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

Jonghyun Kim† Jong Hwan Park‡

December 20, 2022

Abstract

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

In this work, we propose a new NTRU-based key encapsulation mechanism (KEM), called NTRU+, which overcomes nearly all existing drawbacks. NTRU+ is constructed based on two new generic transformations: ACWC₂ and FO⊥ (a variant of the Fujisaki-Okamoto transform). ACWC₂ is used to easily achieve worst-case correctness error, and FO⊥ is used to achieve chosen-ciphertext security without re-encryption. ACWC₂ and FO⊥ are all defined using a randomness-recovery algorithm and an encoding method. In particular, our simple encoding method, semi-generalized one-time pad (SOTP), allows us to sample a message from a natural bit-string space with an arbitrary distribution. We provide four parameter sets for NTRU+ and implementation results using NTT-friendly rings over cyclotomic trinomials.

1 Introduction

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

*This work is submitted to ‘Korean Post-Quantum Cryptography Competition’ (www.kpqc.or.kr).
†Korea University, Seoul, Korea. Email: yoswuk@korea.ac.kr.
‡Sangmyung University, Seoul, Korea. Email: jhpark@smu.ac.kr.
The central design principle of NTRU is described over a ring $R_q = \mathbb{Z}_q[x]/(f(x))$, where $q$ is a positive integer and $f(x)$ is a polynomial. The public key is generated as $h = pg/(pf' + 1)$ if $g$ and $f'$ are sampled according to a narrow distribution $\psi$, $p$ is a positive integer smaller than $q$ (e.g., 3), and the corresponding private key is $f = pf' + 1$. To encrypt message $m$ sampled from message space $\mathcal{M}'$, one creates two polynomials $r$ and $m$, with coefficients drawn from a narrow distribution $\psi$, and computes ciphertext $c = hr + m$ in $R_q$. An (efficient) encoding method may be used by which $m \in \mathcal{M}'$ is encoded into $m$ and $r \in R_q$. Alternatively, it is possible to directly sample $m$ and $r$ from $\psi$, where $m$ is considered a message to be encrypted. To decrypt ciphertext $c$, one computes $cf$ in $R_q$, recovers $m$ by deriving the value of $f'$ modulo $p$, and (if necessary) decodes $m$ to obtain 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, the NTRU (as ordinary public key encryption schemes) must guarantee extremely negligible worst-case correctness error, because otherwise information about the private key may be leaked by adversarial decryption queries (e.g., attacks [15, 5] against lattice-based encryption schemes). Roughly speaking, worst-case correctness error means the probability that decryption fails for any ciphertext that can be generated with all possible messages and randomnesses in their respective spaces. The worst-case correctness error considers that an adversary, $A$, can maliciously choose messages and randomnesses, without sampling normally according to their original distributions (if possible). In the case of NTRU, the failure to decrypt a specific ciphertext $c = hr + m$ provides $A$ 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 the NTRU, two approaches can be used for achieving worst-case correctness error. One 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 [4], 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 [9] is to use an encoding method by which a message $m \in \mathcal{M}'$ is used as a randomness to sample $m$ and $r$ according to $\psi$. Under the Fujisaki-Okamoto (FO) transform [11], 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 worst-case correctness error that is close to an average-case error.

Based on the aforementioned observation, [9] proposed generic (average-case to worst-case) transformation\footnote{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.} that make the average-case correctness error of an underlying scheme nearly close to worst-case error of a transformed scheme. One of their transformations (denoted by ACWC) is based on an encoding method called generalized one-time pad (denoted by GOTP). Roughly speaking, GOTP works as follows: when $m$ is split into two polynomials $m_1||m_2$, a message $m \in \mathcal{M}'$ is first used to sample $r$ and $m_1$ according to $\psi$, and $m_2 = \text{GOTP}(m, G(m_1))$ using a hash function $G$. If the GOTP acts as a sampling function\footnote{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.},
Table 1: Comparison to previous NTRU constructions

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></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 ← {−1, 0, 1}^n</td>
<td>m ← {−1, 0, 1}^λ</td>
<td>m ← {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⊥</td>
<td>ACWC_2 + FO⊥</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>

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

wherein output follows \(\psi\), m and r are created from m following \(\psi\), which can be verified in decryption using the FO transform. However, an ACWC based on the GOTP has two disadvantages in terms of security reduction and message distribution. First, [9] showed that ACWC converts a one-way CPA (OW-CPA) secure underlying scheme into a transformed scheme that is also OW-CPA secure, although their security reduction is loose by causing a security loss factor of \(q_{\text{G}}\), the number of random oracle queries. Second, ACWC requires m to be sampled from message space \(\mathcal{M}'\) according to a special distribution \(\psi'\). Indeed, the NTRU instantiation from ACWC, called ‘NTRU-B’ [9], requires that m should be chosen uniformly at random from \(\mathcal{M}' = \{-1, 0, 1\}^{\lambda}\) for some integer \(\lambda\). Notably, it is difficult to generate exactly uniform numbers from \(\{-1, 0, 1\}\) in a constant time, because of rejection sampling.

1.1 Our Results

We propose a new practical NTRU construction, called ‘NTRU+’, which overcomes the two drawbacks of the previous ACWC. To achieve our goal, we provide a new generic ACWC transformation (denoted by ACWC_2) that works with a simple encoding method. Using ACWC_2, NTRU+ achieves worst-case correctness error that is close to an average-case error of the underlying NTRU. In addition, NTRU+ requires m to be drawn from \(\mathcal{M}' = \{0, 1\}^n\) (for a polynomial degree n) following an arbitrary distribution (with high min-entropy) and is proven to be tightly secure under the same assumptions of NTRU and RLWE. To achieve chosen-ciphertext security, NTRU+ relies on a novel FO-equivalent transform without re-encryption, which makes the decryption algorithm of NTRU+ faster than in the ordinary FO transform. In terms of efficiency, we use the idea of [13] to apply Number Theoretic Transform (NTT) [17] 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 1 lists the main differences between previous NTRU constructions [4] [9] and NTRU+. In the following section, we describe our technique, focusing on these differences.

ACWC_2 Transformation with Tight Reduction. ACWC_2 is a new generic transformation that allows for the aforementioned average-case to worst-case correctness error conversion. However, to apply ACWC_2,
an underlying scheme is required to have 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 [9], SOTP works as follows: first, a message $m \in M'$ is used to sample $r$ based on $\psi$, and then $\mathbf{m} = \text{SOTP}(m, G(r))$ whose coefficients follow $\psi$, using a hash function $G$. In decrypting a ciphertext $c = \text{Enc}(pk, \mathbf{m}; r)$ under a public key $pk$, $\mathbf{m}$ is recovered by a normal decryption algorithm and, using $\mathbf{m}$, $r$ is also recovered by a randomness-recovery algorithm, and then an inverse of SOTP using $G(r)$ and $\mathbf{m}$ gives $m$.

The MR property of an underlying scheme allows us to show that, without causing security loss, ACWC$_2$ transforms an OW-CPA secure scheme into a chosen-plaintext (IND-CPA) secure scheme. The proof idea is simple: unless an IND-CPA adversary $A$ 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 the SOTP. However, whenever $A$ queries $r_i$ to $G$ for $i = 1, \cdots, q_G$, 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.

**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) $(\mathbf{m}, r)$ are correctly generated from $m$ and (2) a (decrypted) ciphertext $c$ is correctly encrypted from $(\mathbf{m}, r)$. 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 [14, 3] and rigidity [2] properties of the underlying schemes.

---

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

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

**Simple SOTP with More Convenient Sampling Distributions.** As mentioned previously, ACWC₂ is based on an efficient construction of SOTP that takes \( m \) and \( G(r) \) 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) \( \psi_{\delta} \) with \( \delta = 1 \), 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 \( \tilde{m} = (m + y_1) - y_2 \) 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., \( \psi_2 \)) by summing up and subtracting more uniformly random bits.

**NTT-Friendly Rings Over Cyclotomic Trinomials.** NTRU+ is instantiated over a polynomial ring \( R_q = \mathbb{Z}_q[x]/\langle f(x) \rangle \), where \( f(x) = x^n - x^{n/2} + 1 \) is a cyclotomic trinomial of degree \( n = 2^4 \cdot 3 \). [18] showed that, with appropriate parametrization of \( n \) and \( q \), such a ring can also provide NTT operation essentially as fast as that over a ring \( R_q = \mathbb{Z}_q[x]/\langle x^n + 1 \rangle \). Moreover, because the choice of a cyclotomic trinomial is moderate, it provides more flexibility to satisfy a certain level of security. Based on these results, we choose four parameter sets for NTRU+, where the polynomial degree \( n \) of \( f(x) = x^n - x^{n/2} + 1 \) is set to be 576, 768, 864, and 1152, and the modulus \( q \) is 3457 for all cases. Table 2 lists the comparison results between finalist NTRU [4], Kyber [22] and NTRU+ in terms of security and efficiency. The (classical) security of NTRU+ is computed using the security analysis script of Kyber [22], considering that all coefficients of each polynomial \( f', g, r, \) and \( m \) are drawn according to the centered binomial distribution \( \psi_1 \), and the implementation results in Table 2 are estimated with AVX2 optimizations. We can observe that NTRU+

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

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

**sec(c): classical security. \( n \): polynomial degree of the ring. \( q \): modulus. \( (pk, ct, sk) \): bytes. \( \log_2 \delta \): worst-case (or perfect) correctness error. \( \psi_1 \): \( \psi \) also expect that, similar to the case of \( \psi_1 \), \( \psi \) 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) \( \psi_{\delta} \) with \( \delta = 1 \), 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 \( \tilde{m} = (m + y_1) - y_2 \) 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., \( \psi_2 \)) by summing up and subtracting more uniformly random bits.**
outperforms the NTRU and Kyber at a similar security level.

1.2 Related Works

The first round NTRUEncrypt [23] submission to the NIST PQC standardization process was an NTRU-based encryption scheme combined with the NAEP padding method [16]. Roughly speaking, NAEP is similar to our SOTP, but the difference is that it does not completely encode $m$ to prevent an adversary $A$ from choosing $m$ maliciously. This follows from the fact that $m := \text{NAEP}(m, G(hr))$ is generated by subtracting two $n$-bit strings $m$ and $G(hr)$ from each other, that is, $m - G(hr)$ by bitwise subtraction, and then assigning them to the coefficients of $m$. Since $m$ can be maliciously chosen by $A$ in NTRUEncrypt, so can be the resulting $m$, regardless of the hash value $G(hr)$.

The finalist NTRU [4], which is a merger of two submissions NTRU-HRSS [21] and NTRU-HPS [23], was submitted as a key encapsulation mechanism (KEM) that provides four parameter sets for perfect correctness. To achieve chosen-ciphertext security, [4] relied on a variant of the SXY [20] conversion, which also avoids re-encryption during decapsulation. Similar to NTRU+, the SXY variant requires the rigidity [2] of an underlying scheme and uses the notion of deterministic public key encryption (DPKE) where $(m, r)$ are all recovered as a message during decryption. In the NTRU construction, recovering $r$ is conceptually the same as the existence of the randomness-recovery algorithm Recover$^r$. Instead of removing re-encryption, the finalist NTRU needs to check whether $(m, r)$ are selected correctly from predefined distributions.

In 2019, Lyubashevsky et al. [18] proposed an efficient NTRU-based KEM, called NTTRU, by applying NTT to the ring defined by a cyclotomic trinomial $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$. [18] showed that NTT can be applied to the ring $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$, using the fact that $x^n - x^{n/2} + 1$ can be factored as $(x^{n/2} - \zeta) \times (x^{n/2} - (1 - \zeta))$ where $\zeta$ is the primitive sixth root of unity modulo $q$. NTTRU was based on the Dent transformation without any encoding method, which resulted in about $2^{-13}$ worst-case correctness error even with $2^{-1230}$ average-case errors. To overcome this large difference, NTTRU was modified to reduce the message space of an underlying scheme, while increasing the size of a ciphertext. This modification is later generalized to the generic ACWC$_0$ in [9].

In 2021, Duman et al. [9] proposed two generic transformations ACWC$_0$ and ACWC by which an average-case correctness error of an underlying scheme is almost equal to worst-case one of a transformed scheme. In particular, ACWC introduced GOTP as an encoding method that prevents $A$ from choosing $m$ adversarially. ACWC$_0$ is simple but requires about 32 bytes of ciphertext expansion, whereas ACWC does not need to expand the ciphertext size. [9] analyzed security from ACWC$_0$ and ACWC in the classical and quantum random oracle model. As mentioned above, however, their NTRU instantiation from ACWC has a drawback that requires a message $m$ to be chosen from a uniformly-random distribution over $\mathcal{M} = \{-1, 0, 1\}^3$. Indeed, it was an open problem [9] to construct a new transformation that permits a different, more easily sampled distribution of a message, while relying on the same security assumptions.

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

$$m := \text{NAEP}(m, G(hr))$$
2 Preliminaries

2.1 Public Key Encryption and Related Properties

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

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

Correctness. We say that $PKE$ has (worst-case) correctness error $δ$ \cite{14} if

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

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

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

where the expectation is taken over $(pk, sk) \leftarrow Gen(1^λ)$, the choice of the random oracles involved (if any), and $m \leftarrow ψ_M$.

Injectivity. \cite{14,3} We say that $PKE$ has injectivity error $μ$ if for all $(pk, sk) \leftarrow Gen(1^λ)$ and $m, m' \in M$ and $r, r' \in R$, we have that

$$\Pr[c = c' \land (m, r) \neq (m', r')|c \leftarrow Enc(pk, m; r) \land c' \leftarrow Enc(pk, m'; r')] \leq μ,$$

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

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

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

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

$$γ(pk, m) ≥ γ.$$

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

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

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

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

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

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

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

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

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 \( \mathcal{M} \). Onewayness under chosen-plaintext attacks (OW-CPA) for message distribution \( \psi_\mathcal{M} \) is defined via the game OW-CPA, which is shown in Figure 2, and the advantage function of adversary \( A \) is

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

**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 \( A \) is

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

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

<table>
<thead>
<tr>
<th>Game IND-CPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: ((pk, sk) \leftarrow \text{Gen}(1^\lambda))</td>
</tr>
<tr>
<td>2: ((m_0, m_1) \leftarrow A_0(pk))</td>
</tr>
<tr>
<td>3: (b \leftarrow {0, 1})</td>
</tr>
<tr>
<td>4: (c^* \leftarrow \text{Enc}(pk, m_b))</td>
</tr>
<tr>
<td>5: (b' \leftarrow A_1(pk, c^*))</td>
</tr>
<tr>
<td>6: \textbf{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 correctness error $\delta$ if

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

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 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 $\text{IND}_K^\text{CCA}_{\text{KEM}}$, as shown in Figure 3, and the advantage function of adversary $A$ is as follows:

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

![Game IND-CCA for KEM](image)

Figure 3: Game IND-CCA for KEM

3 ACWC$_2$ Transformation

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

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

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

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

2. Message-hiding: For all \( x \in \mathcal{X} \), the random variable \( \text{SOTP}(x, u) \), for \( u \leftarrow \psi_U \), has the same distribution as \( \psi_Y \).

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

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

3.2 ACWC

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 \( \psi_M \) and \( \psi_R \), 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 \( \psi_M' \) and \( \psi_R' \) are associated distributions. Let \( \text{SOTP} : \mathcal{M}' \times \mathcal{U} \rightarrow \mathcal{M} \) be a semi-generalized one-time pad for distributions \( \psi_U \) and \( \psi_M \), and let \( G : \mathcal{R} \rightarrow \mathcal{U} \) be a hash function. Assuming that \( \mathcal{R} = \mathcal{R}' \) and \( \psi_R = \psi_R' \), then \( \text{PKE}' = \text{ACWC}_2[\text{PKE}, \text{SOTP}, G] \) is described in Figure 4.

```
Gen'(1^\lambda)
1: (pk, sk) := Gen(1^\lambda)
2: return (pk, sk)

Enc'(pk, m \in M'; r \leftarrow \psi_R)
1: M := SOTP(m, G(r))
2: c := Enc(pk, M; r)
3: return c

Dec'(sk, c)
1: M := Dec(sk, c)
2: r := Recover'(pk, M, c)
3: m := Inv(M, G(r))
4: return m
```

Figure 4: ACWC_2[\text{PKE}, \text{SOTP}, G]

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

**Theorem 3.2 (Average-Case to Worst-Case Correctness error).** Let PKE be RR and have a randomness space $\mathcal{R}$ relative to the distribution $\psi_\mathcal{R}$. Let SOTP : $\mathcal{M}' \times \mathcal{U} \to \mathcal{M}$ be a semi-generalized one-time pad (for distributions $\psi_\mathcal{U}$, $\psi_\mathcal{M}$) and let $G : \mathcal{R} \to \psi_\mathcal{U}$ be a random oracle. If PKE is $\delta$-average-case-correct, then PKE' := ACWC$_2[\text{PKE, SOTP, } G]$ 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 $\|\psi_\mathcal{R}\| := \sqrt{\sum_r \psi_\mathcal{R}(r)^2}$.

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

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

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

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

For any fixed key pair and any real $c$, let $t(pk, sk) := \mu(pk, sk) + \|\psi_\mathcal{R}\| \cdot \sqrt{(c + \ln |\mathcal{M}'|)/2}$, where $\mu(pk, sk) := \Pr_{\mathcal{R}, \mathcal{M}'}[\text{Dec}(sk, \text{Enc}(pk, M(r)) \neq M)]$. Then, we can use helper Lemma 3.3 to argue that

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

To this end, we identify $g(m, r, u)$ in Lemma 3.3 as $g(m, r, u) = (\text{SOTP}(m, u), r)$, and $B$ as $\{(M, r) \in [\text{Dec}(sk, \text{Enc}(pk, M; r)) \neq M]\}$. Note that $\Pr_{r \leftarrow \psi_\mathcal{R}, u \leftarrow \psi_\mathcal{U}}[g(m, r, u) \in B] = \mu(pk, sk)$ holds for all $m \in \mathcal{M}'$ by the message-hiding property of the SOTP.

$$\forall m \in \mathcal{M}, \Pr_{r \leftarrow \psi_\mathcal{R}, u \leftarrow \psi_\mathcal{U}}[g(m, r, u) \in B] = \Pr_{r \leftarrow \psi_\mathcal{R}, u \leftarrow \psi_\mathcal{U}}[(\text{SOTP}(m, u), r) \in B] = \Pr_{r \leftarrow \psi_\mathcal{R}, M \leftarrow \psi_\mathcal{M}}[(M, r) \in B] = \Pr_{r \leftarrow \psi_\mathcal{R}, M \leftarrow \psi_\mathcal{M}}[\text{Dec}(sk, \text{Enc}(pk, M; r) \neq M)] = \mu(pk, sk).$$
Combining Equation (2) with Equation (1) and taking the expectation yields
\[ \delta' \leq \mathbb{E} \left[ \mu(pk, sk) + \|\psi_R\| \cdot \sqrt{(c + \ln |\mathcal{M}'|)/2} + e^{-c} \right] = \delta + \|\psi_R\| \cdot \sqrt{(c + \ln |\mathcal{M}'|)/2} + e^{-c}, \]
and setting \( c := -\ln\|\psi_R\| \) yields the claim in the lemma.

**Lemma 3.3** (Adapting Lemma 3.7 from [9]). Let \( g \) be a function and \( B \) be some set such that
\[ \forall m \in \mathcal{M}', \quad \Pr_{r \leftarrow \psi_R, u \leftarrow U} [g(m, r, u) \in B] = \mu. \]

Let \( G \) be a random function mapping onto \( \mathcal{U} \). Define \( \|\psi_R\| = \sqrt{\sum_r \psi_R(r)^2} \). Then for all fractions except the \( e^{-c} \) of random functions \( G \), we have that \( \forall m \in \mathcal{M}', \)
\[ \Pr_{r \leftarrow \psi_R} [g(m, r, G(r)) \in B] \leq \mu + \|\psi_R\| \cdot \sqrt{(c + \ln |\mathcal{M}'|)/2} \quad (4) \]

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

The probability of Equation (4) for our particular \( m \) is the same as the sum \( \sum_r \psi_R(r)X_r \) and we use the Hoeffding bound to show that this value is not significantly larger than \( \mu \). We define the random variable \( Y_r = \psi_R(r)X_r \). Notice that \( Y_r \in [0, \psi_R(r)] \) and \( \mathbb{E}[\sum_r Y_r] = \mathbb{E}[\sum_r \psi_R(r)X_r] = \sum_r \psi_R(r)p_r = \mu \). By the Hoeffding bound, we have for all positive \( t \),
\[ \Pr[\sum_r Y_r > \mu + t] \leq \exp \left( \frac{-2t^2}{\sum_r \psi_R(r)^2} \right) = \exp \left( \frac{-2t^2}{\|\psi_R\|^2} \right). \quad (5) \]
Setting \( t \geq \|\psi\| \cdot \sqrt{(c + \ln |\mathcal{M}'|)/2}, \) for a fixed \( m \), Equation (4) holds for all, except for the \( e^{-c} \cdot |\mathcal{M}'|^{-1} \) fraction of random functions \( G \). Applying the union bound yields the claim in the lemma.

**Theorem 3.4** (OW-CPA of \( \text{PKE}^{\text{ROM}} \) to IND-CPA of \( \text{ACWC}_2[\text{PKE}, \text{SOTP}, G] \)). Let \( \text{PKE} \) be a public key encryption scheme with RR and MR properties, and injectivity error \( \mu \). For any adversary \( A \) against the IND-CPA security of \( \text{ACWC}_2[\text{PKE}, \text{SOTP}, G] \), making at most \( q_G \) random oracle queries, there exists an adversary \( B \) against the OW-CPA security of \( \text{PKE} \) with
\[ \text{Adv}^{\text{IND-CPA}}_{\text{ACWC}_2[\text{PKE}, \text{SOTP}, G]}(A) \leq \text{Adv}^{\text{OW-CPA}}_{\text{PKE}}(B) + \mu, \]
where the running time of \( B \) is about \( \text{Time}(A) + O(q_G) \).
Proof. We show that there exists an algorithm $B$ which breaks the OW-CPA security of PKE using an algorithm $A = (A_0, A_1)$ that breaks the IND-CPA security of $ACWC_2[PKE, SOTP, G]$.

GAME $G_0$. $G_0$ (see Figure 5) is the original IND-CPA game with $ACWC_2[PKE, SOTP, G]$. In $G_0$, $B$ is given the challenge ciphertext $c^* :=$ Enc$(pk, M^*; r^*)$ for some unknown message $M^*$ and randomness $r^*$. By definition, we have

$$
\Pr[\mathcal{G}_0 \Rightarrow 1] - \frac{1}{2} = \text{Adv}_{ACWC_2[PKE, SOTP, G]}^{\text{IND-CPA}}(A).
$$

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

$$
|\Pr[\mathcal{G}_1 \Rightarrow 1] - \Pr[\mathcal{G}_0 \Rightarrow 1]| \leq \mu.
$$

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

$$
|\Pr[\mathcal{G}_2 \Rightarrow 1] - \Pr[\mathcal{G}_1 \Rightarrow 1]| \leq \Pr[\text{QUERY}].
$$

13
Unless QUERY occurs, \( G(r^*) \) is a uniformly random value that is independent of \( A \)'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, \( B \) can find \( r^* \in \mathcal{L}_r \) such that \( e^* := \text{Enc}(pk, M^*; r^*) \), using the algorithm \( \text{Recover}^m \). Indeed, for each query \( r \) to \( G \), \( B \) checks whether \( \text{Recover}^m(pk, r, e^*) \in \mathcal{M} \). In the QUERY event, there exists \( M^* := \text{Recover}^m(pk, r^*, e^*) \in \mathcal{M} \) which can be the solution to its challenge ciphertext \( e^* \). It follows that

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

which concludes the proof. \( \square \)

**Theorem 3.5** (Classical O2H, Theorem 3 from the eprint version of [1]). Let \( S \subset \mathcal{R} \) be random. Let \( G \) and \( F \) be random functions satisfying \( \forall r \notin S : G(r) = F(r) \). Let \( z \) be a random classical value \((S, G, F, z\) may have an arbitrary joint distribution). Let \( C \) 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 \( \{1, ..., q_G\} \), runs \( C \) right before its \( i \)-th query to \( F \), measures all query input registers, and outputs the set \( T \) of measurement outcomes. Then

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

**Theorem 3.6** (OW-CPA of PKE \( QROM \) IND-CPA of ACWC_2[PKE, SOTP, G]). Let PKE be a public key encryption scheme with RR and MR properties, and injectivity error \( \mu \). 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

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

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 7 to 9 and Theorem 3.5. Before applying Theorem 3.5, we change \( G_0 \) to \( G_2 \). Subsequently, we apply Theorem 3.5 to \( G_2 \) and \( G_3 \). A detailed explanation of the security proof is provided in the followings.

**Game G_0**

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

Figure 7: Game \( G_0 \) for the proof of Theorem 3.6

GAME \( G_0 \). \( G_0 \) (see Figure 7) is the original IND-CPA game with ACWC_2[PKE, SOTP, G]. By definition, we have

\[
\left| \Pr[G_0^A \Rightarrow 1] - \frac{1}{2} \right| = \text{Adv}_\text{ACWC}_2^{\text{IND-CPA}}(A).
\]
Figure 8: Games $G_1$-$G_5$ for the proof of Theorem 3.6

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

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

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

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

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

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

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 Theorem 3.5. In addition, we define $G_5$ by changing the way $M$ is calculated. Instead of computing $M = \text{SOTP}(m_b, u)$, we sample $M \leftarrow \psi_M$. 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 9. As the changes are only conceptual,

$$\Pr[G_5^A \Rightarrow 1] = \Pr[G_6^A \Rightarrow 1].$$
Combining all (in)equalities and bounds, we have

\[ \Pr[G^A_6 \Rightarrow 1] - \Pr[G^A_7 \Rightarrow 1] \leq \mu. \]

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

\[ \text{Adv}_{PKE}^{\text{OW-CPA}}(B) = \Pr[G^A_7 \Rightarrow 1]. \]

Combining all (in)equalities and bounds, we have

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

which concludes the proof. \( \square \)

**Lemma 3.7.** If PKE is \( \gamma \)-spread, then so is PKE’ = ACWC\(_2\)[PKE, 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 \( m \in \mathcal{M}' \), the proof of Lemma 3.7 is completed. \( \square \)

## 4 Chosen-Ciphertext Secure KEM from ACWC\(_2\)

### 4.1 FO Transform with Re-encryption

One can apply the Fujisaki-Okamoto transformation \( \text{FO}^{-} \) to the IND-CPA secure PKE’, shown in Figure 4, to obtain an IND-CCA secure KEM. Figure 10 shows the resultant KEM := \( \text{FO}^{-}[\text{PKE}', H] = \)
(Gen, Encap, Decap), where $H$ is a hash function (modeled as a random oracle). Regarding the correctness error of KEM, KEM preserves the worst-case correctness error of PKE’, as Decap works correctly as long as Dec’ is performed correctly. Regarding the IND-CCA security of KEM, we can use the previous results \(^{[14]}\) and \(^{[8]}\), which are stated in Theorems 4.1 and 4.2, respectively. By combining these results with Theorems 3.4 and 3.6 we can achieve the IND-CCA security of KEM in the classical/quantum random oracle model. In the case of the quantum random oracle model (QROM), we need to further use the fact that IND-CPA generically implies OW-CPA.

![Image](image_url)

Figure 10: KEM = FO\(^⊥\)[PKE’, H]

**Theorem 4.1** (IND-CPA of PKE’ $\xrightarrow{\text{ROM}}$ IND-CCA of KEM \(^{[14]}\)). Let PKE’ be a public key encryption scheme with message space $\mathcal{M}$. Let PKE’ have (worst-case) correctness error $\delta$ and be (weakly) $\gamma$-spread. For any adversary $\mathcal{A}$ making at most $q_D$ decapsulation, $q_H$ hash queries, against the IND-CCA security of KEM, there exists an adversary $\mathcal{B}$ against the IND-CPA security of PKE’ with

$$\text{Adv}^{\text{IND-CCA}}_{\text{KEM}}(\mathcal{A}) \leq 2(\text{Adv}^{\text{IND-CPA}}_{\text{PKE'}}(\mathcal{B}) + q_H|\mathcal{M}|) + q_D2^{-\gamma} + q_H\delta,$$

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

**Theorem 4.2** (OW-CPA of PKE’ $\xrightarrow{\text{QROM}}$ IND-CCA of KEM \(^{[8]}\)). Let PKE’ have (worst-case) correctness error $\delta$ and be (weakly) $\gamma$-spread. For any quantum adversary $\mathcal{A}$, making at most $q_D$ decapsulation, $q_H$ (quantum) hash queries, against the IND-CCA security of KEM, there exists a quantum adversary $\mathcal{B}$ against the OW-CPA security of PKE’ with

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

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

### 4.2 FO-Equivalent Transform Without Re-encryption

The aforementioned FO\(^⊥\) requires the Decap algorithm to perform re-encryption to check if ciphertext $c$ is well-formed. Using $m'$ as a result of Dec’($sk$, $c$), a new randomness $r''$ is obtained from $H(m')$, and Enc'($pk$, $m'$; $r''$) is computed and compared with the (decrypted) ciphertext $c$. In this process, even if ($m'$, $r''$) is the same as ($m$, $r$) used in Encap, it does not guarantee that Enc'($pk$, $m'$; $r''$) = $c$. In other words, there could exist many other ciphertexts $\{c_i\}$ (including $c$ as one of them), all of which are decrypted into
the same \( m' \) and thus have the same randomness \( r'' \) in Decap. In \( \mathsf{FO}^\perp \) (and other FO transformations), there is still no way to find the same \( c \) (honestly) generated in Encap, other than by comparing \( \mathsf{Enc}'(pk, m'; r'') \) and \( c \). In the context of chosen-ciphertext attacks, it is well known that decapsulation queries using \( \{ c_i \} \) can leak information on \( sk \), particularly in lattice-based encryption schemes.

However, we demonstrate that \( \mathsf{FO}^\perp \) based on ACWC\(2 \) can eliminate such ciphertext comparison \( c = \mathsf{Enc}'(pk, m'; r'') \) from Decap, and replace it with a simpler and more efficient comparison \( r' = r'' \). We denote the new \( \mathsf{FO}^\perp \) based on ACWC\(2 \) as \( \mathsf{F_0}^\perp \), as shown in Figure 11. In \( \mathsf{F_0}^\perp \), \( r' \) and \( r'' \) are values generated while performing Decap, where \( r' \) is the output of \( \mathsf{Recover}'(pk, M', c) \) and \( r'' \) is computed from \( H(m') \). Compared to \( \mathsf{FO}^\perp \) in Figure 10, the only change is the boxed area from \( c \neq \mathsf{Enc}'(pk, m'; r'') \) to \( r' \neq r'' \) and the remaining parts are the same. Thus, by proving that equality \( c = \mathsf{Enc}'(pk, m'; r'') \) is equivalent to the equality \( r' = r'' \), we can show that both \( \mathsf{FO}^\perp \) and \( \mathsf{F_0}^\perp \) work identically, and thus achieve the same level of IND-CCA security.

```
Gen(1^\lambda)
1: (pk, sk) := Gen'(1^\lambda)
2: return (pk, sk)

Encap(pk)
1: m \leftarrow \mathcal{M}
2: (r, K) := H(m)
3: c := \mathsf{Enc}'(pk, m; r)
   - M := \mathsf{SOTP}(m, G(r))
   - c := \mathsf{Enc}(pk, M; r)
4: return (K, c)

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

Figure 11: KEM = \( \mathsf{F_0}^\perp [\mathsf{PKE}', H] \)

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

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

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

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

Conversely, assume that \( r' = r'' \) holds in the Decap of \( \mathsf{F_0}^\perp \). The rigidity of the \( \mathsf{SOTP} \) means that \( m' = \mathsf{Inv}(M', G(r')) \) implies \( M' = \mathsf{SOTP}(m', G(r')) \), and thus \( M' = \mathsf{SOTP}(m', G(r'')) \). In addition, the rigidity of \( \mathsf{PKE} \) means that for a ciphertext \( c \) given to Decap, the two equations \( M' = \mathsf{Dec}(sk, c) \) and \( r' = \mathsf{Recover}'(pk, M', c) \) lead to \( \mathsf{Enc}(pk, \mathsf{Dec}(sk, c); r') = c \) and thus \( \mathsf{Enc}(pk, \mathsf{Dec}(sk, c); r'') = c \). Because \( \mathsf{Dec}(sk, c) = M' = \mathsf{SOTP}(m', G(r'')) \), we observe that \( \mathsf{Enc}(pk, \mathsf{SOTP}(m', G(r'')); r'') = c \). Then, \( \mathsf{Enc}(pk, \mathsf{SOTP}(m', G(r'')); r'') \) can be expressed as \( \mathsf{Enc}'(pk, m'; r'') \), which implies that \( \mathsf{Enc}'(pk, m'; r'') = c \). \( \square \)
5 GenNTRU[$\psi_n^1$, $\psi_1^n$] (=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 $\psi_n^1$ is defined over the set $\{-1, 0, 1\}^n$, where each element is selected according to $\psi_1$.

5.1.2 NTT-Friendly Rings over Cyclotomic Trinomials

We use the polynomial ring $R_q := \mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$, where $q$ is a modulus and $n = 2^i3^j$ for some positive integers $i$ and $j$. For a polynomial $f \in R_q$, we use the notation `$f \leftarrow \psi_n^1$' to represent that each coefficient of $f$ is drawn according to the distribution $\psi_1$. In addition, we use the notation `$h \leftarrow R_q$' to show that polynomial $h$ is chosen uniformly at random from $R_q$. Subsequently, to perform NTT over $R_q$, we provide several parameter sets with respect to $(n, q)$.

5.1.3 Other Notations

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

5.2 Description of GenNTRU[$\psi_n^1$, $\psi_1^n$]

We define GenNTRU[$\psi_n^1$, $\psi_1^n$] relative to the distribution $\psi_n^1$ over $R_q$. Since PKE = (Gen, Enc, Dec) should be MR and RR for our ACWC2, Figure 12 shows two additional algorithms Recover$^r$ and Recover$^m$.

<table>
<thead>
<tr>
<th>Gen($1^k$)</th>
<th>Enc($h, m \leftarrow \psi_{1}^n; r \leftarrow \psi_{1}^n$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: $f', g \leftarrow \psi_{1}^n$</td>
<td>1: return $c = hr + m$</td>
</tr>
<tr>
<td>2: $f = 3f' + 1$</td>
<td>Dec($f, c$)</td>
</tr>
<tr>
<td>3: if $f, g$ is not invertible in $R_q$</td>
<td>1: return $m = (cf \mod \pm q) \mod \pm 3$</td>
</tr>
<tr>
<td>4: restart</td>
<td>Recover$^r(h, m, c)$</td>
</tr>
<tr>
<td>5: $h = 3gf^{-1}$</td>
<td>1: return $r = (c - m)h^{-1}$</td>
</tr>
<tr>
<td>6: return $(pk, sk) = (h, f)$</td>
<td>Recover$^m(h, r, c)$</td>
</tr>
<tr>
<td></td>
<td>1: return $m = c - hr$</td>
</tr>
</tbody>
</table>

Figure 12: GenNTRU[$\psi_1^n$] with average-case correctness error
5.3 Security and Other Properties

5.3.1 Cryptographic Assumptions

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

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

**Definition 5.2** (The RLWE problem). Let $\psi$ be a distribution over $R_q$. The RLWE problem $\text{RLWE}_{n,q,\psi}$ 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 $\text{RLWE}_{n,q,\psi}$ is defined as follows:

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

5.3.2 Security Proofs

**Theorem 5.3** (OW-CPA security of $\text{GenNTRU}[\psi^n_1]$). For any adversary $A$, there exist adversaries $B$ and $C$ such that

$$\text{Adv}_{\text{GenNTRU}[\psi^n_1]}^{\text{OW-CPA}}(A) \leq \text{Adv}_{\text{NTRU}}^{n,q,\psi_1^n}(B) + \text{Adv}_{\text{RLWE}}^{n,q,\psi_1^n}(C).$$

**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 $\text{GenNTRU}[\psi^n_1]$. By definition, we have that

$$\text{Adv}_{\text{GenNTRU}[\psi^n_1]}^{\text{OW-CPA}}(A) = \Pr[G_0^A \Rightarrow 1].$$

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

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

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

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

Combining all the probabilities completes the proof. $\square$

**Lemma 5.4** (Spreadness). $\text{GenNTRU}[\psi^n_1]$ is $n$-spread.

**Proof.** For fixed message $m$ and ciphertext $c$, there exists at most one $r$ such that $c = \text{Enc}(h, m; r)$. Suppose there exists $r_1$ 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 $c = \text{Enc}(h, m; r)$, $\Pr[\text{Enc}(h, m; r) = c] = 2^{-n}$. Since it holds for any $(pk, sk) \leftarrow \text{Gen}(1^\lambda)$ and $m \in \mathcal{M}$, $\text{GenNTRU}[\psi^n_1]$ is $n$-spread. $\square$
5.3.3 Average-Case Correctness Error

We analyze the average-case correctness error $\delta$ relative to the distribution $\psi_M = \psi_R = \psi_1^n$ by the template provided in [13]. 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 $R_q$, let $p_i$ be the $i$-th coefficient of $p$, and $|p_i|$ be the absolute value of $p_i$. Then, $((cf)_i \mod \pm q) \mod \pm 3 = m_i$ if the following inequality holds:

$$3(gr + mf') + m_i \leq q - 1 \over 2,$$

where all coefficients of each polynomial are distributed according to $\psi_1^n$. Let $\epsilon_i$ be

$$\epsilon_i = \Pr \left[ 3(gr + mf') + m_i \leq q - 1 \over 2 \right].$$

Then, assuming that each coefficient is independent,

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

Because the coefficients of $m$ have a size at most one,

$$\epsilon_i = \Pr \left[ 3(gr + mf') + m_i \leq q - 1 \over 2 \right] \geq \Pr \left[ |3(gr + mf')|_i + |m_i| \leq q - 1 \over 2 \right] \geq \Pr \left[ |3(gr + mf')|_i + 1 \leq q - 1 \over 2 \right] = \Pr \left[ |gr + mf'|_i \leq q - 3 \over 6 \right] := \epsilon'_i.$$  

Therefore,

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

Here, we analyze $\epsilon'_i = \Pr \left[ |gr + mf'|_i \leq q - 3 \over 6 \right]$. To achieve this, we need to analyze the distribution of $gr + mf'$. By following the analysis in [13], 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

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

In addition, 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

$$c' = ba + b'a' \in \{0, \pm 1, \pm 2\} \text{ where } a, b, a', b' \leftarrow \psi_1. \quad (8)$$
Computing the probability distribution of this sum can be done via a convolution (i.e. polynomial multiplication). Define the polynomial

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

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 \( gr + mf' \) 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 (6), we compute the average-case correctness error \( \delta \) of \( \text{GenNTRU}[\psi_1^n] \).

<table>
<thead>
<tr>
<th>\pm3</th>
<th>\pm2</th>
<th>\pm1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/128</td>
<td>1/32</td>
<td>23/128</td>
<td>9/16</td>
</tr>
</tbody>
</table>

Table 3: Probability distribution of \( c = ab + b'(a + a') \)

<table>
<thead>
<tr>
<th>\pm2</th>
<th>\pm1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/64</td>
<td>3/16</td>
<td>19/32</td>
</tr>
</tbody>
</table>

Table 4: Probability distribution of \( c' = ab + a'b' \)

5.3.4 Rigidity and Injectivity

The rigidity of \( \text{GenNTRU}[\psi_1^n] \) is trivial from the algorithms Dec and Recover \( r \) shown in Figure 12. The injectivity of \( \text{GenNTRU}[\psi_1^n] \) can be easily shown as follows: if there exist two inputs \((m_1, r_1)\) and \((m_2, r_2)\) such that \( \text{Enc}(h, m_1; r_1) = \text{Enc}(h, m_2; r_2) \), the equality indicates that \( (r_1 - r_2)h + (m_1 - m_2) = 0 \), where \( r_1 - r_2 \) and \( m_1 - m_2 \) still have small coefficients of length, at most \( 2\sqrt{n} \). For a lattice set

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

\( (r_1 - r_2, m_1 - m_2) \) becomes an approximate shortest vector in \( \mathcal{L}_0^+ \). 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 \( \mathcal{L}_0^+ \). It is well-known [9] that the approximate SVP over \( \mathcal{L}_0^+ \) is at least as hard as the \( \text{NTRU}_{n,q,\psi_1^n} \) problem (defined above). Hence, if the \( \text{NTRU}_{n,q,\psi_1^n} \) assumption holds, then injectivity of \( \text{GenNTRU}[\psi_1^n] \) also holds.

6 NTRU+

6.1 Instantiation of SOTP

We introduce SOTP : \( \mathcal{M}' \times \mathcal{U} \to \mathcal{M} \), where \( \mathcal{M}' = \{0,1\}^n \), \( \mathcal{U} = \{0,1\}^{2n} \), and \( \mathcal{M} = \{-1,0,1\}^n \) relative to distributions \( \psi_\mathcal{U} = U^{2n} \) and \( \psi_\mathcal{M} = \psi_1^n \). Figure 13 shows the SOTP used for ACWC2.
on the equation of Theorem 3.2 and Equation (6). In certain cases, in the case where GenNTRU worst-case correctness error that is close to the average-case correctness error of NTRU CPA easily obtained by combining Lemma 3.7 with Lemma 5.4. Next, the injectivity of KEM via CPA NTRU+ (\(=\mathbb{PKE}'\))

We obtain CPA-NTRU+ := ACWC2[GenNTRU[\(\psi_1^n\)], SOTP, \(G\)] by applying ACWC2 from Section 3 to GenNTRU[\(\psi_1^n\)]. Because the underlying GenNTRU[\(\psi_1^n\)] provides MR and RR properties, Theorems 3.4 and 3.6 provide us the IND-CPA security of the resulting CPA-NTRU+ in the classical and quantum random oracle models, respectively. Regarding the correctness error, Theorem 3.2 shows that CPA-NTRU+ has worst-case correctness error that is close to the average-case correctness error of GenNTRU[\(\psi_1^n\)]. For instance, in the case where \((n, q) = (768, 3457)\), the worst-case correctness error becomes about \(2^{-379}\), based on the equation of Theorem 3.2 and Equation (6).

Spreadness and Injectivity Properties of CPA-NTRU+. To achieve IND-CCA security of the transformed KEM via \(\mathbb{F}_q^\perp\), we need to show the spreadness and injectivity of CPA-NTRU+. The spreadness can be easily obtained by combining Lemma 3.7 with Lemma 5.4. Next, the injectivity of CPA-NTRU+ can also be proven under the assumption that the \(\text{NTRU}_{n, q, \psi_1^n}\) problem is infeasible, analogous to GenNTRU[\(\psi_1^n\)].
Specifically, if there exist two distinct pairs \((m_1, r_1)\) and \((m_2, r_2)\) such that \(\text{Enc}(pk, m_1; r_1) = \text{Enc}(pk, m_2; r_2)\), this results in the equation \(hr_1 + m_1 = hr_2 + m_2\), where \(m_1 = \text{SOTP}(m_1, G(r_1))\) and \(m_2 = \text{SOTP}(m_2, G(r_2))\). In this case, we have two short polynomials: \(r_1 - r_2\) and \(m_1 - m_2\), which can be a solution of approximate SVP (of length at most \(2\sqrt{n}\)) over \(L_0^\perp\).

6.3 CCA-NTRU+ (=KEM)

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

<table>
<thead>
<tr>
<th>Scheme</th>
<th>(n)</th>
<th>(q)</th>
<th>(pk)</th>
<th>(ct)</th>
<th>(sk)</th>
<th>(\text{sec}(c))</th>
<th>(\text{sec}(q))</th>
<th>(\log_2\delta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NTRU+576</td>
<td>576</td>
<td>3,457</td>
<td>864</td>
<td>864</td>
<td>1,728</td>
<td>115</td>
<td>104</td>
<td>-487</td>
</tr>
<tr>
<td>NTRU+768</td>
<td>768</td>
<td>3,457</td>
<td>1,152</td>
<td>1,152</td>
<td>2,304</td>
<td>164</td>
<td>148</td>
<td>-379</td>
</tr>
<tr>
<td>NTRU+864</td>
<td>864</td>
<td>3,457</td>
<td>1,296</td>
<td>1,296</td>
<td>2,592</td>
<td>188</td>
<td>171</td>
<td>-340</td>
</tr>
<tr>
<td>NTRU+1152</td>
<td>1,152</td>
<td>3,457</td>
<td>1,728</td>
<td>1,728</td>
<td>3,456</td>
<td>264</td>
<td>240</td>
<td>-260</td>
</tr>
</tbody>
</table>

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

Table 5: Parameters for NTRU+
<table>
<thead>
<tr>
<th>NIST Security level</th>
<th>NTRU+576</th>
<th>NTRU+768</th>
<th>NTRU+864</th>
<th>NTRU+1152</th>
</tr>
</thead>
</table>

Core-SVP methodology (primal attack)

<table>
<thead>
<tr>
<th>Lattice attack dim.</th>
<th>1054</th>
<th>1397</th>
<th>1573</th>
<th>2045</th>
</tr>
</thead>
<tbody>
<tr>
<td>BKZ-blocksize</td>
<td>399</td>
<td>560</td>
<td>655</td>
<td>922</td>
</tr>
<tr>
<td>core-SVP classical hardness</td>
<td>116</td>
<td>163</td>
<td>191</td>
<td>269</td>
</tr>
<tr>
<td>core-SVP quantum hardness</td>
<td>105</td>
<td>148</td>
<td>173</td>
<td>244</td>
</tr>
</tbody>
</table>

Core-SVP methodology (dual attack)

<table>
<thead>
<tr>
<th>Lattice attack dim.</th>
<th>1045</th>
<th>1370</th>
<th>1577</th>
<th>2009</th>
</tr>
</thead>
<tbody>
<tr>
<td>BKZ-blocksize</td>
<td>395</td>
<td>553</td>
<td>645</td>
<td>905</td>
</tr>
<tr>
<td>core-SVP classical hardness</td>
<td>115</td>
<td>161</td>
<td>188</td>
<td>264</td>
</tr>
<tr>
<td>core-SVP quantum hardness</td>
<td>104</td>
<td>146</td>
<td>171</td>
<td>240</td>
</tr>
</tbody>
</table>

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

7.2 Concrete Security Strength

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

8 Performance Analysis

8.1 Implementation of NTT

We use NTT to implement polynomial multiplication in a ring. To achieve NTT over the polynomial ring \( \mathbb{Z}_q[x]/(x^n - x^{n/2} + 1) \) with \( n = 2^a3^b \), we use three different types of NTT layers: Radix-2 NTT, Radix-2 NTT to cyclotomic trinomial, and Radix-3 NTT. Given a polynomial in a ring, we first factor the polynomial by adapting the Radix-2 NTT to cyclotomic trinomial layer, which was first used in [18]. This can be viewed as an isomorphism between \( \mathbb{Z}_q[x]/(x^n - x^{n/2} + 1) \) and \( \mathbb{Z}_q[x]/(x^{n/2} - \zeta) \times \mathbb{Z}_q[x]/(x^{n/2} - (1 - \zeta)) \), where \( \zeta \) is a primitive sixth root of unity modulus \( q \). Subsequently, we successively use Radix-3 NTT layers to factor each partitioned polynomial to reach the desired degree of a polynomial. Finally, we use Radix-2 NTT layers until it reaches inertia degree two or three of the polynomials. We use Radix-3 NTT layers before the Radix-2 NTT layers to minimize the size of the predefined tables required to multiply polynomials in the NTT form. Appendix A describes the details of Radix-3 NTT layer.
Table 7: Combinations of NTT layers

<table>
<thead>
<tr>
<th>n</th>
<th>q</th>
<th>Radix-2 to 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>4</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>

8.2 Implementation Results

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

Table 8: Implementation result of NTRU+

<table>
<thead>
<tr>
<th>NTRU+576</th>
<th>Size (Bytes)</th>
<th>Cycles (ref)</th>
<th>Cycles (AVX2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>sk:</td>
<td>1,728</td>
<td>gen: 329,794</td>
<td>gen: 17,123</td>
</tr>
<tr>
<td>pk:</td>
<td>864</td>
<td>encap: 126,378</td>
<td>encap: 13,920</td>
</tr>
<tr>
<td>ct:</td>
<td>864</td>
<td>decap: 179,431</td>
<td>decap: 12,102</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NTRU+768</th>
<th>Size (Bytes)</th>
<th>Cycles (ref)</th>
<th>Cycles (AVX2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>sk:</td>
<td>2,304</td>
<td>gen: 319,640</td>
<td>gen: 15,662</td>
</tr>
<tr>
<td>pk:</td>
<td>1,152</td>
<td>encap: 162,650</td>
<td>encap: 17,451</td>
</tr>
<tr>
<td>ct:</td>
<td>1,152</td>
<td>decap: 235,869</td>
<td>decap: 15,800</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NTRU+864</th>
<th>Size (Bytes)</th>
<th>Cycles (ref)</th>
<th>Cycles (AVX2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>sk:</td>
<td>2,592</td>
<td>gen: 349,226</td>
<td>gen: 13,784</td>
</tr>
<tr>
<td>pk:</td>
<td>1,296</td>
<td>encap: 191,331</td>
<td>encap: 18,964</td>
</tr>
<tr>
<td>ct:</td>
<td>1,296</td>
<td>decap: 277,870</td>
<td>decap: 17,329</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NTRU+1152</th>
<th>Size (Bytes)</th>
<th>Cycles (ref)</th>
<th>Cycles (AVX2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>sk:</td>
<td>3,456</td>
<td>gen: 916,994</td>
<td>gen: 42,206</td>
</tr>
<tr>
<td>pk:</td>
<td>1,728</td>
<td>encap: 252,244</td>
<td>encap: 24,853</td>
</tr>
<tr>
<td>ct:</td>
<td>1,728</td>
<td>decap: 362,996</td>
<td>decap: 23,418</td>
</tr>
</tbody>
</table>
References


A Radix-3 NTT Layer

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

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

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

**Algorithm 1 Radix-3 NTT layer**

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

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

1: $t_1(x) = a_1(x)\alpha$
2: $t_2(x) = a_2(x)\alpha^2$
3: $t_3(x) = t_1(x)\omega$ \hspace{1cm} $\parallel a_1(x)\beta$
4: $t_4(x) = t_2(x)\omega^2$ \hspace{1cm} $\parallel a_2(x)\beta^2$
5: $t_5(x) = t_1(x) + t_2(x)$ \hspace{1cm} $\parallel a_1(x)\alpha + a_2(x)\alpha^2$
6: $t_6(x) = t_3(x) + t_4(x)$ \hspace{1cm} $\parallel a_1(x)\beta + a_2(x)\beta^2$
7: $t_7(x) = t_5(x) + t_6(x)$ \hspace{1cm} $\parallel -a_1(x)\gamma - a_2(x)\gamma^2$
8: $\hat{a}_0(x) = a_0(x) + t_5(x)$ \hspace{1cm} $\parallel a_0(x) + a_1(x)\alpha + a_2(x)\alpha^2$
9: $\hat{a}_1(x) = a_0(x) + t_6(x)$ \hspace{1cm} $\parallel a_0(x) + a_1(x)\beta + a_2(x)\beta^2$
10: $\hat{a}_2(x) = a_0(x) - t_7(x)$ \hspace{1cm} $\parallel a_0(x) + a_1(x)\gamma + a_2(x)\gamma^2$
11: **return** $(\hat{a}_0(x), \hat{a}_1(x), \hat{a}_2(x))$

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

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

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

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

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

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