NTRU+
Algorithm Specifications and Supporting Documentation*

Jonghyun Kim[1] and Jong Hwan Park[2]

[1] Korea University, Seoul, Korea.
yoswuk@korea.ac.kr
[2] Sangmyung University, Seoul, Korea.
jhpark@smu.ac.kr

1 Overview

The NTRU encryption scheme [10] 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 the NTRU is crucially based on the NTRU problem [10], which has withstood significant cryptanalytic attacks over a few decades. Such a longer history than other lattice-based problems (such as Ring/Module-LWE) has been considered as being an important factor in selecting the NTRU as a finalist in the NIST PQC standardization process. While the finalist NTRU [3], which was a merger of two submissions NTRU-HRSS [14] and NTRU-HPS [16], has not been chosen by NIST as one of the first four quantum-resistant cryptographic algorithms, the NTRU has several distinct advantages over other lattice-based competitive schemes like Kyber [15] and Saber [6]. Indeed, the advantages of the NTRU include (1) compact structure of a ciphertext consisting of a single polynomial, and (2) (possibly) faster encryption and decryption without need to sample coefficients of a public key polynomial.

The central design principle behind the NTRU is described over a ring $R_q = \mathbb{Z}_q[x]/(f(x))$, where q is a prime and $f(x)$ is a polynomial. The public key is generated as $\mathbf{h} = p\mathbf{g}/(p\mathbf{f}+1)$[3], where $\mathbf{g}$ and $\mathbf{f}$ are sampled according to a narrow distribution Ψ and p is a small prime (e.g., 3), and the corresponding private key is then $\mathbf{f}' = p\mathbf{f} + 1$. To encrypt a message m sampled from a message space $\mathcal{M}'$, one creates two polynomials $\mathbf{r}$ and $\mathbf{m}$ whose coefficients are also drawn from a narrow distribution ψ, and computes a ciphertext $\mathbf{c} = \mathbf{rh} + \mathbf{m}$ in R_q. An important point is that there could exist an (efficient) encoding/decoding method between the message space $\mathcal{M}'$ and the ring R_q (along with the distribution ψ), so that $m \in \mathcal{M}'$ is encoded into the polynomial $\mathbf{m} \in R_q$ in encryption. Alternatively, it is also possible to directly sample $\mathbf{m}$ from the distribution ψ, without using such an encoding/decoding method. To decrypt the ciphertext $\mathbf{c}$,

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

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

one computes $\mathbf{cf}'$ in R_q, recovers $\mathbf{m}$ by taking the resulting value $\mathbf{cf}'$ modulo p, and (if necessary) decodes $\mathbf{m}$ to obtain the message m. The decryption of the NTRU works correctly if all the coefficients of $p(\mathbf{gr} + \mathbf{fm})$ are smaller than $q/2$, in which case the probability that the NTRU decryption fails is called a *correctness (or decryption) error*.

Average-Case to Worst-Case Correctness Error. Given a pair of public/private keys, roughly speaking, an average-case correctness error is the probability that a decryption fails, when any ciphertext is generated with any message and any randomness which are (*honestly*) drawn from their respective spaces according to their distributions. On the other hand, a worst-case correctness error is the probability that a decryption fails for any ciphertext that can be possibly generated with all messages and all randomnesses in their respective spaces. Naturally, the worst-case correctness error covers the case where a message and a randomness are (*maliciously*) chosen from their spaces, without sampling normally according to their original distributions. In the context of chosen-ciphertext attacks, a public-key encryption scheme must guarantee a very negligible worst-case correctness error, because otherwise information about the private key may be leaked by a number of adversarial decryption queries (especially, in lattice-based encryption schemes [5]). In case of the above NTRU, for example, an event in which a specific ciphertext $\mathbf{c} = \mathbf{rh} + \mathbf{m}$ is not correctly decrypted gives an adversary the information that one of the coefficients of the polynomial $p(\mathbf{gr} + \mathbf{fm})$ is larger than or equal to $q/2$. If the adversary has control over the choice of $\mathbf{r}$ and $\mathbf{m}$, even one such decryption failure may open a path to associated decryption queries to obtain more information about the secret polynomials $\mathbf{g}$ and $\mathbf{f}$. In general, when the Fujisaki-Okamoto (FO) transformation [9] is applied to the NTRU for obtaining chosen-ciphertext security, the randomness $\mathbf{r}$ must be sampled using a hash value of a message. Hence, it is sufficient to consider that the adversary has only control over the message $\mathbf{m}$.

In constructing the NTRU, there have been two approaches to achieve worst-case correctness error. One is not to use an encoding method, in which case $\mathbf{m}$ is expected to be directly drawn from the distribution ψ using the internal randomness of an encryption algorithm. This means that, for an adversary mounting chosen-ciphertext attacks, $\mathbf{m}$ can be chosen maliciously in the space of all possible polynomials that $\mathbf{m}$ takes, in order to make adversarial decryption queries. What is important is that, by issuing a number of decryption queries, the adversary can determine whether a decryption succeeds or not, depending on the selection of $\mathbf{m}$. Thus, a worst-case correctness error should guarantee that all coefficients of $p(\mathbf{gr} + \mathbf{fm})$ are less than $q/2$ for *almost all possible* $\mathbf{m}$ in the message space. Indeed, this approach has been taken by the third-round finalist NTRU [3], where all proposed parameters of the NTRU provide *perfect* correctness error (i.e., the worst-case correctness error becomes zero for all possible $\mathbf{m}$ in its sample space). However, the perfect correctness error requires the modulus q to be relatively larger than the other lattice-based finalists such as Kyber and Saber, thereby causing inefficiency in terms of public-key and ciphertext sizes.

On the other hand, the other approach [8] is to use an encoding method by which (roughly speaking) a message $m \in \mathcal{M}'$ itself is exploited as the randomness to sample $\mathbf{m}$ according to the distribution ψ. If the encoding method has the onewayness[4] property in a sense that it is infeasible to find an input message m corresponding to a targeted $\mathbf{m}$, the adversary has difficulty even sampling a specific $\mathbf{m}$ from a message m. Whenever an adversary makes a decryption query without following the sampling rule (i.e., selects $\mathbf{m}^*$ of its choice), recovering and decoding such $\mathbf{m}^*$ (or its changed polynomial) gives a message m^* independent of what is encoded as $\mathbf{m}^*$. Therefore, re-encrypting m^* as in the FO transformation shows that such a decryption query will always be answered with failure due to the encoding method, regardless of the polynomial $p(\mathbf{gr} + \mathbf{fm})$. This means that $\mathbf{m}$ should be honestly sampled by the adversary, following the sampling rule. Consequently, by disallowing the adversary to have control over the message $\mathbf{m}$ (as well as the randomness $\mathbf{r}$ by the FO transform), the NTRU with an appropriate encoding/decoding method has a worst-case correctness error that is close to an average-case one. Based on this observation, [8] proposed generic transformations[5] (denoted by ACWC) that make an average-case correctness error of an underlying scheme almost equal to be a worst-case one of a transformed scheme.

1.1 Features of NTRU+

ACWC_2 Transformation. NTRU+ is based on our new generic ACWC transformation (denoted by ACWC_2) that works with a simple encoding method and a randomness recovery algorithm. Using ACWC_2, we can prove that (1) a transformed scheme is chosen-plaintext-secure ($\mathsf{IND\text{-}CPA}$) if an underlying scheme is oneway-secure ($\mathsf{OW\text{-}CPA}$), and (2) a worst-case correctness error of a transformed scheme is almost close to that of an underlying scheme. We introduce a so-called semi-generalized one-time pad (denoted by SOTP) as an encoding method, which works by summing up and subtracting random bits according to a centered binomial distribution. Regarding a randomness recovery algorithm (denoted by $\mathsf{Recover}^r$), we use the fact that in the NTRU construction, once $\mathbf{m}$ is obtained via decryption, $\mathbf{r}$ is recovered as $(\mathbf{c} - \mathbf{m})\mathbf{h}^{-1}$.

FO-Equivalent Transform without Re-encryption. To achieve chosen-plaintext-secure ($\mathsf{IND\text{-}CCA}$) NTRU+, we apply the generic FO transform to the ACWC_2-derived NTRU+ that is $\mathsf{IND\text{-}CPA}$-secure. However, by using the feature of the $\mathsf{Recover}^r$ algorithm, we further proceed to remove re-encryption process from the FO transform. The new transform (denoted by $\overline{\mathsf{FO}}$) works functionally the same as the original FO transform, but fits well with the NTRU-based scheme that supports the randomness recovery algorithm. Instead of doing re-encryption, checking the validity of a ciphertext is done by simply comparing

[4] In [8] and our construction, this onewayness property is easy to achieve, when an encoding method is constructed using a hash function.

[5] Their transformations are generic and can be applied to any public-key encryption scheme including the NTRU.

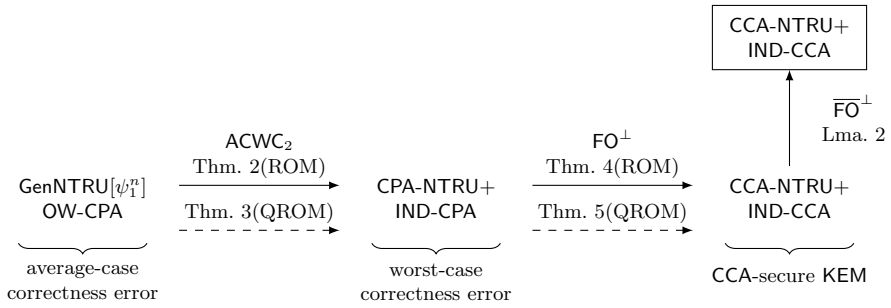

Fig. 1. Overview of correctness errors and security reductions

two randomnesses for equality, resulting in more efficient decryption than the FO transform. Figure 1 shows the overview of NTRU+, in terms of correctness error and security reductions.

NTT-Friendly Rings Over Cyclotomic Trinomials. NTRU+ is instantiated over a polynomial ring $R_q = \mathbb{Z}_q/(f(x))$, where $f(x) = x^n - x^{n/2} + 1$ is a cyclotomic trinomial of degree $n = 2^i 3^j$. With appropriate parametrization of n and q, [13] showed that such a ring can also provide Number Theoretic Transform (NTT) [12] operation essentially as fast as that over a ring $R_q = \mathbb{Z}_q/(f(x))$ with a polynomial $f(x) = x^n + 1$. Moreover, since the choice of a cyclotomic trinomial is moderate, it gives more flexibility to meet a certain level of security. Following these results, NTRU+ recommends four parameter sets, where the 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. Theoretical performance results and security levels according to each parameter set are presented in Table 3 and 5.

More Convenient Sampling Distribution for Messages. Our simple SOTP allows for sampling messages from more convenient message distribution. Indeed, our NTRU+ construction requires a natural uniformly random distribution over $\{0,1\}^n$ for some positive integer n. Thus, as a starting point, a message m is chosen from an arbitrary n-bit $m \in \mathcal{M}' = \{0,1\}^n$ (with high min-entropy). Once m is chosen from the space $\mathcal{M}'$, two polynomials $\mathbf{r}$ and $\mathbf{m}$ are encoded according to the centered binomial distribution ψ_1 obtained by subtracting two uniformly-random bits from each other. Unlike our message sampling, the NTRU construction [8] that instantiates ACWC requires the message space $\mathcal{M}'$ to be uniform over $\{-1,0,1\}$, which is inconvenient for constant-time implementation because of rejection sampling. Also, the finalist NTRU [3] uses two distinct distributions for their message $\mathbf{m}$; one is a fixed-weight ternary distribution where each coefficient is drawn from $\{-1,0,1\}$ and the number of the total (1)s and (-1)s is a fixed value, and the other is the uniformly random distribution over $\{-1,0,1\}$.

2 Preliminaries

2.1 Public Key Encryption and Related Properties

Definition 1 (Public Key Encryption). *A public key encryption scheme* $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with a message space* $\mathcal{M}$ *and a randomness space* $\mathcal{R}$ *consists of following three algorithms:*

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

Correctness. We say that PKE has (worst-case) correctness error δ [11] if

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

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

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

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

Injectivity. We say that PKE is injective if for all $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$, it holds that $\mathsf{Enc}(pk, m; r) = \mathsf{Enc}(pk, m'; r')$ implies $(m, r) = (m', r')$ for all $m, m' \in \mathcal{M}$ and $r, r' \in \mathcal{R}$.

Spreadness. We say that PKE is weakly γ-spread [7] if

$$\mathbb{E}\left[\max_{m \in \mathcal{M}, c \in \mathcal{C}} \Pr\left[\mathsf{Enc}(pk, m) = c\right]\right] \leq 2^{-\gamma},$$

where the probability is taken over the random coins of encryptions and the expectation is taken over $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$.

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

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

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

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

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

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

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

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

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

2.2 Security

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

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

Definition 3 (IND-CPA Security of PKE). *Let* PKE = (Gen, Enc, Dec) *be a public key encryption scheme with a message space* $\mathcal{M}$*. Indistinguishability under chosen-plaintext attacks (*IND-CPA*) is defined via the game* IND-CPA *defined in Figure 2 and the advantage function of an adversary* $\mathcal{A}$ *is*

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

Game OW-CPA	Game IND-CPA
1: $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$.	1: $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$
2: $m \leftarrow \psi_{\mathcal{M}}$	2: $b \leftarrow \{0, 1\}$
3: $c^* \leftarrow \mathsf{Enc}(pk, m)$	3: $(m_0, m_1) \leftarrow \mathcal{A}_1^{\mathsf{G}}(pk)$
4: $m' \leftarrow \mathcal{A}(pk, c^*)$	4: $c^* \leftarrow \mathsf{Enc}(pk, m_b)$
5: **return** $[\![m = m']\!]$	5: $b' \leftarrow \mathcal{A}_2^{\mathsf{G}}(pk, c^*)$
	6: **return** $[\![b = b']\!]$

Fig. 2. Game OW-CPA and Game IND-CPA for PKE

2.3 Key Encapsulation Mechanism

Definition 4 (Key Encapsulation Mechanism). *A key encapsulation mechanism* KEM = (Gen, Encap, Decap) *with a key space* $\mathcal{K}$ *consists of following three algorithms:*

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

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

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

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

Definition 5 (IND-CCA Security of KEM). *Let* KEM = (Gen, Encap, Decap) *be a key encapsulation mechanism with a key space* $\mathcal{K}$*. Indistinguishability under chosen-ciphertext attacks (*IND-CCA*) is defined via the game* IND-CCA *described in Figure 3 and the advantage function of an adversary* $\mathcal{A}$ *is as follows:*

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

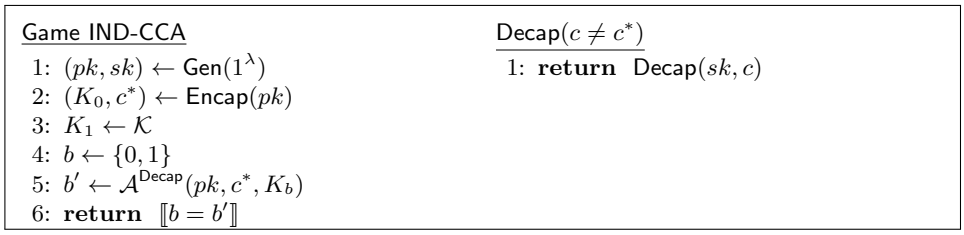

Fig. 3. Game IND-CCA for KEM

3 ACWC_2 Transformation

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

3.1 SOTP

First, we begin by defining a semi generalised one-time pad SOTP as follows:

Definition 6. *Function* $\mathsf{SOTP} : \mathcal{X} \times \mathcal{U} \to \mathcal{Y}$ *is called semi generalized one-time pad (relative to distributions* $\psi_{\mathcal{X}}$*,* $\psi_{\mathcal{U}}$*,* $\psi_{\mathcal{Y}}$*) if*

1. *Decoding: There exists an efficient inversion algorithm* Inv *such that for all* $x \in \mathcal{X}$*,* $u \in \mathcal{U}$*,* $\mathsf{Inv}(\mathsf{SOTP}(x, u), u) = x$*.*
2. *Message-hiding: For all* $x \in \mathcal{X}$*, the random variable* $\mathsf{SOTP}(x, u)$*, for* $u \leftarrow \psi_{\mathcal{U}}$*, has the same distribution as* $\psi_{\mathcal{Y}}$*.*
3. *Rigid: For all* $u \in \mathcal{U}$ *and all* SOTP*-encoded* $y \in \mathcal{Y}$*, it holds that* $\mathsf{SOTP}(\mathsf{Inv}(y, u), u) = y$*.*

Compared to the generalized one-time pad GOTP defined in [8], SOTP does not need to have an additional *randomness-hiding* property, which requires that an output $y = \mathsf{SOTP}(x, u)$ follows the distribution $\psi_{\mathcal{Y}}$ and simultaneously does not leak any information about the used randomness u. The absence of such additional property allows us to design SOTP more flexibly and efficiently than GOTP. Instead, SOTP is required to be *rigid*, which means that for all SOTP-encoded $y \in \mathcal{Y}$ and $u \in \mathcal{U}$, $\mathsf{Inv}(y, u) = x$ implies $\mathsf{SOTP}(x, u) = y$.

3.2 ACWC_2

Let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an underlying public-key encryption scheme with message space $\mathcal{M}$ and randomness space $\mathcal{R}$, where a message $M \in \mathcal{M}$ and a

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

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

$\underline{\mathsf{Enc}'(pk, m \in \mathcal{M}'; r \leftarrow \psi_{\mathcal{R}})}$
1: $M := \mathsf{SOTP}(m, \mathsf{G}(r))$
2: $c := \mathsf{Enc}(pk, M; r)$
3: **return** c

$\underline{\mathsf{Dec}'(sk, c)}$
1: $M := \mathsf{Dec}(sk, c)$
2: $r := \mathsf{Recover}^r(pk, M, c)$
3: $m := \mathsf{Inv}(M, \mathsf{G}(r))$
4: **return** m

Fig. 4. $\mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$

Theorem 1 (Average-Case to Worst-Case Correctness Error). *Let* PKE *be message recoverable and have randomness space* $\mathcal{R}$ *relative to the distribution* $\psi_{\mathcal{R}}$. *Let* $\mathsf{SOTP} : \mathcal{M}' \times \mathcal{U} \to \mathcal{M}$ *be a semi generalized one-time pad (for distributions* $\psi_{\mathcal{M}'}$, $\psi_{\mathcal{U}}$, $\psi_{\mathcal{M}}$*) and* $\mathsf{G} : \mathcal{R} \to \mathcal{U}$ *be a random oracle.*

If PKE *is* δ*-average-case-correct, then* $\mathsf{PKE}' := \mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$ *is* δ'*-worst-case-correct for*

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

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

Since $\mathsf{Recover}^r$ and Inv functions do not affect the correctness error of PKE', the factor that determines the success or failure of decryption is the result of $\mathsf{Dec}(sk, c)$ in Dec'. This means that, in the end, the correctness error of PKE' is determined by the selections of $M \in \mathcal{M}$ and $r \in \mathcal{R}$. We see that r is drawn according to the distribution ψ_R and M is an SOTP-encoded element in $\mathcal{M}$ following the distribution $\psi_{\mathcal{M}}$. We can here view SOTP as a sampling function using an internal randomness $\mathsf{G}(r)$, while hiding m. Eventually, both M and r are chosen according to their respective distributions initially intended. This is the same idea as in ACWC, and overall the proof strategy of Theorem 1 is essentially the same as that of [8] (Lemma 3.6 therein), except for slight modifications to message distribution. For completeness, a proof of Theorem 1 will be given in our full version [ePrint].

Theorem 2 (OW-CPA of PKE $\overset{\mathsf{ROM}}{\Longrightarrow}$ IND-CPA of $\mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$)**.** *Let* PKE *be a public-key encryption scheme with randomness and message recoverable properties. For any adversary $\mathcal{A}$ against the* IND-CPA *security of* $\mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$*, making at most q_{G} random oracle queries, there exists an adversary $\mathcal{B}$ against the* OW-CPA *security of* PKE *with*

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

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

Theorem 3 (OW-CPA of PKE $\overset{\mathsf{QROM}}{\Longrightarrow}$ IND-CPA of $\mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$)**.** *Let* PKE *be a public-key encryption scheme with randomness and message recoverable properties. For any quantum adversary $\mathcal{A}$ against the* IND-CPA *security of* $\mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$ *with query depth at most q_{G}, there exists a quantum adversary $\mathcal{B}$ against the* OW-CPA *security of* PKE *with*

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

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

Lemma 1. *If* PKE *is weakly γ-spread, then so is* $\mathsf{ACWC}_2[\mathsf{PKE}, \mathsf{SOTP}, \mathsf{G}]$.

All proofs of Theorem 2 and 3, and Lemma 1 will be given in our full version [ePrint].

4 Chosen-Ciphertext Secure KEM from ACWC_2

4.1 FO Transform With Re-encryption

One can apply the Fujisaki-Okamoto transformation $\mathsf{FO}^{\perp}$ to the IND-CPA secure PKE' in Figure 4 in order to obtain an IND-CCA secure KEM. Figure 5 shows the resultant $\mathsf{KEM} := \mathsf{FO}^{\perp}[\mathsf{PKE}', \mathsf{H}] = (\mathsf{Gen}, \mathsf{Encap}, \mathsf{Decap})$, where H is a hash function (modeled as a random oracle). Regarding the correctness error of KEM, KEM preserves the worst-case correctness error of PKE', since Decap works correctly as long as Dec' is performed correctly. Regarding the IND-CCA security of KEM, we can make use of the previous results [11] and [7], which are stated in Theorem 4 and 5 below. By combining these results with Theorem 2 and 3, we can achieve the IND-CCA security of KEM in classical/quantum random oracle model, respectively. In case of quantum random oracle model (QROM), we need to further use the fact that IND-CPA implies OW-CPA generically.

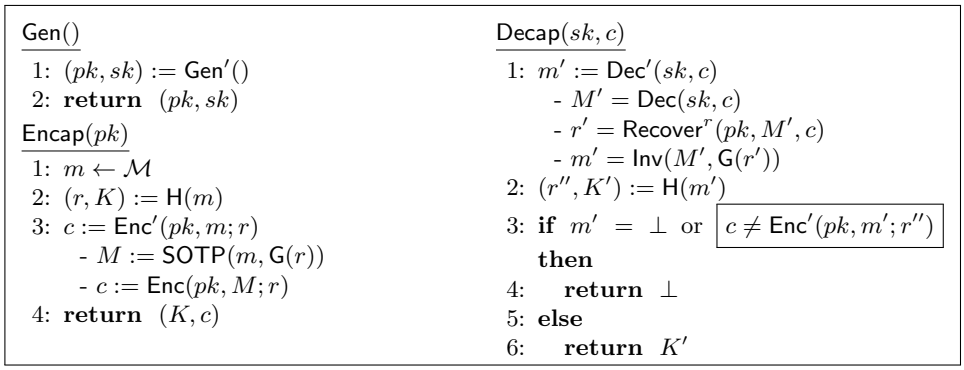

Fig. 5. $\mathsf{KEM} = \mathsf{FO}^{\perp}[\mathsf{PKE}', \mathsf{H}]$

Theorem 4 ($\mathsf{IND\text{-}CPA}$ of $\mathsf{PKE}' \overset{\mathsf{ROM}}{\Longrightarrow}$ $\mathsf{IND\text{-}CCA}$ of KEM[11]). *Let* PKE' *be a public key encryption scheme with message space* $\mathcal{M}$*. Let* PKE' *have (worst-case) correctness error* δ *and be weakly* γ*-spread. For any adversary* $\mathcal{A}$*, making at most* q_{D} *decapsulation,* q_{H} *hash queries, against the* $\mathsf{IND\text{-}CCA}$ *security of* KEM*, there exists an adversary* $\mathcal{B}$ *against the* $\mathsf{IND\text{-}CPA}$ *security of* PKE' *with*

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

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

Theorem 5 ($\mathsf{OW\text{-}CPA}$ of $\mathsf{PKE}' \overset{\mathsf{QROM}}{\Longrightarrow}$ $\mathsf{IND\text{-}CCA}$ of KEM [7]). *Let* PKE' *have (worst-case) correctness error* δ *and be weakly* γ*-spread. For any quantum adversary* $\mathcal{A}$*, making at most* q_{D} *decapsulation,* q_{H} *(quantum) hash queries, against the* $\mathsf{IND\text{-}CCA}$ *security of* KEM*, there exists a quantum adversary* $\mathcal{B}$ *against the* $\mathsf{OW\text{-}CPA}$ *security of* PKE' *with*

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

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

4.2 FO-Equivalent Transform Without Re-encryption

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

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 the information on sk, especially 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 instead replace it with a simpler and much more efficient comparison $r' = r''$. We denote the new $\mathsf{FO}^{\perp}$ based on ACWC_2 as $\overline{\mathsf{FO}}^{\perp}$, which is shown in Figure 6. In $\overline{\mathsf{FO}}^{\perp}$, r' and r'' are values generated while performing Decap, where r' is the output of $\mathsf{Recover}^r(pk, M', c)$ and r'' is computed from $\mathsf{H}(m')$. Compared to $\mathsf{FO}^{\perp}$ in Figure 5, the only change is the boxed area from $c \neq \mathsf{Enc}'(pk, m'; r'')$ to $r' \neq r''$ and the remaining parts are all the same. Thus, by proving that the equality $c = \mathsf{Enc}'(pk, m'; r'')$ is equivalent to the equality $r' = r''$, we can show that both $\mathsf{FO}^{\perp}$ and $\overline{\mathsf{FO}}^{\perp}$ work identically and thus achieve the same level of $\mathsf{IND\text{-}CCA}$ security.

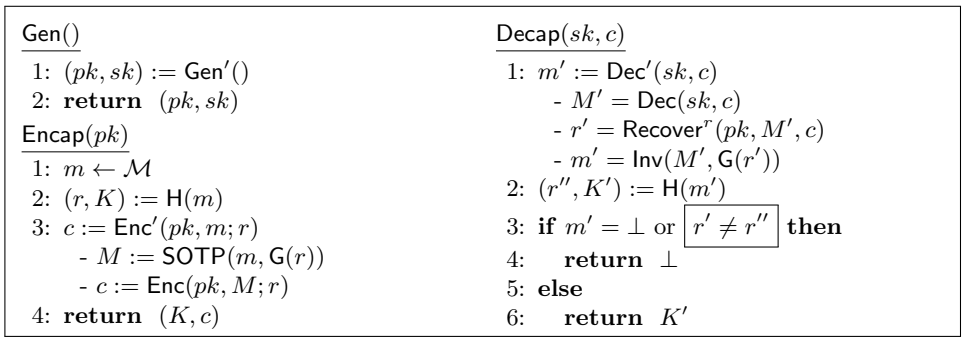

Fig. 6. $\mathsf{KEM} = \overline{\mathsf{FO}}^{\perp}[\mathsf{PKE}', \mathsf{H}]$

Lemma 2. *Let* PKE' *and* PKE *be injective, and let* PKE *and* 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* $\overline{\mathsf{FO}}^{\perp}$.

Proof. Assume that $c = \mathsf{Enc}'(pk, m'; r'')$ holds in Decap of $\mathsf{FO}^{\perp}$. Because PKE' is injective, the pair of (m, r) used in Encap are the same as (m', r''). This is, the injectivity of PKE' guarantees that $m = m'$ and $r = r''$. In this case, the ciphertext c generated by Encap is expressed as $c = \mathsf{Enc}(pk, \mathsf{SOTP}(m', \mathsf{G}(r'')); r'')$.

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

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

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

5 GenNTRU[ψ_1^n] (=PKE)

5.1 Notations

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

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

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

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 $n = 2^i 3^j$ for some positive integers i and j. For a polynomial $f \in R_q$, we use the notation '$f \leftarrow \psi_1^n$' to represent that each coefficient of f is drawn according to the distribution ψ_1. Also, we use the notation '$h \leftarrow R_q$' to show that a polynomial h is chosen uniformly at random from R_q. Later, to perform NTT over R_q, we will provide several parameter sets with respect to (n, q).

Other Notations. For a set $\{0,1\}^\ell$, we denote U^ℓ by the uniformly random distribution over the set $\{0,1\}^\ell$. Let $a \in \mathbb{Z}$ and $q \in \mathbb{Z}$ be an positive integer. We denote $x = a \bmod q$ the unique integer $x \in \{0, \cdots, q-1\}$ which satisfies $q | x - a$. For an odd integer q, we denote $y = a \bmod^{\pm} q$ the unique integer $y \in \{-(q-1)/2, \cdots, (q-1)/2\}$ which satisfies $q | x - a$.

5.2 Description of GenNTRU[ψ_1^n]

We define GenNTRU[ψ_1^n] relative to the distribution ψ_1 over R_q. Since $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ should be message and randomness recoverable for our ACWC_2, Figure 7 includes two additional algorithms $\mathsf{Recover}^r$ and $\mathsf{Recover}^m$.

$\underline{\mathsf{Gen}(1^\lambda)}$
1: $f', g \leftarrow \psi_1^n$
2: $f = 3f' + 1$
3: **if** f, g is not invertible in R_q **then**
4: restart
5: $h = 3gf^{-1}$
6: **return** $(pk, sk) = (h, f)$

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

$\underline{\mathsf{Dec}(f, c)}$
1: **return** $m = cf \mod^{\pm} 3$

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

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

Fig. 7. $\mathsf{GenNTRU}[\psi_1^n]$ with average-case correctness error

5.3 Cryptographic Assumptions

Definition 7 (R_q-$\mathsf{NTRU}_{\psi_1^n}$ assumption). *For a distribution ψ_1^n over the ring R_q and an integer p relatively-prime to q, the R_q-$\mathsf{NTRU}_{\psi_1^n}$ assumption states that $g(pf+1)^{-1}$ is indistinguishable from a uniformly-random element in R_q when g and f are chosen from the distribution ψ_1^n, and $pf+1$ is invertible in R_q.*

Definition 8 (R_q-$\mathsf{LWE}_{\psi_1^n}$). *The $\mathcal{R}$-LWE problem states that given $(h, hr + e)$ where $h \leftarrow R_q$ and $r, e \leftarrow \psi_1^n$, recover e.*

5.4 Security and Other Properties

Theorem 6 (OW-CPA security of $\mathsf{GenNTRU}[\psi_1^n]$). *For any adversary $\mathcal{A}$, there exist adversaries $\mathcal{B}$ and $\mathcal{C}$ such that*

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

Lemma 3 (Spreadness). $\mathsf{GenNTRU}[\psi_1^n]$ *is weakly n-spread.*

All proofs of Theorem 6 and Lemma 3 will be given in our full version [ePrint].

Average-Case Correctness Error. We analyze the average-case correctness error δ relative to the distribution $\psi_{\mathcal{M}} = \psi_{\mathcal{R}} = \psi_1^n$ by following the template given in [13]. We can expand c in the decryption algorithm as follows:

$$c = (hr + m)f = (3gf^{-1}r + m)(3f' + 1) = 3(gr + mf') + m.$$

Let c_i be the i-th coefficient of c, and $|c_i|$ be the absolute value of c_i. Then, $c_i \mod^{\pm} 3 = m_i$ if the following inequality holds:

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

where all the variables are distributed according to ψ_1^n. Let ϵ_i be

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

Then, assuming that each coefficients are independent from each other,

$$\Pr\left[\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) \neq m\right] = 1 - \prod_{i=0}^{n-1} \epsilon_i. \tag{3}$$

Since the coefficients of m have size at most 1,

$$\begin{aligned}
\epsilon_i &= \Pr\left[\left|3(gr + mf') + m\right|_i \leq \frac{q-1}{2}\right] && (4)\\
&\geq \Pr\left[\left|3(gr + mf')\right|_i + \left|m\right|_i \leq \frac{q-1}{2}\right] && (5)\\
&\geq \Pr\left[\left|3(gr + mf')\right|_i + 1 \leq \frac{q-1}{2}\right] && (6)\\
&= \Pr\left[\left|gr + mf'\right|_i \leq \frac{q-3}{6}\right] := \epsilon_i'. && (7)
\end{aligned}$$

Therefore,

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

Now, we analyze $\epsilon_i' = \Pr\left[\left|gr + mf'\right|_i \leq \frac{q-3}{6}\right]$. To 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

$$c = ba + b'(a + a') \in \{0, \pm 1, \pm 2, \pm 3\}, \text{ where } a, b, a, b \leftarrow \psi_1. \tag{9}$$

Also, 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 Eq. 9) and $2i$ independent random variables c' of the form

$$c' = ba + b'a' \in \{0, \pm 1, \pm 2\} \text{ where } a, b, a', b' \leftarrow \psi_1. \tag{10}$$

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

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

where $\theta_j = \Pr[c = j]$ (whose distribution is shown in Table 1) and $\theta'_j = \Pr[c' = j]$ (whose distribution is shown in Table 2). Let $\rho_{i,j}$ be the probability that the degree-i coefficient of $gr + mf'$ is j. Then, ϵ'_i can be computed as

$$\epsilon'_i = \begin{cases} 2 \cdot \sum_{j=(q+3)/6}^{3n} \rho_{i,j} & \text{for } i \in [n/2, n-1], \\ 2 \cdot \sum_{j=(q+3)/6}^{3n-2i} \rho_{i,j} & \text{for } i \in [0, n/2-1], \end{cases} \tag{12}$$

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

± 3	± 2	± 1	0
1/128	1/32	23/128	9/16

Table 1. Probability distribution of $c = ab + b'(a + a')$

± 2	± 1	0
1/64	3/16	19/32

Table 2. Probability distribution of $c' = ab + a'b'$

Rigidity and Injectivity. The rigidity of $\mathsf{GenNTRU}[\psi_1^n]$ is trivial from the algorithms Dec and $\mathsf{Recover}^r$ shown in Figure 7. The injectivity of $\mathsf{GenNTRU}[\psi_1^n]$ can be easily shown as follows: if there exist two inputs (m_1, r_1) and (m_2, r_2) such that $\mathsf{Enc}(h, m_1; r_1) = \mathsf{Enc}(h, m_2; r_2)$, the equality indicates that $(r_1 - r_2)h + (m_1 - m_2) = 0$, where $r_1 - r_2$ and $m_1 - m_2$ have still small coefficients. For a lattice set

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

the short polynomials $r_1 - r_2$ and $m_1 - m_2$ become an approximate shortest vector in $\mathcal{L}_0^\perp$. Thus, if the injectivity is broken against $\mathsf{GenNTRU}[\psi_1^n]$, we can solve the approximate shortest vector problem (SVP) over $\mathcal{L}_0^\perp$. It is well-known that the approximate SVP over $\mathcal{L}_0^\perp$ is at least as hard as the R_q-$\mathsf{NTRU}_{\psi_1^n}$ problem (defined above). Hence, if the R_q-$\mathsf{NTRU}_{\psi_1^n}$ assumption holds, then so is the injectivity of $\mathsf{GenNTRU}[\psi_1^n]$.

6 NTRU+

6.1 Instantiation of SOTP

We introduce $\mathsf{SOTP} : \mathcal{M}' \times \mathcal{U} \to \mathcal{M}$, where $\mathcal{M}' = \{0,1\}^n$, $\mathcal{U} = \{0,1\}^{2n}$, and $\mathcal{M} = \{-1, 0, 1\}^n$ relative to distributions $\psi_{\mathcal{M}'} = U^n$, $\psi_{\mathcal{U}} = U^{2n}$, and $\psi_{\mathcal{M}} = \psi_1^n$. Figure 8 presents SOTP which is used for ACWC_2.

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

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

Fig. 8. SOTP

Message-Hiding and Rigidity. It is easily shown that SOTP is message-hiding because of the one-time pad property, especially for the part $x \oplus u_1$. That is, unless u_1 is known, the message $x \in \mathcal{M}'$ is unconditionally hidden from $y \in \mathcal{M}$. Similarly, $x \oplus u_1$ becomes uniformly random over $\{0,1\}^n$, regardless of the message distribution $\psi_{\mathcal{X}}$, and thus the resulting y follows ψ_1^n. In addition, the rigidity of SOTP is trivial, because $\mathsf{Inv}(y, u) = x$ implies $\mathsf{SOTP}(x, u) = y$.

6.2 CPA-NTRU+ (=PKE′)

We obtain CPA-NTRU+ := $\mathsf{ACWC}_2[\mathsf{GenNTRU}[\psi_1^n], \mathsf{SOTP}, \mathsf{G}]$ by applying ACWC_2 from Section 3 to $\mathsf{GenNTRU}[\psi_1^n]$. Because the underlying $\mathsf{GenNTRU}[\psi_1^n]$ provides message and randomness recoverable properties, Theorem 2 and 3 give us the IND-CPA security of the resulting CPA-NTRU+ in the classical and quantum random oracle model, respectively. Regarding the correctness error, Theorem 1 shows that CPA-NTRU+ has the worst-case correctness error that is almost close to the average-case correctness error of $\mathsf{GenNTRU}[\psi_1^n]$. For instance, in case where $(n, q) = (768, 3457)$, the worst-case correctness error becomes about 2^{-379}, based on the equation of Theorem 1 and Eq. (8).

$\underline{\mathsf{Gen}'(1^\lambda)}$
1: $(pk, sk) := \mathsf{GenNTRU}[\psi_1^n].\mathsf{Gen}(1^\lambda)$
- $\mathbf{f}', \mathbf{g} \leftarrow \psi_1^n$
- $\mathbf{f} = 3\mathbf{f}' + 1$
- **if** $\mathbf{f}$, $\mathbf{g}$ are not invertible in R_q **then** restart
- $(pk, sk) = (\mathbf{h} = 3\mathbf{g}\mathbf{f}^{-1}, \mathbf{f})$
2: **return** (pk, sk)

$\underline{\mathsf{Enc}'(pk, m \in \{0,1\}^n; \mathbf{r} \leftarrow \psi_1^n)}$
1: $\mathbf{m} = \mathsf{SOTP}(m, \mathsf{G}(\mathbf{r}))$
2: $\mathbf{c} = \mathsf{GenNTRU}[\psi_1^n].\mathsf{Enc}(pk, \mathbf{m}; \mathbf{r})$
- $\mathbf{c} = \mathbf{h}\mathbf{r} + \mathbf{m}$
3: **return** $\mathbf{c}$

$\underline{\mathsf{Dec}'(sk, \mathbf{c})}$
1: $\mathbf{m} = \mathsf{GenNTRU}[\psi_1^n].\mathsf{Dec}(sk, \mathbf{c})$
- $\mathbf{m} = \mathbf{c}\mathbf{f} \mod^{\pm} 3$
2: $\mathbf{r} = \mathsf{Recover}^r(pk, \mathbf{c}, \mathbf{m})$
- $\mathbf{r} = (\mathbf{c} - \mathbf{m})\mathbf{h}^{-1}$
3: $m = \mathsf{Inv}(\mathbf{m}, \mathsf{G}(\mathbf{r}))$
4: **return** m

Fig. 9. CPA-NTRU+

Spreadness and Injectivity. To achieve the IND-CCA security of the transformed KEM via $\overline{\mathsf{FO}}^{\perp}$, we need to show the spreadness and injectivity of CPA-NTRU+. The spreadness can be easily obtained by combining Lemma 1 with Lemma 3. Next, the injectivity of CPA-NTRU+ can also be proven under the assumption that the R_q-$\mathsf{NTRU}_{\psi_1^n}$ problem is infeasible, analogously to that of $\mathsf{GenNTRU}[\psi_1^n]$. More precisely, if there exist two pairs $(m_1, \mathbf{r}_1)$ and $(m_2, \mathbf{r}_2)$ such that $\mathsf{Enc}'(pk, m_1; \mathbf{r}_1) = \mathsf{Enc}'(pk, m_2; \mathbf{r}_2)$, this results in the equation $\mathbf{hr}_1 + \mathbf{m}_1 = \mathbf{hr}_2 + \mathbf{m}_2$, where $\mathbf{m}_1 = \mathsf{SOTP}(m_1, \mathsf{G}(\mathbf{r}_1))$ and $\mathbf{m}_2 = \mathsf{SOTP}(m_2, \mathsf{G}(\mathbf{r}_2))$. In that case, we still have two short polynomials $\mathbf{r}_1 - \mathbf{r}_2$ and $\mathbf{m}_1 - \mathbf{m}_2$ that can be a solution of approximate SVP over $\mathcal{L}_0^{\perp}$.

6.3 CCA-NTRU+ (=KEM)

Finally, we can achieve the IND-CCA secure KEM by applying $\overline{\mathsf{FO}}^{\perp}$ to the CPA-NTRU+. We denote such KEM by CCA-NTRU+ $:= \overline{\mathsf{FO}}^{\perp}[\text{CPA-NTRU+}, \mathsf{H}]$. Figure 10 presents the resultant CCA-NTRU+, which is the basis of our specification and implementation in the next section. Putting Theorem 4, Theorem 5 and Lemma 2 altogether, we can achieve the IND-CCA security of CCA-NTRU+. As for correctness error, CCA-NTRU+ preserves the worst-case correctness error of the underlying CPA-NTRU+.

$\underline{\mathsf{Gen}(1^\lambda)}$
1: $\mathbf{f}', \mathbf{g} \leftarrow \psi_1^n$
2: $\mathbf{f} = 3\mathbf{f}' + 1$
3: **if** $\mathbf{f}$, $\mathbf{g}$ are not invertible in R_q **then**
4: restart
5: **return** $(pk, sk) = (\mathbf{h} = 3\mathbf{gf}^{-1}, \mathbf{f})$

$\underline{\mathsf{Encap}(pk)}$
1: $m \leftarrow \{0,1\}^n$
2: $(\mathbf{r}, K) = \mathsf{H}(m)$
3: $\mathbf{m} = \mathsf{SOTP}(m, \mathsf{G}(\mathbf{r}))$
4: $\mathbf{c} = \mathbf{hr} + \mathbf{m}$
5: **return** $(\mathbf{c}, K)$

$\underline{\mathsf{Decap}(sk, \mathbf{c})}$
1: $\mathbf{m} = \mathbf{cf} \mod^{\pm} 3$
2: $\mathbf{r} = (\mathbf{c} - \mathbf{m})\mathbf{h}^{-1}$
3: $m = \mathsf{Inv}(\mathbf{m}, \mathsf{G}(\mathbf{r}))$
4: $(\mathbf{r}', K) = \mathsf{H}(m)$
5: **if** $\mathbf{r} = \mathbf{r}'$ **then**
6: **return** K
7: **else**
8: **return** $\perp$

Fig. 10. CCA-NTRU+

7 Specification

7.1 Notation

Encoding and Decoding We define the function Encode_q in Algorithm 1, which compress the polynomial to $3n/2$ byte array. It assumes that each coefficient of polynomial are stored in 16-bit data type before it is compressed. The design concept of Encode_q to make it efficient when it is implemented with AVX2 instruction set. We define the function Decode_q as the inverse of Encode_q.

Algorithm 1 Encode_q

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

for i from 0 to $\lfloor n/64 \rfloor - 1$ **do**
  for j from 0 to 12 **do**
    $t_0 = f_{64j+i}$
    $t_1 = f_{64j+i+16}$
    $t_2 = f_{64j+i+32}$
    $t_3 = f_{64j+i+48}$
    $b_{96j+2i+0} = t_0$
    $b_{96j+2i+1} = (t_0 >> 8) + (t_1 << 4)$
    $b_{96j+2i+32} = t_1 >> 4$
    $b_{96j+2i+33} = t_2$
    $b_{96j+2i+64} = (t_2 >> 8) + (t_3 << 4)$
    $b_{96j+2i+65} = t_3 >> 4$
if $n = 864$ **then**
  for i from 0 to 7 **do**
    $t_0 = f_{832+i}$
    $t_1 = f_{832+i+8}$
    $t_2 = f_{832+i+16}$
    $t_3 = f_{832+i+24}$
    $b_{1248+2i+0} = t_0$
    $b_{1248+2i+1} = (t_0 >> 8) + (t_1 << 4)$
    $b_{1248+2i+16} = t_1 >> 4$
    $b_{1248+2i+17} = t_2$
    $b_{1248+2i+32} = t_1 >> 4$
    $b_{1248+2i+33} = t_2$
return $(b_0, \cdots, b_{3n/2-1})$

Sampling from a Binomial distribution NTRU+ uses centered binomial distribution with $\eta = 1$ to sample the coefficients of polynomials which is defined in Algorithm 3. We also define $\mathsf{BytesToBits}$ in Algorithm 2 to decide the order of sampled coefficients. $\mathsf{BytesToBits}$ help us to implement the CBD_1 and SOTP efficiently with the AVX2 instructions. We define $\mathsf{BitsToBytes}$ as the inverse of $\mathsf{BytesToBits}$ function.

Algorithm 2 BytesToBits

Input: Byte array $B = (b_0, b_1, \cdots, b_{n/4-1})$
Output: Polynomial $f \in R_q$

$x = \lfloor n/256 \rfloor$
$y = n - 256x$
$(y_0, y_1, y_2, y_4, y_5, y_6, y_7, y_8) :=$ *bit-decompose*(b) // $y = y_0 2^0 + \cdots y_8 2^8$
for i from 0 to $x - 1$ **do**
  for j from 0 to 7 **do**
    $t_1 = b_{32i+4j+3} | b_{32i+4j+2} | b_{32i+4j+1} | b_{32i+4j}$
    for k from 0 to 1 **do**
      for l from 0 to 16 **do**
        $f_{256i+16l+2j+k} = t_1 \& 1$;
        $t_1 >> 1$;
$c_1 = 256x$, $c_2 = 32x$
if $y_8 = 1$ **then**
  for j from 0 to 3 **do**
    $t_1 = b_{c_2+4j+3} | b_{c_2+4j+2} | b_{c_2+4j+1} | b_{c_2+4j}$
    for k from 0 to 1 **do**
      for l from 0 to 16 **do**
        $f_{c_1+8l+2j+k} = t_1 \& 1$;
        $t_1 >> 1$;
$c_1 = 256x + 128y_8$, $c_2 = 32x + 16y_8$
if $y_7 = 1$ **then**
  for j from 0 to 1 **do**
    $t_1 = b_{c_2+4j+3} | b_{c_2+4j+2} | b_{c_2+4j+1} | b_{c_2+4j}$
    for k from 0 to 1 **do**
      for l from 0 to 16 **do**
        $f_{c_1+4l+2j+k} = t_1 \& 1$;
        $t_1 >> 1$;
$c_1 = 256x + 128y_8 + 64y_7$, $c_2 = 32x + 16y_8 + 8y_7$
if $y_6 = 1$ **then**
  $t_1 = b_{c_2+3} | b_{c_2+2} | b_{c_2+1} | b_{c_2}$
  for k from 0 to 1 **do**
    for l from 0 to 16 **do**
      $f_{c_1+2l+k} = t_1 \& 1$;
      $t_1 >> 1$;
return $f_0 + f_1 x + f_2 x^2 + \cdots + f_{n-1} x^{n-1}$

Algorithm 3 $\mathsf{CBD}_1 : \mathcal{B}^{n/4} \rightarrow R_q$

Input: Byte array $B = (b_0, b_1, \cdots, b_{n/4-1})$
Output: Polynomial $y \in R_q$

$(\beta_0, \cdots, \beta_{2n-1}) :=$ BytesToBits(B)
for i from 0 to $n - 1$ **do**
  $f_i := \beta_i - \beta_{i+n}$
return $(f_0, \cdots, f_{n-1})$

Semi-generalized one time pad The function SOTP is identical to CBD_1 except that it computes exclusive or to the first half of the random bytes with the message before sampling centered binomial distribution. Therefore, the function SOTP defined in Algorithm 4 also uses the function BytesToBits as like in CBD_1. We define the function Inv in Algorithm 5 as the inverse of the function SOTP. It uses the function BitsToBytes to recover the bytes.

Algorithm 4 SOTP

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

Algorithm 5 Inv

Input: Polynomial $y \in R_q$
Input: Byte array $B = (b_0, b_1, \cdots, b_{n/4-1})$
Output: Message Byte array $m = (m_0, m_1, \cdots, m_{31})$
1: $(\beta_0, \cdots, \beta_{2n-1}) := \mathsf{BytesToBits}(B)$
2: **for** i from 0 to $n-1$ **do**
3: $\quad m_i := ((f_i + \beta_{i+n}) \& 1) \oplus \beta_i$
$m = \mathsf{BitsToBytes}((m_0, \cdots, m_{n-1}))$
4: **return** m

Symmetric Primitives NTRU+ uses extendable output function XOF and two different hash functions H and G as symmetric primitives. To instantiate XOF, we use AES256-CTR. To instantiate hash function G and H, we use SHA256, SHA512 and XOF as follows.

Algorithm 6 H

Input: Message Byte array $m = (m_0, m_1, \cdots, m_{n/8})$
Output: Byte array $B = (b_0, b_1, \cdots, b_{n/8+31})$
1: $(b_0, \cdots, b_{31}, b_{32}, \cdots b_{63}) := \mathsf{SHA512}(m, n/8)$;
2: $(b_{32}, \cdots b_{n/8+31}) = \mathsf{XOF}((b_{32}, \cdots, b_{63}), n/4)$
3: **return** $(b_0, \cdots b_{n/8+31})$

Algorithm 7 G

Input: Message Byte array $m = (m_0, m_1, \cdots, m_{n/8})$
Output: Byte array $B = (b_0, b_1, \cdots, b_{n/8+31})$
1: $(b_0, \cdots, b_{31}) := \mathsf{SHA256}(m, n/8)$;
2: $(b_0, \cdots b_{n/8-1}) = \mathsf{XOF}((b_0, \cdots, b_{31}), n/4)$
3: **return** $(b_0, \cdots b_{n/8-1})$

Number Theoretic Transform NTRU+ uses number theoretic transform to compute polynomial multiplications and polynomial inverses. We denote NTT the number theoretic transform function and NTT^{-1} the inverse number theoretic transform function. The detailed composition of number theoretic transform used in NTRU+ is described in section 8.2.

7.2 Specification of CCA-NTRU+

Algorithm 8 $\mathsf{Gen}(1^\lambda)$: key generation

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

Algorithm 9 $\mathsf{Encap}(pk)$: encapsulation

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

Algorithm 10 $\mathsf{Decap}(sk, c)$: decapsulation

Input: Secret key $sk \in \mathcal{B}^{\lceil \log_2 q \rceil \cdot n/4}$
Input: Ciphertext $c \in \mathcal{B}^{\lceil \log_2 q \rceil \cdot n/8}$
Output: Shared key $m \in \mathcal{B}^{32}$
$\hat{\mathbf{f}} = \mathsf{Decode}_q(sk)$
$\hat{\mathbf{c}} = \mathsf{Decode}_q(c)$
$\hat{\mathbf{h}}^{-1} = \mathsf{Decode}_q(sk + \lceil \log_2 q \rceil \cdot n/8)$
$\mathbf{m} = \mathsf{NTT}^{-1}(\hat{\mathbf{c}} \circ \hat{\mathbf{f}}) \mod^{\pm} 3$
$\hat{\mathbf{m}} = \mathsf{NTT}(\mathbf{m})$
$\hat{\mathbf{r}} = (\hat{\mathbf{c}} - \hat{\mathbf{m}}) \circ \hat{\mathbf{h}}^{-1}$ // $\mathsf{Recover}^r$
$m := \mathsf{Inv}(\mathbf{m}, \mathsf{G}(\mathsf{Encode}_q(\hat{\mathbf{r}})))$
$(K, \mathbf{r}) := \mathsf{H}(m)$
if $\mathbf{r} = \mathbf{r}'$ **then**
  return K
else
  return $\perp$

7.3 Parameter sets

We define four parameter sets for NTRU+, which we call NTRU+576, NTRU+864, and NTRU+1152. The parameters are listed in Table 3. Note that the table also lists the derived parameter δ, which is the probability that decapsulation of a valid ciphertext fails.

	n	q	sk(byte)	pk(byte)	ct(byte)	sec(c)	sec(q)	δ
NTRU+576	576	3,457	1,728	864	864	115	104	-487
NTRU+768	768	3,457	2,304	1,152	1,152	164	148	-379
NTRU+864	864	3,457	2,592	1,296	1,296	188	171	-340
NTRU+1152	1,152	3,457	3,456	1,728	1,728	264	240	-260

Table 3. Paramter sets for NTRU+

8 Performance analysis

8.1 Description of platform

All benchmarks were obtained on single core of an Intel Core i7-8700K (Coffee Lake) processor clocked at 3700 MHz. The bench-marking machine has 16 GB of RAM. Both implementations were compiled with gcc version 9.4.0 and the compiler flags as indicated in the Makefiles included in the submission package. All cycle counts reported are the average of the cycle counts of 100,000 executions of the respective function.

8.2 Implementation considerations

Implementing the NTT. To implement polynomial multiplication, we use number theoretic transform. To realize number theoretic transform over the polynomial ring $\mathbb{Z}_q[x]/\langle x^n - x^{n/2} + 1\rangle$ with $n = 2^a 3^b$, we use three different types of NTT layers : Radix-2 NTT, Radix-2 NTT with cyclotomic trinomial, and Radix-3. We first adapt one Radix-2 NTT with cyclotomic trinomial layer, which is first used in [13], to factorize the polynomial ring $\mathbb{Z}_q[x]/\langle x^n - x^{n/2} + 1\rangle$ to $\mathbb{Z}_q[x]/\langle x^{n/2} - \psi\rangle \times \mathbb{Z}_q[x]/\langle x^{n/2} - \psi^5\rangle$ where ψ is primitive 6-th root of unity modulus q. After that, we use Radix-3 NTT layers to reduce multiples of 3 in the degree of the polynomial. At last, we use Radix-2 NTT until it reaches degree 2 or 3. Note that we use Radix-3 NTT before Radix-2 NTT to minimize the size of the predefined table required to transform and multiply polynomials.

n	q	Radix-2 with trinomial	Radix-3	Radix-2	inertia degree
576	3457	1	2	4	2
768	3457	1	1	4	2
864	3457	1	2	4	3
1152	3457	1	2	5	2

Table 4. Combinations of NTT layers

For the completeness, we describe the Radix-3 NTT layer used in our implementation. Radix-3 NTT layer is a ring isomorphism from $\mathbb{Z}_q[x]/\langle x^{3n} - \zeta^3\rangle$

to $\mathbb{Z}_q[x]/\langle x^n-\alpha\rangle \times \mathbb{Z}_q[x]/\langle x^n-\beta\rangle \times \mathbb{Z}_q[x]/\langle x^n-\gamma\rangle$ where $\alpha=\zeta$, $\beta=\zeta\omega$, $\gamma=\zeta\omega^2$ (ω is primitive 3rd root of unity in modulus q). To transform $a(x)=a_0(x)+a_1(x)x^n+a_2(x)x^{2n}\in\mathbb{Z}_q[x]\langle x^{3n}-\zeta^3\rangle$($a_0(x)$, $a_1(x)$, and $a_2(x)$ are polynomial with maximum degree $n-1$) to $(\hat{a}_0(x),\hat{a}_1(x),\hat{a}_2(x))\in\mathbb{Z}_q[x]/\langle x^n-\alpha\rangle\times\mathbb{Z}_q[x]/\langle x^n-\beta\rangle\times\mathbb{Z}_q[x]/\langle x^n-\gamma\rangle$, we need to compute following equations.

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

Naively, we can compute above equations with $6n$ multiplications, $6n$ additions with 6 predefined values, α, α^2, β, β^2, γ, and γ^2. We can reduce the amount of computation to $4n$ multiplications, $5n$ additions, n subtractions with only 4 predefined values α, α^2, ω, and ω^2 as described in Algorithm 11. Note that ω, and ω^2 can be reused in the computation of other Radix-3 NTT layers.

Algorithm 11 Radix-3 NTT layer

Input: $a(x)=a_0(x)+a_1(x)x^n+a_2(x)x^{2n}\in\mathbb{Z}_q[x]/\langle x^{3n}-\zeta^3\rangle$
Output: $(\hat{a}_0(x),\hat{a}_1(x),\hat{a}_2(x))\in\mathbb{Z}_q[x]/\langle x^n-\alpha\rangle\times\mathbb{Z}_q[x]/\langle x^n-\beta\rangle\times\mathbb{Z}_q[x]/\langle x^n-\gamma\rangle$

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

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

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

Naively, we can compute above equation with $6n$ multiplications, $6n$ additions with 6 predefined values, α^{-1}, α^{-2}, β^{-1}, β^{-2}, γ^{-1}, and γ^{-2}. We can reduce the over all computations to $4n$ multiplications, $5n$ additions, n subtractions with only 4 predefined values, α^{-1}, α^{-2}, ω^1, and ω^2 as described in Algorithm 12.

Note that ω, and ω^2 can be reused in the computation of other Radix-3 NTT and Radix-3 Inverse NTT layers.

Algorithm 12 Radix-3 Inverse NTT layer

Input: $(\hat{a}_0(x), \hat{a}_1(x), \hat{a}_2(x)) \in \mathbb{Z}_q[x]/\langle x^n - \alpha\rangle \times \mathbb{Z}_q[x]/\langle x^n - \beta\rangle \times \mathbb{Z}_q[x]/\langle x^n - \gamma\rangle$
Output: $3a(x) = 3a_0(x) + 3a_1(x)x^n + 3a_2(x)x^{2n} \in \mathbb{Z}_q[x]/\langle x^{3n} - \zeta^3\rangle$

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

8.3 Performance of reference implementation

Table 5 reports performance results of the reference implementation of NTRU+ together with the sizes of keys and ciphertexts. We also provide the performance result of the AVX2 implementation of NTRU+576.

Table 5. Performance result of NTRU+

NTRU+576			
Sizes (in Bytes)		**K Cycles (ref)**	
sk:	1,728	**gen:**	326
pk:	864	**encap:**	107
ct:	864	**decap:**	161
NTRU+768			
Sizes (in Bytes)		**K Cycles (ref)**	
sk:	2,304	**gen:**	328
pk:	1,152	**encap:**	146
ct:	1,152	**decap:**	218
NTRU+864			
Sizes (in Bytes)		**K Cycles (ref)**	
sk:	2,592	**gen:**	370
pk:	1,296	**encap:**	169
ct:	1,296	**decap:**	260
NTRU+1152			
Sizes (in Bytes)		**K Cycles (ref)**	
sk:	3,456	**gen:**	906
pk:	1,728	**encap:**	224
ct:	1,728	**decap:**	337

9 Security

9.1 Security strength Based on LWE problem

We use the script of Kyber[15] to analyze the security strength of NTRU+. It can be found at `https://github.com/pq-crystals/security-estimates`. Table 6 shows classical and quantum core-SVP-hardness of the four parameter sets of NTRU+. The result of analysis also can be found at `https://github.com/ntruplus/ntruplus/tree/main/scripts/security`.

	NTRU+576	NTRU+768	NTRU+864	NTRU+1152
NIST Security level	1	1	3	5

Core-SVP methodology, Primal attack

lattice attack dim.	1054	1397	1573	2045
BKZ-blocksize	399	560	655	922
core-SVP classical hardness	116	163	191	269
core-SVP quantum hardness	105	148	173	244

Core-SVP methodology, Dual attack

lattice attack dim.	1045	1370	1577	2009
BKZ-blocksize	395	553	645	905
core-SVP classical hardness	115	161	188	264
core-SVP quantum hardness	104	146	171	240

Table 6. Classical and quantum hardness of the different proposed parameter sets of NTRU+ together with the claimed security level.

9.2 Security strength Based on NTRU problem

In case of concrete analysis of LWE problem, there are various security analysis tools including the work of [1], [2], and [4]. However, this is not the case for the NTRU Problem. Since we use the very same assumption with the NTRU[3], the finalist of the 3rd round of NIST PQC Standardization, we can analyzes NTRU problem for our scheme based on the analysis of NTRU[3] in the future work.

10 Conclusion

We propose a key encapsulation mechanism on the hardness of the NTRU problem and RLWE problem. Unlike the finalist NTRU [3] in the NIST PQC finalist, we allow decryption errors and use number theoretic transform to improved the efficiency. To prevent attacker to exploit decryption errors, we constructed new average-case to worst-case correctness generic transformation, called ACWC_2 to

construct IND-CPA secure public key encryption. Also, we applied FO-Equivalent Transform without Re-encryption to construct efficient IND-CCA secure key encapsulation mechanism. Unlike other NTRU based scheme, we use centered binomial distribution so that it is more convenient to sample.

References

1. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. J. Math. Cryptol. **9**(3), 169–203 (2015), http://www.degruyter.com/view/j/jmc.2015.9.issue-3/jmc-2015-0016/jmc-2015-0016.xml
2. Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum key exchange - A new hope. In: Holz, T., Savage, S. (eds.) USENIX Security 2016: 25th USENIX Security Symposium. pp. 327–343. USENIX Association, Austin, TX, USA (Aug 10–12, 2016)
3. Chen, C., Danba, O., Hoffstein, J., Hulsing, A., Rijneveld, J., Schanck, J.M., Schwabe, P., Whyte, W., Zhang, Z., Saito, T., Yamakawa, T., Xagawa, K.: NTRU. Tech. rep., National Institute of Standards and Technology (2020), available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions
4. Dachman-Soled, D., Ducas, L., Gong, H., Rossi, M.: LWE with side information: Attacks and concrete security estimation. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology – CRYPTO 2020, Part II. Lecture Notes in Computer Science, vol. 12171, pp. 329–358. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2020). https://doi.org/10.1007/978-3-030-56880-1_12
5. D'Anvers, J.P., Guo, Q., Johansson, T., Nilsson, A., Vercauteren, F., Verbauwhede, I.: Decryption failure attacks on IND-CCA secure lattice-based schemes. In: Lin, D., Sako, K. (eds.) PKC 2019: 22nd International Conference on Theory and Practice of Public Key Cryptography, Part II. Lecture Notes in Computer Science, vol. 11443, pp. 565–598. Springer, Heidelberg, Germany, Beijing, China (Apr 14–17, 2019). https://doi.org/10.1007/978-3-030-17259-6_19
6. D'Anvers, J.P., Karmakar, A., Roy, S.S., Vercauteren, F., Mera, J.M.B., Beirendonck, M.V., Basso, A.: SABER. Tech. rep., National Institute of Standards and Technology (2020), available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions
7. Don, J., Fehr, S., Majenz, C., Schaffner, C.: Online-extractability in the quantum random-oracle model. In: Dunkelman, O., Dziembowski, S. (eds.) Advances in Cryptology – EUROCRYPT 2022, Part III. Lecture Notes in Computer Science, vol. 13277, pp. 677–706. Springer, Heidelberg, Germany, Trondheim, Norway (May 30 – Jun 3, 2022). https://doi.org/10.1007/978-3-031-07082-2_24
8. Duman, J., Hövelmanns, K., Kiltz, E., Lyubashevsky, V., Seiler, G., Unruh, D.: A thorough treatment of highly-efficient NTRU instantiations. Cryptology ePrint Archive, Report 2021/1352 (2021), https://eprint.iacr.org/2021/1352
9. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M.J. (ed.) Advances in Cryptology – CRYPTO'99. Lecture Notes in Computer Science, vol. 1666, pp. 537–554. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 15–19, 1999). https://doi.org/10.1007/3-540-48405-1_34
10. Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: A ring-based public key cryptosystem. In: Third Algorithmic Number Theory Symposium (ANTS). Lecture

Notes in Computer Science, vol. 1423, pp. 267–288. Springer, Heidelberg, Germany (Jun 1998)
11. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the Fujisaki-Okamoto transformation. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017: 15th Theory of Cryptography Conference, Part I. Lecture Notes in Computer Science, vol. 10677, pp. 341–371. Springer, Heidelberg, Germany, Baltimore, MD, USA (Nov 12–15, 2017). https://doi.org/10.1007/978-3-319-70500-2_12
12. Lyubashevsky, V., Micciancio, D., Peikert, C., Rosen, A.: SWIFFT: A modest proposal for FFT hashing. In: Nyberg, K. (ed.) Fast Software Encryption – FSE 2008. Lecture Notes in Computer Science, vol. 5086, pp. 54–72. Springer, Heidelberg, Germany, Lausanne, Switzerland (Feb 10–13, 2008). https://doi.org/10.1007/978-3-540-71039-4_4
13. Lyubashevsky, V., Seiler, G.: NTTRU: Truly fast NTRU using NTT. IACR Transactions on Cryptographic Hardware and Embedded Systems **2019**(3), 180–201 (2019). https://doi.org/10.13154/tches.v2019.i3.180-201, https://tches.iacr.org/index.php/TCHES/article/view/8293
14. Schanck, J.M., Hulsing, A., Rijneveld, J., Schwabe, P.: NTRU-HRSS-KEM. Tech. rep., National Institute of Standards and Technology (2017), available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions
15. Schwabe, P., Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Seiler, G., Stehlé, D.: CRYSTALS-KYBER. Tech. rep., National Institute of Standards and Technology (2020), available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions
16. Zhang, Z., Chen, C., Hoffstein, J., Whyte, W.: NTRUEncrypt. Tech. rep., National Institute of Standards and Technology (2017), available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions