24, which is used for ACWC_2. We note that, following [27], the values of $y + u_2$ generated by Inv should be checked to determine whether they are 0 or 1.

$\mathsf{Encode}(x \in \mathcal{M}', u \leftarrow U^{2n})$	$\mathsf{Inv}(y \in \mathcal{M}, u \in U^{2n})$
$u = (u_1, u_2) \in \{0,1\}^n \times \{0,1\}^n$	$u = (u_1, u_2) \in \{0,1\}^n \times \{0,1\}^n$
$y = (x \oplus u_1) - u_2 \in \{-1, 0, 1\}^n$	**if** $y + u_2 \notin \{0,1\}^n$, **return** $\perp$
return y	$x = (y + u_2) \oplus u_1 \in \{0,1\}^n$
	return x

Figure 24: SOTP instantiation for NTRU+KEM

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 $y + u_2 \in \{0,1\}^n$.

6.2.2 CPA-NTRU+ (=PKE ′)

We obtain $\mathsf{CPA\text{-}NTRU+} := \mathsf{ACWC}_2$ [$\mathsf{GenNTRU}[\psi_1^n]$,$\mathsf{SOTP}$, G] by applying ACWC_2 from Section 3 to $\mathsf{GenNTRU}[\psi_1^n]$. Because the underlying $\mathsf{GenNTRU}[\psi_1^n]$ provides injectivity, MR, and RR properties, Theorems 3.5 and 3.6 provide us with the $\mathsf{IND\text{-}CPA}$ security of the resulting $\mathsf{CPA\text{-}NTRU+}$ in the classical and quantum random oracle models, respectively. Regarding the correctness error, Theorem 3.2 shows that the worst-case correctness error of $\mathsf{CPA\text{-}NTRU+}$ and the average-case correctness error of $\mathsf{GenNTRU}[\psi_1^n]$ differ by the amount of $\Delta = \|\psi_{\mathcal{R}}\| \cdot (1 + \sqrt{(\ln|\mathcal{M}'| - \ln\|\psi_{\mathcal{R}}\|)/2})$, 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 $\Delta = 2^{-1083}$.

$\underline{\mathsf{Gen}'(1^\lambda)}$	$\underline{\mathsf{Enc}'(pk, m \in \{0,1\}^n; R \leftarrow \{0,1\}^{2n})}$
1: $(pk, sk) := \mathsf{GenNTRU}[\psi_1^n].\mathsf{Gen}(1^\lambda)$	1: $\mathbf{r} \leftarrow \psi_1^n$ using the randomness R
- **repeat**	2: $\mathbf{m} = \mathsf{Encode}(m, \mathsf{G}(\mathbf{r}))$
- $\mathbf{f}', \mathbf{g} \leftarrow \psi_1^n$	3: $\mathbf{c} = \mathsf{GenNTRU}[\psi_1^n].\mathsf{Enc}(pk, \mathbf{m}; \mathbf{r})$
- $\mathbf{f} = 3\mathbf{f}' + 1$	- $\mathbf{c} = \mathbf{hr} + \mathbf{m}$
- **until** $\mathbf{f}$ is invertible in R_q	4: **return** $\mathbf{c}$
- **repeat**	$\underline{\mathsf{Dec}'(sk, \mathbf{c})}$
- $\mathbf{g} \leftarrow \psi_1^n$	1: $\mathbf{m} = \mathsf{GenNTRU}[\psi_1^n].\mathsf{Dec}(sk, \mathbf{c})$
- **until** $\mathbf{g}$ is invertible in R_q	- $\mathbf{m} = (\mathbf{cf} \bmod q) \bmod 3$
- $(pk, sk) = (\mathbf{h} = 3\mathbf{g}\mathbf{f}^{-1} \bmod q, \mathbf{f})$	2: $\mathbf{r} = \mathsf{RRec}(pk, \mathbf{c}, \mathbf{m})$
2: **return** (pk, sk)	- $\mathbf{r} = (\mathbf{c} - \mathbf{m})\mathbf{h}^{-1}$
	3: $m = \mathsf{Inv}(\mathbf{m}, \mathsf{G}(\mathbf{r}))$
	4: **if** $m = \perp$ or $\mathbf{r} \notin \{-1,0,1\}^n$, **return** $\perp$
	5: **return** m

Figure 25: CPA-NTRU+

Spreadness Properties of $\mathsf{CPA\text{-}NTRU+}$**.** To achieve $\mathsf{IND\text{-}CCA}$ security of the KEM and PKE via $\overline{\mathsf{FO}}^{\perp}_{\mathsf{KEM}}$ and $\overline{\mathsf{FO}}^{\perp}_{\mathsf{PKE}}$, we need to show the spreadness of $\mathsf{CPA\text{-}NTRU+}$. The spreadness can be easily obtained by combining Lemma 3.7 with Lemma 6.2.

6.3 NTRU+KEM

Finally, we achieve $\mathsf{IND\text{-}CCA}$ secure KEM by applying $\overline{\mathsf{FO}}^{\perp}_{\mathsf{KEM}}$ to $\mathsf{CPA\text{-}NTRU+}$. We denote such KEM by $\mathsf{NTRU+KEM} := \overline{\mathsf{FO}}^{\perp}_{\mathsf{KEM}}[\mathsf{CPA\text{-}NTRU+}, \mathsf{H}_{\mathsf{KEM}}]$. Figure 26 shows the resultant $\mathsf{NTRU+KEM}$, which is the basis of our implementation in the next section. By combining Theorems 4.1, 4.2, and Lemma 4.3, we can achieve $\mathsf{IND\text{-}CCA}$ security of $\mathsf{NTRU+KEM}$. As for the correctness error, $\mathsf{NTRU+KEM}$ preserves the worst-case correctness error of the underlying $\mathsf{CPA\text{-}NTRU+}$.

6.4 NTRU+PKE

Finally, we achieve $\mathsf{IND\text{-}CCA}$ secure PKE by applying $\overline{\mathsf{FO}}^{\perp}_{\mathsf{PKE}}$ to $\mathsf{CPA\text{-}NTRU+}$. We denote such PKE by $\mathsf{NTRU{+}PKE} := \overline{\mathsf{FO}}^{\perp}_{\mathsf{KEM}}[\mathsf{CPA\text{-}NTRU+}, \mathsf{H}_{\mathsf{PKE}}]$. Figure 27 shows the resultant $\mathsf{NTRU{+}PKE}$, which is the basis of our implementation in the next section. By combining Theorems 5.1, 5.2, and Lemma 5.3, we can achieve $\mathsf{IND\text{-}CCA}$ security of $\mathsf{NTRU{+}PKE}$. As in $\mathsf{NTRU{+}KEM}$, $\mathsf{NTRU{+}PKE}$ preserves the worst-case correctness error of the underlying $\mathsf{CPA\text{-}NTRU+}$.

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

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

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

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

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

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

Figure 26: NTRU+KEM

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

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

$\underline{\mathsf{Enc}(pk, m \in \{0,1\}^{\ell_m})}$

$r \leftarrow \{0,1\}^{\ell_r}$
$\tilde{m} = m || r \in \{0,1\}^{n = \ell_m + \ell_r}$
$R = \mathsf{H}_{\mathsf{PKE}}(\tilde{m})$
$\mathbf{r} \leftarrow \psi_1^n$ using the randomness R
$\mathbf{m} = \mathsf{Encode}(\tilde{m}, \mathsf{G}(\mathbf{r}))$
$\mathbf{c} = \mathbf{h}\mathbf{r} + \mathbf{m}$
return c

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

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

Figure 27: NTRU+PKE

7 Algorithm Specification

7.1 Preliminaries and notation

Symmetric primitives. $\mathsf{NTRU+}\{\mathsf{KEM},\mathsf{PKE}\}$ use four different hash functions: F, G, $\mathsf{H}_{\mathsf{KEM}}$, and $\mathsf{H}_{\mathsf{PKE}}$. We instantiate these functions with $\mathsf{SHA256}$ and $\mathsf{SHAKE256}$ as described in Algorithms 1, 2, 3, and 4. We also use $\mathsf{SHAKE256}$ as an extendable output function (XOF).

Algorithm 1: F

Require: Byte array $m = (m_0, m_1, \cdots, m_{3n/2-1})$
Ensure: Byte array $B = (b_0, b_1, \cdots, b_{31})$
1: $(b_0, \cdots, b_{31}) := \mathsf{SHA256}(0\text{x}00 || m)$;
2: **return** $(b_0, \cdots b_{31})$

Algorithm 2: G

Require: Byte array $m = (m_0, m_1, \cdots, m_{n/8-1})$
Ensure: Byte array $B = (b_0, b_1, \cdots, b_{n/8+31})$
1: $(b_0, \cdots b_{n/4-1}) := \mathsf{SHAKE256}(0\text{x}01 || m, n/4)$;
2: **return** $(b_0, \cdots b_{n/4-1})$

Algorithm 3: $\mathsf{H}_{\mathsf{KEM}}$

Require: Byte array $m = (m_0, m_1, \cdots, m_{n/8-1})$
Ensure: Byte array $B = (b_0, b_1, \cdots, b_{n/4+31})$
1: $(b_0, \cdots b_{n/4+31}) := \mathsf{SHAKE256}(0\text{x}02 || m, n/4 + 32)$;
2: **return** $(b_0, \cdots b_{n/4+31})$

Algorithm 4: $\mathsf{H}_{\mathsf{PKE}}$

Require: Byte array $m = (m_0, m_1, \cdots, m_{n/8-1})$
Ensure: Byte array $B = (b_0, b_1, \cdots, b_{n/8+31})$
1: $(b_0, \cdots, b_{n/4-1}) := \mathsf{SHAKE256}(0\text{x}03 || m, n/4)$;
2: **return** $(b_0, \cdots b_{n/4-1})$

Sampling from a Binomial distribution. $\mathsf{NTRU+}\{\mathsf{KEM},\mathsf{PKE}\}$ use a centered binomial distribution with $\eta = 1$ for sampling the coefficients of polynomials, as defined in Algorithm 5. Additionally, we introduce the $\mathsf{BytesToBits}$ function in Algorithm 6, which determines the order of sampled coefficients. $\mathsf{BytesToBits}$ plays a crucial role in the efficient implementation of CBD_1 and SOTP using AVX2 instructions. We also define $\mathsf{BitsToBytes}$ as the inverse function of $\mathsf{BytesToBits}$.

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

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

Algorithm 6: BytesToBits

Require: Byte array $B = (b_0, b_1, \cdots, b_{n/8-1}) \in \mathcal{B}^{n/8}$
Ensure: Bit array $f = (f_0, \cdots, f_{n-1}) \in \{0,1\}^n$

$s = \lfloor n/256 \rfloor$
$r = n - 256s$
$(r_0, r_1, r_2, r_4, r_5, r_6, r_7) := \textsf{bit-decompose}(r)$ // $r = r_0 2^0 + \cdots r_7 2^7$
for i from 0 to $s-1$ **do**
  for j from 0 to 7 **do**
    $t = b_{32i+4j+3} | b_{32i+4j+2} | b_{32i+4j+1} | b_{32i+4j}$
    for k from 0 to 1 **do**
      for l from 0 to 15 **do**
        $f_{256i+16l+2j+k} = t \& 1;$
        $t = t >> 1;$
$c_1 = 256s$, $c_2 = 32s$
if $r_7 = 1$
  for j from 0 to 3 **do**
    $t = b_{c_2+4j+3} | b_{c_2+4j+2} | b_{c_2+4j+1} | b_{c_2+4j}$
    for k from 0 to 1 **do**
      for l from 0 to 16 **do**
        $f_{c_1+8l+2j+k} = t \& 1;$
        $t = t >> 1;$
$c_1 = c_1 + 128r_7$, $c_2 = c_2 + 16r_7$
if $r_6 = 1$
  for j from 0 to 1 **do**
    $t = b_{c_2+4j+3} | b_{c_2+4j+2} | b_{c_2+4j+1} | b_{c_2+4j}$
    for k from 0 to 1 **do**
      for l from 0 to 15 **do**
        $f_{c_1+4l+2j+k} = t \& 1;$
        $t = t >> 1;$
$c_1 = c_1 + 64r_6$, $c_2 = c_2 + 8r_6$
if $r_5 = 1$
  $t = b_{c_2+3} | b_{c_2+2} | b_{c_2+1} | b_{c_2}$
  for k from 0 to 1 **do**
    for l from 0 to 15 **do**
      $f_{c_1+2l+k} = t \& 1;$
      $t = t >> 1;$
return $f = (f_0, \cdots, f_{n-1})$

Semi-generalized one time pad The Encode function of $\mathsf{SOTP} = (\mathsf{Encode}, \mathsf{Inv})$ is nearly identical to CBD_1, differing only in that it applies an exclusive OR operation to the first half of the random bytes and the message before sampling from the centered binomial distribution. Consequently, Encode, as defined in Algorithm 7, also utilizes the BytesToBits function, just like CBD_1. Additionally, we introduce the Inv function in Algorithm 8, which serves as the inverse of the Encode function and utilizes the BitsToBytes function for byte recovery.

Algorithm 7: 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 $\mathbf{f} \in R_q$
$(\beta_0, \cdots, \beta_{n-1}) := \mathsf{BytesToBits}((b_0, \cdots, b_{n/8-1}))$
$(\beta_n, \cdots, \beta_{2n-1}) := \mathsf{BytesToBits}((b_{n/8}, \cdots, b_{n/4-1}))$
$(m_0, \cdots, m_{n-1}) := \mathsf{BytesToBits}(m)$
for i from 0 to $n-1$ **do**
    $f_i := (m_i \oplus \beta_i) - \beta_{i+n}$
return $\mathbf{f} = f_0 + f_1 x + f_2 x^2 + \cdots + f_{n-1} x^{n-1}$

Algorithm 8: Inv

Require: Polynomial $\mathbf{f} \in R_q$
Require: Byte array $B = (b_0, b_1, \cdots, b_{n/4-1})$
Ensure: Message Byte array $m = (m_0, m_1, \cdots, m_{31})$
$(\beta_0, \cdots, \beta_{n-1}) := \mathsf{BytesToBits}((b_0, \cdots, b_{n/8-1}))$
$(\beta_n, \cdots, \beta_{2n-1}) := \mathsf{BytesToBits}((b_{n/8}, \cdots, b_{n/4-1}))$
for i from 0 to $n-1$ **do**
    if $f_i + \beta_{i+n} \notin \{0, 1\}$, **return** $\perp$ // Refer to line 8 in Algorithm 17
    $m_i := ((f_i + \beta_{i+n}) \& 1) \oplus \beta_i$
$m = \mathsf{BitsToBytes}((m_0, \cdots, m_{n-1}))$
return m

Encoding and Decoding. We introduce the Encode_m function in Algorithm 9 to encode a byte array with a length equal or less than $\ell_m - 1$ to a byte array with length ℓ_m. Additionally, the Decode_m function, defined in Algorithm 10, serves as the inverse of Encode_m.

Algorithm 9: Encode_m

Require: Byte array $B = (b_0, \cdots, b_{\ell-1}) \in \mathcal{B}^{\ell}$
Ensure: Byte array $B' = (b_0, \cdots, b_{\ell_m-1}) \in \mathcal{B}^{\ell_m}$
if $\ell_m - 1 < \ell$, **return** $\perp$
return $B' = (\underbrace{b_0, \cdots, b_{\ell-1}}_{\ell \text{ bytes}}, \text{0xff}, \underbrace{0, \quad \cdots \quad, 0}_{\ell_m-\ell-1 \text{ bytes}})$

Algorithm 10: Decode_m

Require: Byte array $B = (b_0, \cdots, b_{\ell_m-1}) \in \mathcal{B}^{\ell_m}$
Ensure: Byte array $B' = (b'_0, \cdots, b'_{\ell-1}) \in \mathcal{B}^{\ell}$
for $i = \ell_m - 1; i \geq 0; i$-- **do**
    if $b_i = 0$, continue;
    else if $b_i =$ 0xff, $\ell = i$ break;
    else, return $\perp$
if $i = -1$, **return** $\perp$
return $B' = (b'_0, \cdots, b'_{\ell-1}) = (b_0, \cdots, b_{\ell-1})$

To encode polynomials in R_q into a $3n/2$ byte array, we introduce the Encode_q function in Algorithms 11 and 12. This function assumes each coefficient of the polynomial belongs to $\{0, \ldots, q-1\}$ and is stored as a 16-bit datum. Additionally, we define the Decode_q function in Algorithms 13 and 14 as the inverse of Encode_q. max_j in Algorithm 11 and 13 is defined as $max_j = 8$ for NTRU+{KEM, PKE}576, $max_j = 11$ for NTRU+{KEM, PKE}768, and $max_j = 17$ for NTRU+{KEM, PKE}1152.

Algorithm 11: Encode_q
for NTRU+{KEM, PKE}576, NTRU+{KEM, PKE}768, and NTRU+{KEM, PKE}1152

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

for i from 0 to 15 **do**
  for j from 0 to max_j **do**
    for k from 0 to 3 **do**
      $t_k = f_{64j+i+16k}$
    $b_{96j+2i} = t_0$
    $b_{96j+2i+1} = (t_0 >> 8) + (t_1 << 4)$
    $b_{96j+2i+32} = t_1 >> 4$
    $b_{96j+2i+33} = t_2$
    $b_{96j+2i+64} = (t_2 >> 8) + (t_3 << 4)$
    $b_{96j+2i+65} = t_3 >> 4$
return $(b_0, \cdots, b_{3n/2-1})$

Algorithm 12: Encode_q for NTRU+{KEM, PKE}864

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

for i from 0 to 15 **do**
  for j from 0 to 12 **do**
    for k from 0 to 3 **do**
      $t_k = f_{64j+i+16k}$
    $b_{96j+2i} = t_0$
    $b_{96j+2i+1} = (t_0 >> 8) + (t_1 << 4)$
    $b_{96j+2i+32} = t_1 >> 4$
    $b_{96j+2i+33} = t_2$
    $b_{96j+2i+64} = (t_2 >> 8) + (t_3 << 4)$
    $b_{96j+2i+65} = t_3 >> 4$
for i from 0 to 7 **do**
  for k from 0 to 3 **do**
    $t_k = f_{832+i+8k}$
  $b_{1248+2i} = t_0$
  $b_{1248+2i+1} = (t_0 >> 8) + (t_1 << 4)$
  $b_{1248+2i+16} = t_1 >> 4$
  $b_{1248+2i+17} = t_2$
  $b_{1248+2i+32} = (t_2 >> 8) + (t_3 << 4)$
  $b_{1248+2i+33} = t_3 >> 4$
return $(b_0, \cdots, b_{3n/2-1})$

Algorithm 13: Decode_q
for $\mathsf{NTRU+}\{\mathsf{KEM},\mathsf{PKE}\}576$, $\mathsf{NTRU+}\{\mathsf{KEM},\mathsf{PKE}\}768$, and $\mathsf{NTRU+}\{\mathsf{KEM},\mathsf{PKE}\}1152$

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

for i from 0 to 15 **do**
  for j from 0 to max_j **do**
    $t_0 = b_{96j+2i}$
    $t_1 = b_{96j+2i+1}$
    $t_2 = b_{96j+2i+32}$
    $t_3 = b_{96j+2i+33}$
    $t_4 = b_{96j+2i+64}$
    $t_5 = b_{96j+2i+65}$
    $f_{64j+i} = t_0|(t_1\&0\text{xf}) << 8$
    $f_{64j+i+16} = t_1 >> 4|t_2 << 4$
    $f_{64j+i+32} = t_3|(t_4\&0\text{xf}) << 8$
    $f_{64j+i+48} = t_4 >> 4|t_5 << 4$
return $\mathbf{f} = (f_0, \cdots, f_{n-1})$

Algorithm 14: Decode_q for $\mathsf{NTRU+}\{\mathsf{KEM},\mathsf{PKE}\}864$

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

for i from 0 to 15 **do**
  for j from 0 to 12 **do**
    $t_0 = b_{96j+2i}$
    $t_1 = b_{96j+2i+1}$
    $t_2 = b_{96j+2i+32}$
    $t_3 = b_{96j+2i+33}$
    $t_4 = b_{96j+2i+64}$
    $t_5 = b_{96j+2i+65}$
    $f_{64j+i} = t_0|(t_1\&0\text{xf}) << 8$
    $f_{64j+i+16} = t_1 >> 4|t_2 << 4$
    $f_{64j+i+32} = t_3|(t_4\&0\text{xf}) << 8$
    $f_{64j+i+48} = t_4 >> 4|t_5 << 4$
for i from 0 to 15 **do**
  $t_0 = b_{1248+2i}$
  $t_1 = b_{1248+2i+1}$
  $t_2 = b_{1248+2i+16}$
  $t_3 = b_{1248+2i+17}$
  $t_4 = b_{1248+2i+32}$
  $t_5 = b_{1248+2i+33}$
  $f_{832+i} = t_0|(t_1\&0\text{xf}) << 8$
  $f_{832+i+8} = t_1 >> 4|t_2 << 4$
  $f_{832+i+16} = t_3|(t_4\&0\text{xf}) << 8$
  $f_{832+i+24} = t_4 >> 4|t_5 << 4$
return $\mathbf{f} = (f_0, \cdots, f_{n-1})$

- NTRU+{KEM, PKE}576

```
index[144] = {1, 217, 109, 325, 55, 271, 163, 379, 19, 235, 127, 343, 73, 289,
181, 397, 37, 253, 145, 361, 91, 307, 199, 415, 7, 223, 115, 331, 61, 277, 169,
385, 25, 241, 133, 349, 79, 295, 187, 403, 43, 259, 151, 367, 97, 313, 205, 421,
13, 229, 121, 337, 67, 283, 175, 391, 31, 247, 139, 355, 85, 301, 193, 409, 49,
265, 157, 373, 103, 319, 211, 427, 5, 221, 113, 329, 59, 275, 167, 383, 23, 239,
131, 347, 77, 293, 185, 401, 41, 257, 149, 365, 95, 311, 203, 419, 11, 227, 119,
335, 65, 281, 173, 389, 29, 245, 137, 353, 83, 299, 191, 407, 47, 263, 155, 371,
101, 317, 209, 425, 17, 233, 125, 341, 71, 287, 179, 395, 35, 251, 143, 359, 89,
305, 197, 413, 53, 269, 161, 377, 107, 323, 215, 431};
```

- NTRU+{KEM, PKE}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+{KEM, PKE}864 and NTRU+{KEM, PKE}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 28: Index for the NTT

n	q	Radix-2 for cyclotomic trinomial	Radix-3	Radix-2	d	ζ	$\ell = 3n/d$
576	3457	1	2	3	4	81	432
768	3457	1	1	5	4	22	576
864	3457	1	2	4	3	9	864
1152	3457	1	2	4	4	9	864

ζ : primitive ℓ-th root of unity modulo q

Table 5: Combinations of NTT layers

Polynomial rings and Number Theoretic Transform. We define two quotient rings: $R = \mathbb{Z}[x]/\langle x^n - x^{n/2} + 1\rangle$ and $R_q = \mathbb{Z}_q[x]/\langle x^n - x^{n/2} + 1\rangle$, where $n = 2^a 3^b$ with $a, b \in \mathbb{N} \cup \{0\}$ such that $x^n - x^{n/2} + 1$ is the $3n$-th cyclotomic polynomial. To efficiently perform computations within the ring R_q, we reduce the computations to the product of smaller rings, denoted as $\prod_{i=0}^{n/d-1} \mathbb{Z}_q[x]/\langle x^d - \zeta_i\rangle$, using the Number Theoretic Transform (NTT). To implement NTT efficiently, we combine three different NTT layers in the following sequence: Radix-2 NTT layer for the cyclotomic trinomial, Radix-3 NTT layer, and then Radix-2 NTT layer. We choose to use Radix-3 NTT layers before Radix-2 NTT layers to minimize the size of pre-computation table. The initial Radix-2 NTT layer for the cyclotomic trinomial, as introduced by [30], establishes a ring isomorphism from $\mathbb{Z}_q[x]/\langle x^n - x^{n/2} + 1\rangle$ to the product ring $\mathbb{Z}_q[x]/\langle x^{n/2} - \zeta\rangle \times \mathbb{Z}_q[x]/\langle x^{n/2} - \zeta^5\rangle$, where ζ denotes a primitive sixth root of unity modulo q. Subsequently, we use Radix-3 NTT layers to establish isomorphisms from $\mathbb{Z}_q[x]/\langle x^n - \alpha^3\rangle$ to the product ring $\mathbb{Z}_q[x]/\langle x^{n/3} - \alpha\rangle \times \mathbb{Z}_q[x]/\langle x^{n/3} - \alpha\omega\rangle \times \mathbb{Z}_q[x]/\langle x^{n/3} - \alpha\omega^2\rangle$, where ω denotes a primitive third root of unity modulo q. In the final step, we use Radix-2 NTT layers to establish isomorphisms from $\mathbb{Z}_q[x]/\langle x^n - \zeta^2\rangle$ to the product ring $\mathbb{Z}_q[x]/\langle x^{n/2} - \zeta\rangle \times \mathbb{Z}_q[x]/\langle x^{n/2} + \zeta\rangle$. Table 5 presents comprehensive information, including the number of applied NTT layers and the resulting degree d of component rings in the product rings for various parameter sets. Note that, for the successful implementation of NTT, it requires a primitive ℓ-th root of unity ζ modulo q, where $\ell = 3n/d$. The values of ℓ and ζ for each parameter are also included in Table 5.

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

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

where the array `index` is defined in Figure 28. In this document, we denote NTT as the number theoretic transform function and NTT^{-1} as the inverse number theoretic transform function.

Multiplication in the NTT domain. After transforming polynomials in R_q into elements of the product rings, multiplication is performed within each component ring $\mathbb{Z}_q[x]/\langle x^d - \zeta_i\rangle$. 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 $\mathbb{Z}_q[x]/\langle x^d - \zeta_i\rangle$.

For $d = 2$, the product $a(x)b(x)$ is computed as follows:

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

which can be represented in matrix form as:

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

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

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

In matrix form, this is equivalent to:

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

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

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

The corresponding matrix form is:

$$c(x) = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3\zeta_i & a_2\zeta_i & a_1\zeta_i \\ a_1 & a_0 & a_3\zeta_i & a_2\zeta_i \\ a_2 & a_1 & a_0 & a_3\zeta_i \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}.$$

Inversion in the NTT domain. In the NTT domain, inversion is performed within each component ring $\mathbb{Z}_q[x]/\langle x^d - \zeta_i\rangle$, similar to multiplication. To find the inverse $b(x) = \sum_{j=0}^{d-1} b_jx^j$ of a polynomial $a(x) = \sum_{j=0}^{d-1} a_jx^j \in \mathbb{Z}_q[x]/\langle x^d - \zeta_i\rangle$, we use matrix representations.

For $d = 2$, the inverse $b(x)$ is computed as:

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

For $d = 3$, the inverse $b(x)$ is:

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

where

$$a_0' = a_0^2 - \zeta_i a_1 a_2, \qquad a_1' = \zeta_i a_2^2 - a_0 a_1, \qquad a_2' = a_1^2 - a_0 a_2$$

and

$$d = a_0(a_0^2 - \zeta_i a_1 a_2) + \zeta_i a_1(a_1^2 - a_0 a_2) + \zeta_i a_2(\zeta_i a_2^2 - a_0 a_1) = a_0 a_0' + \zeta_i(a_1 a_2' + a_2 a_1').$$

For $d = 4$, finding the inverse of $a(x)$ through matrix inversion is more complex. Instead, we follow the method in [38], reducing the problem of inversion in the ring $\mathbb{Z}_q[x]/\langle x^4 - \zeta_i \rangle$ to inversion in $\mathbb{Z}_q[z]/\langle z^2 - \zeta_i \rangle$, where $z = x^2$. Thus, $a(x) \in \mathbb{Z}_q[y]/\langle x^4 - \zeta_i \rangle$ is rewritten as:

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

The product of $a(x) = \hat{a}_0(z) + \hat{a}_1(z)x$ and $b(x) = \hat{b}_0(z) + \hat{b}_1(z)x$ is:

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

which can be expressed in matrix form as:

$$c(x) = \begin{bmatrix} \hat{c}_0 \\ \hat{c}_1 \end{bmatrix} = \begin{bmatrix} \hat{a}_0(z) & \hat{a}_1(z)z \\ \hat{a}_1(z) & \hat{a}_0(z) \end{bmatrix} \begin{bmatrix} b_0(z) \\ b_1(z) \end{bmatrix}.$$

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

$$\begin{aligned} \begin{bmatrix} \hat{a}_0(z) \\ \hat{a}_1(z) \end{bmatrix} = \begin{bmatrix} \hat{a}_0(z) & \hat{a}_1(z)z \\ \hat{a}_1(z) & \hat{a}_0(z) \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} &= \frac{1}{\hat{a}_0^2(z) - \hat{a}_1^2(z)z} \begin{bmatrix} \hat{a}_0(z) & -\hat{a}_1(z)z \\ -\hat{a}_1(z) & \hat{a}_0(z) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ &= \frac{1}{a_0^2(z) - a_1^2(z)z} \begin{bmatrix} a_0(z) \\ -a_1(z) \end{bmatrix} \in \mathbb{Z}_q[z]/\langle z^2 - \zeta_i \rangle^{1\times 2}. \end{aligned}$$

The inverse of $\hat{a}_0^2(z) - \hat{a}_1^2(z)z \in \mathbb{Z}_q[z]/\langle z^2 - \zeta_i \rangle$ can be computed using the case of $d = 2$. After performing the necessary operations in $\mathbb{Z}_q[z]/\langle z^2 - \zeta_i \rangle$, the final result is obtained by substituting $z = x^2$.

In all cases, we need to compute the multiplicative inverse modulo q. To mitigate the risk of side-channel attacks, we opt for Fermat's Little Theorem rather than the extended Euclidean algorithm. Fermat's Little Theorem states that if a is coprime with q, then $a^{q-1} \equiv 1 \pmod{q}$. Using this theorem, we can compute the inverse of a by calculating $a^{q-2} \bmod q$.

7.2 Specification of NTRU+

7.2.1 NTRU+KEM

We describe our $\mathsf{NTRU{+}KEM}$. Unlike $\mathsf{NTRU{+}KEM}$ in section 6.3, we apply a slightly tweaked $\overline{\mathsf{FO}}^{\perp}_{\mathsf{KEM}}$ to resist the multi-target attacks. Algorithms 15, 16, and 17 define the key generation, encapsulation, and decapsulation of $\mathsf{NTRU{+}KEM}$. Note that, in the key generation algorithm, we multiply $\hat{\mathbf{h}}$ and $\hat{\mathbf{h}}^{-1}$ by 2^{16} to account for the Montgomery reduction.

Algorithm 15: $\mathsf{Gen}(1^{\lambda})$: key generation

Ensure: Public key $pk \in \mathcal{B}^{\lceil \log_2 q \rceil \cdot n/8}$
Ensure: Secret key $sk \in \mathcal{B}^{\lceil \log_2 q \rceil \cdot n/4}$

```
repeat
    d ← B^32
    f := XOF(d, n/4)
    f' := CBD_1(f)
    f := 3f' + 1
    f̂ := NTT(f)
until f is invertible in R_q
repeat
    d ← B^32
    g := XOF(d, n/4)
    g' := CBD_1(g)
    g := 3g'
    ĝ := NTT(g)
until g is invertible in R_q
ĥ := ĝ ∘ f̂^{-1}
pk := Encode_q(2^16 · ĥ)
sk := Encode_q(f̂)||Encode_q(2^16 · ĥ^{-1})||F(pk)
return (pk, sk)
```

Algorithm 16: $\mathsf{Encap}(pk)$: 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 ← B^{n/8}
(K, r) := H(m, F(pk))
r := CBD_1(r)
r̂ = NTT(r)
m = Encode(m, G(Encode_q(r̂)))
m̂ = NTT(m)
2^16 · ĥ := Decode_q(pk)
ĉ = ĥ ∘ r̂ + m̂
c := Encode_q(ĉ)
return (c, K)
```

Algorithm 17: $\mathsf{Decap}(sk, c)$: decapsulation

Require: Secret key $sk \in \mathcal{B}^{\lceil \log_2 q \rceil \cdot n/4+32}$
Require: Ciphertext $c \in \mathcal{B}^{\lceil \log_2 q \rceil \cdot n/8}$
Ensure: Shared key $m \in \mathcal{B}^{32}$
1: Parse $sk = (sk_1, sk_2, sk_3) \in \mathcal{B}^{\lceil \log_2 q \rceil \cdot n/8} \times \mathcal{B}^{\lceil \log_2 q \rceil \cdot n/8} \times \mathcal{B}^{32}$
2: $\hat{\mathbf{f}} = \mathsf{Decode}_q(sk_1)$
3: $\hat{\mathbf{c}} = \mathsf{Decode}_q(c)$
4: $\mathbf{m} = \mathsf{NTT}^{-1}(\hat{\mathbf{c}} \circ \hat{\mathbf{f}}) \mod {}^{\pm}3$
5: $\hat{\mathbf{m}} = \mathsf{NTT}(\mathbf{m})$
6: $2^{16} \cdot \hat{\mathbf{h}}^{-1} = \mathsf{Decode}_q(sk_2)$
7: $\hat{\mathbf{r}} = (\hat{\mathbf{c}} - \hat{\mathbf{m}}) \circ \hat{\mathbf{h}}^{-1}$ // RRec
8: $m' := \mathsf{Inv}(\mathbf{m}, \mathsf{G}(\mathsf{Encode}_q(\hat{\mathbf{r}})))$ // Checking if $m' = \perp$ is done 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'

7.2.2 NTRU+ PKE

Finally, we specify our NTRU+PKE for the KpqC competition. As in NTRU+KEM, we apply a slightly tweaked $\overline{\mathsf{FO}}^{\perp}_{\mathsf{PKE}}$ in order to resist the multi-target attacks. Algorithms 18, 19, and 20 define the key generation, encryption, and decryption of NTRU+PKE, respectively.

Algorithm 18: $\mathsf{Gen}(1^\lambda)$: key generation

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

Algorithm 19: $\mathsf{Enc}(pk, m)$: encryption

Require: Public key $pk \in \mathcal{B}^{\lceil \log_2 q \rceil \cdot n/8}$
Require: Message $m \in \mathcal{B}^{\leq \ell_m - 1}$
Ensure: Ciphertext $c \in \mathcal{B}^{\lceil \log_2 q \rceil \cdot n/8}$

$m = \mathsf{Encode}_m(m) \in \mathcal{B}^{\ell_m}$
$r \leftarrow \mathcal{B}^{\ell_r}$
$\tilde{m} = m || r \in \mathcal{B}^{n/8}$ // $n/8 = \ell_m + \ell_r$
$r := \mathsf{H}_{\mathsf{PKE}}(\tilde{m}, \mathsf{F}(pk))$
$\mathbf{r} := \mathsf{CBD}_1(r)$
$\hat{\mathbf{r}} = \mathsf{NTT}(\mathbf{r})$
$\mathbf{m} = \mathsf{Encode}(\tilde{m}, \mathsf{G}(\mathsf{Encode}_q(\hat{\mathbf{r}})))$
$\hat{\mathbf{m}} = \mathsf{NTT}(\mathbf{m})$
$2^{16} \cdot \hat{\mathbf{h}} := \mathsf{Decode}_q(pk)$
$\hat{\mathbf{c}} = \hat{\mathbf{h}} \circ \hat{\mathbf{r}} + \hat{\mathbf{m}}$
$c := \mathsf{Encode}_q(\hat{\mathbf{c}})$
return c

Algorithm 20: $\mathsf{Dec}(sk, c)$: decryption

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: Message $m \in \mathcal{B}^{\leq \ell_m - 1}$

1: Parse $sk = (sk_1, sk_2, sk_3) \in \mathcal{B}^{\lceil \log_2 q \rceil \cdot n/8} \times \mathcal{B}^{\lceil \log_2 q \rceil \cdot n/8} \times \mathcal{B}^{32}$
2: $\hat{\mathbf{f}} = \mathsf{Decode}_q(sk_1)$
3: $\hat{\mathbf{c}} = \mathsf{Decode}_q(c)$
4: $\mathbf{m} = \mathsf{NTT}^{-1}(\hat{\mathbf{c}} \circ \hat{\mathbf{f}}) \mod 3$
5: $\hat{\mathbf{m}} = \mathsf{NTT}(\mathbf{m})$
6: $2^{16} \cdot \hat{\mathbf{h}}^{-1} = \mathsf{Decode}_q(sk_2)$
7: $\hat{\mathbf{r}} = (\hat{\mathbf{c}} - \hat{\mathbf{m}}) \circ \hat{\mathbf{h}}^{-1}$ // RRec
8: $\tilde{m}' = (\tilde{m}'_0, \cdots, \tilde{m}'_{n-1}) := \mathsf{Inv}(\mathbf{m}, \mathsf{G}(\mathsf{Encode}_q(\hat{\mathbf{r}})))$ // Checking if $\tilde{m}' = \perp$ is done in line 12
9: $r' := \mathsf{H}_{\mathsf{PKE}}(\tilde{m}', sk_3)$
10: $\mathbf{r}' := \mathsf{CBD}_1(r')$
11: $\hat{\mathbf{r}}' = \mathsf{NTT}(\mathbf{r}')$
12: **if** $\tilde{m}' = \perp$ or $\hat{\mathbf{r}} \neq \hat{\mathbf{r}}'$, **return** $\perp$ // Check if $\tilde{m}' = \perp$ or $\mathbf{r}' \notin R_q$
13: **else**, **return** $\mathsf{Decode}_m((\tilde{m}'_0, \cdots, \tilde{m}'_{\ell_m - 1}))$

8 Parameters and Security Analysis

We define four parameter sets for $\mathsf{NTRU+}\{\mathsf{KEM},\mathsf{PKE}\}$, which are listed in Table 7 and 8, respectively. We call them $\mathsf{NTRU+}\{\mathsf{KEM},\mathsf{PKE}\}\{576, 768, 864, 1152\}$, respectively, depending on the degree of the polynomial $x^n - x^{n/2} + 1$. In all parameter sets, the modulus q is set to 3457, and the coefficients of $\mathbf{m}$ and $\mathbf{r}$ are sampled according to the distribution ψ_1^n (i.e., $\psi_{\mathcal{R}} = \psi_{\mathcal{M}} = \psi_1^n$). For each set of $(n, q, \psi_1^n, \mathcal{M}' = \{0,1\}^n)$, the worst-case correctness error δ' is calculated by adding the average-case correctness error δ of $\mathsf{GenNTRU}[\psi_1^n]$ and the value $\Delta = \|\psi_{\mathcal{R}}\| \cdot (1 + \sqrt{(\ln|\mathcal{M}'| - \ln\|\psi_{\mathcal{R}}\|)/2})$ using the equation from Theorem 3.2. Since Δ is negligible for all parameter sets, the worst-case correctness error of $\mathsf{NTRU+}\{\mathsf{KEM},\mathsf{PKE}\}$ is almost equal to the average-case correctness error of each corresponding $\mathsf{GenNTRU}[\psi_1^n]$ as expected.

Scheme	classical		quantum	
	LWE	NTRU	LWE	NTRU
$\mathsf{NTRU+}\{\mathsf{KEM},\mathsf{PKE}\}576$	115	114	102	101
$\mathsf{NTRU+}\{\mathsf{KEM},\mathsf{PKE}\}768$	164	164	144	144
$\mathsf{NTRU+}\{\mathsf{KEM},\mathsf{PKE}\}864$	189	189	167	166
$\mathsf{NTRU+}\{\mathsf{KEM},\mathsf{PKE}\}1152$	263	266	234	233

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

To estimate the concrete security level of $\mathsf{NTRU+}\{\mathsf{KEM},\mathsf{PKE}\}$, we analyze the hardness of the two problems $\mathsf{RLWE}_{n,q,\psi_1^n}$ and $\mathsf{NTRU}_{n,q,\psi_1^n}$ based on each parameter set. For the RLWE problem, we employ the Lattice estimator [1], which uses the BKZ lattice reduction algorithm [11] for the best-known lattice attacks such as the primal [2] and dual [28] attacks. Next, for the NTRU problem, we use the NTRU estimator provided by the finalist NTRU [10], which is based on the primal attack and Howgrave-Graham's hybrid attack [22] 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 [25] on a (reduced) sub-lattice. As a result, the concrete security level of the NTRU problem is almost the same as that of the RLWE problem. Table 6 shows the resulting security levels relative to the RLWE and NTRU problems, depending on each $\mathsf{NTRU+}\{\mathsf{KEM},\mathsf{PKE}\}$ parameter set. For the cost model of the BKZ algorithm, we employ $2^{0.292\beta}$ [4] and $2^{0.257\beta}$ [9] for the classical and quantum settings, respectively.

Recently, Lee *et al.* [26] proposed a combinatorial attack that improves upon May's Meet-LWE attack [31] and analyzed the concrete security level of $\mathsf{NTRU+}\{\mathsf{KEM},\mathsf{PKE}\}$. Their analysis demonstrated that the security of $\mathsf{NTRU+}\{\mathsf{KEM},\mathsf{PKE}\}$ against their combinatorial attack does not degrade below the level predicted by the above Lattice and NTRU estimators.

9 Performance Analysis

All benchmarks were obtained on a single core of an Intel Core i7-8700K (Coffee Lake) processor clocked at 3700 MHz. The benchmarking machine was equipped with 16 GB of RAM. Implementations were compiled using gcc version 11.4.0. Table 7 and 8 list the execution time of the reference and AVX2 implementations of $\mathsf{NTRU+}\{\mathsf{KEM},\mathsf{PKE}\}$, NTRU, and KYBER, along with the security level, the size of the secret key, public key, and ciphertext. The execution time was measured as the average cycle counts of 100,000 executions for the respective algorithms. The source code for $\mathsf{NTRU+}\{\mathsf{KEM},\mathsf{PKE}\}$ can be downloaded from https://github.com/ntruplus/ntruplus.

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

Scheme	security level		n	q	pk	ct	sk	$\log_2 \delta'$	reference			AVX2		
	classical	quantum							Gen	Encap	Decap	Gen	Encap	Decap
NTRU+KEM576	114	101	576	3457	864	864	1760	-487	167	80	95	24	22	13
NTRU+KEM768	164	144	768	3457	1152	1152	2336	-379	192	101	121	26	27	16
NTRU+KEM864	189	166	864	3457	1296	1296	2624	-340	238	123	148	28	30	19
NTRU+KEM1152	263	233	1152	3457	1728	1728	3488	-260	370	162	196	41	39	26
KYBER512	118	104	512	3329	800	768	1632	-139	116	137	158	36	39	24
KYBER768	182	160	768	3329	1184	1088	2400	-164	182	202	230	51	55	37
KYBER1024	255	224	1024	3329	1568	1568	3168	-174	270	321	359	65	73	52
ntruhps2048509	104	93	509	2048	699	699	935	$-\infty$	8031	746	1384	376	262	33
ntruhrss701	133	119	701	8192	1138	1138	1450	$-\infty$	14684	1030	2623	365	166	52
ntruhps2048677	144	127	677	2048	930	930	1234	$-\infty$	13882	1206	2441	545	348	49
ntruhps4096821	178	158	821	4096	1230	1230	1590	$-\infty$	20385	1644	3519	702	423	62

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

Table 8: Comparison between NTRU+PKE, finalist NTRU, and KYBER

Scheme	Security		n	q	pk	ct	sk	(ℓ_m, ℓ_r)	$\log_2 \delta'$	reference			AVX2		
	c	q								Gen	Enc	Dec	Gen	Enc	Dec
NTRU+PKE576	114	101	576	3457	864	864	1760	(33,39)	-487	167	80	96	24	22	13
NTRU+PKE768	164	144	768	3457	1152	1152	2336	(33,63)	-379	192	100	121	26	26	16
NTRU+PKE864	189	166	864	3457	1296	1296	2624	(33,75)	-340	238	124	149	29	31	20
NTRU+PKE1152	263	233	1152	3457	1728	1728	3488	(33,111)	-260	371	162	198	42	39	26
KYBER512	118	104	512	3329	800	816	1632	N/A	-139	116	140	161	36	42	27
KYBER768	182	160	768	3329	1184	1232	2400	N/A	-164	183	206	234	51	59	41
KYBER1024	255	224	1024	3329	1568	1616	3168	N/A	-174	272	323	362	65	77	56
ntruhps2048509	104	93	509	2048	699	747	935	N/A	$-\infty$	8044	751	1393	377	265	37
ntruhrss701	133	119	701	8192	1138	1186	1450	N/A	$-\infty$	14683	1032	2630	364	170	56
ntruhps2048677	144	127	677	2048	930	978	1234	N/A	$-\infty$	13901	1206	2451	544	351	53
ntruhps4096821	178	158	821	4096	1230	1278	1590	N/A	$-\infty$	20392	1651	3532	703	424	66

c: classical security level. q: quantum security level. n: polynomial degree of the ring. q: modulus. $(pk, ct, sk, \ell_m, \ell_r)$: bytes.
δ': worst-case (or perfect) correctness error. (Gen, Enc, Dec): K cycles of reference or AVX2 implementations.
$*$: means that 32-byte messages are encrypted using AES-256-GCM.

When comparing NTRU and NTRU+KEM, Table 7 shows that both schemes have similar bandwidth (consisting of a public key and a ciphertext) at comparable security levels. For instance, NTRU+KEM864 at the 189-bit security level requires a bandwidth of 2,592 bytes, and ntruhps4096821 at the 178-bit security level requires a bandwidth of 2,460 bytes. In terms of storage cost with respect to the secret key, NTRU+KEM requires almost twice as much storage cost as NTRU. This is because NTRU+KEM stores $(\mathbf{f}, \mathbf{h}^{-1}, \mathsf{F}(pk))$ as a secret key rather than only $\mathbf{f}$. However, in terms of execution time, NTRU+KEM outperforms NTRU, primarily depending on whether NTT-friendly rings are used.

When comparing KYBER and NTRU+KEM, the bandwidth of NTRU+KEM is slightly larger than that of KYBER at similar security levels. This is because KYBER uses a rounding technique to reduce the size of a ciphertext. In terms of efficiency, Table 7 shows that, at similar security levels, the key generation of NTRU+KEM is slower than that of KYBER in the reference implementation. However, the encapsulation and decapsulation of NTRU+KEM is faster than that of KYBER in both the reference and 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] 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.

[5] Daniel J. Bernstein. Curve25519: New Diffie-Hellman speed records. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *PKC 2006: 9th International Conference on Theory and Practice of Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 207–228, New York, NY, USA, April 24–26, 2006. Springer, Berlin, Heidelberg, Germany.

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

[7] 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.

[8] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 41–69, Seoul, South Korea, December 4–8, 2011. Springer, Berlin, Heidelberg, Germany.

[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. How to enhance the security of public-key encryption at minimum cost. In Hideki Imai and Yuliang Zheng, editors, *PKC'99: 2nd International Workshop on*

Theory and Practice in Public Key Cryptography, volume 1560 of *Lecture Notes in Computer Science*, pages 53–68, Kamakura, Japan, March 1–3, 1999. Springer, Berlin, Heidelberg, Germany.

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

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

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

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

[28] 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.

[29] 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.

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

[31] 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.

[32] 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.

[33] 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.

[34] Michael Scott. Slothful reduction. Cryptology ePrint Archive, Report 2017/437, 2017.

[35] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004.

[36] Dominique Unruh. Quantum position verification in the random oracle model. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 1–18, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Berlin, Heidelberg, Germany.

[37] Dominique Unruh. Revocable quantum timed-release encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 129–146, Copenhagen, Denmark, May 11–15, 2014. Springer, Berlin, Heidelberg, Germany.

[38] 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.

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

$$x^{576} - x^{288} + 1 = (x^{288} - \zeta^{72})(x^{288} - \zeta^{360}).$$

Here, $\zeta^{\ell/6} = \zeta^{72}$ represents a primitive sixth root of unity modulo q. Consequently, we can observe that we can apply a Radix-3 NTT layer because both $x^{288} - \zeta^{72}$ and $x^{288} - \zeta^{360}$ can be factorized as:

$$\begin{aligned} x^{288} - \zeta^{72} &= (x^{96} - \zeta^{24})(x^{96} - \zeta^{24}\omega)(x^{96} - \zeta^{24}\omega^2) = (x^{96} - \zeta^{24})(x^{96} - \zeta^{168})(x^{96} - \zeta^{312}) \\ x^{288} - \zeta^{360} &= (x^{96} - \zeta^{120})(x^{96} - \zeta^{120}\omega)(x^{96} - \zeta^{120}\omega^2) = (x^{96} - \zeta^{120})(x^{96} - \zeta^{264})(x^{96} - \zeta^{408}). \end{aligned}$$

Here, $\omega = \zeta^{\ell/3} = \zeta^{144}$ is a primitive third root of unity modulo q. Similarly, we can observe that we can apply a Radix-2 NTT layer because both $x^{96} - \zeta^{24}$ and $x^{96} - \zeta^{120}$ can be further factored by half. For example, $x^{96} - \zeta^{32}$ can be factored as:

$$x^{96} - \zeta^{24} = (x^{48} - \zeta^{12})(x^{48} + \zeta^{12}) = (x^{48} - \zeta^{12})(x^{48} - \zeta^{12}\zeta^{\ell/2}) = (x^{48} - \zeta^{12})(x^{48} - \zeta^{228})$$

Here, $\zeta^{\ell/2} = \zeta^{216}$ is a primitive second root of unity modulo q. If we continue this process, we can factor the polynomial $x^{576} - x^{288} + 1$ all the way down to the degree $d = 4$.

B Radix-3 NTT layer

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

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

Naively, these equations might appear to require $2n$ multiplications and $2n$ additions, using six predefined values: α, α^2, β, β^2, γ, and γ^2. Nevertheless, by following the techniques in [19], we can significantly reduce this computational load to n multiplications, n additions, and $4n/3$ subtractions, by using only three predefined values: α, α^2, and ω, as described in Algorithm 21.

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

Algorithm 21: 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: $(\hat{a}_0(x), \hat{a}_1(x), \hat{a}_2(x)) \in \mathbb{Z}_q[x]/\langle x^{n/3} - \alpha \rangle \times \mathbb{Z}_q[x]/\langle x^{n/3} - \beta \rangle \times \mathbb{Z}_q[x]/\langle x^{n/3} - \gamma \rangle$
$t_1(x) = a_1(x)\alpha$
$t_2(x) = a_2(x)\alpha^2$
$t_3(x) = (t_1(x) - t_2(x))w$
$\hat{a}_2(x) = a_0(x) - t_1(x) + t_3(x)$
$\hat{a}_1(x) = a_0(x) - t_1(x) + t_3(x)$
$\hat{a}_0(x) = a_0(x) - t_1(x) + t_3(x)$
return $(\hat{a}_0(x), \hat{a}_1(x), \hat{a}_2(x))$

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

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

Naively, these equations might appear to require $2n$ multiplications and $2n$ additions, using six predefined values: α^{-1}, α^{-2}, β^{-1}, β^{-2}, γ^{-1}, and γ^{-2}. Nevertheless, by following the techniques in [19], we can significantly reduce this computational load to n multiplications, n additions, and $4n/3$ subtractions, by employing only four predefined values: α^{-1}, α^{-2}, and ω, as described in in Algorithm 22.

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

Algorithm 22: 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: $3a(x) = 3a_0(x) + 3a_1(x)x^{n/3} + 3a_2(x)x^{2n/3} \in \mathbb{Z}_q[x]/\langle x^n - \alpha^3 \rangle$
$t_1(x) = w(\hat{a}_1(x) - \hat{a}_2(x))$
$t_2(x) = \hat{a}_0(x) - \hat{a}_1(x) - t_1(x)$
$t_3(x) = \hat{a}_0(x) - \hat{a}_2(x) + t_1(x)$
$3a_0(x) = \hat{a}_0(x) + \hat{a}_1(x) + \hat{a}_2(x)$
$3a_1(x) = t_2(x)\alpha^{-1}$
$3a_2(x) = t_3(x)\alpha^{-2}$
return $3a(x) = 3a_0(x) + 3a_1(x)x^{n/3} + 3a_2(x)x^{2n/3}$

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

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