NTRU+: Compact Construction of NTRU Using Simple Encoding Method

Jonghyun Kim‡  Jong Hwan Park§

August 11, 2023

Abstract

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

In this work, we propose a new NTRU-based key encapsulation mechanism (KEM), called NTRU+, which overcomes nearly all existing drawbacks. NTRU+ is constructed based on two new generic transformations: \text{ACWC}_2 and \text{FO}^⊥ (a variant of the Fujisaki-Okamoto transform). \text{ACWC}_2 is used to easily achieve worst-case correctness error, while \text{FO}^⊥ is used to achieve chosen-ciphertext security without re-encryption. Both \text{ACWC}_2 and \text{FO}^⊥ are defined using a randomness-recovery algorithm and an encoding method. In particular, our simple encoding method, the semi-generalized one-time pad (SOTP), allows us to sample a message from a natural bit-string space with an arbitrary distribution. We provide four parameter sets for NTRU+ and present implementation results using NTT-friendly rings over cyclotomic trinomials.

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

1 Introduction

The NTRU encryption scheme [14] 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 [14], 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 [5] 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 [24] and Saber [9]. Specifically, the advantages of NTRU include: (1) the compact structure

\footnote{This work is submitted to ‘Korean Post-Quantum Cryptography Competition’ (www.kpqc.or.kr).}

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

\footnote{Korea University, Seoul, Korea. Email: yoswuk@korea.ac.kr.}

\footnote{Sangmyung University, Seoul, Korea. Email: jhpark@smu.ac.kr.}
of a ciphertext consisting of a single polynomial, and (2) (possibly) faster encryption and decryption without the need to sample the coefficients of a public key polynomial.

The central design principle of NTRU is described over a ring \( R_q = \mathbb{Z}_q[x]/(f(x)) \), where \( q \) is a positive integer and \( f(x) \) is a polynomial. The public key is generated as \( h = pg/(pf' + 1) \), where \( g \) and \( f' \) are sampled according to a narrow distribution \( \psi \), \( p \) is a positive integer that is coprime with \( q \) and smaller than \( q \) (e.g., 3), and the corresponding private key is \( f = pf' + 1 \). To encrypt a message \( m \) sampled from the message space \( M' \), one creates two polynomials \( r \) and \( m \), with coefficients drawn from a narrow distribution \( \psi \), and computes the ciphertext \( c = hr + m \) in \( R_q \). An (efficient) encoding method may be used to encode \( m \in M' \) into \( m \) and \( r \in R_q \). Alternatively, it is possible to directly sample \( m \) and \( r \) from \( \psi \), where \( m \) is considered as the message to be encrypted. To decrypt the ciphertext \( c \), one computes \( cf \) in \( R_q \), recovers \( m \) by deriving the value \( cf' \mod p \), and (if necessary) decodes \( m \) to obtain the message \( m \). The decryption of NTRU works correctly if all the coefficients of the polynomial \( p(gr + f'm) + m \) 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, 8]. Roughly speaking, the worst-case correctness error refers to the probability that decryption fails for any ciphertext that can be generated with all possible messages and randomness in their respective spaces. The worst-case correctness error error considers that an adversary, \( A \), 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 \( c = hr + m \) provides \( A \) with the information that one of the coefficients of \( p(gr + f'm) + m \) is larger than or equal to \( q/2 \). If \( A \) has control over the choice of \( r \) and \( m \), even one such decryption failure may open a path to associated decryption queries to obtain more information about secret polynomials \( g \) and \( f \).

When designing NTRU, two approaches can be used to achieve worst-case correctness error. One approach is to draw \( m \) and \( r \) directly from \( \psi \), while setting the modulus \( q \) to be relatively large. The larger \( q \) guarantees a high probability that all coefficients of \( p(gr + f'm) + m \) are less than \( q/2 \) for nearly all possible \( m \) and \( r \) 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 [5], wherein all recommended parameters provide perfect correctness error (i.e., the worst-case correctness error becomes zero for all possible \( m \) and \( r \)). By contrast, the other approach [12] is to use an encoding method by which a message \( m \in M' \) is used as a randomness to sample \( m \) and \( r \) according to \( \psi \). Under the Fujisaki-Okamoto (FO) transform [13], decrypting a ciphertext \( c \) 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 \( m \) and \( r \) should be honestly sampled by \( A \) according to \( \psi \). Consequently, by disallowing \( A \) to have control over \( m \) and \( r \), the NTRU with an encoding method has a worst-case correctness error that is close to an average-case error.

Based on the aforementioned observation, [12] proposed generic (average-case to worst-case) transformation\(^1\) that make the average-case correctness error of an underlying scheme nearly close to the worst-case error of a transformed scheme. One of their transformations (denoted by ACWC) is based on an encoding method called the generalized one-time pad (denoted by GOTP). Roughly speaking, GOTP works as fol-

\(^1\)There is another way of creating the public key as \( h = pg/f \), but we focus on setting \( h = pg/(pf' + 1) \) 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.
to achieve chosen-ciphertext security, high min-entropy, and is proven to be tightly
m which utilizes a simple encoding method. By using ACWC it was an open problem \[\text{exactly uniformly random numbers from } \{\pm 1 \}\]
error close to the average-case error of the underlying NTRU. Additionally, NTRU
We propose a new practical NTRU construction called “NTRU+” that addresses the two drawbacks of the
message distribution. First, \[\text{perfect uniformity over ternary strings } m\] wherein the output follows \(\psi\), \(m\) and \(r\) are created from \(m\) following \(\psi\), which can be verified in decryption using the FO transform. Specifically, for two inputs \(m\) and \(G(m_1)\) that are sampled from \(\{\pm 1, 0\}\) for some integer \(\lambda\), \(m_2 \in \{\pm 1, 0\}^\lambda\) is computed by doing the component-wise exclusive-or modulo 3 of two ternary strings \(m\) and \(G(m_1)\). Thus, if \(G(m_1)\) follows a uniformly random distribution \(\psi\) over \(\{\pm 1, 0\}\), \(m\) is hidden from \(m_2\) because of the one-time pad property.
However, an ACWC based on the GOTP has two disadvantages in terms of security reduction and message distribution. First, [12] showed that ACWC converts a one-way CPA (OW-CPA) secure underlying scheme into a transformed one that is still OW-CPA secure, besides the fact that their security reduction is loose\(^B\) by causing a security loss factor of \(q_C\), the number of random oracle queries. Second, ACWC forces even a message \(m \in M'\) to follow a specific distribution because their security analysis of ACWC requires GOTP to have the additional randomness-hiding property, meaning that \(G(m_1)\) should also be hidden from the output \(m_2\). Indeed, the NTRU instantiation from ACWC, called ‘NTRU-B’ [12], requires that \(m\) be chosen uniformly at random from \(M' = \{\pm 1, 0\}^\lambda\). Notably, it is difficult to generate exactly uniformly random numbers from \(\{\pm 1, 0\}\) in constant time due to rejection sampling. Therefore, it was an open problem [12] to construct a new transformation that permits a different, more easily sampled distribution of a message while relying on the same security assumptions.

1.1 Our Results
We propose a new practical NTRU construction called “NTRU+” that addresses the two drawbacks of the previous ACWC. To achieve this, we introduce a new generic ACWC transformation, denoted as ACWC\(_2\), which utilizes a simple encoding method. By using ACWC\(_2\), NTRU+ achieves a worst-case correctness error close to the average-case error of the underlying NTRU. Additionally, NTRU+ requires the message \(m\) to be drawn from \(M' = \{0, 1\}^n\) (for a polynomial degree \(n\)), following an arbitrary distribution with high min-entropy, and is proven to be tightly secure under the same assumptions as NTRU-B, the NTRU and RLWE assumptions. To achieve chosen-ciphertext security, NTRU+ relies on a novel FO-equivalent

\[\text{correctness error (for a polynomial degree } n)\]
\[\text{worst-case correctness error}\]
\[\text{uniform/Fixed-weight}\]
\[\text{Arbitrary}\]
\[\text{Yes}\]
\[\text{Yes}\]
\[\text{Yes}\]
\[\text{No}\]
\[\text{Yes}\]

Table 1: Comparison to previous NTRU constructions

\(n\): polynomial degree of the ring. \(\lambda\): length of the message. DPKE: deterministic public key encryption.
SXY variant: SXY transformation [23] described in the finalist NTRU.

\[\text{NTRU}\ [6]\]
\[\text{NTRU-B} \ [12]\]
\[\text{NTRU+}\]

<table>
<thead>
<tr>
<th>Scheme</th>
<th>NTRU</th>
<th>NTRU-B</th>
<th>NTRU+</th>
</tr>
</thead>
<tbody>
<tr>
<td>NTT-friendly</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Correctness error</td>
<td>Perfect</td>
<td>Worst-case</td>
<td>Worst-case</td>
</tr>
<tr>
<td>(m, r)-encoding</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Message set</td>
<td>(m, r \leftarrow {-1, 0, 1}^n)</td>
<td>(m \leftarrow {-1, 0, 1}^\lambda)</td>
<td>(m \leftarrow {0, 1}^n)</td>
</tr>
<tr>
<td>Message distribution</td>
<td>Uniform/Fixed-weight</td>
<td>Uniform</td>
<td>Arbitrary</td>
</tr>
<tr>
<td>CCA transform</td>
<td>DPKE + SXY variant</td>
<td>ACWC + FO(\perp)</td>
<td>ACWC(_2) + FO(\perp)</td>
</tr>
<tr>
<td>Assumptions</td>
<td>NTRU, RLWE</td>
<td>NTRU, RLWE</td>
<td>NTRU, RLWE</td>
</tr>
<tr>
<td>Tight reduction</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

\(3\) [12] introduced a new security notion, \(q\)-OW-CPA, which states that an adversary outputs a set \(Q\) with a maximum size of \(q\) and wins if the correct message corresponding to a challenged ciphertext belongs to \(Q\). We believe that \(q\)-OW-CPA causes a security loss of \(q\).
transform without re-encryption, which makes the decryption algorithm of NTRU+ faster than in the ordinary FO transform. In terms of efficiency, we use the idea from [22] to apply the Number Theoretic Transform (NTT) to NTRU+ and therefore instantiate NTRU+ over a ring $R_q = \mathbb{Z}_q[x]/(f(x))$, where $f(x) = x^n - x^{n/2} + 1$ is a cyclotomic trinomial. By selecting appropriate $(n, q)$ and $\psi$, we suggest four parameter sets for NTRU+ and provide the implementation results for NTRU+ in each parameter set. Table \ref{tab:comparison} lists the main differences between the previous NTRU constructions \cite{ACWC0,ACWC1} and NTRU+. In the following section, we describe our technique, focusing on these differences.

**ACWC 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. Additionally, ACWC$_2$ involves an encoding method called semi-generalized one-time pad (denoted by SOTP). In contrast to the GOTP in \cite{ACWC0,ACWC1}, SOTP works in a generic sense as follows: first, a message $m \in \mathcal{M}'$ is used to sample $r$ based on $\psi$, and then $m = SOTP(m, G(r))$ is computed, where the coefficients follow $\psi$, using a hash function $G$. When decrypting a ciphertext $c = \text{Enc}(pk, m; r)$ under a public key $pk$, $m$ is recovered by a normal decryption algorithm, and using $m$, $r$ is also recovered by a randomness-recovery algorithm. Finally, an inverse of SOTP using $G(r)$ and $m$ 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 $A$ queries $r$ to a (classical) random oracle $G$, $A$ does not obtain any information on $m_b$ (that $A$ submits) for $b \in \{0, 1\}$ because of the basic message-hiding property of SOTP. However, whenever $A$ queries $r_i$ to $G$ for $i = 1, \ldots, q_G$, a reductionist can check whether each $r_i$ is the randomness used for its OW-CPA challenge ciphertext using a message-recovery algorithm. Therefore, the reductionist can find the exact $r_i$ among the $q_G$ number of queries if $A$ queries $r_i$ (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 \ref{tab:comparison} presents a detailed comparison between previous ACWC transformations and our new ACWC$_3$. Unlike the previous ACWC based on GOTP, \cite{ACWC0} 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 $c = (\text{Enc}(pk, m; r), F(m) \oplus m)$ using a hash function $F$, which causes the ciphertext

<table>
<thead>
<tr>
<th>Transformation</th>
<th>ACWC$_0$ \cite{ACWC0}</th>
<th>ACWC$_1$ \cite{ACWC1}</th>
<th>ACWC$_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message encoding</td>
<td>No</td>
<td>SOTP</td>
<td>SOTP</td>
</tr>
<tr>
<td>Message distribution</td>
<td>Arbitrary</td>
<td>Uniform</td>
<td>Arbitrary</td>
</tr>
<tr>
<td>Ciphertext expansion</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Tight reduction</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Underlying PKE</td>
<td>Any</td>
<td>Any</td>
<td>Injective + MR + RR</td>
</tr>
</tbody>
</table>


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

Table 2: Comparison to previous ACWC transformations
expansion of $F(m) \oplus m$, whereas such a ciphertext redundancy does not occur in ACWC and ACWC$_2$. Like ACWC$_2$, ACWC$_0$ transforms any OW-CPA secure scheme into an IND-CPA secure one, but their security reduction is not tight as in ACWC. ACWC$_0$ and ACWC$_2$ requires no specific message distribution, whereas ACWC requires $m \in \mathcal{M}$ to be sampled according to a uniformly random distribution from $\mathcal{M}$. ACWC$_0$ and ACWC work for any OW-CPA secure scheme, but ACWC$_2$ works for any OW-CPA secure scheme satisfying injectivity, MR, and RR properties.

**FO-Equivalent Transform without Re-encryption.** To achieve chosen-ciphertext (IND-CCA) security, we apply the generic transform $\text{FO}^\perp$ to the ACWC$_2$-derived scheme, which is IND-CPA secure. As with other FO-transformed schemes, the resulting scheme from ACWC$_2$ and $\text{FO}^\perp$ is still required to perform re-encryption in the decryption process to check if (1) $(m, r)$ are correctly generated from $m$ and (2) a (decrypted) ciphertext $c$ is correctly encrypted from $(m, r)$. However, by using the RR property of the underlying scheme, we further remove the re-encryption process from $\text{FO}^\perp$. Instead, the more advanced transform (denoted by $\overline{\text{FO}}^\perp$) simply checks whether $r$ from the randomness-recovery algorithm is the same as the (new) randomness $r'$ created from $m$. We show that $\overline{\text{FO}}^\perp$ is functionally identical to $\text{FO}^\perp$ by proving that the randomness-checking process in $\overline{\text{FO}}^\perp$ is equivalent to the re-encryption process $\text{FO}^\perp$. The equivalence proof relies mainly on the injectivity [13, 14] and rigidity [15] properties of the underlying schemes. As a result, although the RR property seems to incur some additional decryption cost, it ends up making the decryption algorithm faster than the original FO transform. Figure 1 presents an overview of security reductions from OW-CPA to IND-CCA.

**Simple SOTP Instantiation with More Convenient Sampling Distributions.** As mentioned previously, ACWC$_2$ is based on an efficient construction of SOTP that takes $m$ and $G(r)$ as inputs and outputs $m = \text{SOTP}(m, G(r))$. In particular, computing $m = \text{SOTP}(m, G(r))$ requires that each coefficient of $m$ should follow $\psi$, while preserving the message-hiding property. To achieve this, we set $\psi$ as the centered binomial distribution (CBD) $\psi_k$ with $k = 1$, which is easily obtained by subtracting two uniformly random bits from each other. For a polynomial degree $n$ and hash function $G : \{0, 1\}^* \rightarrow \{0, 1\}^{2n}$, $m$ is chosen from the message space $\mathcal{M}' = \{0, 1\}^n$ for an arbitrary distribution (with high min-entropy) and $G(r) = y_1\|y_2 \in \{0, 1\}^n \times \{0, 1\}^n$. 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 $m$. 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 $m$ follows $\psi_1$. Since $r$ is also sampled from a hash value of $m$ according to $\psi_1$, all sampling distributions in NTRU+ are easy to implement. We can also expect that, similar to the case of $\psi_1$, 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.

![Figure 1: Overview of security reductions](image-url)
NTT-Friendly Rings Over Cyclotomic Trinomials. NTRU+ is instantiated over a polynomial ring \( R_q = \mathbb{Z}_q[x]/(f(x)) \), where \( f(x) = x^n - x^{n/2} + 1 \) is a cyclotomic trinomial of degree \( n = 2^{31} \). \cite{Dum02} 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 \( R_q = \mathbb{Z}_q[x]/(x^n + 1) \). Moreover, because the choice of a cyclotomic trinomial is moderate, it provides more flexibility to satisfy a certain level of security. Based on these results, we choose four parameter sets for NTRU+, where the polynomial degree \( n \) of \( f(x) = x^n - x^{n/2} + 1 \) is set to be 576, 768, 864, and 1152, and the modulus \( q \) is 3457 for all cases. Table 6 lists the comparison results between finalist NTRU \cite{Lyubashevsky2019}, KYBER, KYBER-90s \cite{Collins2020}, and NTRU+ in terms of security and efficiency. To estimate the concrete security level of NTRU+, we use the Lattice estimator \cite{Happel2021} for the RLWE problem and the NTRU estimator \cite{Happel2021} for the NTRU problem, considering that all coefficients of each polynomial \( f', g, r \), and \( m \) are drawn according to the centered binomial distribution \( \psi_1 \). The implementation results in Table 6 are estimated with reference and AVX2 optimizations. We can observe that NTRU+ outperforms NTRU at a similar security level.

1.2 Related Works

The first-round NTRUEncrypt \cite{Dum02} submission to the NIST PQC standardization process was an NTRU-based encryption scheme with the NAEP padding method \cite{Shamir1997}. Roughly speaking, NAEP is similar to our SOTP, but the difference is that it does not completely encode \( m \) to prevent an adversary \( A \) from choosing \( m \) maliciously. This is due to the fact that \( m := \text{NAEP}(m, G(hr)) \) is generated by subtracting two \( n \)-bit strings \( m \) and \( G(hr) \) from each other, i.e., \( m = G(hr) \) by bitwise subtraction, and then assigning them to the coefficients of \( m \). Since \( m \) can be maliciously chosen by \( A \) in NTRUEncrypt, \( m \) can also be maliciously chosen, regardless of \( G(hr) \).

The finalist NTRU \cite{Lyubashevsky2019} was submitted as a key encapsulation mechanism (KEM) that provides four parameter sets for perfect correctness. To achieve chosen-ciphertext security, \cite{Lyubashevsky2019} relied on a variant of the SXY \cite{Shamir2017} conversion, which also avoids re-encryption during decapsulation. Similar to NTRU+, the SXY variant requires the rigidity \cite{Happel2020} of an underlying scheme and uses the notion of deterministic public key encryption (DPKE) where \( (m, r) \) are all recovered as a message during decryption. In the NTRU construction, the recovery of \( r \) 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 \( (m, r) \) are selected correctly from predefined distributions.

In 2019, Lyubashevsky et al. \cite{Lyubashevsky2019} proposed an efficient NTRU-based KEM called NTTRU by applying NTT to the ring defined by a cyclotomic trinomial \( \mathbb{Z}_q[x]/(x^n - x^{n/2} + 1) \). NTTRU was based on the Dent \cite{Happel2021} 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 ACWC0 in \cite{Dum02}.

In 2021, Duman et al. \cite{Dum02} proposed two generic transformations, ACWC0 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 GOTA as an encoding method to prevent \( A \) from adversarially choosing \( m \). While ACWC0 is simple, it requires a ciphertext expansion of 32 bytes. On the other hand, ACWC does not require an expansion of the ciphertext size. The security of ACWC0 and ACWC was analyzed in both the classical and quantum random oracle models \cite{Dum02}. However, their NTRU instantiation using ACWC has the drawback of requiring the message \( m \) to be chosen from a uniformly random distribution over \( \mathcal{M}' = \{-1, 0, 1\}^\lambda \).
2 Preliminaries

2.1 Public Key Encryption and Related Properties

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

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

**Correctness.** We say that $\text{PKE}$ has a (worst-case) correctness error $\delta$ \cite{15} if

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

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

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

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

**Injectivity.** \cite{15, 5} We say that $\text{PKE}$ is $\mu$-injective if for all $(pk, sk) \leftarrow \text{Gen}(1^\lambda)$ and $m, m' \in \mathcal{M}$ and $r, r' \in \mathcal{R}$, we have that

$$\Pr[c = c' \land (m, r) \neq (m', r')] \leq \mu,$$

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

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

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

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

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

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

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

$$\Pr \left[ \forall m' \in \text{Pred}(pk, c) : \text{RRec}(pk, m', c) \notin \mathcal{R} \lor \text{Enc}(pk, m'; \text{RRec}(pk, m', c)) \neq c | c \leftarrow \text{Enc}(pk, m; r) \right] = 0,$$
where the probability is taken over $c \leftarrow \text{Enc}(pk, m; r)$ and $\text{Pre}^m(pk, c) := \{m \in \mathcal{M} | \exists r \in \mathcal{R} : \text{Enc}(pk, m; r) = c\}$. Additionally, it is required that $\text{RRec}$ returns $\bot$ if $\text{RRec}(pk, m', c) \notin \mathcal{R}$ or $\text{Enc}(pk, m'; \text{RRec}(pk, m', c)) \neq c$.

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

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

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

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

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

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

### 2.2 Security

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

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

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

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

<table>
<thead>
<tr>
<th>Game OW-CPA</th>
<th>Game IND-CPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: $(pk, sk) \leftarrow \text{Gen}(1^\lambda)$</td>
<td>1: $(pk, sk) \leftarrow \text{Gen}(1^\lambda)$</td>
</tr>
<tr>
<td>2: $m \leftarrow \psi_{\mathcal{M}}$</td>
<td>2: $(m_0, m_1) \leftarrow \mathcal{A}_0(pk)$</td>
</tr>
<tr>
<td>3: $c^* \leftarrow \text{Enc}(pk, m)$</td>
<td>3: $b \leftarrow {0, 1}$</td>
</tr>
<tr>
<td>4: $m' \leftarrow \mathcal{A}(pk, c^*)$</td>
<td>4: $c^* \leftarrow \text{Enc}(pk, m_b)$</td>
</tr>
<tr>
<td>5: return $[m = m']$</td>
<td>5: $b' \leftarrow \mathcal{A}_1(pk, c^*)$</td>
</tr>
<tr>
<td>6: return $[b = b']$</td>
<td>6: return $[b = b']$</td>
</tr>
</tbody>
</table>

Figure 2: Game OW-CPA and Game IND-CPA for PKE
2.3 Key Encapsulation Mechanism

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

- \( \text{Gen}(1^\lambda) \): The key generation algorithm \( \text{Gen} \) is a randomized algorithm that takes a security parameter \( \lambda \) as input and outputs a pair of public key and secret key, \((pk, sk)\).

- \( \text{Encap}(pk) \): The encapsulation algorithm \( \text{Encap} \) is a randomized algorithm that takes a public key \( pk \) as input, and outputs a ciphertext \( c \) and a key \( K \in \mathcal{K} \).

- \( \text{Decap}(sk, c) \): The decryption algorithm \( \text{Decap} \) is a deterministic algorithm that takes a secret key \( sk \) and ciphertext \( c \) as input, and outputs a key \( K \in \mathcal{K} \).

Correctness. We say that \( \text{KEM} \) has a correctness error \( \delta \) if

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

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

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

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

<table>
<thead>
<tr>
<th>Game ( \text{IND-CCA} )</th>
<th>( \text{Decap}(c \neq c^*) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: ((pk, sk) \leftarrow \text{Gen}(1^\lambda))</td>
<td>1: return ( \text{Decap}(sk, c) )</td>
</tr>
<tr>
<td>2: ((K_0, c^*) \leftarrow \text{Encap}(pk))</td>
<td></td>
</tr>
<tr>
<td>3: (K_1 \leftarrow \mathcal{K})</td>
<td></td>
</tr>
<tr>
<td>4: (b \leftarrow {0, 1})</td>
<td></td>
</tr>
<tr>
<td>5: (b' \leftarrow A^\text{Decap}(pk, c^*, K_b))</td>
<td></td>
</tr>
<tr>
<td>6: return ([b = b'])</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3: Game \( \text{IND-CCA} \) for \( \text{KEM} \)

3 ACWC\textsubscript{2} Transformation

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

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

1. **Decoding:** There exists an efficient algorithm $\text{Inv}$ such that for all $x \in \mathcal{X}$, $u \in \mathcal{U}$, $\text{Inv}(\text{SOTP}(x, u), u) = x$.

2. **Message-hiding:** For all $x \in \mathcal{X}$, the random variable $\text{SOTP}(x, u)$, for $u \sim \psi_{\mathcal{U}}$, has the same distribution as $\psi_{\mathcal{Y}}$.

3. **Rigid:** For all $u \in \mathcal{U}$ and all $y \in \mathcal{Y}$ encoded with respect to $u$, it holds that $\text{SOTP}(\text{Inv}(y, u), u) = y$.

In contrast to the GOTP defined in [12], SOTP does not need to have an additional randomness-hiding property, which requires that the output $y = \text{SOTP}(x, u)$ follows the distribution $\psi_{\mathcal{Y}}$ and simultaneously does not leak any information about 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 $\delta_s$-rigid, which means that for all $u \in \mathcal{U}$ and all $y \in \mathcal{Y}$ encoded with respect to $u$, $\text{Inv}(y, u) = x$ implies that $\text{SOTP}(x, u) = y$, except with a probability of at most $\delta_s$.

3.2 ACWC$_2$

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

<table>
<thead>
<tr>
<th>Gen$'(1^\lambda)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: $(pk, sk) := \text{Gen}(1^\lambda)$</td>
</tr>
<tr>
<td>2: return $(pk, sk)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enc$'(pk, m \in \mathcal{M}'; r \sim \psi_{\mathcal{R}})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: $M := \text{SOTP}(m, G(r))$</td>
</tr>
<tr>
<td>2: $c := \text{Enc}(pk, M; r)$</td>
</tr>
<tr>
<td>3: return $c$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dec$'(sk, c)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: $M := \text{Dec}(sk, c)$</td>
</tr>
<tr>
<td>2: $r := \text{RRec}(pk, M, c)$</td>
</tr>
<tr>
<td>3: $m := \text{Inv}(M, G(r))$</td>
</tr>
<tr>
<td>4: return $m$</td>
</tr>
</tbody>
</table>

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

Under the condition that $\text{Dec}(sk, c)$ in $\text{Dec}'$ yields the same $M$ as in $\text{Enc}$, the deterministic $\text{RRec}$ and $\text{Inv}$ functions do not affect the correctness error of $\text{PKE}'$. Thus, the factor that determines the success or failure of $\text{Dec}'(sk, c)$ is the result of $\text{Dec}(sk, c)$ in $\text{Dec}'$. This means that the correctness error of $\text{PKE}$ is straightforwardly transferred to that of $\text{PKE}'$, and eventually determined by how randomness $r \in \mathcal{R}$ and message $m \in \mathcal{M}$ are sampled in $\text{PKE}'$. We see that $r$ is drawn according to the distribution $\psi_{\mathcal{R}}$ and $M$ is an SOTP-encoded element in $\mathcal{M}$. Because every output of $G$ is independently $\psi_{\mathcal{U}}$-distributed, we can
expect that the message-hiding property of SOTP makes \( M \) follow the distribution \( \psi_M \) 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 \( \text{PKE}' \), 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 \( \text{PKE}' \) (transformed by \( \text{ACWC}_2 \)) is close to the average-case correctness of \( \text{PKE} \). This is the same idea as in ACWC, and the proof strategy of Theorem 3.2 is essentially the same as that of [12] (Lemma 3.6 therein), except for slight modifications to the message distribution.

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

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

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

\end{theorem}

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

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

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

\[ \delta'(pk, sk) = \mathbb{E}\left[ \max_{m \in \mathcal{M}'} \text{Pr}[\text{Dec}'(sk, \text{Enc}'(pk, m)) \neq m]\right] \]
\[ \leq t + \text{Pr}_G\left[ \max_{m \in \mathcal{M}'} \text{Pr}[\text{Dec}'(sk, \text{Enc}'(pk, m)) \neq m] \geq t\right] \]
\[ \leq t + \text{Pr}_G\left[ \max_{m \in \mathcal{M}'} \text{Pr}[\text{Dec}'(sk, \text{Enc}(pk, \tilde{m}); r) \neq m] \geq t\right], \tag{1} \]

where \( \tilde{m} = \text{SOTP}(m, G(r)) \). 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 \( \mu(pk, sk) := \text{Pr}_{\mathcal{M}', r}[\text{Dec}(sk, \text{Enc}(pk, M; r)) \neq M] \). Then, we can use the helper Lemma 3.4 to argue that

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

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

11
\[ B = \mu(pk, sk) \] holds for all \( m \in \mathcal{M}' \) by the message-hiding property of the SOTP. For all \( m \in \mathcal{M}' \),

\[
\Pr_{r \leftarrow \psi_R, u \leftarrow \psi_U} \left[ g(m, r, u) \in B \right] = \mu(pk, sk)
\]

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

\[
\delta' \leq \mathbb{E} \left[ \mu(pk, sk) + \|\psi_R\| \cdot \sqrt{(c + \ln |\mathcal{M}'|)/2 + e^{-c}} \right]
\]

and setting \( c := -\ln\|\psi_R\| \) 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:

\[
\mathbb{E}[f(X)] = \sum_{x} f(x) \Pr[X = x] \\
= \sum_{f(x) < t} f(x) \Pr[X = x] + \sum_{f(x) \geq t} f(x) \Pr[X = x] \\
\leq \sum_{f(x) < t} t \Pr[X = x] + \sum_{f(x) \geq t} f(x) \Pr[X = x] \\
\leq t + \sum_{f(x) \geq t} f(x) \Pr[X = x] \\
\leq t + \sum_{f(x) \geq t} \Pr[X = x] = t + \Pr[f(X) \geq t]
\]

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

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

\[
\forall m \in \mathcal{M}', \quad \Pr_{r \leftarrow \psi_R, u \leftarrow \psi_U} \left[ g(m, r, u) \in B \right] \leq \mu
\]

(3)
for some $\mu \in [0,1]$. Let $G : \mathcal{R} \to \mathcal{U}$ 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 $\forall m \in \mathcal{M}'$,

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

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

**Proof.** Let us fix a specific $m \in \mathcal{M}'$, and for each $r \in \mathcal{R}$, define $p_r := \Pr_{u \leftarrow \psi_\mathcal{U}} [g(m, r, u) \in B]$. By the assumption of $g$ in Equation (4), we know that $\sum_r \psi_\mathcal{R}(r)p_r \leq \mu$. For each $r$, define a random variable $X_r$ whose value is determined as follows: $G$ chooses a random $u = G(r)$ and then checks whether $g(m, r, G(r)) \in B$; 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 $\mu$. We define the random variable $Y_r = \psi_\mathcal{R}(r)X_r$. Notice that $Y_r \in [0, \psi_\mathcal{R}(r)]$, and $\mathbb{E}[\sum_r Y_r] = \mathbb{E}[\sum_r \psi_\mathcal{R}(r)X_r] = \sum_r \psi_\mathcal{R}(r)p_r \leq \mu$. By the Hoeffding bound, we have for all positive $t$,

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

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

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

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

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

---

**Figure 5:** Game $G_0$ of Theorems 3.5 and 3.7

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

13
\begin{figure}[h]
\centering
\begin{tabular}{|c|c|}
\hline
\textbf{B}(pk, c^*) & \textbf{G}(r) \\
\hline
1: \( \mathcal{L}_G, \mathcal{L}_r := \emptyset \) & 1: if \( \exists (r, u) \in \mathcal{L}_G \) \\
2: \( b \leftarrow \{0, 1\} \) & 2: return \( u \) \\
3: \( (m_0, m_1) \leftarrow \mathcal{A}_0^G(pk) \) & 3: else \\
4: \( b' \leftarrow \mathcal{A}_0^G(pk, c^*) \) & 4: \( u \leftarrow \psi_M \) \\
5: \textbf{for} \( r \in \mathcal{L}_r \) \textbf{do} & 5: \( \mathcal{L}_G := \mathcal{L}_G \cap \{(r, u)\} \) \\
6: \( \textbf{if} \ M \in \mathcal{M} \) & 6: \( \mathcal{L}_r := \mathcal{L}_r \cap \{r\} \) \\
7: \textbf{return} \( M \in \mathcal{M} \) & 7: return \( u \) \\
8: \textbf{return} \( M \leftarrow \psi_M \) & \\
\hline
\end{tabular}
\caption{Adversary \( \mathcal{B} \) for the proof of Theorem 3.5}
\end{figure}

\textbf{Proof.} We show that there exists an algorithm \( \mathcal{B} \) (see Figure 6) which breaks the OW-CPA security of PKE using an algorithm \( \mathcal{A} = (A_0, A_1) \) that breaks the IND-CPA security of ACWC\( _2 \)PKE, SOTP, G.

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

By the definition of the IND-CPA game, we have

\[ \left| \Pr[G_0^A \Rightarrow 1] - \frac{1}{2} \right| = \text{Adv}^{\text{IND-CPA}_{\text{ACWC}_2PKE,SOTP,G}}(\mathcal{A}). \]

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

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

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

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

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

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

which concludes the proof. \( \square \)

\textbf{Lemma 3.6} (Classical O2H, Theorem 3 from the eprint version of [3]). Let \( S \subset \mathcal{R} \) be random. Let \( G \) and \( F \) be random functions satisfying \( \forall r \notin S: G(r) = F(r) \). Let \( z \) be a random classical value \( (S, G, F, z \text{ may have an arbitrary joint distribution}) \). Let \( C \) be a quantum oracle algorithm with query depth \( q_G \), expecting
input $z$. Let $D$ be the algorithm that, on input $z$, samples a uniform $i$ from $\{1, \ldots, q_G\}$, runs $C$ right before its $i$-th query to $F$, measures all query input registers, and outputs the set $T$ of measurement outcomes. Then

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

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

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

and the running time of $B$ is about that of $A$.

**Proof.** To prove this theorem, we use a sequence of games $G_0$ to $G_7$ defined in Figures 5, 6 and 8, and Lemma 5.6. Before applying Lemma 5.6, we change $G_0$ to $G_2$. Subsequently, we apply Lemma 5.6 to $G_2$ and $G_3$. A detailed explanation of the security proof is provided in the following.

GAME $G_0$. $G_0$ (see Figure 5) is the original IND-CPA game with ACWC$_2$ [PKE, SOTP, $G$]. By definition, we have

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

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

$$\Pr[G_0^A \Rightarrow 1] = \Pr[G_1^A \Rightarrow 1].$$
GAME $G_6$. We change the way $G$ is defined in $G_2$. Rather than choosing $G$ uniformly, we choose $F$ and $u$ uniformly and then set $G := F(r := u)$. Here, $G = F(r := u)$ is the same function as $F$, except that it returns $u$ on input $r$. Because the distributions of $G$ and $u$ remain unchanged, we have

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

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

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

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

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

GAME $G_4$ and $G_5$. We define $G_4$ according to Lemma 3.8. In addition, we define $G_5$ by changing the way $M$ is calculated. Instead of computing $M = \text{SOTP}(m_b, u)$, we sample $M \leftarrow \psi_M$. By contrast, in $G_4$, since $u$ is sampled from $\psi_M$ and used only for computing SOTP$(m_b, u)$, the message-hiding property of SOTP shows that $M = \text{SOTP}(m_b, u)$ follows the distribution $\psi_M$. Therefore,

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

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

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

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

$$|\Pr[G_6^A \Rightarrow 1] - \Pr[G_7^A \Rightarrow 1]| \leq \mu.$$
We can observe that in $G_7$, $B$ wins if there exists $r \in T$ such that $m^* := \text{MRec}(pk, r, c^*) \in M$, as the solution of its challenge ciphertext $c^*$. Therefore, we have

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

Combining all (in)equalities and bounds, we have

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

which concludes the proof.

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

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

## 4 IND-CCA Secure KEM from ACWC$_2$

### 4.1 FO Transform with Re-encryption

One can apply the Fujisaki-Okamoto transformation $\text{FO}^{\perp}$ to the IND-CPA secure PKE', as shown in Figure 9, to obtain an IND-CCA secure KEM. Figure 9 shows the resultant KEM := $\text{FO}^{\perp}[\text{PKE}', H] = (\text{Gen}, \text{Encap}, \text{Decap})$, 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 [13], which are stated in Theorems 4.1 and 4.2, respectively. By combining these results with Theorems 4.3 and 4.4, 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.

```
\begin{verbatim}
Gen(1^\lambda)
  1: (pk, sk) := Gen'(1^\lambda)
  2: return (pk, sk)
Encap(pk)
  1: m \leftarrow M
  2: (r, K) := H(m)
  3: c := Enc'(pk, m; r)
  - M := \text{SOTP}(m, G(r))
  - c := Enc(pk, M; r)
  4: return (K, c)
\end{verbatim}

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

Figure 9: KEM = \text{FO}^{\perp}[\text{PKE}', H]
```
Theorem 4.1 (IND-CPA of PKE′ \(\xrightarrow{\text{ROM}}\) IND-CCA of KEM \([12]\)). Let PKE′ be a public key encryption scheme with a message space \(\mathcal{M}\). Let PKE′ have (worst-case) correctness error \(\delta\) and be (weakly) \(\gamma\)-spread. For any adversary \(A\) making at most \(q_D\) decapsulation and \(q_H\) hash queries, against the IND-CCA security of KEM, there exists an adversary \(B\) against the IND-CCA security of PKE′ with

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

where the running time of \(B\) is about that of \(A\).

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

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

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

4.2 FO-Equivalent Transform Without Re-encryption

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

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

Figure 10: KEM = FO\(\perp\)[PKE′, H]

18
However, we demonstrate that FO\(^{-}\) based on ACWC\(_2\) can eliminate the need for ciphertext comparison \(c = \text{Enc}(pk, m'; r'')\) in Decap, and instead replace it with a simpler and more efficient comparison \(r' = r''\).

We denote the new FO\(^{-}\) based on ACWC\(_2\) as \(\overline{\text{FO}}\(^{-}\)\), as shown in Figure 11. In \(\overline{\text{FO}}\(^{-}\), \(r'\) and \(r''\) are values generated during the execution of Decap, where \(r'\) is the output of \(\text{RRec}(pk, M', c)\) and \(r''\) is computed from \(H(m')\). The only change compared to FO\(^{-}\) in Figure 9 is the boxed area, replacing \(c = \text{Enc}(pk, m'; r'')\) with \(r' \neq r''\), while the remaining parts remain the same. By proving that the equality \(c = \text{Enc}(pk, m'; r'')\) is equivalent to the equality \(r' = r''\), we can show that both FO\(^{-}\) and \(\overline{\text{FO}}\(^{-}\) work identically and achieve the same level of IND-CCA security.

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

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

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

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

\[\text{Decap}\]

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

5.1 **Notations**

5.1.1 **Centered Binomial Distribution** \(\psi_k\)

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

- \(b_1, \ldots, b_k \leftarrow \{0, 1\}\), \(b'_1, \ldots, b'_k \leftarrow \{0, 1\}\).

- Return \(\sum_{i=1}^{k} (b_i - b'_i)\).

Hereafter, in our NTRU construction, we use \(\psi_1\) over the set \(\{-1, 0, 1\}\). For a positive integer \(n\), the distribution \(\psi_1^n\) is defined over the set \(\{-1, 0, 1\}^n\), where each element is selected according to \(\psi_1\).
5.1.2 Other Notations

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

5.2 Description of GenNTRU[\( \psi^n_q \)]

Figure 11 defines GenNTRU[\( \psi^n_q \)] relative to the distribution \( \psi^n_q \) over \( R_q \). Since GenNTRU[\( \psi^n_q \)] should be MR and RR for our ACWC2, Figure 11 shows two additional algorithms RRRec and MRec.

We notice that RRRec(h, m, c) is necessary for performing ACWC2 where \( r \) should be recovered from \( c \) once \( m \) is obtained. The RR property guarantees that such a randomness-recovery process works well, because for a ciphertext \( c = \text{Enc}(h, m, r) = hr + m \) we see that RRRec(h, m, c) = \( (c - m) h^{-1} = r \in R \). On the other hand, MRec(h, r, c) is only used for proving IND-CPA security of the ACWC2-transformed scheme. The security analysis requires that for a challenge ciphertext \( c^* = \text{Enc}(h, m^*, r^*) = hr^* + m^* \) the algorithm MRec(h, r^*, c^*) returns the corresponding message \( m^* \) if a queried \( r^* \) was used for \( c^* \). The MR property guarantees that once \( r^* \) is given, MRec(h, r^*, c^*) = \( c^* - hr^* = m^* \in M \).

5.3 Security and Other Properties

5.3.1 Cryptographic Assumptions

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

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

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

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

### 5.3.2 Security Proofs

**Theorem 5.3 (OW-CPA security of GenNTRU[ψ_1^q]).** For any adversary \( A \), there exist adversaries \( B \) and \( C \) such that

\[ \text{Adv}^{\text{OW-CPA}}_{\text{GenNTRU}[ψ_1^q]}(A) \leq \text{Adv}^{\text{NTRU}}_{n,q,ψ_1}(B) + \text{Adv}^{\text{RLWE}}_{n,q,ψ_1}(C). \]

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

GAME \( G_0 \). In \( G_0 \), we have the original OW-CPA game with GenNTRU[ψ_1^q]. By the definition of the advantage function of the adversary \( A \) against the OW-CPA game, we have that

\[ \text{Adv}^{\text{OW-CPA}}_{\text{GenNTRU}[ψ_1^q]}(A) = \Pr[G_0^A \Rightarrow 1]. \]

GAME \( G_1 \). In \( G_1 \), the public key \( h \) in Gen is replaced by \( h \leftarrow R_q \). Therefore, distinguishing \( G_1 \) from \( G_0 \) is equivalent to solving the NTRU_{n,q,ψ_1^q} problem. More precisely, there exists an adversary \( B \) with the same running time as that of \( A \) such that

\[ |\Pr[G_0^A \Rightarrow 1] - \Pr[G_1^A \Rightarrow 1]| \leq \text{Adv}^{\text{NTRU}}_{n,q,ψ_1^q}(B). \]

Since \( h \leftarrow R_q \) is now changed to a uniformly random polynomial from \( R_q \), \( G_1 \) is equivalent to solving an RLWE_{n,q,ψ_1^q} problem. Therefore,

\[ \Pr[G_1^A \Rightarrow 1] = \text{Adv}^{\text{RLWE}}_{n,q,ψ_1^q}(C). \]

Combining all the probabilities completes the proof. \( \square \)

**Lemma 5.4 (Spreadness).** GenNTRU[ψ_1^q] is \( n \)-spread.

**Proof.** For a fixed message \( m \) and ciphertext \( c \), there exists at most one \( r \) such that \( c = \text{Enc}(h, m; r) \). Suppose there exist \( r_1 \) and \( r_2 \) such that \( c = \text{Enc}(h, m; r_1) = \text{Enc}(h, m; r_2) \). Based on this assumption, \( hr_1 + m = hr_2 + m \) holds. By subtracting \( m \) and multiplying \( h^{-1} \) on both sides of the equation, we obtain \( r = r' \). Therefore, there exists at most one \( r \) such that \( c = \text{Enc}(h, m; r) \).

For fixed \( m \), to maximize \( \Pr[\text{Enc}(h, m; r) = c] \), we need to choose \( c \) such that \( c = \text{Enc}(h, m; r) \) for \( r = 0 \). Since there exists only one \( r \) such that \( c = \text{Enc}(h, m; r) \), we have \( \Pr[\text{Enc}(h, m; r) = c] = 2^{-n} \). Since this holds for any \((pk, sk) \leftarrow \text{Gen}(1^λ)\) and \( m \in M \), \( \text{GenNTRU}[ψ_1^q] \) is \( n \)-spread. \( \square \)

### 5.3.3 Average-Case Correctness Error

We analyze the average-case correctness error \( δ \) relative to the distribution \( ψ_M = ψ_R = ψ_1^q \) using the template provided in [22]. We can expand \( cf \) in the decryption algorithm as follows:

\[ cf = (hr + m)f = (3gf^{-1}r + m)(3f' + 1) = 3(gr + mf') + m. \]
For a polynomial \( p \) in \( \mathbb{R}[x] \), let \( p_i \) be the \( i \)-th coefficient of \( p \), and \( |p_i| \) be the absolute value of \( p_i \). Then, \( ((cf) \mod q) \mod 3 = m_i \) if the following inequality holds:

\[
|3(gr + mf') + m|_i \leq \frac{q-1}{2},
\]

where all coefficients of each polynomial are distributed according to \( \psi^n_1 \). Let \( \epsilon_i \) be

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

Then, assuming that each coefficient is independent,

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

Because the coefficients of \( m \) have a size at most one,

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

Therefore,

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

Now, we analyze \( \epsilon'_i = \Pr \left[ |gr + mf'|_i \leq \frac{q-3}{6} \right] \). To achieve this, we need to analyze the distribution of \( gr + mf' \). By following the analysis in [22], we can check that for \( i \in [n/2, n] \), the degree-\( i \) coefficient of \( gr + mf' \) is the sum of \( n \) independent random variables:

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

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

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

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

\[
\rho_i(X) = \begin{cases} 
\sum_{j=-3n}^{3n} \rho_{i,j} X^j = \left( \sum_{j=-3}^{3} \theta_j X^j \right)^n & \text{for } i = [n/2, n - 1], \\
\sum_{j=-(3n-2i)}^{3n-2i} \rho_{i,j} X^j = \left( \sum_{j=-3}^{3} \theta_j X^j \right)^{n-2i} \left( \sum_{j=-2}^{2} \theta'_j X^j \right)^{2i} & \text{for } i = [0, n/2 - 1].
\end{cases}
\] \hfill (9)
where \( \theta_j = \Pr[c = j] \) (distribution is shown in Table 3) and \( \theta'_j = \Pr[c' = j] \) (distribution is shown in Table 4). Let \( \rho_{i,j} \) be the probability that the degree-\( i \) coefficient of \( \mathbf{g} \mathbf{r} + \mathbf{m} \mathbf{l}' \) is \( j \). Then, \( \epsilon'_i \) 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 \( \epsilon'_i \) into Equation (5), we compute the average-case correctness error \( \delta \) of \( \text{GenNTRU}[\psi_1^n] \).

### 5.3.4 Injectivity and rigidity

The injectivity of \( \text{GenNTRU}[\psi_1^n] \) can be easily shown as follows: if there exist two inputs \((\mathbf{m}_1, \mathbf{r}_1)\) and \((\mathbf{m}_2, \mathbf{r}_2)\) such that \( \text{Enc}(\mathbf{h}, \mathbf{m}_1; \mathbf{r}_1) = \text{Enc}(\mathbf{h}, \mathbf{m}_2; \mathbf{r}_2) \), the equality indicates that \( (\mathbf{r}_1 - \mathbf{r}_2) \mathbf{h} + (\mathbf{m}_1 - \mathbf{m}_2) = 0 \), 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

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

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

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

### 6 NTRU+

#### 6.1 Instantiation of SOTP

We introduce SOTP : \( \mathcal{M}' \times U \rightarrow \mathcal{M} \), where \( \mathcal{M}' = \{0, 1\}^n, \mathcal{U} = \{0, 1\}^{2n} \), and \( \mathcal{M} = \{-1, 0, 1\}^n \) relative to distributions \( \psi_{\mathcal{U}} = U^{2n} \) and \( \psi_{\mathcal{M}} = \psi_{\psi_1^n}^\perp \). Figure 12 shows SOTP used for ACWC2. We notice that, following [20], the values of \( y + u_2 \) generated by the Inv algorithm should be checked to determine they are 0 or 1.

<table>
<thead>
<tr>
<th>SOTP (( x \in \mathcal{M}', u \leftarrow U^{2n} ))</th>
<th>Inv (( y \in \mathcal{M}, u \in U^{2n} ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: ( u = (u_1, u_2) \in {0, 1}^n \times {0, 1}^n )</td>
<td>1: ( u = (u_1, u_2) \in {0, 1}^n \times {0, 1}^n )</td>
</tr>
<tr>
<td>2: ( y = (x \oplus u_1) - u_2 \in {-1, 0, 1}^n )</td>
<td>2: if ( y + u_2 \notin {0, 1}^n ), return ( \perp )</td>
</tr>
<tr>
<td>3: return ( y )</td>
<td>3: ( x = (y + u_2) \oplus u_1 \in {0, 1}^n )</td>
</tr>
<tr>
<td>4: return ( x )</td>
<td></td>
</tr>
</tbody>
</table>

Figure 12: SOTP instantiation for NTRU+
Specifically, if there exist two distinct pairs regardless of the message distribution and quantum random oracle models, respectively. Regarding the correctness error, Theorem 6.3 NTRU+ (=KEM) be proven under the assumption that the NTRU n,q,ψ results in the equation ⊥ because of the one-time pad property, particularly for part x ⊕ u1. That is, unless u1 is known, the message x ∈ M is unconditionally hidden from y ∈ M. Similarly, x ⊕ u1 becomes uniformly random over {0, 1}n, regardless of the message distribution ψM', and thus the resulting y follows ψ1 1. In addition, the rigidity of the SOTP is trivial because ln(y, u) = x implies that SOTP(x, u) = y.

6.2 CPA-NTRU+ (=PKE ’)

We obtain CPA-NTRU+ := ACWC2 [GenNTRU [ψ1 1],SOTP, G] by applying ACWC2 from Section 6.1 to GenNTRU[ψ1 1]. Because the underlying GenNTRU[ψ1 1] provides injectivity, MR, and RR properties. Theorems 6.3 and 6.7 provide us with the IND-CPA security of the resulting CPA-NTRU+ in the classical and quantum random oracle models, respectively. Regarding the correctness error, Theorem 6.2 shows that the worst-case correctness error of CPA-NTRU+ and the average-case correctness error of GenNTRU[ψ1 1] differ by the amount of ∆ = ∥ψ∥(1 + √(ln |M′| − ln∥ψ∥))/2), where ψ and M′ are specified by ψ1 1 and {0, 1}n, respectively. For instance, when n = 768, we obtain about ∆ = 2−1083.

\[
\text{Gen}(1^λ) \quad \text{Enc}'(pk, m ∈ \{0, 1\}^n; r ← ψ1 1) \\
1: (pk, sk) := \text{GenNTRU}[ψ1 1].\text{Gen}(1^λ) \\
\quad - f', g ← ψ^n_1 \\
\quad - f = 3f' + 1 \\
\quad - \text{if } f', g \text{ are not invertible in } Rq \text{ then restart} \\
\quad - (pk, sk) = (h = 3gf^{-1} \mod q, f) \\
2: \text{return } (pk, sk) \\
\text{Dec}(sk, c) \\
1: m = \text{GenNTRU}[ψ1 1].\text{Dec}(sk, c) \\
\quad - m = (cf \mod q) \mod 3 \\
2: r = RRec(pk, c, m) \\
\quad - r = (c - m)h^{-1} \\
3: m = \text{ln}(m, G(r)) \\
4: \text{return } m
\]

Figure 13: CPA-NTRU+

Spreadness and Injectivity Properties of CPA-NTRU+. To achieve IND-CCA security of the transformed KEM via FO⊥, we need to show the spreadness and injectivity of CPA-NTRU+. The spreadness can be easily obtained by combining Lemma 6.8 with Lemma 6.3. Next, the injectivity of CPA-NTRU+ can also be proven under the assumption that the NTRU0,q,ψ problem is infeasible, analogous to GenNTRU[ψ1 1]. Specifically, if there exist two distinct pairs (m1, r1) and (m2, r2) such that Enc′(pk, m1; r1) = Enc′(pk, m2; r2), this results in the equation hr1 + m1 = hr2 + m2, where m1 = SOTP(m1, G(r1)) and m2 = SOTP(m2, G(r2)). In this case, we have two short polynomials: r1 − r2 and m1 − m2, which can be a solution of the approximate SVP (of length at most 2√Ł) over L⊥.

6.3 NTRU+ (=KEM)

Finally, we achieve IND-CCA secure KEM by applying FO⊥ to CPA-NTRU+. We denote such KEM by NTRU+ := FO⊥[CPA-NTRU+, H]. Figure 14 shows the resultant NTRU+, which is the basis of our implementation in the next section. By combining Theorems 6.1, 6.2 and Lemma 6.3, we can achieve
IND-CCA security of NTRU+. As for the correctness error, NTRU+ preserves the worst-case correctness error of the underlying CPA-NTRU+.

\begin{align*}
\text{Gen} (1^\lambda) & \\
1: & f', g \leftarrow \psi_1^n \\
2: & f = 3f' + 1 \\
3: & \text{if } f, g \text{ are not invertible in } R_q \\
4: & \text{restart} \\
5: & \text{return } (pk, sk) = (h = 3gf^{-1}, f) \\
\text{Encap} (pk) & \\
1: & m \leftarrow \{0, 1\}^n \\
2: & (r, K) = H(m) \\
3: & m = \text{SOTP} (m, G(r)) \\
4: & c = hr + m \\
5: & \text{return } (c, K) \\
\text{Decap} (sk, c) & \\
1: & m = (cf \mod q) \mod 3 \\
2: & r = (c - m)h^{-1} \\
3: & m = \text{Inv}(m, G(r)) \\
4: & (r', K) = H(m) \\
5: & \text{if } r = r' \\
6: & \text{return } K \\
7: & \text{else} \\
8: & \text{return } \bot
\end{align*}

Figure 14: NTRU+

7 Parameters and Security Analysis

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

<table>
<thead>
<tr>
<th>Scheme</th>
<th>classical</th>
<th>quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LWE</td>
<td>NTRU</td>
</tr>
<tr>
<td>NTRU+576</td>
<td>115</td>
<td>114</td>
</tr>
<tr>
<td>NTRU+768</td>
<td>164</td>
<td>164</td>
</tr>
<tr>
<td>NTRU+864</td>
<td>189</td>
<td>189</td>
</tr>
<tr>
<td>NTRU+1152</td>
<td>263</td>
<td>265</td>
</tr>
</tbody>
</table>

Table 5: Concrete Security Level relative to LWE and NTRU problems

To estimate the concrete security level of NTRU+, we analyze the hardness of the two problems \( \text{RLWE}_{n,q,\psi_1^n} \) and \( \text{NTRU}_{n,q,\psi_1^n} \) based on each parameter set. For the RLWE problem, we employ the Lattice estimator [1], which uses the BKZ lattice reduction algorithm [7] for the best-known lattice attacks such as the primal [2] and dual [21] attacks. Next, for the NTRU problem, we use the NTRU estimator provided by the finalist NTRU [6], which is based on the primal attack and Howgrave-Graham’s hybrid attack [16] 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 [12] 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 5 shows the resulting security levels relative to the RLWE and NTRU problems, depending on each NTRU parameter set.

8 Implementation of NTT

We employ NTT to implement polynomial multiplication within a ring. To achieve NTT over the polynomial ring \( \mathbb{Z}_q[x]/\langle x^n - x^{n/2} + 1 \rangle \) where \( n = 2^a3^b \), we utilize three distinct types of NTT layers: Radix-2 NTT, Radix-2 NTT for cyclotomic trinomial, and Radix-3 NTT. Our approach begins with factoring the polynomial through the adaptation of the Radix-2 NTT for cyclotomic trinomial layer, as initially introduced in [22]. This transformation establishes an isomorphism between two polynomial rings: \( \mathbb{Z}_q[x]/\langle x^n - x^{n/2} + 1 \rangle \) and the product ring \( \mathbb{Z}_q[x]/\langle x^{n/2} - \zeta \rangle \times \mathbb{Z}_q[x]/\langle x^{n/2} - (1 - \zeta) \rangle \), where \( \zeta \) denotes a primitive sixth root of unity modulo \( q \). Subsequently, we employ Radix-3 NTT layers to successively factor each partitioned polynomial until the desired polynomial degree is reached. Finally, we use Radix-2 NTT layers until the polynomials reach inertia degree two or three. Adopting Radix-3 NTT layers before Radix-2 NTT layers aims to minimize the size of the precomputed tables required for polynomial multiplication in the NTT form.

<table>
<thead>
<tr>
<th>( n )</th>
<th>( q )</th>
<th>Radix-2 for cyclotomic trinomial</th>
<th>Radix-3</th>
<th>Radix-2</th>
<th>inertia degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>576</td>
<td>3457</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>768</td>
<td>3457</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>864</td>
<td>3457</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>1152</td>
<td>3457</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 6: Combinations of NTT layers

8.1 Radix-3 NTT Layer

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

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

Naively, these equations might appear to require \( 6n \) multiplications and \( 6n \) additions, using six predefined values: \( \alpha, \alpha^2, \beta, \beta^2, \gamma, \text{ and } \gamma^2 \). Nevertheless, we can significantly reduce this computational load to \( 4n \)
multiplications, 5\(n\) additions, and \(n\) subtractions, by employing only four predefined values: \(\alpha\), \(\alpha^2\), \(\omega\), and \(\omega^2\), as delineated in Algorithm I.

Algorithm 1 Radix-3 NTT layer

**Require:** \(a(x) = a_0(x) + a_1(x)x^n + a_2(x)x^{2n} \in \mathbb{Z}_q[x]/\langle x^{3n} - \zeta^3 \rangle\)

**Ensure:** \((\hat{a}_0(x), \hat{a}_1(x), \hat{a}_2(x)) \in \mathbb{Z}_q[x]/\langle x^n - \alpha \rangle \times \mathbb{Z}_q[x]/\langle x^n - \beta \rangle \times \mathbb{Z}_q[x]/\langle x^n - \gamma \rangle\)

\[
\begin{align*}
1: & \quad t_1(x) = a_1(x) \alpha \\
2: & \quad t_2(x) = a_2(x) \alpha^2 \\
3: & \quad t_3(x) = t_1(x) \omega \quad \text{//} \quad a_1(x) \beta \\
4: & \quad t_4(x) = t_2(x) \omega^2 \quad \text{//} \quad a_2(x) \beta^2 \\
5: & \quad t_5(x) = t_1(x) + t_2(x) \quad \text{//} \quad a_1(x) \alpha + a_2(x) \alpha^2 \\
6: & \quad t_6(x) = t_3(x) + t_4(x) \quad \text{//} \quad a_1(x) \beta + a_2(x) \beta^2 \\
7: & \quad t_7(x) = t_5(x) + t_6(x) \quad \text{//} \quad -a_1(x) \gamma - a_2(x) \gamma^2 \\
8: & \quad \hat{a}_0(x) = a_0(x) + t_5(x) \quad \text{//} \quad a_0(x) + a_1(x) \alpha + a_2(x) \alpha^2 \\
9: & \quad \hat{a}_1(x) = a_0(x) + t_6(x) \quad \text{//} \quad a_0(x) + a_1(x) \beta + a_2(x) \beta^2 \\
10: & \quad \hat{a}_2(x) = a_0(x) - t_7(x) \quad \text{//} \quad a_0(x) + a_1(x) \gamma + a_2(x) \gamma^2 \\
11: & \quad \textbf{return} \quad (\hat{a}_0(x), \hat{a}_1(x), \hat{a}_2(x))
\end{align*}
\]

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

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

Naively, these equations might appear to require \(6n\) multiplications and \(6n\) additions, using six predefined values: \(\alpha^{-1}, \alpha^{-2}, \beta^{-1}, \beta^{-2}, \gamma^{-1}\), and \(\gamma^{-2}\). Nevertheless, we can significantly reduce this computational load to \(4n\) multiplications, \(5n\) additions, and \(n\) subtractions, by employing only four predefined values: \(\alpha^{-1}, \alpha^{-2}, \omega\), and \(\omega^2\), as described in in Algorithm II.

Algorithm 2 Radix-3 Inverse NTT layer

**Require:** \((\hat{a}_0(x), \hat{a}_1(x), \hat{a}_2(x)) \in \mathbb{Z}_q[x]/\langle x^n - \alpha \rangle \times \mathbb{Z}_q[x]/\langle x^n - \beta \rangle \times \mathbb{Z}_q[x]/\langle x^n - \gamma \rangle\)

**Ensure:** \(3a(x) = 3a_0(x) + 3a_1(x)x^n + 3a_2(x)x^{2n} \in \mathbb{Z}_q[x]/\langle x^{3n} - \zeta^3 \rangle\)

\[
\begin{align*}
1: & \quad t_1(x) = \hat{a}_1(x) + \hat{a}_2(x) \\
2: & \quad t_2(x) = \hat{a}_1(x) \omega^2 \\
3: & \quad t_3(x) = \hat{a}_2(x) \omega \quad \text{//} \quad \hat{a}_1(x) \omega^{-2} \\
4: & \quad t_4(x) = t_2(x) + t_3(x) \\
5: & \quad t_5(x) = t_1(x) + t_4(x) \quad \text{//} \quad \hat{a}_1(x) \omega^{-1} + a_2(x) \omega^{-2} \\
6: & \quad t_6(x) = \hat{a}_0(x) + t_4(x) \quad \text{//} \quad \hat{a}_0(x) + \hat{a}_1(x) \omega^{-1} + a_2(x) \omega^{-2} \\
7: & \quad t_7(x) = \hat{a}_0(x) - t_5(x) \quad \text{//} \quad \hat{a}_0(x) + a_1(x) \omega^{-2} + a_2(x) \omega^{-4} \\
8: & \quad 3a_0(x) = \hat{a}_0(x) + t_1(x) \quad \text{//} \quad \hat{a}_0(x) + \hat{a}_1(x) + \hat{a}_2(x) \\
9: & \quad 3a_1(x) = t_6(x) \alpha^{-1} \quad \text{//} \quad \hat{a}_0(x) \alpha^{-1} + \hat{a}_1(x) \beta^{-1} + \hat{a}_2(x) \gamma^{-1} \\
10: & \quad 3a_2(x) = t_7(x) \alpha^{-2} \quad \text{//} \quad a_0(x) \alpha^{-2} + a_1(x) \beta^{-2} + a_2(x) \gamma^{-2} \\
11: & \quad \textbf{return} \quad 3a(x) = 3a_0(x) + 3a_1(x)x^n + 3a_2(x)x^{2n}
\end{align*}
\]
9 Performance Analysis

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

Typically, the number of cycles increases as $n$ increases, but this is not the case for the key generation of NTRU+ in the AVX2 implementation. NTRU+768 and NTRU+864 generate public/secret keys faster than NTRU+576. The main reason for this is because the key generation cycles are dominated by whether the inverse of a random polynomial exists. For NTRU+576 whose ring structure can be decomposed with a product of degree-1 polynomials, there exists the inverse of a random polynomial with probability $(1 - (1/3457))^{576} \approx 0.85$. However, for NTRU+768 and NTRU+864 whose ring structures can only be decomposed with a product of degree-2 and degree-3 polynomials, respectively, the probability becomes $(1 - (1/3457^2))^{768/2} \approx 0.99$ in NTRU+768 and $(1 - (1/3457^3))^{864/3} \approx 0.99$ in NTRU+864. Since the key generation algorithm continues until it finds two invertible polynomials corresponding to $f$ and $g$, we can expect that the key generation of NTRU+576 requires more CPU cycles compared to that of NTRU+768 and NTRU+864. In addition, the reason why NTRU+864 generates keys faster than NTRU+768 in the AVX2 implementation can be found in the number of modular inversions required to calculate a polynomial inverse. Indeed, the number of modular inversions is $384(=768/2)$ in NTRU+768 and $288(=864/3)$ in NTRU+864, which explains the inconsistent patterns of the key generation cycles between NTRU+768 and NTRU+864.

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

When comparing KYBER (KYBER-90s) and NTRU+, the bandwidth of NTRU+ is slightly larger than that of KYBER at similar security levels. This is because KYBER uses a rounding technique to reduce the size of a ciphertext. In terms of efficiency, it would be fairer to compare KYBER-90s (rather than KYBER) and NTRU+, because both schemes commonly use AES256-CTR as an eXtendable-Output Function (XOF) to expand randomness from a seed. We notice that KYBER uses SHAKE-128 as its XOF. Generally, SHAKE-128 is faster than AES256-CTR in the reference implementation, but the situation is reversed in the AVX2 implementation due to the existence of assembly instructions designed for AES. Table 7 shows that, at similar security levels, the key generation of NTRU+ is slower than that of KYBER-90s in the reference implementation. However, the encapsulation and decapsulation of NTRU+ is faster than that of KYBER-90s in both the reference and AVX2 implementations.
<table>
<thead>
<tr>
<th>Scheme</th>
<th>security level</th>
<th>$n$</th>
<th>$q$</th>
<th>$pk$</th>
<th>$ct$</th>
<th>$sk$</th>
<th>$\log_2 \delta'$</th>
<th>reference (Gen, Encap, Decap)</th>
<th>AVX2 (Gen, Encap, Decap)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NTRU+576</td>
<td>classical</td>
<td>114</td>
<td>105</td>
<td>576</td>
<td>3457</td>
<td>864</td>
<td>864</td>
<td>1728</td>
<td>18, 14, 12</td>
</tr>
<tr>
<td></td>
<td>quantum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NTRU+768</td>
<td>classical</td>
<td>164</td>
<td>149</td>
<td>768</td>
<td>3457</td>
<td>1152</td>
<td>1152</td>
<td>2304</td>
<td>16, 18, 16</td>
</tr>
<tr>
<td></td>
<td>quantum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NTRU+864</td>
<td>classical</td>
<td>189</td>
<td>172</td>
<td>864</td>
<td>3457</td>
<td>1296</td>
<td>1296</td>
<td>2592</td>
<td>14, 19, 18</td>
</tr>
<tr>
<td></td>
<td>quantum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NTRU+1152</td>
<td>classical</td>
<td>263</td>
<td>241</td>
<td>1152</td>
<td>3457</td>
<td>1728</td>
<td>1728</td>
<td>3456</td>
<td>43, 26, 24</td>
</tr>
<tr>
<td></td>
<td>quantum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KYBER512</td>
<td>classical</td>
<td>118</td>
<td>108</td>
<td>512</td>
<td>3329</td>
<td>800</td>
<td>768</td>
<td>1632</td>
<td>27, 35, 26</td>
</tr>
<tr>
<td></td>
<td>quantum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KYBER512-90s</td>
<td>classical</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>quantum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KYBER768</td>
<td>classical</td>
<td>182</td>
<td>165</td>
<td>768</td>
<td>3329</td>
<td>1184</td>
<td>1088</td>
<td>2400</td>
<td>43, 55, 42</td>
</tr>
<tr>
<td></td>
<td>quantum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KYBER768-90s</td>
<td>classical</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>quantum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KYBER1024</td>
<td>classical</td>
<td>255</td>
<td>231</td>
<td>1024</td>
<td>3329</td>
<td>1568</td>
<td>1568</td>
<td>3168</td>
<td>61, 80, 64</td>
</tr>
<tr>
<td></td>
<td>quantum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KYBER1024-90s</td>
<td>classical</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>quantum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ntruhps2048509</td>
<td></td>
<td>106</td>
<td>96</td>
<td>509</td>
<td>2048</td>
<td>699</td>
<td>699</td>
<td>935</td>
<td>195, 84, 33</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ntruhrss701</td>
<td></td>
<td>137</td>
<td>124</td>
<td>701</td>
<td>8192</td>
<td>1138</td>
<td>1138</td>
<td>1450</td>
<td>257, 60, 51</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ntruhps2048677</td>
<td></td>
<td>144</td>
<td>131</td>
<td>677</td>
<td>2048</td>
<td>930</td>
<td>930</td>
<td>1234</td>
<td>307, 114, 49</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ntruhps4096821</td>
<td></td>
<td>178</td>
<td>162</td>
<td>821</td>
<td>4096</td>
<td>1230</td>
<td>1230</td>
<td>1590</td>
<td>417, 136, 62</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

\((\text{Gen}, \text{Encap}, \text{Decap})\): K cycles of reference or AVX2 implementations.
References


