How Compact Can NTRU Encryption Be? Heuristic Frontiers and Practical Schemes

Yijian Liu[1,2], Yu Zhang[1,2], Xianhui Lu[1,2](✉), Yao Cheng, and Yongjian Yin[1,2]

[1] State Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
{liuyijian, luxianhui}@iie.ac.cn

Abstract. NTRU is one of the longest-tested lattice-based public-key encryption families and is often viewed as a compact alternative to (R/M)-LWE. Yet, after three decades of research, its potential for compactness remains an open area for further exploration: recent designs such as NEV (Asiacrypt 2023) and DAWN (Asiacrypt 2025) suggest that there is still room for improvement. This raises a natural question: Has NTRU reached its compactness limit? If not, how compact can it be while still remaining secure and efficient?

Motivated by this question, we aim to formalize a unified relationship between compactness and efficiency under the required security level. We present a common two-stage view of NTRU decryption. In the first stage, the decoder constructs a small set of candidate wrap-around errors. In the second stage, it verifies these candidates using either algebraic redundancy or trapdoor-derived distributional information. We introduce *Free Candidate Localization* (FCL), a generic first-stage method that ranks coordinates by their proximity to the centered boundary. FCL could be used in most lattice-based encryptions; we instantiate it in ML-KEM to achieve a 10% smaller ciphertext at the cost of a 5% slower overall runtime in the reference C implementation under NIST-I.

We organize modern NTRU encryptions into two frameworks. *NTRU with Encoding* leverages algebraic structure via auxiliary quotient rings. *NTRU with Trapdoor* exploits geometric structure through the NTRU trapdoor, then verifies candidate corrections using distributional tests. These frameworks give a common language for existing NTRU designs and for the compactness searches in this paper. Within an explicit search model, we derive heuristic compactness frontiers for both frameworks. At NIST-I, the encoding frontier yields a total public-key plus ciphertext size of 812 bytes, and the trapdoor frontier yields a size of 754 bytes, 15%/21% smaller than the previous lowest size of 964 bytes in DAWN.

Furthermore, we propose END, an instantiation of NTRU with Trapdoor plus FCL. At NIST-I, END has a 384-byte ciphertext, which is exactly half the size of the 768-byte ciphertext of ML-KEM-512, and is 12–19% smaller than the shortest prior NTRU-style ciphertexts in BAT (TCHES 2022) and DAWN. In our reference C implementation, the combined encapsulation and decapsulation cost of END-512 is about 3% higher than that of ML-KEM-512.

Keywords: lattice-based cryptography · NTRU · PKE · KEM

1 Introduction

NTRU, proposed by Hoffstein, Pipher, and Silverman [24], is one of the earliest lattice-based encryption schemes. NTRU remains attractive because its ciphertext can be represented by a single ring element. Ring-LWE and Module-LWE ciphertexts typically contain two components, but the message-bearing component can often be aggressively rounded; compactness therefore has to be compared at the encoded-byte level rather than by component count alone. During the NIST post-quantum cryptography (PQC) standardization process, two NTRU variants demonstrated significant promise: NTRU Prime [5] advanced to the third round of candidates, while NTRU [9] (originating from [24], a merger of NTRU-HPS and NTRU-HRSS) progressed to the third round of finalists. Despite their advancement, neither was ultimately selected; Kyber [8] (ML-KEM [39]) was eventually standardized as the lattice-based KEM. Nevertheless, the cryptographic community continues to actively explore and enhance NTRU structures, with numerous improvements and variants proposed in recent years [32,18,14,41,30], enhancing the competitiveness of NTRU-based schemes.

Among modern NTRU decryption methods, there are two main approaches for extracting the message: one focuses on algebraic form, and the other on a geometric view. Most NTRU encryptions follow the original NTRU [24] to encode the message into some subspace via integer coefficients in the original NTRU, or polynomial encodings in NEV [41] and DAWN [30]. The other way to extract the message through the NTRU trapdoor without an encoding structure has only one representative example: BAT [18], since generating the NTRU trapdoor is expensive, making key generation much slower than that of other schemes.

To understand the decryption more generally, we use a two-stage view. The first stage constructs a small candidate set for the wrap-around error. The second stage verifies these candidates and selects the unique one that gives a valid message or valid secret distribution. In algebraic constructions, verification comes from quotient-ring redundancy in the message encoding. In trapdoor constructions, verification comes from the distribution of the vector returned by a trapdoor closest-vector procedure. This separation is useful because the same first-stage candidate-generation method can be combined with different verification rules. Most NTRU-based schemes choose the ring, modulus, distributions, and encoding so that there's no wrap-around error except with negligible or zero probability, the candidate in the first stage is $\{0\}$, and there's no second stage. DAWN [30] may instead tolerate rare wrap events and recover correctly when the wrapped error is a monomial. Concretely, DAWN uses side information from the quotient to obtain a candidate set of five wrap-error candidates in the first stage and then finds the unique correct one via side information from the centered residual.

1.1 Our Contributions

This work asks how compact NTRU-based encryption can be while maintaining efficient decryption and meeting the required security level.

– **A universal technique.** We introduce FCL, a generic candidate-localization method for centered lattice decryption residues. FCL outputs a small, publicly sized family of possible wrap-around errors. We give its candidate-set construction, a failure-probability estimate under an explicit coefficient model, and a list-decoding interface that separates candidate generation from verification. This technique can be applied to Ring/Module-LWE or NTRU decryption residues. We instantiate it on ML-KEM to obtain a 10% smaller ciphertext by the cost of 5% total runtime.
– **Two generic frameworks.** We formalize two decryption frameworks. NTRU with Encoding uses auxiliary quotient rings to obtain algebraic side information on the wrap-around error. NTRU with Trapdoor uses an NTRU trapdoor to reduce decryption to a structured closest-vector problem and then verifies candidate corrections by distributional tests. We also provide a specialization table that explains how existing NTRU-style schemes fit within these frameworks at the design-pattern level.
– **Two heuristic frontiers.** We derive compactness frontiers within a fixed search model: power-of-two cyclotomic rings, fixed-target security, fixed-decryption-failure target, fixed public candidate-trial budget, and the coefficient-independence heuristic used in the DFR calculation. These frontiers are heuristic design frontiers inside the stated model. In this model, the NTRU-with-Encoding frontier at NIST-I yields a total public-key plus ciphertext size of 812 bytes, and the NTRU-with-Trapdoor frontier yields 754 bytes, 15%/21% smaller than the previous lowest size of 964 bytes in DAWN [30].
– **A practical encryption scheme.** We instantiate NTRU with Trapdoor as END[1]. At NIST-I, END-512 has a 384-byte ciphertext and a 514-byte public key. Its ciphertext is exactly half the size of ML-KEM-512 and is 12–19% smaller than BAT and DAWN at the same security level. Also, the encapsulation and decapsulation time of END is comparable to that of those schemes. The tradeoff is key generation: END applies the slow NTRU trapdoor generation similar to that in BAT [7]. This makes it most appropriate in settings where the public key is reused over many encapsulations. The comparisons of size and performance are detailed in Table 4 and Table 5.

1.2 Technical Overview

We start by modeling the NTRU decryption through a three-term decomposition

$$\boldsymbol{v} = \boldsymbol{\eta} + \boldsymbol{\mu} - q \cdot \boldsymbol{e},$$

where $\boldsymbol{\eta}$ is the noise term induced by the secret key and encryption randomness, $\boldsymbol{\mu}$ is the encoded message term, and $\boldsymbol{e}$ records the wrap-around error. The term $\boldsymbol{e}$ is sparse when the parameters are close to correct decryption but slightly beyond the ordinary no-wrap regime. A decoder can therefore try to recover $\boldsymbol{e}$ by searching over a small candidate family. The challenge is to make this family small enough for implementation and large enough to contain the true error with probability at least $1 - 2^{-\lambda_{\mathrm{DFR}}}$.

[1] END stands for **E**xplore **N**TRU **D**ecryption.

Free Candidate Localization. For the first stage to build a candidate family, we introduce FCL, which uses the centered residue $\boldsymbol{v}$ itself. If an unreduced coefficient wraps around modulo q, the centered residue often lands close to one of the two centered boundaries $\pm\frac{q}{2}$. FCL ranks coordinates by this boundary distance and keeps the most likely positions. From these positions, it builds a candidate family with bounded weight and bounded coefficient magnitude. The size of this family is public once the parameters are fixed, so the enumeration can be implemented without secret-dependent loop lengths. FCL is most informative when the unwrapped coefficient distribution is roughly symmetric, and has limited mass near the centered boundary. These conditions hold for the NTRU parameter sets studied in this paper and for most lattice decryption residues.

NTRU with Encoding. The encoding framework uses public polynomials $t, w \in \mathbb{Z}[x]$ and quotient rings $\mathbb{Z}_{p_i}[x]/\langle\phi_i\rangle$. The parameters are chosen so that multiplication by t kills the noise term in the decoding and auxiliary rings, while multiplication by w preserves the message in the decoding ring and kills the message in the auxiliary rings. After decryption, the decoder sees

$$t\eta + w\mu - q \cdot e \quad \text{in } \mathcal{R}_\varphi.$$

Reducing this expression modulo the decoding ring gives information about $w\mu - q\cdot e$. Reducing it modulo the auxiliary rings gives information about $-q\cdot e$. These residues restrict the candidate set for e. FCL can then further restrict the possible support of e, and an unembedding map recovers the original message after the right candidate is selected. The resulting framework is useful as an organizing principle. It captures the design pattern behind most existing NTRU decoders except BAT, and it explains the DAWN-style point as a frontier point under a tight candidate-trial budget.

NTRU with Trapdoor. The trapdoor framework uses a trapdoor basis of the NTRU lattice. A Babai nearest-plane computation removes the dominant noise term and returns a polynomial that equals the desired μ plus offsets determined by two rounding errors. FCL gives candidates for these two rounding errors. The candidates are verified using distributional distinguishment: the correct correction yields a vector whose secret and message distributions match those specified by the scheme, whereas incorrect corrections often leave visible range or norm violations. NTRU with Trapdoor generalizes the structure of BAT [18], involves a candidate search process, and offers greater compactness than the encoding framework in our search model, because the trapdoor provides geometric side information that does not consume message-space redundancy. The price is slow key generation due to the computation of the trapdoor basis.

The practical scheme. The main practical construction is END. It instantiates NTRU with Trapdoor, uses FCL for both rounding-error terms, and uses two fixed public candidate bounds. The implementation evaluates all candidates up

to the public bound, uses branchless score computation and constant-time winner selection, and includes masking for the sensitive arithmetic. At NIST-I, the ciphertext is 384 bytes, which is exactly half of ML-KEM. Although it has slow key generation, its encapsulation and decapsulation performance is comparable to that of ML-KEM [39], NEV [41], and DAWN [30].

2 Preliminaries

2.1 Notations

For a polynomial $\varphi \in \mathbb{Z}[x]$, let $\mathcal{R}_\varphi := \mathbb{Z}[x]/\langle\varphi\rangle, \mathcal{R}_{\varphi,q} := \mathcal{R}_\varphi/q\mathcal{R}_\varphi \cong \mathbb{Z}_q[x]/\langle\varphi\rangle$. Throughout the paper, elements of $\mathbb{Z}_q$ and $\mathcal{R}_{\varphi,q}$ are represented by their centered representatives. Thus absolute values and inequalities involving coefficients of elements modulo q always refer to these centered representatives. This convention avoids repeatedly writing the centered representative map. Lowercase letters denote ring elements, and bold lowercase letters denote coefficient vectors. For $a = \sum_i a_{[i]}x^i$, the notation $a_{[i]}$ denotes the coefficient of x^i. For a vector $\boldsymbol{a}$, the notation $\boldsymbol{a}_{[i]}$ denotes its i-th entry. We use $\lfloor\cdot\rceil$ for coefficient-wise rounding to the nearest integer, and write $[\![a]\!] := a - \lfloor a\rceil$ for the coefficient-wise fractional part. When x is invertible in $\mathcal{R}_\varphi$, we define the conjugate $\bar{a}$ of $a \in \mathcal{R}_\varphi$ by $\bar{a} := a(x^{-1})$. Let $n = \deg(\varphi)$. We identify each element of $\mathcal{R}_\varphi$ with its representative of degree $< n$ and define

$$\mathsf{poly2vec}(a) = (a_{[0]}, \ldots, a_{[n-1]}) \in \mathbb{Z}^n, \qquad \mathsf{vec2poly}(\boldsymbol{v}) = \sum_{i=0}^{n-1} \boldsymbol{v}_{[i]}x^i.$$

For a set $\mathcal{S}$ of polynomials and a quotient ideal $\langle p, \phi\rangle$, we write

$$\mathcal{S} \bmod \langle p, \phi\rangle := \{\, s \bmod \langle p, \phi\rangle : s \in \mathcal{S} \,\}$$

for its image under reduction, and its cardinality is denoted by $|\mathcal{S}|$. We write $a \xleftarrow{\$} \mathcal{D}$ for sampling a from a distribution $\mathcal{D}$, and $\mathcal{U}(\mathcal{S})$ for the uniform distribution on a finite set $\mathcal{S}$. We denote by $\mathcal{T}_{n,k}$ the ternary distribution over $\mathcal{R}_\varphi$ ($n := \deg(\varphi)$) with fixed Hamming weight: polynomials contain exactly k coefficients equal to 1, exactly k coefficients equal to -1, and the remaining $n - 2k$ coefficients are 0.

2.2 Embeddings

We use deterministic and randomized embeddings.

Definition 1 (Deterministic embedding). *A deterministic embedding from $\mathcal{S}_1$ to $\mathcal{S}_2$ is an injective map*

$$\mathsf{Emb}_{\mathcal{S}_1 \hookrightarrow \mathcal{S}_2} : \mathcal{S}_1 \to \mathcal{S}_2.$$

Definition 2 (Randomized embedding). *Let $\mathcal{S}_1$ be a message space, $\mathcal{S}_r$ a randomness space, and $\mathcal{S}_2$ a target space, and let $\varepsilon_{\mathrm{emb}} \geq 0$. A randomized embedding is a map*

$$\mathsf{Emb}^{\mathsf{rand}}_{\mathcal{S}_1 \Rightarrow \mathcal{S}_2} : \mathcal{S}_1 \times \mathcal{S}_r \to \mathcal{S}_2$$

such that the image sets

$$\mathcal{S}_m := \{\mathsf{Emb}^{\mathsf{rand}}_{\mathcal{S}_1 \Rightarrow \mathcal{S}_2}(m; r) : r \in \mathcal{S}_r\}$$

are pairwise disjoint, and the distribution

$$\mathsf{Emb}^{\mathsf{rand}}_{\mathcal{S}_1 \Rightarrow \mathcal{S}_2}(m; r), \qquad (m, r) \leftarrow \mathcal{U}(\mathcal{S}_1) \times \mathcal{U}(\mathcal{S}_r),$$

has statistical distance at most $\varepsilon_{\mathrm{emb}}$ from the target distribution used in the security analysis, usually $\mathcal{U}(\mathcal{S}_2)$.

The disjointness condition ensures that the embedded value determines the message if it lies in the valid image. The statistical-distance condition ensures that replacing the embedded message distribution by the target distribution changes the security experiment by at most $\varepsilon_{\mathrm{emb}}$.

2.3 Error Sets

For the quotient ring $\mathcal{R}_\varphi$, let $n := \deg(\varphi)$. For $\boldsymbol{z} \in \mathbb{Z}^n$, define the Hamming weight $\mathsf{wt}(\boldsymbol{z}) := \big|\{j : \boldsymbol{z}_{[j]} \neq 0\}\big|$. We define the signed-permutation group

$$\mathcal{H}_n := \{\mathrm{diag}(d_0, \ldots, d_{n-1}) \cdot P_\pi : d_j \in \{\pm 1\}, \pi \in \mathrm{S}_n\},$$

where S_n is the symmetric group on n elements. For $\boldsymbol{z} \in \mathbb{Z}^n$, define its orbit (as polynomials) by

$$\mathcal{E}_{\mathrm{orb}}(\boldsymbol{z}) := \big\{\mathsf{vec2poly}(H\boldsymbol{z}) : H \in \mathcal{H}_n\big\} \subseteq \mathcal{R}_\varphi. \tag{1}$$

Throughout the paper, B_0 bounds sparsity and B_1 bounds coefficient magnitude. Define

$$\mathcal{Z}_{B_0,B_1} := \{\boldsymbol{z} \in \mathbb{Z}^n : \mathsf{wt}(\boldsymbol{z}) \leq B_0,\ \|\boldsymbol{z}\|_\infty \leq B_1\},$$

and the corresponding error set in $\mathcal{R}_\varphi$ by

$$\mathcal{E}_{B_0,B_1} = \bigcup_{\boldsymbol{z} \in \mathcal{Z}_{B_0,B_1}} \mathcal{E}_{\mathrm{orb}}(\boldsymbol{z}). \tag{2}$$

This viewpoint is useful when the induced error distribution is (approximately) invariant under signed permutations, since then all elements in a fixed orbit $\mathcal{E}_{\mathrm{orb}}(\boldsymbol{z})$ occur with the same probability.

2.4 Coefficient Independence Heuristic

Heuristic 1 (Coefficient independence) *Let $a \xleftarrow{\$} \mathcal{D}_a$ and $b \xleftarrow{\$} \mathcal{D}_b$ be independently sampled elements of $\mathcal{R}_\varphi$. For the products used in the DFR estimates, we model the coefficients of $a \cdot b \in \mathcal{R}_\varphi$ by independent samples from the corresponding one-dimensional marginal distribution.*

This heuristic is the central heuristic in our DFR calculations because it reduces an n-dimensional tail estimate to a one-dimensional estimate, using binomial or union bounds. It is commonly used for average-case noise analysis (e.g., in BFV [6] and HElib [12]), but it is not a theorem. Ring structure can introduce coefficient dependencies, and dependency-aware estimates can either improve or worsen a concrete DFR estimate. For example, D'Anvers, Vercauteren, and Verbauwhede show that the independence assumption is usually suitable for schemes without error correction, but can underestimate the DFR when error-correcting codes are used [13]. Figure 1 of that work shows close agreement between the independence-model and experimental curves in the low-wrap-count regime (fewer than ten wraps). Also, Paiva *et al.* provided methods for analyzing compact lattice KEMs without relying on coefficient independence on the specific encoding in [33]. A more recent work [29] proposed a norm-wise model to analyze dependencies among coefficients. The settings that permit three or four wraps occur only in the exploratory frontier of Section 5; every concrete END correction stage has $B_0 \leq 2$, as shown in Table 2.

For END-512 and END-1024, we additionally performed dependency stress tests following dependency-amplifying patterns proposed in [20] We prescribe a consecutive-sign bad-secret-key pattern whose probability matches the claimed DFR, and a consecutive extremal bad-compression-error pattern whose offline-search cost matches the security target. In 100 trials for each parameter set, we observed no three-wrap cases, only a few two-wrap cases, and every wrap lay among the FCL candidates.[2]

3 Free Candidate Localization

Free Candidate Localization (FCL) is a candidate-generation method for centered lattice decryption residues. Given a residue modulo q, it identifies coordinates that are likely to have wrapped. FCL is not a standalone decoder or an error-correction mechanism by itself. It only produces a candidate family for the wrap-around error. Error correction is obtained only after FCL is paired with an independent algebraic, distributional, or hash verification rule that selects the correct candidate.

FCL localizes candidate wraps; correction still requires a scheme-specific verification rule. DAWN and BAT can be directly retrofitted by increasing compression, enumerating FCL candidates, and appending a hash tag, as illustrated by

[2] https://github.com/Icarid-Liu/Explore_NTRU_Decryption

FCL-ML-KEM in Appendix A. The gain is limited because the hash tag consumes ciphertext space, and their original structures were not co-designed for verification. Larger gains require jointly redesigning the structure, parameters, and verification rule. END performs this trapdoor-side co-design rather than being BAT plus FCL; NwE likewise redesigns the DAWN-style encoding, embedding, and verification.

HILA5's SafeBits [37] and FCL share the observation that distance to a public decision boundary contains reliability information. SafeBits selects coefficients close to reconciliation-region centers, hence far from decoding boundaries, and communicates a selection subset so that reliable bits can be extracted. FCL is instead receiver-side post-decryption localization: it uses the centered residual available during decryption, ranks coefficients close to the boundary as likely wrap locations, and outputs a bounded family of candidate error patterns. FCL does not decide which candidate is correct; separating localization from verification allows the same candidate-generation method to be paired with algebraic quotient verification, trapdoor-derived distributional verification, or hash verification, and to be instantiated across NTRU, RLWE, and MLWE.

3.1 Three-term model

Let q be the modulus and let $\boldsymbol{v} \in \mathbb{Z}_q^n$ be the centered vector obtained during decryption. In the applications considered in this paper, $\boldsymbol{v}$ is the centered reduction of an unreduced vector

$$\boldsymbol{u} = \boldsymbol{\eta} + \boldsymbol{\mu} \in \mathbb{Z}^n,$$

where $\boldsymbol{\eta}$ is the noise term and $\boldsymbol{\mu}$ is the message term. Define the wrap-around vector $\boldsymbol{e} \in \mathbb{Z}^n$ by

$$\boldsymbol{e} := \left\lfloor \frac{\boldsymbol{u}}{q} \right\rceil, \qquad \boldsymbol{v} = \boldsymbol{\eta} + \boldsymbol{\mu} - q \cdot \boldsymbol{e}. \tag{3}$$

A coordinate i is called wrapped if $\boldsymbol{e}_{[i]} \neq 0$. For the concrete parameter sets below, $B_1 = 1$, so the relevant bounded wrap vectors are sparse ternary; the general definition retains B_1 to state the framework uniformly. For each coordinate, define the boundary-distance score

$$d_i := \lfloor q/2 \rfloor - |\boldsymbol{v}_{[i]}|.$$

Small d_i means that the centered residue lies close to the boundary. FCL is useful when the unwrapped coefficients are roughly symmetric and have limited mass near the boundary. These conditions hold for most lattice-based encryptions. If the unwrapped distribution has a nonzero mean, the score should be applied after recentering by the mean.

3.2 FCL candidate generation

Let $\mathcal{I} \subseteq \{0, \ldots, n-1\}$ be the set of indices that may contain a wrap. Usually $\mathcal{I} = \{0, \ldots, n-1\}$, but side information may reduce it. Given an integer $m_0 \leq |\mathcal{I}|$, FCL returns the m_0 indices in $\mathcal{I}$ with the smallest boundary-distance scores.

Algorithm 1 Free Candidate Localization (FCL)

Input: centered residue $\boldsymbol{v} \in \mathbb{Z}_q^n$, modulus q, index set $\mathcal{I}$, output size $m_0 < |\mathcal{I}|$
Output: index set $\mathcal{S}_{\mathrm{FCL}} \subseteq \mathcal{I}$ of size m_0
1: compute $d_i \leftarrow \lfloor q/2 \rfloor - |\boldsymbol{v}_{[i]}|$ for all $i \in \mathcal{I}$
2: order $\mathcal{I}$ as $i_0, \ldots, i_{|\mathcal{I}|-1}$ so that $d_{i_0} \le \cdots \le d_{i_{|\mathcal{I}|-1}}$
3: **return** $\{i_0, \ldots, i_{m_0-1}\}$

Given bounds B_0 and B_1, the corresponding candidate error family is

$$\mathcal{E}_{\mathrm{cand}}(\boldsymbol{v}, q, m_0, B_0, B_1, \mathcal{I}) = \left\{ \sum_{i \in I} \sigma_i x^i : I \subseteq \mathcal{S}_{\mathrm{FCL}},\ |I| \le B_0,\ 1 \le |\sigma_i| \le B_1 \right\}.$$

The zero error is included by taking $I = \varnothing$. Its worst-case size is

$$1 + \sum_{j=1}^{B_0} \binom{m_0}{j} (2B_1)^j.$$

All parameters in this expression are public. Hence, an implementation can enumerate the full public bound and avoid secret-dependent loop lengths.

3.3 Failure-probability estimate

Assume that the true wrap support W has size m_1, with $m_1 \le B_0$, and that $W \subseteq \mathcal{I}$. Let $N = |\mathcal{I}|$. Fix a wrapped index $j \in W$ and condition on the unreduced coefficient $\boldsymbol{u}_{[j]} = y$. Let X denote the absolute value of an unwrapped coefficient, conditioned on being unwrapped. Define

$$p_{|u|,q}(y) := \Pr\big[X \ge |y \bmod q| \ \big|\ X \le \lfloor q/2 \rfloor \big].$$

Equivalently, if $F_{|u|}$ is the CDF of $|\boldsymbol{u}_{[i]}|$ for an unwrapped index, then

$$p_{|u|,q}(y) = \frac{F_{|u|}(\lfloor q/2 \rfloor) - F_{|u|}(|y \bmod q| - 1)}{F_{|u|}(\lfloor q/2 \rfloor)}.$$

If the wrapped coordinate j is excluded from the m_0 FCL positions, then at least $m_0 - m_1 + 1$ unwrapped coordinates must be ranked no worse than j. The fixed tie-breaking rule can only make this condition more conservative. Under the coefficient-independence heuristic, the number of such unwrapped coordinates is dominated by a binomial random variable with $N - m_1$ trials and success probability $p_{|u|,q}(y)$. Therefore

$$\begin{aligned} &\Pr\big[j \notin \mathcal{S}_{\mathrm{FCL}} \mid \boldsymbol{u}_{[j]} = y\big] \\ &\le \sum_{k=m_0-m_1+1}^{N-m_1} \binom{N-m_1}{k} p_{|u|,q}(y)^k \big(1 - p_{|u|,q}(y)\big)^{N-m_1-k}. \end{aligned} \tag{4}$$

Averaging over the conditional distribution of y among wrapped coefficients gives

$$\begin{aligned}&\Pr[j \notin \mathcal{S}_{\mathrm{FCL}}]\\ &\leq \sum_{|y|\geq \lfloor q/2\rfloor+1} \Pr\left[j \notin \mathcal{S}_{\mathrm{FCL}} \mid \boldsymbol{u}_{[j]} = y\right] \Pr\left[\boldsymbol{u}_{[j]} = y \mid |\boldsymbol{u}_{[j]}| \geq \lfloor q/2\rfloor + 1\right]. \end{aligned} \tag{5}$$

By a union bound over the m_1 wrapped coordinates,

$$\Pr[\text{FCL fails}] \leq m_1 \cdot \Pr[j \notin \mathcal{S}_{\mathrm{FCL}}]. \tag{6}$$

The inequality in (4) is intentional: it ignores additional constraints involving the other wrapped coordinates and treats ties conservatively.

3.4 Verification interface

FCL only gives a list, and the decoder must verify candidates. Let $\mathcal{C}$ denote the public context used by the verification procedure, such as quotient-ring residues, public distributions, public thresholds, or precomputed tables. For a candidate error $\tilde{e}$, the decoder forms a corrected value and derives a candidate object $\widehat{X}(\tilde{e})$. The verification rule is a predicate

$$\mathsf{Verify}(\mathcal{C}, \widehat{X}) \in \{0, 1\}.$$

It may be a deterministic algebraic validity test or a distributional test.

Let $\tilde{e}^\star$ be the true error. We say that Verify has false-reject and false-accept parameters $(\varepsilon_{\mathrm{fr}}, \varepsilon_{\mathrm{fa}})$ if

$$\Pr[\mathsf{Verify}(\mathcal{C}, \widehat{X}(\tilde{e}^\star)) = 0] \leq \varepsilon_{\mathrm{fr}},$$

and

$$\max_{\tilde{e} \neq \tilde{e}^\star} \Pr[\mathsf{Verify}(\mathcal{C}, \widehat{X}(\tilde{e})) = 1] \leq \varepsilon_{\mathrm{fa}},$$

where the maximum is over the finite candidate family used by the decoder. If the decoder enumerates a family $\mathcal{E}$, outputs the first accepted candidate, and outputs $\perp$ if no candidate is accepted, then

$$\mathrm{DFR} \leq \Pr[\text{FCL fails}] + \varepsilon_{\mathrm{fr}} + |\mathcal{E}|\varepsilon_{\mathrm{fa}}. \tag{7}$$

This bound is conservative and sufficient for the parameter searches in Section 5.

3.5 Use in Ring-LWE and Module-LWE decryption

FCL is not specific to NTRU. In Ring/Module-LWE encryption, decryption computes a centered noisy message residue and then decodes the coefficients based on their distance from a decision boundary. If a failure is caused by a small number of coefficients crossing such a boundary, the same boundary-distance ranking can localize likely failing coordinates before applying redundancy, reconciliation, or an error-correcting layer. This gives a generic way to turn a coefficient-wise failure pattern into a small candidate family, provided an independent verification rule is available. We provide an instantiation of ML-KEM with FCL in Appendix A, yielding a ciphertext that is 10% smaller than ML-KEM's at about 5% cost under the independence heuristic in NIST-I security.

4 NTRU Frameworks

This section lifts the vector view of Section 3 back to NTRU rings. We freely convert between ring elements and coefficient vectors using poly2vec and vec2poly. Our motivation is that even when a scheme exhibits occasional decryption failures, these failures are often non-catastrophic, and only a small number of coefficients wrap. This motivates the addition of a limited second-stage decoding procedure that searches over a small candidate error set (e.g., produced by FCL) and uses independent verification to uniquely select the valid message. We present two generic decryption frameworks that capture most NTRU-based public-key encryptions: *NTRU with Encoding*, and *NTRU with Trapdoor*. The former exploits algebraic structure; auxiliary quotient rings provide projections that narrow the error-candidate space, while the latter adopts a geometric viewpoint and solves a structured closest vector problem (CVP) using a trapdoor basis. The common decryption template is

$$\mathsf{Ciphertext} \longrightarrow \mathsf{CandidateErrorSet} \longrightarrow \mathsf{Verify} \longrightarrow \mathsf{RecoverMessage}.$$

The frameworks differ in how they obtain side information for the second and third steps. NTRU with Encoding uses auxiliary quotient rings and message redundancy. NTRU with Trapdoor uses FCL and distributional checks. For simplicity, we state all frameworks under the one-way chosen plaintext security (OW-CPA) assumption; standard transforms (e.g., Fujisaki–Okamoto) are orthogonal to our discussion. We assume all instantiations target λ-bit security, and we separately target a DFR bound $2^{-\lambda_{\mathrm{DFR}}}$, as typically required to make CCA transforms conservative in practice.

4.1 NTRU with Encoding

NTRU with Encoding uses public quotient rings to separate the noise, message, and wrap-around terms. Ciphertexts live in $\mathcal{R}_{\varphi,q}$. Fix quotient rings

$$\mathcal{R}_{\phi_i,p_i} := \mathbb{Z}_{p_i}[x]/\langle \phi_i \rangle, \qquad 1 \le i \le \ell,$$

where each p_i is coprime to q, and each ϕ_i divides φ in $\mathbb{Z}_{p_i}[x]$. The first quotient ring is the decoding ring, and the rings with $i \ge 2$ are auxiliary rings. Let

$$\pi_i : \mathcal{R}_{\varphi} \to \mathcal{R}_{\phi_i,p_i}$$

be the natural reduction map. The framework has public polynomials $t, w \in \mathbb{Z}[x]$ satisfying

$$\begin{aligned} t &\equiv 0 \pmod{\langle p_i, \phi_i \rangle} \quad \text{for } 1 \le i \le \ell, \\ w &\equiv 0 \pmod{\langle p_i, \phi_i \rangle} \quad \text{for } 2 \le i \le \ell, \qquad w \not\equiv 0 \pmod{\langle p_1, \phi_1 \rangle}. \end{aligned}$$

We also require w to be invertible in $\mathcal{R}_{\varphi,q}$. These are public parameters.

Lemma 1 (Encoding residue decomposition). *Let (g, f, s, μ) be sampled as in Figure 1, let $f' = tw^{-1}f + 1$, and let $c = tw^{-1}hs + \mu \pmod{\langle q, \varphi\rangle}$. If c'_0 is the centered representative of $wf'c$, then there is a unique coefficient vector $e \in \mathbb{Z}^n$ such that*

$$c'_0 = t(gs + f\mu) + w\mu - q \cdot e \quad \textit{in} \ \ \mathcal{R}_\varphi. \tag{8}$$

Moreover,

$$\pi_1(c'_0) = \pi_1(w\mu - q \cdot e), \qquad \pi_i(c'_0) = \pi_i(-q \cdot e) \quad (2 \le i \le \ell).$$

Proof. The congruence $f'h \equiv g \pmod{\langle q, \varphi\rangle}$ gives

$$wf'c \equiv tgs + wf'\mu \equiv tgs + tf\mu + w\mu \pmod{\langle q, \varphi\rangle}.$$

Centering subtracts $q \cdot e$ coefficient-wise. The reductions follow from the defining congruences for t and w.

The effective message ring is the largest public quotient of the decoding ring on which multiplication by w is invertible. Let $\overline{w}$ be the reduction of w in $\mathbb{Z}_{p_1}[x]$. Choose a public divisor $\phi_{\mathcal{M}} \mid \phi_1$ such that $\gcd(\overline{w}, \phi_{\mathcal{M}}) = 1$, and set

$$\mathcal{R}_{\mathcal{M}} := \mathbb{Z}_{p_1}[x]/\langle \phi_{\mathcal{M}} \rangle.$$

When ϕ_1 is square-free, this is equivalently obtained by removing the common factor $\gcd(\overline{w}, \phi_1)$; in non-square-free cases, all primary factors shared with $\overline{w}$ must be removed. Let $\pi_{\mathcal{M}}$ be the projection from the decoding ring to $\mathcal{R}_{\mathcal{M}}$. The framework uses the map

$$\psi(m; r) := \pi_{\mathcal{M}}\Big(\mathsf{Emb}^{\mathsf{rand}}_{\mathcal{M} \Rightarrow \mathcal{S}_\mu}(m; r) \bmod \langle p_1, \phi_1 \rangle\Big),$$

and requires a left inverse $\mathsf{Unemb}_{\mathcal{M}' \to \mathcal{M}}$ on the image

$$\mathcal{M}' := \{\psi(m; r) : m \in \mathcal{M}, \ r \in \mathcal{S}_r\}$$

such that $\mathsf{Unemb}_{\mathcal{M}' \to \mathcal{M}}(\psi(m; r)) = m$ for all valid (m, r). The map ψ need not itself be a randomized embedding in the sense of Section 2; only this left-inverse property is required for recovery.

4.1.1 Auxiliary residues and FCL. Let $\mathcal{E}_{\mathrm{err}} := \mathcal{E}_{B_0, B_1}$. For $2 \le i \le \ell$, Lemma 1 gives $c'_i = \pi_i(-q \cdot e)$. Define

$$\mathsf{err2res} : \mathcal{E}_{\mathrm{err}} \to \prod_{i=2}^{\ell} \mathcal{R}_{\phi_i, p_i}, \qquad \mathsf{err2res}(\tilde{e}) := \big(\pi_i(-q \cdot \tilde{e})\big)_{2 \le i \le \ell},$$

and $\mathcal{C}_{\mathrm{err}} := \mathsf{err2res}(\mathcal{E}_{\mathrm{err}})$. For $\boldsymbol{c} \in \mathcal{C}_{\mathrm{err}}$, set

$$\mathsf{res2err}(\boldsymbol{c}) := \{\tilde{e} \in \mathcal{E}_{\mathrm{err}} : \mathsf{err2res}(\tilde{e}) = \boldsymbol{c}\},$$
$$\mathcal{E}_{\mathrm{res}}(\boldsymbol{c}) := \{\pi_{\mathcal{M}}(-q \cdot \tilde{e}) : \tilde{e} \in \mathsf{res2err}(\boldsymbol{c})\}.$$

Let

$$C_0(\boldsymbol{c}) := \max_{\tilde{e}\in \mathsf{res2err}(\boldsymbol{c})} \mathsf{wt}(\mathsf{poly2vec}(\tilde{e})), \quad C_1(\boldsymbol{c}) := \max_{\tilde{e}\in \mathsf{res2err}(\boldsymbol{c})} \|\mathsf{poly2vec}(\tilde{e})\|_\infty,$$

and

$$\mathcal{I}(\boldsymbol{c}) := \bigcup_{\tilde{e}\in \mathsf{res2err}(\boldsymbol{c})} \{j : \tilde{e}_{[j]} \neq 0\}.$$

For a public FCL output size $m_0(\boldsymbol{c})$, define the FCL-induced shift set in the effective message ring by

$$\mathcal{E}^{\lambda_{\mathrm{DFR}}}_{\mathrm{cand,res}}(\boldsymbol{v}, \boldsymbol{c}) := \{\pi_{\mathcal{M}}(-q \cdot \tilde{e}) : \tilde{e} \in \mathcal{E}_{\mathrm{cand}}(\boldsymbol{v}, q, m_0(\boldsymbol{c}), C_0(\boldsymbol{c}), C_1(\boldsymbol{c}), \mathcal{I}(\boldsymbol{c}))\}.$$

The effective list size after both filters is

$$L_{\mathrm{Enc}}(\boldsymbol{v}, \boldsymbol{c}) := \left|\mathcal{E}_{\mathrm{res}}(\boldsymbol{c}) \cap \mathcal{E}^{\lambda_{\mathrm{DFR}}}_{\mathrm{cand,res}}(\boldsymbol{v}, \boldsymbol{c})\right|. \tag{9}$$

We then present the generic framework of *NTRU with Encoding* in Figure 1.

Fig. 1: **Generic NTRU with Encoding**

KeyGen:

1 : $g \xleftarrow{\$} \mathcal{D}_g$
2 : $f \xleftarrow{\$} \mathcal{D}_f$
3 : $f' \leftarrow tw^{-1}f + 1 \pmod{\langle q, \varphi\rangle}$
4 : **if** f' is not invertible in $\mathcal{R}_{\varphi,q}$ restart
5 : $h \leftarrow g(f')^{-1} \pmod{\langle q, \varphi\rangle}$
6 : **return** $(pk := h,\ sk := f')$

Encrypt$(pk := h, m)$:

1 : $s \xleftarrow{\$} \mathcal{D}_s$
2 : $\mu \leftarrow \mathsf{Emb}^{\mathsf{rand}}_{\mathcal{M}\Rightarrow\mathcal{S}_\mu}(m)$
3 : $c \leftarrow tw^{-1}hs + \mu \pmod{\langle q, \varphi\rangle}$
4 : **return** c

Decrypt$(sk := f', c)$:

1 : $c'_0 \leftarrow wf'c \pmod{\langle q, \varphi\rangle}$
2 : $m \leftarrow \mathsf{Decode}_{\mathrm{Enc}}(c'_0, t, w, \{(p_i, \phi_i)\}_{1\le i\le \ell})$
3 : **return** m

Lemma 2 (Encoding correctness after candidate containment). *Fix a decryption instance with true message representative $\psi(m; r)$, true wrap error $e \in \mathcal{E}_{\mathrm{err}}$, residue $\boldsymbol{c} = \mathsf{err2res}(e)$, and centered vector $\boldsymbol{v} = \mathsf{poly2vec}(c'_0)$. Assume that e is contained in $\mathcal{E}_{\mathrm{aux}} \cap \mathcal{E}_{\mathrm{FCL}}$. If, for every wrong candidate $\tilde{e} \in \mathcal{E}_{\mathrm{aux}} \cap \mathcal{E}_{\mathrm{FCL}}$ with $\tilde{e} \neq e$,*

$$\psi(m; r) + \overline{w}^{-1}\pi_{\mathcal{M}}\big(q \cdot (\tilde{e} - e)\big) \notin \mathcal{M}', \tag{10}$$

then Algorithm 2 returns m.

Algorithm 2 $\mathsf{Decode}_{\mathrm{Enc}}$

Input: centered residue c_0', public $t, w, \{(p_i, \phi_i)\}_{1 \le i \le \ell}$, FCL parameters (m_0, B_0, B_1)
Output: message m

for $i = 1$ to ℓ **do**
  $c_i' \leftarrow c_0' \bmod \langle p_i, \phi_i \rangle$
end for
$\mathcal{E}_{\mathrm{err}} \leftarrow \mathcal{E}_{B_0,B_1}$
$\mathcal{E}_{\mathrm{aux}} \leftarrow \{\tilde{e} \in \mathcal{E}_{\mathrm{err}} : -q \cdot \tilde{e} \equiv c_i' \pmod{\langle p_i, \phi_i \rangle} \text{ for all } 2 \le i \le \ell\}$
$\mathcal{I} \leftarrow \{j : \tilde{e}_{[j]} \neq 0 \text{ for some } \tilde{e} \in \mathcal{E}_{\mathrm{aux}}\}$
$\mathcal{E}_{\mathrm{FCL}} \leftarrow \mathcal{E}_{\mathrm{cand}}(\mathsf{poly2vec}(c_0'), q, m_0, B_0, B_1, \mathcal{I})$
for $\tilde{e} \in \mathcal{E}_{\mathrm{aux}} \cap \mathcal{E}_{\mathrm{FCL}}$ **do**
  $\tilde{\mu}' \leftarrow \overline{w}^{-1} \pi_{\mathcal{M}}(c_1' + q \cdot \tilde{e})$
  if $\tilde{\mu}' \in \mathcal{M}'$ **then**
    $m \leftarrow \tilde{\mu}'$
  end if
end for
return m

Proof. For a candidate $\tilde{e}$, Lemma 1 gives

$$\overline{w}^{-1} \pi_{\mathcal{M}}(c_1' + q \cdot \tilde{e}) = \psi(m; r) + \overline{w}^{-1} \pi_{\mathcal{M}}\big(q \cdot (\tilde{e} - e)\big).$$

The true candidate $\tilde{e} = e$ gives $\psi(m; r) \in \mathcal{M}'$ and unembeds to m. Condition (10) prevents every wrong candidate from being accepted. The algorithm, therefore, collects exactly the message m.

Candidate containment and candidate uniqueness are separate requirements. Lemma 1 and the auxiliary identities place the true error e in $\mathcal{E}_{\mathrm{aux}}$, while a successful FCL event places e in $\mathcal{E}_{\mathrm{FCL}}$. Once $e \in \mathcal{E}_{\mathrm{aux}} \cap \mathcal{E}_{\mathrm{FCL}}$, Condition (10) is the uniqueness requirement: every nonzero candidate shift must map no valid projected message to another valid projected message. NwE checks this condition concretely in Appendix C: $\Delta_{\mathrm{ver}} > 2K_{\mathrm{ver}}$, so the false-accept probability is zero over its checked candidate family.

Corollary 1 (Packing screen for algebraic verification). *Let $\mathcal{V}$ be the distributional support of $\mathsf{poly2vec}(c_0')$. A necessary packing condition for a disjoint-shift verification rule with worst-case list size L_{Enc} is*

$$|\mathcal{M}'| \cdot \max_{\boldsymbol{c} \in \mathcal{C}_{\mathrm{err}},\, \boldsymbol{v} \in \mathcal{V}} L_{\mathrm{Enc}}(\boldsymbol{v}, \boldsymbol{c}) \le |\mathcal{R}_{\mathcal{M}}|. \tag{11}$$

When the concrete embedding and the concrete choice of w are checked to satisfy the disjoint-shift condition (10) for all candidates counted by L_{Enc}, this packing screen becomes a sufficient correctness check. Without that concrete disjointness check, Equation (11) should be read only as a feasibility screen, not as a proof of unique decoding.

4.1.2 Heuristic average-case estimate. The previous lemma is algebraic. To estimate average verification failure in the search, we additionally use the signed-permutation and placement heuristics. Assume the induced error distribution is invariant under signed permutations, so all errors in a fixed orbit $\mathcal{E}_{\mathrm{orb}}(\boldsymbol{z})$ of Equation (1) have equal probability. Define

$$\omega(\boldsymbol{v}, \boldsymbol{c}) := \max\left(\frac{|\mathcal{M}'| \cdot L_{\mathrm{Enc}}(\boldsymbol{v}, \boldsymbol{c})}{|\mathcal{R}_{\mathcal{M}}|} - 1, 0\right).$$

Under the heuristic that the valid projected message set behaves like a random subset of $\mathcal{R}_{\mathcal{M}}$ with respect to the candidate shifts, the average decoding-failure contribution from falsely accepted candidates is estimated by

$$\Pr_{\text{encoding}}[\text{Dec. fails}] \leq \sum_{\boldsymbol{z} \in \mathcal{Z}^{\mathrm{rep}}_{B_0, B_1}} \Pr[e \in \mathcal{E}_{\mathrm{orb}}(\boldsymbol{z})] \cdot \mathbb{E}_{\boldsymbol{v}, e}[\omega(\boldsymbol{v}, \mathsf{err2res}(e)) \mid e \in \mathcal{E}_{\mathrm{orb}}(\boldsymbol{z})], \tag{12}$$

where $\mathcal{Z}^{\mathrm{rep}}_{B_0,B_1}$ contains one canonical representative per absolute-value multi-set, for example sorted nonnegative vectors

$$\mathcal{Z}^{\mathrm{rep}}_{B_0,B_1} = \{\boldsymbol{z} \in \mathbb{Z}^n_{\geq 0} \mid B_1 \geq \boldsymbol{z}_{[0]} \geq \boldsymbol{z}_{[1]} \geq \cdots \geq \boldsymbol{z}_{[n-1]} \geq 0, \mathrm{wt}(\boldsymbol{z}) \leq B_0\}$$

This estimate is not used as a theorem; it is the heuristic verification term used only by the exploratory frontier search when an exact disjointness check is not performed. NwE does not use this random-placement model: Appendix C checks the concrete distance condition $\Delta_{\mathrm{ver}} > 2K_{\mathrm{ver}}$.

4.1.3 Design requirements. An instantiation of NTRU with Encoding should satisfy

$$\Pr[e \notin \mathcal{E}_{\mathrm{err}}] + \Pr[\text{FCL fails} \mid e \in \mathcal{E}_{\mathrm{err}}] + \Pr_{\text{encoding}}[\text{Dec. fails}] \leq 2^{-\lambda_{\mathrm{DFR}}}.$$

For efficiency, the concrete parameter set either enforces the worst-case bound

$$\max_{\boldsymbol{c} \in \mathcal{C}_{\mathrm{err}},\, \boldsymbol{v} \in \mathcal{V}} L_{\mathrm{Enc}}(\boldsymbol{v}, \boldsymbol{c}) \leq B,$$

or the high-probability variant

$$\Pr\big[L_{\mathrm{Enc}}(\boldsymbol{V}, \mathsf{err2res}(e)) > B\big] \leq 2^{-\lambda_{\mathrm{DFR}}}.$$

We note that the NTRU with Encoding framework can be extended in a straightforward way to RLWE-style encryption. Since the resulting compactness advantage is less significant than in the NTRU setting, we present this extension only in Appendix B, together with an FCL-based instantiation. We show the visualized workflow of *NTRU with Encoding* in Figure 2.

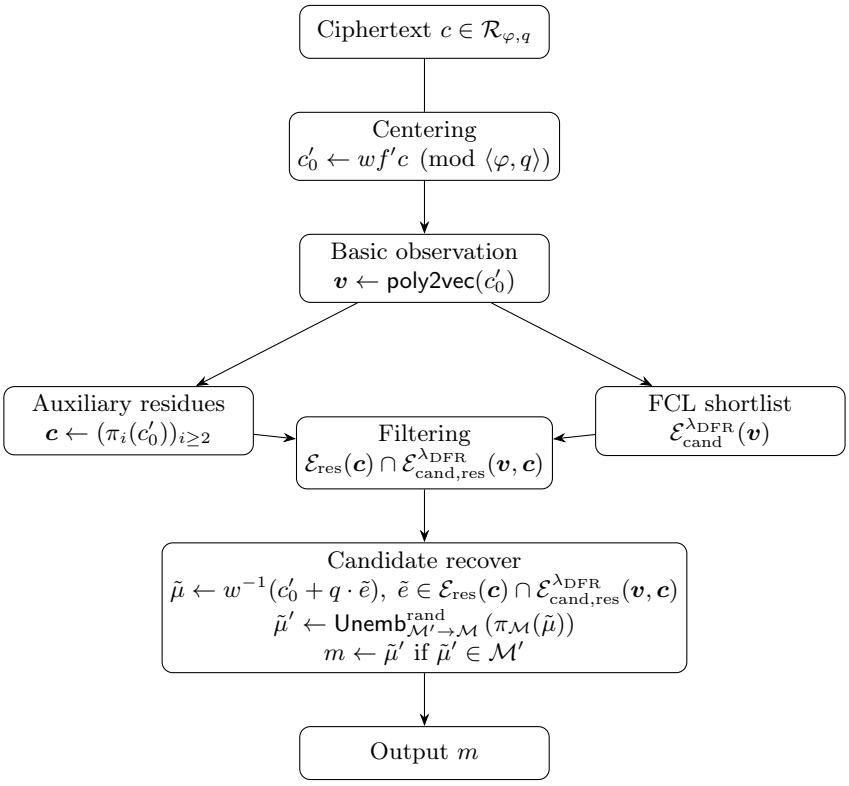

Fig. 2: *NTRU with Encoding* decryption workflow (auxiliary rings + FCL).

4.2 NTRU with Trapdoor

NTRU with Trapdoor removes the need for algebraic message encoding. Given $h \in \mathcal{R}_{\varphi,q}$, define

$$\mathcal{L}_{\mathrm{NTRU}} := \{(u, v) \in \mathcal{R}_\varphi^2 : u \equiv hv \pmod{\langle q, \varphi \rangle}\}.$$

The public basis and the private trapdoor basis are

$$\mathbf{B}_{h,q} = \begin{pmatrix} h & 1 \\ q & 0 \end{pmatrix}, \qquad \mathbf{B}_{g,f} = \begin{pmatrix} g & f \\ G & F \end{pmatrix},$$

where $h \equiv gf^{-1} \pmod{\langle q, \varphi \rangle}$ and $gF - fG \equiv q \pmod{\langle \varphi \rangle}$. For a ciphertext $c \equiv hs + \mu$, the vector $(c, 0)$ decomposes as a lattice vector plus the hidden offset $(\mu, -s)$. Thus, decryption is a structured CVP problem, not an ordinary BDD problem with a generic target. The target always lies in $\mathcal{R}_{\varphi,q} \times \{0\}$, so correctness depends on the chosen trapdoor decoding procedure, as shown in Figure 3.

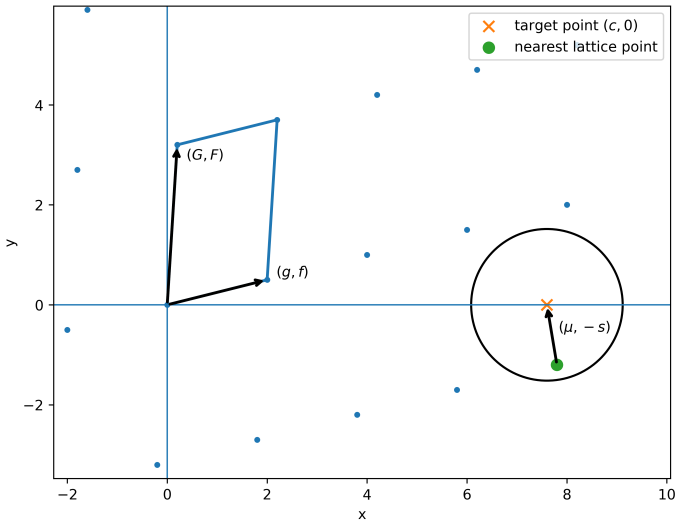

Fig. 3: Structured CVP view of NTRU decryption.

4.2.1 BNP rounding identity. We instantiate solveCVP with Babai's nearest plane (BNP). Let

$$\mathbf{B}^*_{g,f} = \begin{pmatrix} g & f \\ G^* & F^* \end{pmatrix}, \qquad (G^*, F^*) := (G, F) - \alpha_{\mathrm{gf}} \cdot (g, f), \qquad \alpha_{\mathrm{gf}} := \frac{G\bar{g} + F\bar{f}}{g\bar{g} + f\bar{f}}.$$

Define

$$e_{\mathrm{gf}} := \frac{gs + f\mu}{q}, \qquad e_{\mathrm{GF}} := \frac{G^*s + F^*\mu}{q}.$$

The centered residues used by FCL are

$$c_{\mathrm{gf}} = gs + f\mu - q \cdot \lfloor e_{\mathrm{gf}} \rceil,$$
$$c_{\mathrm{GF}} = G^*s + F^*\mu - q \cdot \lfloor e_{\mathrm{GF}} \rceil.$$

For BNP, a sufficient zero-rounding condition is

$$\lfloor e_{\mathrm{gf}} \rceil = 0 \quad \text{and} \quad \lfloor e_{\mathrm{GF}} \rceil = 0,$$

equivalently $\|gs + f\mu\|_\infty < \frac{q}{2}$ and $\|G^*s + F^*\mu\|_\infty < \frac{q}{2}$. By defining

$$e_{\mathrm{mid}} := \alpha_{\mathrm{gf}} \lfloor e_{\mathrm{gf}} \rceil, \qquad e_{\mathrm{GF,mid}} := \lfloor e_{\mathrm{GF}} \rceil + \lfloor [\![e_{\mathrm{GF}}]\!] + [\![e_{\mathrm{mid}}]\!] \rceil$$

FCL extends this condition by searching bounded sets

$$\lfloor e_{\mathrm{gf}} \rceil \in \mathcal{E}_{\mathrm{err,gf}} := \mathcal{E}_{B_{\mathrm{gf},0}, B_{\mathrm{gf},1}}, \qquad e_{\mathrm{GF,mid}} \in \mathcal{E}_{\mathrm{err,GF}} := \mathcal{E}_{B_{\mathrm{GF},0}, B_{\mathrm{GF},1}}.$$

The exact BNP output contains a rounding-interaction term. Then we have,

$$(\mu', -s') = (\mu, -s) + \lfloor e_{\mathrm{gf}} \rceil \cdot (G, F) - (e_{\mathrm{GF,mid}} + \lfloor e_{\mathrm{mid}} \rceil) \cdot (g, f). \tag{13}$$

We present the generic framework of *NTRU with Trapdoor* in Figure 4.

Fig. 4: **Generic NTRU with Trapdoor**

KeyGen:	Decrypt($sk := \mathbf{B}_{g,f}, c$):
$g \xleftarrow{\$} \mathcal{D}_g, \quad f \xleftarrow{\$} \mathcal{D}_f$	$(\mu', -s') \leftarrow \mathsf{solveCVP}(\mathbf{B}_{g,f}, (c, 0))$
$h \leftarrow gf^{-1} \pmod{\langle q, \varphi \rangle}$	$c_{\mathrm{gf}} \leftarrow fc \pmod{\langle q, \varphi \rangle}$
$(G, F) \leftarrow \mathsf{TrapdoorGen}(g, f, q)$	$c_{\mathrm{GF}} \leftarrow F^* c \pmod{\langle q, \varphi \rangle}$
return $(pk := h,\ sk := \mathbf{B}_{g,f})$	$(\mu, -s) \leftarrow \mathsf{Decode}_{\mathrm{Trap}}(\mathbf{B}_{g,f}, (\mu', -s'), c_{\mathrm{gf}}, c_{\mathrm{GF}})$
	$m \leftarrow \mathsf{Unemb}^{\mathsf{rand}}_{\mathcal{S}_\mu \to \mathcal{M}}(\mu)$
Encrypt($pk := h, m$):	**return** m
$s \xleftarrow{\$} \mathcal{D}_s$	
$\mu \leftarrow \mathsf{Emb}^{\mathsf{rand}}_{\mathcal{M} \Rightarrow \mathcal{S}_\mu}(m)$	
$c \leftarrow hs + \mu \pmod{\langle q, \varphi \rangle}$	
return c	

Algorithm 3 $\mathsf{Decode}_{\mathrm{Trap}}$

Input: trapdoor basis $\mathbf{B}_{g,f} = \begin{pmatrix} g & f \\ G & F \end{pmatrix}$, initial CVP output $(\mu', -s')$, centered residues $c_{\mathrm{gf}}, c_{\mathrm{GF}}$, modulus q, FCL parameters $(m_{\mathrm{gf}}, B_{\mathrm{gf},0}, B_{\mathrm{gf},1})$ and $(m_{\mathrm{GF}}, B_{\mathrm{GF},0}, B_{\mathrm{GF},1})$, public score functions $\mathsf{CheckDist}_{\mathrm{gf}}$ and $\mathsf{CheckDist}_{\mathrm{GF}}$

Output: corrected vector $(\mu, -s)$

$\alpha_{\mathrm{gf}} \leftarrow \frac{G\bar{g} + F\bar{f}}{g\bar{g} + f\bar{f}}$
$\mathcal{I}_n \leftarrow \{0, \ldots, n-1\}$
$\mathcal{E}_{\mathrm{gf}} \leftarrow \mathcal{E}_{\mathrm{cand}}(\mathsf{poly2vec}(c_{\mathrm{gf}}), q, m_{\mathrm{gf}}, B_{\mathrm{gf},0}, B_{\mathrm{gf},1}, \mathcal{I}_n)$
$\tilde{e}^\star_{\mathrm{gf}} \leftarrow \arg\min_{\tilde{e}_{\mathrm{gf}} \in \mathcal{E}_{\mathrm{gf}}} \mathsf{CheckDist}_{\mathrm{gf}}((\mu', -s') - \tilde{e}_{\mathrm{gf}} \cdot (G, F))$
$(\mu'', -s'') \leftarrow (\mu', -s') - \tilde{e}^\star_{\mathrm{gf}} \cdot (G, F)$
$\mathcal{E}_{\mathrm{GF}} \leftarrow \mathcal{E}_{\mathrm{cand}}(\mathsf{poly2vec}(c_{\mathrm{GF}}), q, m_{\mathrm{GF}}, B_{\mathrm{GF},0}, B_{\mathrm{GF},1}, \mathcal{I}_n)$
$\tilde{e}^\star_{\mathrm{GF}} \leftarrow \arg\min_{\tilde{e}_{\mathrm{GF}} \in \mathcal{E}_{\mathrm{GF}}} \mathsf{CheckDist}_{\mathrm{GF}}((\mu'', -s'') + \lfloor \tilde{e}_{\mathrm{GF}} + \alpha_{\mathrm{gf}} \tilde{e}^\star_{\mathrm{gf}} \rceil \cdot (g, f))$
$(\mu, -s) \leftarrow (\mu'', -s'') + \lfloor \tilde{e}^\star_{\mathrm{GF}} + \alpha_{\mathrm{gf}} \tilde{e}^\star_{\mathrm{gf}} \rceil \cdot (g, f)$
return $(\mu, -s)$

4.2.2 FCL and distributional verification. Let $\boldsymbol{V}_{\mathrm{gf}} = \mathsf{poly2vec}(c_{\mathrm{gf}})$ and $\boldsymbol{V}_{\mathrm{GF}} = \mathsf{poly2vec}(c_{\mathrm{GF}})$. FCL gives

$$\mathcal{E}_{\mathrm{cand,gf}}(\boldsymbol{V}_{\mathrm{gf}}) := \mathcal{E}_{\mathrm{cand}}(\boldsymbol{V}_{\mathrm{gf}}, q, m_{\mathrm{gf}}, B_{\mathrm{gf},0}, B_{\mathrm{gf},1}, \mathcal{I}_n),$$
$$\mathcal{E}_{\mathrm{cand,GF}}(\boldsymbol{V}_{\mathrm{GF}}) := \mathcal{E}_{\mathrm{cand}}(\boldsymbol{V}_{\mathrm{GF}}, q, m_{\mathrm{GF}}, B_{\mathrm{GF},0}, B_{\mathrm{GF},1}, \mathcal{I}_n).$$

The score functions $\mathsf{CheckDist}_{\mathrm{gf}}$ and $\mathsf{CheckDist}_{\mathrm{GF}}$ are public deterministic functions. They are not part of FCL. They estimate whether a corrected pair has

the expected message and secret distribution. The stage-failure probabilities are

$$\begin{aligned}
\varepsilon_{\mathrm{gf}} &:= \Pr\left[\tilde{e}^{\star}_{\mathrm{gf}} \neq \lfloor e_{\mathrm{gf}} \rceil \mid \lfloor e_{\mathrm{gf}} \rceil \in \mathcal{E}_{\mathrm{err,gf}}\right], \\
\varepsilon_{\mathrm{GF}} &:= \Pr[\tilde{e}^{\star}_{\mathrm{GF}} \neq e_{\mathrm{GF,mid}} \mid e_{\mathrm{GF,mid}} \in \mathcal{E}_{\mathrm{err,GF}}], \\
\Pr_{\text{trapdoor}}[\text{Dec. fails}] &:= \varepsilon_{\mathrm{gf}} + \varepsilon_{\mathrm{GF}}.
\end{aligned}$$

4.2.3 Design requirements.

A trapdoor instantiation should satisfy

$$\Pr\left[\lfloor e_{\mathrm{gf}} \rceil \notin \mathcal{E}_{\mathrm{err,gf}}\right] + \Pr\left[e_{\mathrm{GF,mid}} \notin \mathcal{E}_{\mathrm{err,GF}}\right] + \Pr\left[\text{FCL fails}\right] + \Pr_{\text{trapdoor}}[\text{Dec. fails}] \leq 2^{-\lambda_{\mathrm{DFR}}}.$$

For efficiency, the public candidate bounds should give

$$1 + \sum_{j=1}^{B_{\mathrm{gf},0}} \binom{m_{\mathrm{gf}}}{j} (2B_{\mathrm{gf},1})^j \leq B_{\mathrm{gf}}, \qquad 1 + \sum_{j=1}^{B_{\mathrm{GF},0}} \binom{m_{\mathrm{GF}}}{j} (2B_{\mathrm{GF},1})^j \leq B_{\mathrm{GF}},$$

with $B_{\mathrm{gf}} + B_{\mathrm{GF}}$ fixed by the parameter set. We also provide a visualized workflow of *NTRU with Trapdoor* in Figure 5.

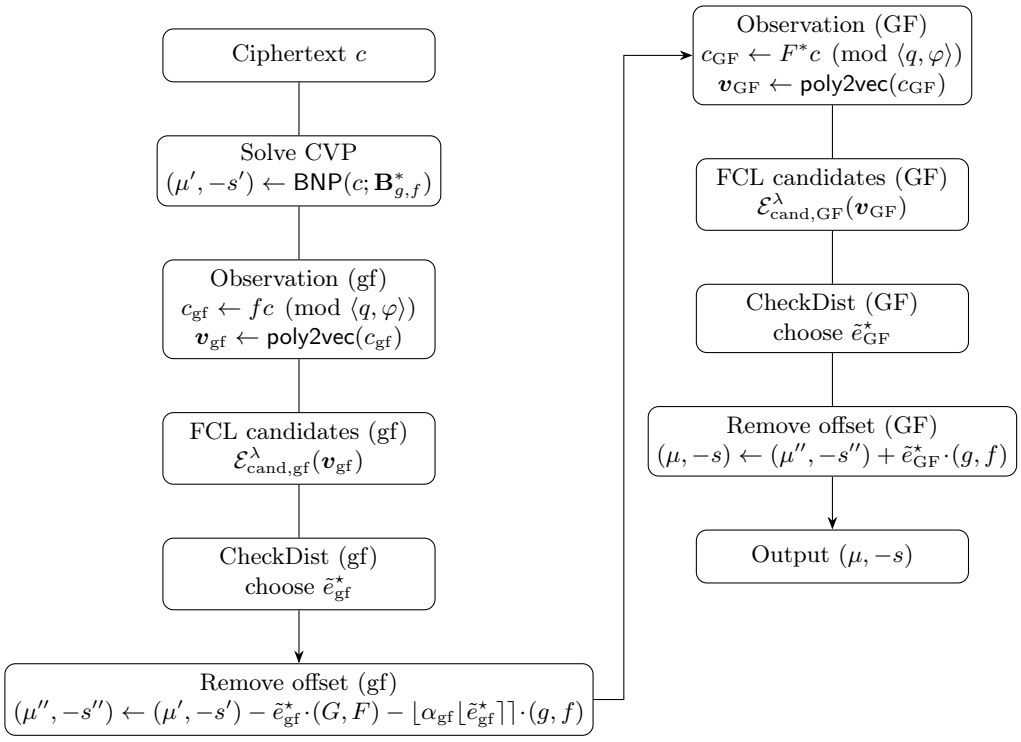

Fig. 5: *NTRU with Trapdoor* decryption workflow (BNP + two-step FCL).

4.3 Framework Coverage

Table 1 summarizes how representative NTRU-style schemes fit into our two decryption frameworks. In the "Framework" column, "Encoding" means that the scheme follows the *NTRU with Encoding* framework, while "Trapdoor" means that the scheme follows the *NTRU with Trapdoor* framework. In the "Encoding (p_1, p_2, t, w)" column, we write $\perp$ when the scheme does not use an extra auxiliary quotient; among the schemes listed here, NTRU-HPS and DAWN are the only two schemes with an auxiliary quotient. In the "Embedding" column, we use sspRLWE for both NTRU-A and NEV: NTRU-A states its plaintext-security assumption as RLWE2, whereas NEV generalizes this assumption to sspRLWE, and we use the latter name for uniformity. In the "Assumption" column, "SK" denotes the assumption used for secret-key security, and "PT" denotes the assumption used for plaintext security. Finally, in the "Scope" column, "framework" means that the scheme is covered at the design level, while "frontier" means that it is also covered by the narrower heuristic compactness-frontier model of Section 5. Thus, the table records decryption-pattern coverage; the compactness frontiers use the more restricted power-of-two cyclotomic search model.

Table 1: Coverage of representative NTRU-style public encryption schemes.

Scheme	Framework	Main ring	Encoding (p_1, p_2, t, w) or CVP-solving algorithm	Embedding	Auxiliary	Assumption	Scope
NTRU-HPS [43]	encoding	$\mathbb{Z}[x]/\langle x^p - 1\rangle$ p is a prime	$(3, \perp, 3, 1)$	identity	side information from a quotient $\mathbb{Z}[x]/\langle x - 1\rangle$	SK: NTRU PT: RLWE	framework
NTRU-HRSS [38]	encoding	$\mathbb{Z}[x]/\langle x^p - 1\rangle$ p is a prime	$(3, \perp, 3, 1)$	lift by $x - 1$	sampling on subset with non-negative correlation	SK: NTRU PT: RLWE	framework
NTRU Prime [4]	encoding	$\mathbb{Z}[x]/\langle x^p - x - 1\rangle$ p is a prime	$(3, \perp, 3, 1)$	identity	$\perp$	SK: NTRU PT: RLWR	framework
NTTRU [32]	encoding	$\mathbb{Z}[x]/\langle x^n - x^{n/2} + 1\rangle$ $n = 2^i \cdot 3^j$	$(3, \perp, 3, 1)$	ACWC_0	$\perp$	SK: NTRU PT: RLWE	framework
NTRU-A [15]	encoding	$\mathbb{Z}[x]/\langle x^n - x^{n/2} + 1\rangle$ $n = 2^i \cdot 3^j$	$(2, \perp, 2, 1)$	sspRLWE encoding	$\perp$	SK: NTRU PT: sspRLWE	framework
NTRU-B [15]	encoding	$\mathbb{Z}[x]/\langle x^n - x^{n/2} + 1\rangle$ $n = 2^i \cdot 3^j$	$(3, \perp, 3, 1)$	ACWC (GOTP)	$\perp$	SK: NTRU PT: RLWE	framework
NTRU-C [15]	encoding	$\mathbb{Z}[x]/\langle x^n - x^{n/2} + 1\rangle$ $n = 2^i \cdot 3^j$	$(3, \perp, 3, 1)$	ACWC_0	$\perp$	SK: NTRU PT: RLWE	framework
CTRU [27]	encoding	$\mathbb{Z}[x]/\langle x^n - x^{n/2} + 1\rangle$ $n = 2^i \cdot 3^j$	$(q, \perp, 1, \frac{q}{2})$	E_8 lattice encoding	$\perp$	SK: NTRU PT: RLWE	framework
CNTR [27]	encoding	$\mathbb{Z}[x]/\langle x^n - x^{n/2} + 1\rangle$ $n = 2^i \cdot 3^j$	$(q, \perp, 1, \frac{q}{2})$	E_8 lattice encoding	$\perp$	SK: NTRU PT: RLWR	framework
NTRU+ [14]	encoding	$\mathbb{Z}[x]/\langle x^n - x^{n/2} + 1\rangle$ $n = 2^i \cdot 3^j$	$(3, \perp, 3, 1)$	ACWC (SOTP)	$\perp$	SK: NTRU PT: RLWE	framework
NEV [41]	encoding	$\mathbb{Z}[x]/\langle x^n + 1\rangle$ $n = 2^i$	$(q, \perp, 1 - x^{n/k}, \frac{q+1}{2} \cdot \sum_{i=0}^{k-1} x^{in/k})$ $k = 2$ or 4	identity	$\perp$	SK: NTRU PT: RLWE	frontier
NEV′ [41]	encoding	$\mathbb{Z}[x]/\langle x^n + 1\rangle$ $n = 2^i$	$(q, \perp, 1, \frac{q+1}{2} \cdot \sum_{i=0}^{k-1} x^{in/k})$ $k = 2$ or 4	sspRLWE encoding	$\perp$	SK: NTRU PT: sspRLWE	frontier
MNTRU [1]	encoding	$\mathbb{Z}[x]/\langle x^n + 1\rangle$ $n = 2^i$	$(2, \perp, 2, 1)$	identity	$\perp$	SK: MNTRU PT: MLWE	framework
DAWN [30]	encoding	$\mathbb{Z}[x]/\langle x^n + 1\rangle$ $n = 2^i$	$(2, 2, x^{n/2} + 1, x^{n/4} + 1)$	identity	side information from a quotient $\mathbb{Z}[x]/\langle 2, x^{n/4} + 1\rangle$	SK: NTRU PT: RLWE	frontier
BAT [18]	trapdoor	$\mathbb{Z}[x]/\langle x^n + 1\rangle$ $n = 2^i$	Babai nearest plane	ACWC_0	$\perp$	SK: NTRU PT: RLWR	frontier
END (this work)	trapdoor	$\mathbb{Z}[x]/\langle x^n + 1\rangle$ $n = 2^i$	Babai nearest plane	ACWC (GOTP)	FCL with distributional verification	SK: NTRU PT: RLWR	frontier

5 Heuristic Compactness Frontiers

This section studies how small the public key and ciphertext can be within the two decryption frameworks. The results are heuristic compactness frontiers under the following search model. We restrict the search to power-of-two cyclotomic rings. We do not include trinomial cyclotomic rings, module variants, or ECC-assisted message recovery. Trinomial cyclotomic rings introduce additional ring-specific error terms; we therefore treat them as framework-level examples but exclude them from the frontier search. For each parameter point, we require a target lattice-estimator security level, a target DFR, and a public bound on the number of candidate trials. The compactness metric is the sum of the public-key and ciphertext sizes. Aiming for NIST-I security, we set $\lambda = \lambda_{\mathrm{DFR}} = 128$, use $n = 512$ whenever it satisfies the security estimate, and require the number of candidate trials to be at most n. The n-candidate limit is an engineering budget rather than a mathematical threshold. Increasing it can improve compactness, but it raises verification cost and exposes more simultaneous wraps to dependency effects. The main computation in NTRU encryption is polynomial multiplication, which is $O(n \log n)$ when an NTT is available; we therefore keep verification as a sub-cost relative to this operation and avoid a candidate budget large enough to amplify dependency effects substantially. For concrete scale, the FCL-ML-KEM point in Appendix A has $(m_0, B_0, B_1) = (2, 2, 1)$, hence nine candidate errors including zero; it reduces the ciphertext from 768 to 688 bytes with about a 5% increase in total reference-C runtime. The NIST-I encoding and trapdoor frontier points below use 473 and 478 candidate trials, respectively, and yield public-key-plus-ciphertext totals of 812 and 754 bytes. For these unimplemented frontier points we report candidate-trial count as the verification-cost measure rather than extrapolating cycle counts. If the candidate budget is made unbounded, the optimization problem changes: the compactness frontier approaches a packing limit rather than an efficient-decryption frontier. We provide all the scripts to calculate the probabilities in the anonymous repository[3].

5.1 Encoding frontier

For NTRU with Encoding, we search over quotient-ring families $\{(p_i, \phi_i)\}_{1 \le i \le \ell}$ and public polynomials t, w. The constraints are: the decoding ring must support at least 2^λ messages after the required embedding and unembedding; the auxiliary rings must reduce the error-candidate fibers enough to meet the candidate-trial budget; and the DFR terms from the wrap bound, FCL miss probability, and candidate verification must sum to at most $2^{-\lambda_{\mathrm{DFR}}}$.

5.1.1 Quotient ring series selection For fixed (n, q) and fixed distributions at the targeted security level, the remaining degrees of freedom in *NTRU with Encoding* are the choice of quotient-ring series $\{(p_i, \phi_i)\}_{1 \le i \le \ell}$ and the associated

[3] https://anonymous.4open.science/r/Explore_NTRU_Decryption-2F3F/

polynomials (t, w) that enable (i) a DFR less than $2^{-\lambda_{\mathrm{DFR}}}$ and (ii) a small candidate list after FCL and auxiliary filtering. A naive search over primes p and factors $\phi \mid \varphi$ in $\mathbb{Z}_p[x]$ is expensive. We therefore use the following arithmetic filter driven by t. Fix φ and a candidate $t \in \mathbb{Z}[x]$, and let

$$r_t \;:=\; \varphi \bmod \langle t \rangle \in \mathbb{Z}[x], \qquad T_\varphi(t) \;:=\; \gcd(\{(r_t)_{[i]}\}_{0 \le i \le \deg(r_t)}) \in \mathbb{Z}_{\ge 0},$$

If a prime p divides $T_\varphi(t)$, then $r_t \equiv 0 \pmod{p}$, hence $\varphi \equiv 0 \bmod \langle p, t \rangle$, i.e., t divides φ in $\mathbb{Z}_p[x]$. Consequently, every irreducible factor of t in $\mathbb{Z}_p[x]$ is also a factor of φ in $\mathbb{Z}_p[x]$ and can be used as some ϕ_i. Therefore, for a fixed t, admissible primes p_i must lie among the prime divisors of $T_\varphi(t)$. This reduces the search space from "all primes" to a small set determined by t. For each prime $p_i \mid T_\varphi(t)$ we factor t in $\mathbb{Z}_p[x]$ and choose:

- a decoding factor ϕ_1 to define $\mathcal{R}_{\phi_1,p_1}$ such that the effective message ring $\mathcal{R}_{\phi_{\mathcal{M}},p_1}$ has size at least 2^λ, and
- optionally, additional auxiliary factors $\{\phi_i\}_{i\ge 2}$ to define $\mathcal{R}_{\phi_i,p_i}$ that are cheap to reduce into and that split the error set into small fibers.

We then require $t \equiv 0 \pmod{\langle p_i, \phi_i \rangle}$ for all $1 \le i \le \ell$ and $w \equiv 0 \pmod{\langle p_i, \phi_i \rangle}$ for all $2 \le i \le \ell$ but $w \not\equiv 0 \pmod{\langle p_1, \phi_1 \rangle}$ as clarified in Section 4.1. In practice, once $\{(p_i, \phi_i)\}_{i\ge 2}$ is fixed, one convenient way to obtain such a w is to pick w from the ideal $\bigcap_{i=2}^{\ell} \langle p_i, \phi_i \rangle$ and explicitly reject choices that fall into $\langle p_1, \phi_1 \rangle$, while enforcing that w is invertible in $\mathcal{R}_{\varphi,q}$. Also, we need w to separate the right guess from the wrong ones in practice.

After ensuring quotient rings, we can design the corresponding Emb and Unemb such that: i) security: $\mathsf{Emb}^{\mathrm{rand}}_{\mathcal{M}\Rightarrow\mathcal{S}_\mu}(m) \overset{\mathrm{stat}}{\approx} \mathcal{U}(\mathcal{S}_\mu)$ for $m \leftarrow \mathcal{U}(\mathcal{M})$; ii) recoverable: there exists $\mathsf{Unemb}_{\mathcal{M}'\to\mathcal{M}}$ that allows us to recover m after decoding; iii) efficiency: the computation of Emb and Unemb are polynomial time, thus we can sample μ from m fast. We construct concrete embedding pairs in Appendix C.

For each candidate series, we evaluate the DFR components and efficiency components as in Section 4.1. We note that when B_0 and B_1 are large, the average-case probability in Equation (12) is expensive to calculate; therefore, we use the worst-case probability in Equation (11) to obtain the probability. We obtain the parameter setting for the heuristic frontier of *NTRU with Encoding* as follows and provide a script to calculate all probabilities.

5.1.2 Concrete heuristic frontier In this model, the smallest NIST-I encoding point we find has a public key and a ciphertext of 406 bytes each, for a total of 812 bytes. The parameters are

$$n = 512, \quad q = 81, \quad \varphi = x^n + 1,$$
$$\mathcal{D}_g = \mathcal{D}_f = \mathcal{D}_s = \mathcal{U}(\mathcal{S}_\mu) = \mathcal{T}_{512,32},$$
$$p_1 = 2, \quad \phi_1 = x^{n/2} + 1, \qquad p_2 = 2, \quad \phi_2 = x^{n/4} + 1, \qquad (B_0, B_1) = (4, 1).$$

The estimated components are

$$\Pr_{\text{encoding}}[\text{Dec. fails}] = 0, \qquad \Pr[e \notin \mathcal{E}_{\mathrm{err}}] \le 2^{-155.5}, \qquad \Pr[\text{FCL fails}] \le 2^{-132.1},$$

and the verification failure is zero for this quotient choice. The estimated security level is 128.2 bits, and the candidate set has size $473 < n$. This choice of quotient ring coincides with the DAWN-style point. This should not be interpreted as saying that DAWN is universally optimal. Rather, within the stated power-of-two encoding model and under the n-candidate budget, DAWN lies on the efficient side of the frontier. To obtain a smaller encoding-based scheme, the search finds choices with more aggressive quotient information, but these choices require more candidate trials. To achieve faster decoding with fewer candidate trials, the search requires larger parameter counts. This is the intended meaning of the encoding frontier.

Two regimes appear in the search. If $\overline{w}$ is not invertible in the decoding ring, multiplication by w maps the message into a proper image. This creates algebraic redundancy that can help verification, but it also reduces the effective message space. If $\overline{w}$ is invertible after projection to the effective message ring, then redundancy must come from the embedding and unembedding mechanism rather than from the image of multiplication by w. The encoding case study in Appendix C illustrates the second regime with two auxiliary rings.

(i) $\gcd(w, \phi_1) \neq 1$ in $\mathbb{Z}_{p_1}[x]$. In this case, multiplication by w induces a proper image in $\mathbb{Z}_{p_1}[x]/\langle\phi_1\rangle$, so the effective message space is a strict subset of $\mathbb{Z}_{p_1}[x]/\langle\phi_1\rangle$. This redundancy can help distinguish incorrect error guesses, and it comes at the cost of reducing the message space. In the context of KEM, the message space is always redundant. Beyond the trivial choice $p_1 = 2$, the next parameter choice we found with this property is $p_1 = 7$ and $\phi_1 = x^{n/2} - 3x^{n/4} + 1$. However, this choice substantially amplifies the noise term, which in turn requires a large number of candidate trials, exceeding our trial budget.

(ii) $\gcd(w, \phi_1) = 1$ in $\mathbb{Z}_{p_1}[x]$. Here, w is invertible modulo ϕ_1, so the map from m to the decoding ring is a bijection on $\mathbb{Z}_{p_1}[x]/\langle\phi_1\rangle$. As a result, any redundancy coming from the message distribution (i.e., the fact that valid messages occupy a strict subset of the ambient ring due to encoding) cannot be exploited in the same "algebraic subspace" manner as in the former case. Instead, this redundancy must be surfaced through the embedding/unembedding mechanism to strengthen verification. NwE provides an instantiation in this regime.

Overall, when the efficiency bound (candidate-trial budget) is tight, the available algebraic structure is limited. This motivates the use of fast-decoding techniques that remain effective even when multiple candidate trials are allowed. In this sense, we partially address the question raised in DAWN [30]: DAWN's parameter choice is essentially optimal under the constraint of a single candidate trial. Achieving higher decoding efficiency within the *NTRU with Encoding* framework, therefore, requires paying the cost of the number of candidate trials.

5.2 Trapdoor frontier

For NTRU with Trapdoor, the search fixes the NTRU trapdoor structure and varies n, q, the sparse distributions, and the public FCL bounds for the two

rounding terms. The verification rule is distributional: the correct correction produces a vector that matches the secret and message distributions, whereas incorrect corrections introduce norm or range violations.

5.2.1 Distribution based verification We choose different CheckDist functions for $\lfloor e_{\mathrm{gf}} \rceil$ and $e_{\mathrm{GF,mid}}$. For $\lfloor e_{\mathrm{gf}} \rceil$, the gap of norm between the right guess and the wrong guess is large enough; therefore, we can compute the norm corresponding to the guess $\tilde{e}_{\mathrm{gf}}$, and choose the minimum one as $\tilde{e}^{\star}_{\mathrm{gf}}$. Then we need to estimate the gap between the norms of right and wrong guesses. We rewrite Equation (13) as

$$\begin{aligned}(\mu',\ -s') =&(\mu, -s) + \lfloor e_{\mathrm{gf}} \rceil \cdot (G^*, F^*) - \lfloor e_{\mathrm{GF}} \rceil \cdot (g, f) \\ &+ \Big([\![e_{\mathrm{mid}}]\!] - \big\lfloor [\![e_{\mathrm{mid}}]\!] + [\![e_{\mathrm{GF}}]\!] \big\rceil \Big) \cdot (g, f). \end{aligned} \tag{14}$$

We treat $\lfloor e_{\mathrm{gf}} \rceil$ and $\lfloor e_{\mathrm{GF}} \rceil$ as bounded discrete variables, and model the fractional parts $[\![e_{\mathrm{GF}}]\!]$ and $[\![e_{\mathrm{mid}}]\!]$ as approximately i.i.d. uniform on $[-\frac{1}{2}, \frac{1}{2})$. Combined with the heuristic that

$$\big\| (G^*, F^*) \big\|_2 \cdot \big\| (g, f) \big\|_2 \approx \frac{q \exp(1)}{2}$$

from [36], [23], and the BAT analysis [18], we can estimate the distribution of the norm $\big\| (\tilde{\mu}, \tilde{s}) \big\|_2$. This trapdoor-norm model is distinct from the coefficient-independence and random-placement heuristics. Since (g, f, G^*, F^*) are fixed by key generation and cannot be searched offline by the adversary, average-case behavior over honestly generated keys is the relevant notion here, unlike randomness in publicly searchable encryption. For the guess $\tilde{e}_{\mathrm{gf}}$, let $\Delta_{e_{\mathrm{gf}}} := \lfloor e_{\mathrm{gf}} \rceil - \tilde{e}_{\mathrm{gf}}$, then we define

$$\begin{aligned}\boldsymbol{v}_{\mathrm{guess,gf}}(\mu, s, g, f, \Delta_{e_{\mathrm{gf}}}) :=&(\mu, -s) + \Delta_{e_{\mathrm{gf}}} \cdot (G^*, F^*) - \lfloor e_{\mathrm{GF}} \rceil \cdot (g, f) \\ &+ \Big([\![\Delta_{e_{\mathrm{mid}}}]\!] - \big\lfloor [\![\Delta_{e_{\mathrm{mid}}}]\!] + [\![e_{\mathrm{GF}}]\!] \big\rceil \Big) \cdot (g, f),\end{aligned}$$

where $\Delta_{e_{\mathrm{mid}}} = \frac{G\bar{g} + F\bar{f}}{g\bar{g} + f\bar{f}} \Delta_{e_{\mathrm{gf}}}$. We need to distinguish the correct guess from all the other guesses. For the right guess, we have

$$\boldsymbol{v}_{\mathrm{guess,gf}}(\mu, s, g, f, 0) = (\mu, -s) - \lfloor e_{\mathrm{GF}} \rceil \cdot (g, f).$$

For the commonly used power-of-2 cyclotomic ring, the closest guess means that $\Delta_{e_{\mathrm{gf}}} \in \{\pm x^i\}_{0 \le i < n}$ and such polynomials wouldn't affect the norm, while for some other rings like $\mathcal{R}_{x^n - x^{n/2} + 1}$, the tri-cyclotomic ring, such monomials may increase or decrease the norm by a little, therefore, we cannot give a formal definition of the closest wrong guess. Define

$$\begin{aligned}N_{\mathrm{right}}(\mu, s, g, f) &:= \|\boldsymbol{v}_{\mathrm{guess,gf}}(\mu, s, g, f, 0)\|_2, \\ N_{\mathrm{wrong}}(\mu, s, g, f) &:= \min_{\Delta_{e_{\mathrm{gf}}} \neq 0} \|\boldsymbol{v}_{\mathrm{guess,gf}}(\mu, s, g, f, \Delta_{e_{\mathrm{gf}}})\|_2.\end{aligned}$$

Then, we can compute the probability of the wrong decoding as

$$\Pr\left[\text{Dec. fails on} \lfloor e_{\mathrm{gf}} \rceil\right] :=$$
$$\Pr\left[N_{\mathrm{right}}(\mu, s, g, f) \geq N_{\mathrm{wrong}}(\mu, s, g, f) \mid (\mu, s, g, f) \xleftarrow{\$} \mathcal{U}(\mathcal{S}_\mu) \times \mathcal{D}_s \times \mathcal{D}_g \times \mathcal{D}_f\right].$$

After finding the right $\lfloor e_{\mathrm{gf}} \rceil$, we do it iteratively on $\lfloor e_{\mathrm{GF}} \rceil$. Similarly, let $\Delta_{e_{\mathrm{GF}}} := \lfloor e_{\mathrm{GF}} \rceil - \tilde{e}_{\mathrm{GF}}$ and we can define

$$\boldsymbol{v}_{\mathrm{guess,GF}}(\mu, s, g, f, \Delta_{e_{\mathrm{GF}}}) := (\mu, -s) - \Delta_{e_{\mathrm{GF}}} \cdot (g, f),$$

However, when identifying the right $\tilde{e}_{\mathrm{GF}}$, the norm gap between the correct and incorrect guesses may not be large, so we need specific methods for specific distributions. We use the fixed sparse ternary distribution $\mathcal{T}_{n,k}$ as an example; we can easily check the number of 1s and -1s, then obtain the probability

$$\Pr\left[\text{Dec. fails on } e_{\mathrm{GF,mid}}\right] := \Pr\left[\exists\, \Delta_{e_{\mathrm{GF}}} \text{ s.t. } \boldsymbol{v}_{\mathrm{guess,GF}}(\mu, s, g, f, \Delta_{e_{\mathrm{GF}}}) \in \mathcal{T}_{n,k}\right].$$

5.2.2 Concrete heuristic frontier The smallest NIST-I trapdoor point found by this search has public key and ciphertext size 377 bytes each, hence total size 754 bytes. The parameters are

$$\begin{aligned} &n = 512, \quad q = 59, \quad \varphi = x^n + 1, \\ &\mathcal{D}_g = \mathcal{D}_f = \mathcal{D}_s = \mathcal{U}(\mathcal{S}_\mu) = \mathcal{T}_{512,26}, \\ &(B_{\mathrm{gf},0}, B_{\mathrm{gf},1}) = (3, 1), \qquad (B_{\mathrm{GF},0}, B_{\mathrm{GF},1}) = (2, 1). \end{aligned}$$

The estimated DFR components are

$$\begin{aligned} &\Pr[\lfloor e_{\mathrm{gf}} \rceil \notin \mathcal{E}_{\mathrm{err,gf}}] \leq 2^{-189.7}, \qquad \Pr[e_{\mathrm{GF,mid}} \notin \mathcal{E}_{\mathrm{err,GF}}] \leq 2^{-165.8}, \\ &\Pr_{\text{trapdoor}}[\text{Dec. fails}] \leq 2^{-251.8}, \qquad \Pr[\text{FCL fails}] \leq 2^{-129.2}. \end{aligned}$$

The estimated security level is 131.3 bits, and the candidate set size is $478 < n$.

6 END: A Smaller Trapdoor-Based NTRU KEM

We now instantiate NTRU with Trapdoor as a practical KEM, denoted END. The goal is to retain the compactness advantage suggested by the trapdoor frontier while using conservative parameters, a standard KEM transform, and implementable constant-time candidate verification.

The construction has three layers. The deterministic public-key encryption layer (Algorithms 7–9) exposes the trapdoor decoding problem, runs the fixed precision variant of Babai's nearest plane as in Zalcon [19] and BAT [18], and then performs two FCL correction rounds. The randomized PKE layer (Algorithms 10–11) applies the average-case to worst-case (ACWC) transform proposed in [15]. The KEM layer (Algorithms 12–14) applies the Fujisaki-Okamoto transform with re-encryption in decapsulation. The security proof is given in Appendix D under decision NTRU and search Ring-LWR assumptions.

6.1 Trapdoor decoding and verification

To select the unique correct representative among the candidates, END uses two distribution-based verification scores, $\mathsf{CheckDist}_{\mathrm{gf}}$ and $\mathsf{CheckDist}_{\mathrm{GF}}$. The score $\mathsf{CheckDist}_{\mathrm{gf}}$ implements the squared-norm check from Section 5.2 (Algorithm 4). The first verification score is

$$\mathsf{CheckDist}_{\mathrm{gf}}(\tilde{\mu}, -\tilde{s}) = \|\mathsf{poly2vec}(\tilde{\mu})\|_2^2 + \|\mathsf{poly2vec}(\tilde{s})\|_2^2.$$

The correct gf-correction gives a vector close to the expected message and secret distributions, while an incorrect correction adds a trapdoor-basis component and usually increases the norm.

The second verification score checks the final secret and message ranges:

$$\begin{aligned}&\mathsf{CheckDist}_{\mathrm{GF}}(\tilde{\mu}_{\mathrm{final}}, -\tilde{s}_{\mathrm{final}})\\ &= \max\left(\|\mathsf{poly2vec}(\tilde{\mu}_{\mathrm{final}})\|_\infty - \left\lfloor \frac{q}{2d_e} \right\rceil, 0\right) + \sum_{i=0}^{n-1} (\tilde{s}_{\mathrm{final}})_{[i]}((\tilde{s}_{\mathrm{final}})_{[i]} - 1).\end{aligned}$$

The score $\mathsf{CheckDist}_{\mathrm{GF}}$ enforces coefficient-range constraints implied by the chosen secret and message distributions (Algorithm 5).

Algorithm 4 $\mathsf{END.CheckDist}_{\mathrm{gf}}$

Input: vector $(\mu', -s') \in \mathcal{R}_\varphi^2$
Output: score S_{gf}
1: **return** $S_{\mathrm{gf}} := \|\mathsf{poly2vec}(\mu')\|_2^2 + \|\mathsf{poly2vec}(s')\|_2^2$

Algorithm 5 $\mathsf{END.CheckDist}_{\mathrm{GF}}$

Input: candidate vector $(\mu'', -s'') \in \mathcal{R}_\varphi^2$
Output: score S_{GF}
1: **return** $S_{\mathrm{GF}} := \max\left(\|\mathsf{poly2vec}(\mu'')\|_\infty - \lfloor \frac{q}{2d_e} \rceil, 0\right) + \sum_{i=0}^{n-1} (s'')_{[i]}((s'')_{[i]} - 1)$

This is natural in our setting: the secret satisfies $s_i \in \{0, 1\}$, and the message coefficients μ_i arise from compression and are designed to be close to uniform over a bounded interval. Accordingly, $\mathsf{CheckDist}_{\mathrm{GF}}$ can be viewed as a range-violation score, assigning larger penalties to candidates whose coefficients fall outside the permitted ranges. In our implementation, when the overall decryption-failure probability Pr[Dec. fails] is extremely small, we exploit this redundancy to accelerate verification by evaluating only the terms involving s in $\mathsf{CheckDist}_{\mathrm{gf}}$ and $\mathsf{CheckDist}_{\mathrm{GF}}$. In Table 3, we report the estimates of DFRs corresponding to $\lfloor e_{\mathrm{gf}} \rceil$ and $e_{\mathrm{GF,mid}}$; for example, $(-133, -130)$ means $\Pr\left[\lfloor e_{\mathrm{gf}} \rceil \notin \mathcal{E}_{\mathrm{err,gf}}\right] \leq 2^{-133}$, and $\Pr\left[e_{\mathrm{GF,mid}} \notin \mathcal{E}_{\mathrm{err,GF}}\right] \leq 2^{-130}$.

Algorithm 6 $\mathsf{END.Decode}_{\mathrm{Trap}}$

Input: private key $sk := (g, f, G, F, w)$, decompressed ciphertext c
Output: corrected pair $(\mu, -s)$

$(G_Q, F_Q) \leftarrow Q \cdot (G, F) - w \cdot (g, f)$
$c_{\mathrm{gf}} \leftarrow fc \pmod{\langle q, \varphi \rangle}$
$c_{\mathrm{GF}} \leftarrow Q \cdot Fc - wc_{\mathrm{gf}} \pmod{\langle qQ, \varphi \rangle}$
$(\mu', -s') \leftarrow \frac{1}{qQ} \cdot (c_{\mathrm{GF}}, -c_{\mathrm{gf}}) \cdot \begin{pmatrix} g & f \\ G_Q & F_Q \end{pmatrix}$
$\mathcal{I}_n \leftarrow \{0, \ldots, n-1\}$
$\mathcal{E}_{\mathrm{gf}} \leftarrow \mathcal{E}_{\mathrm{cand}}(\mathsf{poly2vec}(c_{\mathrm{gf}}), q, m_{\mathrm{gf}}, B_{\mathrm{gf},0}, B_{\mathrm{gf},1}, \mathcal{I}_n)$
$\tilde{e}^{\star}_{\mathrm{gf}} \leftarrow \underset{\tilde{e}_{\mathrm{gf}} \in \mathcal{E}_{\mathrm{gf}}}{\arg\min}\, \mathsf{END.CheckDist}_{\mathrm{gf}}\left((\mu', -s') + (\lfloor \frac{1}{qQ} \cdot (c_{\mathrm{GF}} - q \cdot w\tilde{e}_{\mathrm{gf}}) \rceil, -\tilde{e}_{\mathrm{gf}}) \cdot \begin{pmatrix} g & f \\ G & F \end{pmatrix} \right)$
$\tilde{c}_{\mathrm{gf}} \leftarrow c_{\mathrm{gf}} + q \cdot \tilde{e}^{\star}_{\mathrm{gf}} \pmod{\langle \varphi \rangle}$
$\tilde{c}_{\mathrm{GF}} \leftarrow c_{\mathrm{GF}} - q \cdot w\tilde{e}^{\star}_{\mathrm{gf}} \pmod{\langle qQ, \varphi \rangle}$
$(\mu'', -s'') \leftarrow \frac{1}{qQ} \cdot (\tilde{c}_{\mathrm{GF}}, -\tilde{c}_{\mathrm{gf}}) \cdot \begin{pmatrix} g & f \\ G_Q & F_Q \end{pmatrix}$
$\mathcal{E}_{\mathrm{GF}} \leftarrow \mathcal{E}_{\mathrm{cand}}(\mathsf{poly2vec}(\tilde{c}_{\mathrm{GF}}), qQ, m_{\mathrm{GF}}, B_{\mathrm{GF},0}, B_{\mathrm{GF},1}, \mathcal{I}_n)$
$\tilde{e}^{\star}_{\mathrm{GF}} \leftarrow \underset{\tilde{e}_{\mathrm{GF}} \in \mathcal{E}_{\mathrm{GF}}}{\arg\min}\, \mathsf{END.CheckDist}_{\mathrm{GF}}\left((\mu'', -s'') + \tilde{e}_{\mathrm{GF}} \cdot (g, f) \right)$
$(\mu, -s) \leftarrow (\mu'', -s'') + \tilde{e}^{\star}_{\mathrm{GF}} \cdot (g, f)$
return $(\mu, -s)$

Algorithm 7 $\mathsf{END.DPKE.KeyGen}$

Output: public key pk, private key sk

$g \xleftarrow{\$} \mathcal{T}_{n,k_g}$
$f \xleftarrow{\$} \mathcal{T}_{n,k_f}$
if g or f is not invertible in $\mathcal{R}_{\varphi,q}$ **then**
    go back to step 1
$h \leftarrow gf^{-1} \pmod{\langle q, \varphi \rangle}$
compute $(G, F) \in \mathcal{R}^2_{\varphi}$ s.t. $gF - fG \equiv q$ ▷ *follow the algorithm in [35]*
$w \leftarrow \lfloor Q \cdot \frac{G\bar{g} + \gamma F\bar{f}}{g\bar{g} + \gamma f\bar{f}} \rceil$
return $(pk := h, sk := (g, f, G, F, w))$

Algorithm 8 $\mathsf{END.DPKE.Encrypt}$

Input: public key $pk := h$, secret s
Output: ciphertext c

$c \leftarrow \lfloor \frac{d_e}{q} \cdot (hs \pmod{\langle q, \varphi \rangle}) \rceil$
return c

6.2 Parameter Selection

We provide two concrete parameter sets targeting NIST-I/V security. The table reports the ring dimension n, the small decryption modulus q, the precision modulus Q, the trapdoor tuning parameter γ, the distribution parameters $(k_g, k_f), d_e$, and the bounds used in FCL. We choose Q to be as small as pos-

Algorithm 9 END.DPKE.Decrypt

Input: private key $sk := (g, f, G, F, w)$, ciphertext c
Output: secret s
1: $(\mu, -s) \leftarrow \mathsf{END.Decode}_{\mathrm{Trap}}(sk, \lfloor \frac{q}{d_e} \cdot c \rceil)$
2: **return** s

Algorithm 10 END.PKE.Encrypt

Input: public key $pk := h$, message $m \in \mathcal{M}$, randomness $r \in \mathcal{M}$
Output: ciphertext c
1: $s \leftarrow m + \mathsf{Hash}_1(r) + x^{n/2} r \pmod{\langle 2 \rangle}$
2: $c \leftarrow \mathsf{END.DPKE.Encrypt}(pk, s)$
3: **return** c

Algorithm 11 END.PKE.Decrypt

Input: private key sk, ciphertext c
Output: message m
1: $s \leftarrow \mathsf{END.DPKE.Decrypt}(sk, c)$
2: $m' \leftarrow s \pmod{\langle x^{n/2} \rangle}$
3: $r \leftarrow x^{n/2}(s - m') \pmod{\langle 2, x^n + 1 \rangle}$
4: $m \leftarrow m' + \mathsf{Hash}_1(r) \pmod{\langle 2 \rangle}$
5: **return** m

Algorithm 12 END.KEM.KeyGen

Output: public key pk, private key sk
1: $(pk, sk') \leftarrow \mathsf{END.DPKE.KeyGen}$
2: $k \xleftarrow{\$} \mathcal{U}(\mathcal{M})$
3: **return** $(pk, sk := (k, pk, sk'))$

Algorithm 13 END.KEM.Encapsulation

Input: public key pk
Output: ciphertext c, shared key K
1: $m \xleftarrow{\$} \mathcal{U}(\mathcal{M})$
2: $c \leftarrow \mathsf{END.PKE.Encrypt}(pk, m, \mathsf{Hash}_2(m))$
3: $K \leftarrow \mathsf{Hash}_3(m, c)$
4: **return** (c, K)

sible while still supporting an NTT implementation and meeting the DFR constraints. We choose γ to balance the scaling of G^*s and $F^*\mu$, since μ originates from compression and typically has larger variance than s. We note that the frontier point is a heuristic lower-size design point; END uses more conservative, implementation-driven parameters and therefore has a larger total size.

Algorithm 14 END.KEM.Decapsulation

Input: private key $sk := (k, pk, sk')$, ciphertext c
Output: shared key K
1: $m \leftarrow \mathsf{END.PKE.Decrypt}(sk', c)$
2: **if** $c = \mathsf{END.PKE.Encrypt}(pk, m, \mathsf{Hash}_2(m))$
3: $\quad K \leftarrow \mathsf{Hash}_3(m, c)$
4: **else**
5: $\quad K \leftarrow \mathsf{Hash}_3(k, c)$
6: **return** K

6.3 Size, performance, and implementation

Table 4 compares public-key and ciphertext sizes. Table 5 reports cycle counts. We provide a reference C implementation and an AVX2 implementation. The benchmarks were measured single-threaded on an Intel Core i9-11900K CPU at 3.5 GHz, with TurboBoost and hyperthreading disabled. The main cost difference is key generation. END key generation is much slower than ML-KEM and ordinary NTRU encryption because it computes an NTRU trapdoor completion (G, F). This cost is comparable in nature to BAT. The construction is therefore best suited to settings where the public key is reused across many encapsulations. The encapsulation and decapsulation costs are competitive. For END-512 in reference C, the combined cost of encapsulation and decapsulation is about 3.2% higher than ML-KEM. The ciphertext is 384 bytes, exactly half the size of the 768-byte ML-KEM-512 ciphertext. The implementation uses fixed public candidate bounds, evaluates all candidates up to those bounds, computes verification scores without secret-dependent branches, and uses constant-time winner selection. We use TIMECOP[4] to test timing independence of the submitted implementation paths, which gives evidence that the reported implementation is not using secret-dependent early exits or timing-variable candidate enumeration. For each baseline, we use the side-channel-countermeasure or constant-time path provided by its implementation; for ML-KEM, we use the pq-crystals Kyber implementation.[5] Thus Table 5 compares the protected END implementation with protected baseline implementations.

For the concrete security estimates in Table 4, we estimated the listed schemes with `lattice-estimator` at commit `3e48ef4` and `red_cost_model=MATZOV`. We additionally checked END with the enhanced lattice estimator [6] from [25] and the PrimalMeetLWE estimator [7] from [22]. Neither check lowers END's minimum from the standard lattice estimator. The resulting END minima are 152.7 bits for END-512 and 287.1 bits for END-1024, the highest values among the schemes listed at the corresponding levels. These estimates use the sparse ternary NTRU secret distributions specified by (k_g, k_f) in Table 2.

[4] https://www.post-apocalyptic-crypto.org/timecop/
[5] https://github.com/pq-crystals/kyber
[6] https://github.com/identitymapping/enhanced_lattice-estimator
[7] https://github.com/yonghaason/PrimalMeetLWE/tree/main/estimator

Table 2: Suggested parameters for END

Scheme	n	q	Q	γ	(k_g, k_f)	d_e	$(B_{\mathrm{gf},0}, B_{\mathrm{gf},1})$	$(B_{\mathrm{GF},0}, B_{\mathrm{GF},1})$	PK	CT
END-512	512	257	12289	4	$(72, 72)$	64	$(2, 1)$	$(2, 1)$	514	384
END-1024	1024	257	12289	3	$(96, 96)$	90	$(2, 1)$	$(1, 1)$	1027	832

Table 3: DFRs of END

Scheme	$\log_2(\Pr[\lfloor e \rceil \notin \mathcal{E}_{\mathrm{err}}])$	$\log_2(\Pr[\text{FCL fails}])$	$\log_2(\Pr[\text{Dec. fails}])$	$\log_2(\mathrm{DFR})$
END-512	$(-147, -138)$	$(-133, -130)$	-138	-130
END-1024	$(-185, -161)$	$(-169, -157)$	-296	-157

Table 4: Comparison between END, NTRU KEMs, and ML-KEM in size

Scheme	NIST Sec.	Ciphertext (bytes)	Public Key (bytes)	Sec. Level ($\log_2$)	Dec. Failure ($\log_2$)	Compression Ratio (CT/PK)
END-512	NIST-I	**384**	**514**	**152.7**	-130	-/-
DAWN-α-512		436	615	139.4	-133	12%/16%
DAWN-β-512		450	514	133.5	-130	15%/0%
BAT-512		473	521	144.6	-146	19%/1%
NEV-512		615	615	136.7	-138	37%/16%
ML-KEM-512		768	800	139.7	-138	50%/36%
END-1024	NIST-V	**832**	**1027**	**287.1**	-157	-/-
DAWN-α-1024		973	1229	269.7	-166	14%/16%
BAT-1024		1006	1230	286.4	-138	17%/16%
DAWN-β-1024		1027	1027	252.9	-146	19%/0%
NEV-1024		1229	1229	267	-152	32%/16%
ML-KEM-1024		1568	1568	262.3	-174	47%/35%

Table 5: Comparison between END, NTRU KEMs, and ML-KEM in performance

Scheme	NIST Sec.	REF			AVX2		
		Keygen (kcycles)	Encaps (kcycles)	Decaps (kcycles)	Keygen (kcycles)	Encaps (kcycles)	Decaps (kcycles)
END-512	NIST-I	10, 408	23	267	6, 651	12	71
DAWN-α-512		92	42	45	22	20	19
DAWN-β-512		71	35	48	23	22	23
BAT-512		29, 579	33	233	23, 543	9	50
NEV-512		95	88	113	21	33	26
ML-KEM-512		107	122	159	21	22	24
END-1024	NIST-V	41, 597	56	766	30, 811	23	160
DAWN-α-1024		138	77	113	46	36	38
DAWN-β-1024		135	88	127	46	42	47
BAT-1024		155, 582	65	444	109, 412	20	107
NEV-1024		208	183	257	37	64	50
ML-KEM-1024		271	284	348	47	46	51

References

1. Bai, S., Jangir, H., Lin, H., Ngo, T., Wen, W., Zheng, J.: Compact Encryption Based on Module-NTRU Problems. In: Post-Quantum Cryptography: 15th International Workshop, PQCrypto 2024, Oxford, UK, June 12–14, 2024, Proceedings, Part I. p. 371–405. Springer-Verlag, Berlin, Heidelberg (2024). https://doi.org/10.1007/978-3-031-62743-9_13, https://doi.org/10.1007/978-3-031-62743-9_13
2. Banerjee, A., Brenner, H., Leurent, G., Peikert, C., Rosen, A.: SPRING: Fast Pseudorandom Functions from Rounded Ring Products. In: Cid, C., Rechberger, C. (eds.) Fast Software Encryption. pp. 38–57. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
3. Banerjee, A., Peikert, C., Rosen, A.: Pseudorandom Functions and Lattices. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology – EUROCRYPT 2012. pp. 719–737. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
4. Bernstein, D.J., Chuengsatiansup, C., Lange, T., van Vredendaal, C.: NTRU Prime. Tech. rep., National Institute of Standards and Technology (2017), available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions
5. Bernstein, D.J., Chuengsatiansup, C., Lange, T., van Vredendaal, C.: NTRU Prime: Reducing Attack Surface at Low Cost. In: Adams, C., Camenisch, J. (eds.) Selected Areas in Cryptography – SAC 2017. pp. 235–260. Springer International Publishing, Cham (2018)
6. Biasioli, B., Marcolla, C., Calderini, M., Mono, J.: Improving and Automating BFV Parameters Selection: An Average-Case Approach. Cryptology ePrint Archive, Paper 2023/600 (2023), https://eprint.iacr.org/2023/600
7. Blair, W., Araujo, F., Taylor, T., Jang, J.: Automated synthesis of effect graph policies for microservice-aware stateful system call specialization. In: 2024 IEEE Symposium on Security and Privacy. pp. 4554–4572. IEEE Computer Society Press, San Francisco, CA, USA (May 19–23, 2024). https://doi.org/10.1109/SP54263.2024.00064
8. Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehle, D.: CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM. In: 2018 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 353–367 (2018). https://doi.org/10.1109/EuroSP.2018.00032
9. Chen, C., Damba, O., rey Ho stein, J., Hulsing, A., Rijneveld, J., Schanck, J.M., Schwabe, P., Whyte, W., Zhang, Z.: Ntru: Algorithm specifications and supporting documentation. Brown University and Onboard security company, Wilmington, USA (2019)
10. Chen, J., Chen, R., Wang, Y., Jiang, H., Peng, C., Huang, X., He, D., Chen, X.: Tighter proofs for PKE-to-KEM transformations under average-case decryption error and without γ-spread. Cryptology ePrint Archive, Paper 2026/468 (2026), https://eprint.iacr.org/2026/468
11. Chen, L., Zhang, Z., Zhang, Z.: On the Hardness of the Computational Ring-LWR Problem and Its Applications. In: Peyrin, T., Galbraith, S. (eds.) Advances in Cryptology – ASIACRYPT 2018. pp. 435–464. Springer International Publishing, Cham (2018)
12. Costache, A., Nürnberger, L., Player, R.: Optimisations and Tradeoffs for HElib. In: Rosulek, M. (ed.) Topics in Cryptology – CT-RSA 2023. pp. 29–53. Springer International Publishing, Cham (2023)

13. D Anvers, J.P., Vercauteren, F., Verbauwhede, I., Ding, J., Steinwandt, R.: The Impact of Error Dependencies on Ring/Mod-LWE/LWR Based Schemes. Post-Quantum Cryptography **11505** (2019-07-14)
14. Duman, J., Hövelmanns, K., Kiltz, E., Lyubashevsky, V., Seiler, G., Unruh, D.: A Thorough Treatment of Highly-Efficient NTRU Instantiations. In: Public-Key Cryptography – PKC 2023: 26th IACR International Conference on Practice and Theory of Public-Key Cryptography, Atlanta, GA, USA, May 7–10, 2023, Proceedings, Part I. p. 65–94. Springer-Verlag, Berlin, Heidelberg (2023). https://doi.org/10.1007/978-3-031-31368-4_3
15. Duman, J., Hövelmanns, K., Kiltz, E., Lyubashevsky, V., Seiler, G., Unruh, D.: A Thorough Treatment of Highly-Efficient NTRU Instantiations. In: Public-Key Cryptography – PKC 2023: 26th IACR International Conference on Practice and Theory of Public-Key Cryptography, Atlanta, GA, USA, May 7–10, 2023, Proceedings, Part I. p. 65–94. Springer-Verlag, Berlin, Heidelberg (2023). https://doi.org/10.1007/978-3-031-31368-4_3, https://doi.org/10.1007/978-3-031-31368-4_3
16. D'Anvers, J.P., Karmakar, A., Sinha Roy, S., Vercauteren, F.: Saber: Module-LWR Based Key Exchange, CPA-Secure Encryption and CCA-Secure KEM. pp. 282–305 (04 2018). https://doi.org/10.1007/978-3-319-89339-6_16
17. Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU (2019)
18. Fouque, P.A., Kirchner, P., Pornin, T., Yu, Y.: BAT: Small and fast KEM over NTRU lattices. IACR Transactions on Cryptographic Hardware and Embedded Systems **2022**(2), 240–265 (2022). https://doi.org/10.46586/tches.v2022.i2.240-265
19. Fouque, P., Gérard, F., Rossi, M., Yu, Y.: Zalcon: an alternative fpa-free NTRU sampler for Falcon. (2021)
20. Guo, Q., Johansson, T., Yang, J.: A Novel CCA Attack Using Decryption Errors Against LAC. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology – ASIACRYPT 2019. pp. 82–111. Springer International Publishing, Cham (2019)
21. Heimberger, L., Kales, D., Lolato, R., Mir, O., Ramacher, S., Rechberger, C.: Leap: A Fast, Lattice-Based OPRF with Application to Private Set Intersection. In: Advances in Cryptology – EUROCRYPT 2025: 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4–8, 2025, Proceedings, Part VII. p. 254–283. Springer-Verlag, Berlin, Heidelberg (2025). https://doi.org/10.1007/978-3-031-91098-2_10, https://doi.org/10.1007/978-3-031-91098-2_10
22. Hhan, M., Kim, J., Lee, C., Son, Y.: A hybrid of lattice-reduction and meet-lwe via near-collision on babai's plane: A hybrid of lattice-reduction and meet-lwe... **94**(4) (Apr 2026). https://doi.org/10.1007/s10623-026-01813-1, https://doi.org/10.1007/s10623-026-01813-1
23. Hoffstein, J., Howgrave-Graham, N., Pipher, J., Silverman, J.H., Whyte, W.: NTRUSign: Digital Signatures Using the NTRU Lattice. In: Joye, M. (ed.) Topics in Cryptology — CT-RSA 2003. pp. 122–140. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)
24. Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: A ring-based public key cryptosystem. In: Buhler, J.P. (ed.) Algorithmic Number Theory. pp. 267–288. Springer Berlin Heidelberg, Berlin, Heidelberg (1998)
25. Hou, J., Jiang, H., Ogilvie, T.: Careful with the Ring! Concrete Hardness Gaps Between LWE and MLWE. In: Advances in Cryptology – CRYPTO 2026 (2026)

26. Jiang, H., Zhang, Z., Chen, L., Wang, H., Ma, Z.: IND-CCA-Secure Key Encapsulation Mechanism in the Quantum Random Oracle Model, Revisited. In: CRYPTO (3). Lecture Notes in Computer Science, vol. 10993, pp. 96–125. Springer (2018)
27. Liang, Z., Fang, B., Zheng, J., Zhao, Y.: Compact and efficient kems over ntru lattices. Comput. Stand. Interfaces **89**(C) (Apr 2024). https://doi.org/10.1016/j.csi.2023.103828
28. Liu, F.H., Wang, Z.: Rounding in the Rings. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology – CRYPTO 2020. pp. 296–326. Springer International Publishing, Cham (2020)
29. Liu, Y., Ge, J., Zhang, Y., Wang, J., Lu, X.: A Primer on Dependency in Polynomial Product: Identify, Exploit, and Trim. Cryptology ePrint Archive, Paper 2026/802 (2026), https://eprint.iacr.org/2026/802
30. Liu, Y., Zhang, Y., Lu, X., Cheng, Y., Yin, Y.: DAWN: Smaller and Faster NTRU Encryption via Double Encoding. In: Hanaoka, G., Yang, B.Y. (eds.) Advances in Cryptology – ASIACRYPT 2025. pp. 396–427. Springer Nature Singapore, Singapore (2026)
31. Lu, X., Liu, Y., Zhang, Z., Jia, D., Xue, H., He, J., Li, B., Wang, K.: LAC: Practical Ring-LWE Based Public-Key Encryption with Byte-Level Modulus. Cryptology ePrint Archive, Paper 2018/1009 (2018), https://eprint.iacr.org/2018/1009
32. Lyubashevsky, V., Seiler, G.: NTTRU: Truly Fast NTRU Using NTT. IACR Transactions on Cryptographic Hardware and Embedded Systems **2019**(3), 180–201 (May 2019). https://doi.org/10.13154/tches.v2019.i3.180-201
33. Paiva, T.B., Simplício Jr., M.A., Hafiz, S.M., Yildiz, B., Cominetti, E.L., Ogawa, H.S.: Tailorable codes for lattice-based KEMs with applications to compact ML-KEM instantiations. IACR Transactions on Cryptographic Hardware and Embedded Systems **2025**(3), 139–163 (2025). https://doi.org/10.46586/tches.v2025.i3.139-163
34. Pellet-Mary, A., Stehlé, D.: On the Hardness of the NTRU Problem. In: Tibouchi, M., Wang, H. (eds.) Advances in Cryptology – ASIACRYPT 2021. pp. 3–35. Springer International Publishing, Cham (2021)
35. Pornin, T., Prest, T.: More Efficient Algorithms for the NTRU Key Generation Using the Field Norm. In: Lin, D., Sako, K. (eds.) Public-Key Cryptography – PKC 2019. pp. 504–533. Springer International Publishing, Cham (2019)
36. Prest, T.: Gaussian Sampling in Lattice-Based Cryptography. (2015)
37. Saarinen, M.J.O.: Hila5: On reliability, reconciliation, and error correction for ring-lwe encryption. In: Adams, C., Camenisch, J. (eds.) Selected Areas in Cryptography – SAC 2017. pp. 192–212. Springer International Publishing, Cham (2018)
38. 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/post-quantum-cryptography-standardization/round-1-submissions
39. of Standards, N.I., (U.S.), T.: FIPS-203: Module-lattice-based key-encapsulation mechanism standard (2024). https://doi.org/10.6028/nist.fips.203, https://csrc.nist.gov/pubs/fips/203/final
40. Stehlé, D., Steinfeld, R.: Making NTRU as Secure as Worst-Case Problems over Ideal Lattices. In: Paterson, K.G. (ed.) Advances in Cryptology – EUROCRYPT 2011. pp. 27–47. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
41. Zhang, J., Feng, D., Yan, D.: NEV: Faster and smaller NTRU encryption using vector decoding. In: Guo, J., Steinfeld, R. (eds.) Advances in Cryptology – ASIACRYPT 2023, Part VII. Lecture Notes in Computer Science, vol. 14444,

pp. 157–189. Springer, Singapore, Singapore, Guangzhou, China (Dec 4–8, 2023). https://doi.org/10.1007/978-981-99-8739-9_6
42. Zhang, Y., Lu, X., Liu, Y., Yin, Y., Wang, K.: LEAP: High-Performance Lattice-Based Pseudorandom Number Generator. IACR Transactions on Symmetric Cryptology **2025**(3), 151–182 (Sep 2025). https://doi.org/10.46586/tosc.v2025.i3.151-182, https://tosc.iacr.org/index.php/ToSC/article/view/12468
43. 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/post-quantum-cryptography-standardization/round-1-submissions

A ML-KEM with FCL

For MLWE-based encryptions, their designs already exploit redundancy in the message space by using the ciphertext component carrying the message that has a lower-dimensional ring structure than in typical RLWE-based encryption at the same security level. For example, ML-KEM [39] uses $\mathbb{Z}[x]/(x^{256}+1)$, whereas LAC [31] uses $\mathbb{Z}[x]/(x^{512}+1)$ at NIST-I. As a result, auxiliary quotient information brings a limited compactness gain in the MLWE setting. FCL, however, can still be used as a lightweight post-decryption candidate-localization mechanism paired with hash verification. We therefore give an FCL-based instantiation for ML-KEM. The only changes are in PKE encryption and decryption, described in Algorithms 15–16. We use λ to denote the target security bits, e.g., $\lambda = 128$ for NIST-I. We instantiate Hash using SHA3-256, which is consistent with ML-KEM.

Algorithm 15 FCL-ML-KEM.PKE.Encrypt$(\mathrm{ek}_{\mathrm{PKE}}, m, r)$

Input: encryption key $\mathrm{ek}_{\mathrm{PKE}} \in \mathbb{B}^{384k+32}$
Input: message $m \in \mathbb{B}^{32}$
Input: randomness $r \in \mathbb{B}^{32}$
Output: Ciphertext $c \in \mathbb{B}^{32(d_u k + d_v)+\lambda/8}$

$N \leftarrow 0$
$\hat{\mathbf{t}} \leftarrow \mathsf{ByteDecode}(\mathrm{ek}_{\mathrm{PKE}}[0:384k])$
$\rho \leftarrow \mathrm{ek}_{\mathrm{PKE}}[384k:384k+32]$
for i from 0 to $k-1$ **do**
  for j **from** 0 **to** $k-1$ **do**
    $\hat{\mathbf{A}}[i][j] \leftarrow \mathsf{SampleNTT}(\rho \| j \| i)$
  end for
end for
for i from 0 to $k-1$ **do**
  $\mathbf{y}[i] \leftarrow \mathsf{SamplePolyCBD}_{\eta_1}(\mathsf{PRF}_{\eta_1}(r, N))$
  $N \leftarrow N+1$
end for
for i from 0 to $k-1$ **do**
  $\mathbf{e}_1[i] \leftarrow \mathsf{SamplePolyCBD}_{\eta_2}(\mathsf{PRF}_{\eta_2}(r, N))$
  $N \leftarrow N+1$
end for
$e_2 \leftarrow \mathsf{SamplePolyCBD}_{\eta_2}(\mathsf{PRF}_{\eta_2}(r, N))$
$\hat{\mathbf{y}} \leftarrow \mathsf{NTT}(\mathbf{y})$
$\mathbf{u} \leftarrow \mathsf{NTT}^{-1}(\hat{\mathbf{A}}^T \circ \hat{\mathbf{y}}) + \mathbf{e}_1$
$\mu \leftarrow \mathsf{Decompress}_1(\mathsf{ByteDecode}_1(m))$
$v \leftarrow \mathsf{NTT}^{-1}(\hat{\mathbf{t}}^\top \circ \hat{\mathbf{y}}) + e_2 + \mu$
$c_1 \leftarrow \mathsf{ByteEncode}_{d_u}(\mathsf{Compress}_{d_u}(\mathbf{u}))$
$c_2 \leftarrow \mathsf{ByteEncode}_{d_v}(\mathsf{Compress}_{d_v}(v))$
$h \leftarrow \mathsf{Hash}(m)[0:\lambda/8]$
return $c = (c_1 \| c_2 \| h)$

Algorithm 16 FCL-ML-KEM.PKE.Decrypt($\mathrm{dk}_{\mathrm{PKE}}, c$)

Input: decryption key $\mathrm{dk}_{\mathrm{PKE}} \in \mathbb{B}^{384k}$
Input: ciphertext $c \in \mathbb{B}^{32(d_u k + d_v) + \lambda/8}$
Output: message $m \in \mathbb{B}^{32}$

1: $c_1 \leftarrow c[0 : 32d_u k]$
2: $c_2 \leftarrow c[32d_u k : 32(d_u k + d_v)]$
3: $\mathbf{u}' \leftarrow \mathsf{Decompress}_{d_u}(\mathsf{ByteDecode}_{d_u}(c_1))$
4: $v' \leftarrow \mathsf{Decompress}_{d_v}(\mathsf{ByteDecode}_{d_v}(c_2))$
5: $\hat{\mathbf{s}} \leftarrow \mathsf{ByteDecode}_{12}(\mathrm{dk}_{\mathrm{PKE}})$
6: $w \leftarrow v' - \mathsf{NTT}^{-1}(\hat{\mathbf{s}}^{\top} \circ \mathsf{NTT}(\mathbf{u}'))$
7: $m \leftarrow \mathsf{ByteEncode}_1(\mathsf{Compress}_1(w))$
8: $h \leftarrow c[32(d_u k + d_v) : 32(d_u k + d_v) + \lambda/8]$
9: $\mathcal{S}_{\mathrm{FCL}} \leftarrow \mathsf{FCL}(\mathsf{poly2vec}(w), q, \{0, 1, \ldots, n-1\}, B_0)$
10: error $:= (0, \ldots, 0) \in \mathbb{B}^{32}$
11: **for** i from 0 to $2^{B_0} - 1$
12: error $\leftarrow (0, \ldots, 0) \in \mathbb{B}^{32}$
13: index $:= i$
14: **for** j from 0 to $B_0 - 1$
15: error $[\mathcal{S}_{\mathrm{FCL}}[j]] \leftarrow \lfloor \frac{\mathrm{index}}{2^j} \rfloor \pmod 2$
16: **end for**
17: **if** $\mathsf{Hash}(\mathsf{XOR}(m', \mathrm{error})) = h$ **then**
18: mask $\leftarrow 1$
19: **else**
20: mask $\leftarrow 0$
21: **end if**
22: $m \leftarrow \mathsf{XOR}(m, \mathsf{AND}(-\mathrm{mask}, \mathrm{error}))$
23: **end for**
24: **return** m

The construction keeps the ML-KEM structure unchanged, appends a hash tag computed by a secure hash function Hash in ciphertext, and adds a verification after the usual decryption procedure. We set the FCL output size m_0 equal to B_0 for simplicity. We use XOR and AND for the bit-wise operations. Steps 15–20 of Algorithm 16 implement the constant-time masking used for verification.

From a security perspective, this instantiation is analyzed under the standard coefficient-independence heuristic. This heuristic has only a mild effect in the MLWE setting, but it becomes more significant for NTRU/RLWE-style schemes because of the dependency effects studied in [29]. The hash tag has different implications for the internal PKE and for the final KEM. As a standalone PKE, the tagged construction is no longer IND-CPA secure, whereas the tagged PKE is used only as a one-way primitive for random high-entropy messages. In the KEM setting, the relevant notion is that the encrypted message is sampled uniformly from $\mathbb{B}^{32}$. Under the random-oracle/preimage-resistance heuristic for Hash, the tag gives only an offline equality test for guessed messages, contributing a generic $Q_H/2^{\lambda}$ loss. Therefore, the tag degrades the standalone PKE claim from IND-

CPA to OW-CPA, but it does not degrade the final KEM security target: the standard FO-style transform can still yield an IND-CCA-secure KEM from an OW-CPA internal encryption primitive, assuming the usual correctness and re-encryption checks are included.

From a practical perspective, we implement FCL-ML-KEM at NIST-I security level. Starting from ML-KEM's compression parameters $(d_u, d_v) = (10, 4)$, we reduce them to $(9, 3)$, and apply FCL with $(B_0, B_1) = (2, 1)$ that involves the additional error rate $\Pr[\text{FCL fails}] \leq 2^{-150}$. This reduces the ciphertext size from 768 bytes to 688 bytes, a reduction of about 10%, while increasing the total running time by about 5% in the reference C implementation.

Table 6: Comparison between ML-KEM and FCL-ML-KEM

Scheme	PK (bytes)	CT (bytes)	DFR	Keygen (cycles)	Encaps (cycles)	Decaps (cycles)
ML-KEM-512	800	768	2^{-139}	107, 057	123, 615	161, 918
FCL-ML-KEM-512	800	688	2^{-139}	107, 141	129, 176	178, 146

B RLWE with Encoding

The *RLWE with Encoding* framework mirrors *NTRU with Encoding*, with two differences that are typical in RLWE-PKE design: (i) messages need not "double" as noise for security, so deterministic embeddings (including ECC-style choices) are natural, and (ii) ciphertext component c_2 is often compressed for compactness and balance. We introduce the framework *RLWE with Encoding* as follows.

We keep the same auxiliary rings $\{\mathcal{R}_{\phi_i,p_i}\}_{0\leq i\leq \ell}$ and the same decoding interface (t, w, Decode), and additionally assume t is invertible in $\mathcal{R}_{\varphi,q}$ (since t^{-1} is used to inject the message term). Let $e_{c_2} := \lfloor \frac{q}{d_2} \cdot \lfloor \frac{d_2}{q} \cdot c_2 \rceil \rceil - c_2$ denote the deterministic compression noise. Then the initial decryption expression again has three components:

$$c_0' = t\,(er - e_1 s + e_2 + e_{c_2}) + \overset{\text{noise}}{} \quad \overset{\text{message}}{w\,\mu} \quad - \quad \overset{\text{error}}{q \cdot e} \quad \text{in} \quad \mathcal{R}_\varphi,$$

where $q \cdot e$ captures the wrap-around error incurred when interpreting reductions modulo q in $\mathcal{R}_\varphi$. All correctness/efficiency constraints follow the same template as *NTRU with Encoding*. When there's no extra ECC structure on the message, it's complex to consider the combination of ECC without limitations, and this encoding framework here only gives the result easily extended from NTRU to RLWE. The process is similar to *NTRU with Encoding*, thus we omit it.

For the heuristic frontier of the size, we reuse the same quotient-series for *NTRU with Encoding* in Section 5.1.2 and additionally account for compression

Fig. 6: **Generic RLWE Encryption with Encoding**

KeyGen:

1 : $a \xleftarrow{\$} \mathcal{U}(\mathcal{R}_{\varphi,q})$
2 : $s \xleftarrow{\$} \mathcal{D}_s$
3 : $e \xleftarrow{\$} \mathcal{D}_e$
4 : $b \leftarrow as + e \pmod{\langle q, \varphi \rangle}$
5 : **return** $(pk := (a, b),\ sk := s)$

Decrypt$(c := (c_1, \hat{c}_2),\ sk := s)$:

1 : $c_2 \leftarrow \lfloor \frac{q}{d_2} \cdot \hat{c}_2 \rceil$
2 : $c_0' \leftarrow t\,(c_2 - c_1 s) \pmod{\langle q, \varphi \rangle}$
3 : **for** i **from** 1 **to** ℓ **do**
$c_i' \leftarrow c_0' \pmod{\langle p_i, \phi_i \rangle}$
4 : $m' \leftarrow \mathsf{Decode}_{\mathrm{Enc}}(c_0', t, w, \{(p_i, \phi_i)\}_{1 \le i \le \ell})$
5 : $m \leftarrow \mathsf{Unemb}_{\mathcal{M}' \to \mathcal{M}}(m')$
6 : **return** m

Encrypt$(m,\ pk := (a, b))$:

1 : $r \xleftarrow{\$} \mathcal{D}_r$
2 : $e_1 \xleftarrow{\$} \mathcal{D}_{e_1}$
3 : $e_2 \xleftarrow{\$} \mathcal{D}_{e_2}$
4 : $\mu \leftarrow \mathsf{Emb}_{\mathcal{M} \hookrightarrow \mathcal{S}_\mu}(m)$
5 : $c_1 \leftarrow ar + e_1 \pmod{\langle q, \varphi \rangle}$
6 : $c_2 \leftarrow br + e_2 + t^{-1} w\,\mu \pmod{\langle q, \varphi \rangle}$
7 : $\hat{c}_2 \leftarrow \lfloor \frac{d_2}{q} \cdot c_2 \rceil$
8 : **return** $(c_1, \hat{c}_2)$

noise in c_2 when estimating correctness in the *RLWE with Encoding* framework. We obtain the parameter setting of *RLWE with Encoding* as

$$n = 512, \quad q = 83, \quad d_2 = 11, \quad \varphi = x^n + 1,$$
$$\mathcal{D}_s = \mathcal{D}_e = \mathcal{D}_r = \mathcal{D}_{e_1} = \mathcal{D}_{e_2} = \mathcal{U}(\mathcal{S}_\mu) = \mathcal{T}_{512,30},$$
$$p_1 = 2, \quad \phi_1 = x^{n/2} + 1, \qquad p_2 = 2, \quad \phi_2 = x^{n/4} + 1, \qquad (B_0, B_1) = (4, 1).$$

The estimated components are

$$\Pr_{\text{encoding}}[\text{Dec. fails}] = 0, \qquad \Pr[e \notin \mathcal{E}_{\text{err}}] \le 2^{-150.4}, \qquad \Pr[\text{FCL fails}] \le 2^{-130.9},$$

Additionally, we set a compression of e_{c_2} from $\mathbb{Z}_{83}$ to $\mathbb{Z}_{11}$, and the size of the candidate error set needs to be traversed is $473 < n$. The ciphertext size is 631 bytes, and the public key is 441 bytes, where a is stored as a seed of 16 bytes. We note that some existing RLWE schemes achieve approximate ciphertexts without auxiliary quotient, such as LAC [31], since they use additional ECC structures, which are excluded from our search. We leave the integration of ECC into *RLWE with Encoding* for future work.

C NwE: An Instantiation of NTRU with Encoding

This appendix gives a concrete instantiation of the *NTRU with Encoding* framework. Compared with DAWN [30], the construction uses different auxiliary quo-

tients and an explicit embedding/unembedding pair. We work with the public parameters

$$p_1 = 2, \qquad \phi_1 = x^{n/2} + 1, \qquad p_2 = 3, \qquad \phi_2 = x^{n/2} + x^{n/4} - 1.$$

Following the previous notations, let π_1 and π_2 denote reduction modulo $\langle p_1, \phi_1 \rangle$ and $\langle p_2, \phi_2 \rangle$, respectively.

NwE encodes a λ-bit message into a sparse ternary vector μ. Decryption first computes a residue of the form $w\mu - q \cdot e$, then intersects an auxiliary-derived candidate set with an FCL-derived candidate set. For each surviving candidate error $\tilde{e}$, the decoder computes

$$\mu' \leftarrow w^{-1}(c_1' - \tilde{e}) \pmod{\langle 2, \phi_1 \rangle}$$

and applies a weight check. If exactly one candidate passes this check, NwE.Unembed returns the corresponding message; otherwise, decryption returns $\perp$.

Algorithms 17 and 18 define a randomized embedding of λ-bit strings into the projected message space, together with a deterministic left inverse on the image of the embedding. The maps Bits2Int and Int2Bits are inverse bijections between $\{0,1\}^\lambda$ and $\mathbb{Z}_{2^\lambda}$. The function $\mathsf{UnrankSubset}(N, a, I)$ returns the subset of $[0, N-1]$ of size a with combinatorial rank I, and RankSubset denotes its inverse. Let $N = n/2$. For $\mu \xleftarrow{\$} \mathcal{T}_{n,k_\mu}$, the projection $\pi_1(\mu)$ identifies the two coefficients indexed by i and $i+N$, and $\mathsf{wt}(\pi_1(\mu))$ is the number of pairs $\{i, i+N\}$ in which exactly one of the two positions is nonzero. We precompute the distribution of $\mathsf{wt}(\pi_1(\mu))$ induced by $\mu \leftarrow \mathcal{T}_{n,k_\mu}$, and denote it by $\mathcal{D}_{\mu,\mathrm{res}}$. The embedding samples $\mathsf{wt}(\pi_1(\mu))$ from this distribution, subject to the admissibility conditions

$$\mathsf{wt}(\pi_1(\mu)) \le K_{\mathrm{ver}}, \qquad 2^\lambda \le \binom{N}{\mathsf{wt}(\pi_1(\mu))}, \qquad \mathsf{wt}(\pi_1(\mu)) \equiv 0 \pmod 2.$$

The last condition ensures that the remaining nonzero positions can be placed in colliding pairs. Rejection sampling of this form is common in lattice-based constructions, for example in the key-generation checks of Falcon [17] and BAT [18]. In the present setting, however, the rejection also changes the message distribution relative to the unconditioned target distribution $\mathcal{T}_{n,k_\mu}$. We therefore account for the resulting statistical distance explicitly.

Lemma 3 (Embedding statistical distance). *Let $N = n/2$, $k = k_\mu$, and let $\mathsf{wt}(\pi_1(\mu))$ be the projected weight induced by $\mu \xleftarrow{\$} \mathcal{T}_{n,k}$. For feasible a, define*

$$p_a := \Pr[\mathsf{wt}(\pi_1(\mu)) = a] = \frac{\binom{N}{a} 2^a \binom{N-a}{(2k-a)/2}}{\binom{2N}{2k}}, \qquad C_a := \binom{N}{a}, \qquad S_a := \left\lfloor C_a 2^{-\lambda} \right\rfloor.$$

Let $\mathcal{W}$ be the admissible set of projected weights used by NwE.Embed, and let

$$\rho := 1 - \sum_{a \in \mathcal{W}} p_a$$

be the rejection probability with respect to the unconditioned target distribution $\mathcal{T}_{n,k}$*. Assume that, conditioned on* $\mathsf{wt}(\pi_1(\mu)) = a$*, the embedding samples according to the conditional distribution induced by* $\mathcal{T}_{n,k}$*, except that the projected-support rank is restricted to the first* $2^\lambda S_a$ *ranks.*

Then, for $m \leftarrow \mathcal{U}(\{0,1\}^\lambda)$*, the statistical distance between the output distribution of* $\mathsf{NwE.Embed}(m)$ *and the unconditioned target distribution* $\mathcal{T}_{n,k}$ *satisfies*

$$\varepsilon_{\mathrm{emb}} \geq \rho,$$

and more precisely is bounded by

$$\varepsilon_{\mathrm{emb}} \leq \rho + \sum_{a \in \mathcal{W}} p_a \left(1 - \frac{2^\lambda S_a}{C_a}\right) = \rho + \sum_{a \in \mathcal{W}} p_a \frac{C_a \bmod 2^\lambda}{C_a}.$$

In particular,

$$\varepsilon_{\mathrm{emb}} < \rho + \sum_{a \in \mathcal{W}} p_a \frac{2^\lambda}{\binom{N}{a}}.$$

If the target distribution is instead defined as $\mathcal{T}_{n,k}$ *conditioned on* $\mathsf{wt}(\pi_1(\mu)) \in \mathcal{W}$*, then the rejection term is removed and the same rank-truncation bound holds with* p_a *replaced by* $p_a/(1-\rho)$*.*

For the suggested NwE parameters, the rejection probability is small but not λ-level negligible. In NwE-512 we have $\rho \approx 2^{-50}$, and in NwE-1024 we have $\rho \approx 2^{-67}$. Since the embedding never outputs samples with $\mathsf{wt}(\pi_1(\mu)) \notin \mathcal{W}$, the statistical distance from the unconditioned target distribution is at least ρ. Thus, the embedding is not $2^{-\lambda}$-close to $\mathcal{T}_{n,k_\mu}$ when $\lambda = 128$. For this reason, NwE should be viewed as a proof-of-concept instantiation of the NTRU-with-Encoding framework rather than as a construction with a standard distributional security reduction. Its concrete security should be interpreted heuristically, relative to the conditioned message distribution and to the checked decoder candidate family.

Algorithms 19 and 20 give key generation and encryption. Algorithm 21 gives the corresponding decoder. The decoder first constructs an auxiliary-derived set of admissible error residues. It then constructs an FCL-derived set of error residues over the reduced index set, intersects the two sets, and finally applies the residue-weight verification predicate. For both NwE-512 and NwE-1024, we choose

$$t = x^n - 2x^{n/2} - 1.$$

The polynomial w is chosen separately for the two parameter sets in order to suppress decoding failures. For NwE-512, we use

$$w = x^{256} + 3x^{159} + x^{128} + 3x^{97} - 1,$$

and for NwE-1024, we use

$$w = x^{512} + 3x^{455} + 3x^{397} + x^{256} - 1.$$

In both cases, $w \equiv 0 \pmod{\langle 3, \phi_2 \rangle}$. Hence the auxiliary quotient removes the message term and exposes a constraint on the error residue.

The verification threshold is denoted by K_{ver}. In the parameter search, the false-accept term is checked by evaluating

$$\Delta_{\mathrm{ver}} := \min_{\delta \in \Delta_{\mathrm{false}}} \mathsf{wt}\big(\mathsf{poly2vec}\left(w^{-1}\pi_1(\delta)\right)\big),$$

where Δ_{false} is the nonzero difference set between the true error residue and the false candidate residues considered by the decoder. The checked values satisfy

$$\Delta_{\mathrm{ver}} \geq 171 \quad \text{for NwE-512}, \qquad \Delta_{\mathrm{ver}} \geq 261 \quad \text{for NwE-1024}.$$

With $K_{\mathrm{ver}} = 80$ and $K_{\mathrm{ver}} = 128$, respectively, this gives $\Delta_{\mathrm{ver}} > 2K_{\mathrm{ver}}$. Therefore, within the checked candidate family, the residue-weight verification test admits no false acceptances, and the corresponding false-accept term is $\varepsilon_{\mathrm{fa}} = 0$.

Table 7: Suggested parameters for NwE.

Scheme	n	q	w	(k_g, k_f)	(k_s, k_μ)	(B_0, B_1)	K_{ver}	PK/CT
NwE-512	512	193	$x^{256} + 3x^{159} + x^{128} + 3x^{97} - 1$	$(40, 40)$	$(40, 40)$	$(1, 1)$	80	501/501
NwE-1024	1024	193	$x^{512} + 3x^{455} + 3x^{397} + x^{256} - 1$	$(72, 72)$	$(80, 64)$	$(1, 1)$	128	1000/1000

Table 8: DFR estimates and security bits for NwE.

Scheme	$\log_2 \Pr[\lfloor e \rceil \notin \mathcal{E}_{\mathrm{err}}]$	$\log_2 \Pr[\text{FCL fails}]$	$\log_2 \Pr[\text{Dec. fails}]$	$\log_2(\text{DFR})$	$\log_2(\text{Sec. bits})$
NwE-512	-250	-251	$-\infty$	-249	132
NwE-1024	-152	-156	$-\infty$	-152	257

Algorithm 17 NwE.Embed

Input: message $m \in \{0,1\}^\lambda$
Output: $\mu \in \{-1,0,1\}^n$ with exactly k_μ coefficients $+1$ and k_μ coefficients -1

repeat
  $a \leftarrow \mathcal{D}_{\mu,\mathrm{res}}$
until $a \le K_{\mathrm{ver}}$, $2^\lambda \le \binom{n/2}{a}$, and $2k_\mu - a$ is even
$M \leftarrow \mathsf{Bits2Int}(m)$
$k_{\mu,\mathrm{col}} \leftarrow (2k_\mu - a)/2$
$S \leftarrow \left\lfloor \binom{n/2}{a} 2^{-\lambda} \right\rfloor$
$u \xleftarrow{\$} \mathcal{U}(\mathbb{Z}_S)$
$I \leftarrow u \cdot 2^\lambda + M$
$O \leftarrow \mathsf{UnrankSubset}(n/2, a, I)$ ▷ $O \subseteq [0, n/2-1]$, $|O| = a$
$E \leftarrow [0, n/2-1] \setminus O$
$T \xleftarrow{\$} \binom{E}{k_{\mu,\mathrm{col}}}$
$\mathcal{S} \leftarrow \varnothing$
for $i \in O$ **do**
  $b \xleftarrow{\$} \mathcal{U}(\{0,1\})$
  if $b = 0$ **then**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{i\}$
  else
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{i + n/2\}$
  end if
end for
for $i \in T$ **do**
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{i, i + n/2\}$
end for
$P \xleftarrow{\$} \binom{\mathcal{S}}{k_\mu}$
$\mu \leftarrow \mathbf{0} \in \mathbb{Z}^n$
for $j \in \mathcal{S}$ **do**
  if $j \in P$ **then**
    $\mu_{[j]} \leftarrow 1$
  else
    $\mu_{[j]} \leftarrow -1$
  end if
end for
return μ

Algorithm 18 NwE.Unembed

Input: residue $\mu_{\text{res}} \in \mathbb{Z}[x]/\langle 2, x^{n/2}+1\rangle$
Output: message $m \in \{0,1\}^{\lambda}$ or $\perp$
1: $O \leftarrow \{i \in [0, n/2-1] : (\mu_{\text{res}})_{[i]} = 1\}$
2: **if** $|O|$ is not an admissible residue weight **then**
3: **return** $\perp$
4: **end if**
5: $I \leftarrow \mathsf{RankSubset}(n/2, |O|, O)$
6: $M \leftarrow I \bmod 2^{\lambda}$
7: **return** $\mathsf{Int2Bits}_{\lambda}(M)$

Algorithm 19 NwE.PKE.KeyGen

Output: public key pk, private key sk
1: **repeat**
2: $g \xleftarrow{\$} \mathcal{T}_{n,k_g}$
3: $f \xleftarrow{\$} \mathcal{T}_{n,k_f}$
4: $f' \leftarrow tw^{-1}f + 1 \pmod{\langle q, \varphi\rangle}$
5: **until** f' is invertible in $\mathcal{R}_{\varphi,q}$ and all parameter-specific key checks pass
6: $h \leftarrow g(f')^{-1} \pmod{\langle q, \varphi\rangle}$
7: **return** $(pk := h, sk := f')$

Algorithm 20 NwE.PKE.Encrypt

Input: public key $pk := h$, message m
Output: ciphertext c
1: $\mu \leftarrow \mathsf{NwE.Embed}(m)$
2: $s \xleftarrow{\$} \mathcal{T}_{n,k_s}$
3: $c \leftarrow tw^{-1}hs + \mu \pmod{\langle q, \varphi\rangle}$
4: **return** c

Algorithm 21 NwE.PKE.Decrypt

Input: private key $sk := f'$, ciphertext c
Output: message m or $\perp$

$c'_0 \leftarrow wf'c \pmod{\langle q, \varphi \rangle}$
$c'_1 \leftarrow \pi_1(c'_0)$
$c'_2 \leftarrow \pi_2(c'_0)$
$\mathcal{S}_{\mathrm{err,aux,res}} \leftarrow \{\pi_1(-q \cdot e) : e \in \mathcal{E}_{B_0,B_1} \text{ and } \pi_2(-q \cdot e) = c'_2\}$
$\mathcal{I}_{\mathrm{err}} \leftarrow \bigcup_{\tilde{e} \in \mathcal{S}_{\mathrm{err,aux,res}}} \{i : (\tilde{e})_{[i]} \neq 0\}$
$\mathcal{S}_{\mathrm{err,FCL,res}} \leftarrow \bigcup_{j=0}^{B_0} \left\{ \pi_1(-q \cdot \tilde{e}) : \tilde{e} \in \mathcal{E}_{\mathrm{cand}}^{\lambda_{\mathrm{DFR}}} \left(\mathsf{poly2vec}(c'_0), q, m_0, j, B_1, \mathcal{I}_{\mathrm{err}} \right) \right\}$
$\mathcal{S}_{\mathrm{err,res}} \leftarrow \mathcal{S}_{\mathrm{err,aux,res}} \cap \mathcal{S}_{\mathrm{err,FCL,res}}$
$\mathcal{M}_{\mathrm{acc}} \leftarrow \varnothing$
for $\tilde{e}^\star \in \mathcal{S}_{\mathrm{err,res}}$ **do**
    $\mu' \leftarrow w^{-1}(c'_1 - \tilde{e}^\star) \pmod{\langle 2, \phi_1 \rangle}$
    if $\mathsf{wt}(\mathsf{poly2vec}(\tilde{\mu})) \leq K_{\mathrm{ver}}$ **then**
        $\tilde{m} \leftarrow \mathsf{NwE.Unembed}(\tilde{\mu})$
        $\mathcal{M}_{\mathrm{acc}} \leftarrow \mathcal{M}_{\mathrm{acc}} \cup \{\tilde{m}\}$
    end if
end for
if $|\mathcal{M}_{\mathrm{acc}}| = 1$ **then**
    return the unique element of $\mathcal{M}_{\mathrm{acc}}$
else
    return $\perp$
end if

D Security Proof of END

This appendix proves the security of the END KEM from Section 6. The proof uses two assumptions. The public key is handled by a decision NTRU assumption. The ciphertext part of the deterministic encryption layer is handled by a search Ring-LWR assumption for the compressed NTRU sample. Correctness is treated separately through the DFR estimates in Table 3. The encoding-framework case study follows the same proof pattern as DAWN-style NTRU encryption and is not the main KEM theorem.

D.1 Hardness Assumption

Decision NTRU assumption. Let $\mathcal{D}$ be a distribution over $\mathcal{R}_\varphi$. Sample $f, g \leftarrow \mathcal{D} \cap \mathcal{R}^\times_{\varphi,q}$, and sample $u \leftarrow \mathcal{U}(\mathcal{R}^\times_{\varphi,q})$. The decision NTRU advantage of an adversary $\mathcal{A}$ is

$$\mathrm{Adv}^{\mathrm{NTRU}}_{\mathcal{R}_\varphi,q,\mathcal{D}}(\mathcal{A}) = \left|\Pr[\mathcal{A}(gf^{-1}) = 1] - \Pr[\mathcal{A}(u) = 1]\right|.$$

We assume this advantage is at most ϵ_{NTRU} for all adversaries running in time t_{NTRU}. The decision NTRU assumption is called $(\epsilon_{\mathsf{NTRU}}, t_{\mathsf{NTRU}})$-hard if for any adversary with time t_{NTRU}, we have $\mathrm{Adv}^{\mathrm{NTRU}}_{\mathcal{R}_\varphi,q,\mathcal{X}}(\mathcal{A}) \leq \epsilon_{\mathsf{NTRU}}$.

For the case we consider, with $\varphi = x^n + 1$, where $n = 2^l$ and $l \in \mathbb{N}$, and $\mathcal{D} = \mathcal{T}_{n,k_g}$, there has been considerable research on the hardness of this type of decision NTRU problem. In a power of 2 cyclotomic ring, the work in [40] demonstrated that when $\mathcal{X}$ is a discrete Gaussian distribution with a standard deviation slightly greater than $\sqrt{q}$, the advantage of distinguishing is exponentially small. Furthermore, [34] shows that a variant of the search NTRU problem, which restricts $||g||, ||f||$ to approximate $\sqrt{q}$, is at least as hard as the worst-case approximate Shortest Vector Problem on ideal lattices, offering a solid foundation. The decision NTRU problem with narrow distribution is also referred to as the decisional small polynomial ratio (DSPR) assumption, which has been widely used and investigated in [17], [41], [18], [30].

Search Ring-LWR assumption. Let $a \leftarrow \mathcal{U}(\mathcal{R}^\times_{\varphi,q})$ and $s \leftarrow \mathcal{D} \cap \mathcal{R}^\times_{\varphi,q}$. Let $\lfloor \frac{p}{q} \cdot as \rceil$ denote the public rounding used in the DPKE encryption algorithm. The search Ring-LWR advantage of an adversary $\mathcal{A}$ is

$$\mathrm{Adv}^{\mathrm{RLWR}}_{\mathcal{R}_\varphi,q,p,\mathcal{D}}(\mathcal{A}) = \Pr[\mathcal{A}(a, \lfloor \tfrac{p}{q} \cdot as \rceil) = s].$$

We assume this advantage is at most ϵ_{RLWR} for all adversaries running in time t_{RLWR}. The search Ring-LWR assumption is called $(\epsilon_{\mathsf{RLWR}}, t_{\mathsf{RLWR}})$-hard if for any adversary with time t_{RLWR}, we have $\mathrm{Adv}^{\mathrm{RLWR}}_{\mathcal{R}_\varphi,q,k,\mathcal{D}}(\mathcal{A}) \leq \epsilon_{\mathsf{RLWR}}$.

For the case we consider, with $\varphi = x^n + 1$, where $n = 2^l, l \in \mathbb{N}$, $p = d_e$, and $\mathcal{D} = \mathcal{U}(\mathbb{Z}_2)$. The Ring-LWR problem was first proposed in [3], [11] presented

a reduction from decision Ring-LWE to a variant called Computational Ring-LWR. [28] proposed a reduction from search Ring-LWE to search Ring-LWR, which provides confidence in the security of Ring-LWR. The Ring-LWR with narrow secret distributions has been used in many practical schemes, such as SPRING [2], Saber [16], BAT [18], Leap OPRF [21], and LEAP PRG [42].

D.2 KEM Security

We first clarify the definitions of primitives and transformations as follows.

D.2.1 Primitives

Definition 3 (PKE). *A PKE scheme* $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *consists of a triple of polynomial-time algorithms with the security parameter* λ.

- Gen *is a key generation algorithm. It is probabilistic and outputs a public-secret key pair* (pk, sk) *on input* 1^λ.
- Enc *is a probabilistic encryption algorithm. On input* pk*, a message* m*, it outputs a ciphertext* c*. Alternatively, the* Enc *algorithm can also take an explicit randomness value* r *as an additional input.*
- Dec *is a deterministic decryption algorithm. On input* sk *and a ciphertext* c*, it outputs either a message* m *or a rejection symbol* $\perp$.

We define the following correctness and security notions:

- ***Correctness Error.*** *A PKE scheme is said to be* δ-average-case correct *if*

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

where the expectation is over $(pk, sk) \leftarrow \mathsf{Gen}$*, the randomness of* Enc*, and* $m \xleftarrow{\$} \mathcal{M}$*, with* $\mathcal{M}$ *being the message space.*
The scheme is δ-worst-case correct *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 over $(pk, sk) \leftarrow \mathsf{Gen}$ *and the randomness of* Enc.
- $\mathsf{OW\text{-}CPA}$ ***Security.*** *A PKE scheme is* $\mathsf{OW\text{-}CPA}$ *secure (one-way under chosen-plaintext attacks) if, for any PPT (quantum) adversary* $\mathcal{A}$*, the advantage*

$$\mathrm{Adv}^{\mathrm{OW\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A}) := \Pr\left[m' = m^* \;\middle|\; \begin{array}{l} (sk, pk) \xleftarrow{\$} \mathsf{Gen}, \\ m^* \xleftarrow{\$} \mathcal{M}, \\ c^* = \mathsf{Enc}(pk, m^*), \\ m' \leftarrow \mathcal{A}(pk, c^*) \end{array}\right]$$

is negligible.

A PKE scheme is called a DPKE *scheme if the encryption algorithm* Enc *is deterministic and uses no randomness (i.e., the randomness space is empty). The notions of* OW-CPA *security and δ-average-case correctness also apply to DPKE schemes.*

Definition 4 (KEM). *A KEM scheme* $\mathsf{KEM} = (\mathsf{KGen}, \mathsf{Encaps}, \mathsf{Decaps})$ *consists of a triple of polynomial-time algorithms with the security parameter λ.*

- KGen *is a key generation algorithm. It is probabilistic and, on input 1^λ, outputs a public/secret key pair (pk, sk).*
- Encaps *is a probabilistic encapsulation algorithm. On input pk, it outputs a pair (c, K), where c is a ciphertext and K is a key.*
- Decaps *is a deterministic decapsulation algorithm. On input (sk, c), it outputs a key K.*

We define the following correctness and security notion:

- ***Correctness Error.*** *The KEM scheme is called δ-correct if:*

$$\mathbb{E}\left[\Pr[\mathsf{Decaps}(sk, c) \neq K, (c, K) \leftarrow \mathsf{Encaps}(pk)\right] \leq \delta,$$

  where the expectation is over $(pk, sk) \leftarrow \mathsf{KGen}$ and the randomness of Encaps.
- IND-CCA ***Security.*** *The KEM scheme is* IND-CCA *secure if for any PPT (quantum) adversary $\mathcal{A}$, the advantage*

$$\mathrm{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}}(\mathcal{A}) := \left| \Pr\left[b' = b \;\middle|\; \begin{array}{l} (pk, sk) \leftarrow \mathsf{KGen}(1^\lambda), \\ (c^*, K_0^*) \leftarrow \mathsf{Encaps}(pk), \\ K_1^* \xleftarrow{\$} \mathcal{K}, \\ b \xleftarrow{\$} \{0,1\}, \\ b' \leftarrow \mathcal{A}^{O_{\mathsf{Decaps}}(sk, \cdot \neq c^*)}(pk, c^*, K_b^*) \end{array} \right] - \frac{1}{2} \right|$$

  is negligible. The oracle $O_{\mathsf{Decaps}}(sk, \cdot \neq c^*)$ returns $\mathsf{Decaps}(sk, c)$ for all $c \neq c^*$.

D.2.2 Transformations

Transformation ACWC: Let $\mathsf{DPKE} = (\mathsf{Gen}_0, \mathsf{Enc}_0, \mathsf{Dec}_0)$ be a DPKE scheme with message space $\mathcal{M}' = \mathcal{M} \times \mathcal{M}$. Let $\mathbf{H}_1 : \mathcal{M} \to \mathcal{M}$ be a random oracle. The algorithms of the resulting PKE scheme $\mathsf{PKE} := \mathsf{ACWC}[\mathsf{DPKE}, \mathbf{H}_1] = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ are defined in Figure7.

Transformation $\mathsf{FO}^\not\perp$: Let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a PKE scheme with message space $\mathcal{M}$, and randomness space $\mathcal{S}_0$. The algorithms of the KEM scheme $\mathsf{KEM} := \mathsf{FO}^\not\perp[\mathsf{PKE}, \mathbf{H}_2, \mathbf{H}_3] = (\mathsf{KGen}, \mathsf{Encaps}, \mathsf{Decaps})$ are defined in Figure 8, where $\mathbf{H}_2 : \{0,1\}^* \to \mathcal{S}_0, \mathbf{H}_3 : \{0,1\}^* \to \mathcal{S}_1$ are random oracles, and $\mathcal{S}_1$ denotes the key space of the KEM scheme.

The IND-CCA security proof for END.KEM proceeds in three steps:

Gen:	$\mathsf{Enc}(pk, m, r)$:	$\mathsf{Dec}(sk, c)$:
1 : $(pk, sk) \xleftarrow{\$} \mathsf{Gen}_0$	1 : $c \leftarrow \mathsf{Enc}_0(pk, r \Vert (m \oplus \mathbf{H}_1(r)))$	1 : $r \Vert r' := \mathsf{Dec}_0(sk', c)$
2 : **return** (pk, sk)	2 : **return** c	2 : $m = r' \oplus \mathbf{H}_1(r)$
		3 : **return** m

Fig. 7: Transformation ACWC

KGen:	$\mathsf{Encaps}(pk)$:	$\mathsf{Decaps}(sk, c)$:
1 : $(pk, sk') \xleftarrow{\$} \mathsf{Gen}$	1 : $m \xleftarrow{\$} \mathcal{M}$	1 : Parse $sk = (k, pk, sk')$
2 : $k \xleftarrow{\$} \mathcal{M}, sk := (k, pk, sk')$	2 : $c \leftarrow \mathsf{Enc}(pk, m, \mathbf{H}_2(m))$	2 : $m' := \mathsf{Dec}(sk', c)$
3 : **return** (pk, sk)	3 : $K := \mathbf{H}_3(m, c)$	3 : **if** $c = \mathsf{Enc}(pk, m', \mathbf{H}_2(m')))$
	4 : **return** (c, K)	4 : **return** $K := \mathbf{H}_3(m', c)$
		5 : **else return** $K := \mathbf{H}_3(k, c)$

Fig. 8: Transformation $\mathsf{FO}^{\not\perp}$

- Step 1. We reduce the OW-CPA security of END.DPKE to hardness assumptions, as formalized in Lemma 4.
- Step 2. We derive the worst-case correctness of END.PKE from the average-case correctness of END.DPKE, following Lemma 5. We clarify that the average-case correctness parameter δ of END.DPKE discussed here corresponds to the decryption failure probability analyzed in Section 5.1.2, and the specific values of δ are provided in Table 3. Additionally, we reduce the OW-CPA security of END.PKE to that of END.DPKE, as shown in Lemma 6.
- Step 3. Finally, we reduce the IND-CCA security of END.KEM to the OW-CPA security of END.PKE. The loss in this reduction is related to the worst-case correctness of END.PKE, as discussed in Lemma 7.

Lemma 4 (OW-CPA security of END.DPKE). *Assume decision NTRU is* $(\epsilon_{\mathsf{NTRU}}, t_{\mathsf{NTRU}})$*-hard and search Ring-LWR is* $(\epsilon_{\mathsf{RLWR}}, t_{\mathsf{RLWR}})$*-hard for the parameters of END. Then any adversary against END.DPKE running in the corresponding reduction time has advantage at most* $\epsilon_{\mathsf{NTRU}} + \epsilon_{\mathsf{RLWR}}$*.*

Proof. Let $\mathcal{F}$ be an adversary against the DPKE scheme. The adversary $\mathcal{F}$ can obtain the public key pk and make encryption queries. We replace pk with $\mathbf{a}$, sampled from $\mathcal{U}(\mathcal{R}^{\times}_{\varphi,q})$. According to the decision NTRU assumption, the adversary $\mathcal{F}$ cannot detect this modification. We then use $\mathcal{F}$ to break the search Ring-LWR assumption. For the search Ring-LWR challenge, we give $\mathcal{F}$ the modified public key $\mathbf{a}$ and the challenge. We answer $\mathcal{F}$'s encryption queries according to the procedure in Algorithm 8. If $\mathcal{F}$ succeeds, we have broken the search Ring-LWR assumption.

Lemma 5 ([15]DPKE δ-*average-case-correct* $\Rightarrow$ ACWC[DPKE, $\mathbf{H}_1$] δ'-*worst-case-correct*). *Let* DPKE *be a DPKE scheme that is* δ*-average-case-correct with*

message space $\mathcal{M}' = \mathcal{M} \times \mathcal{M}$*, where* $\psi_{\mathcal{M}'} = \psi_{\mathcal{M}} \times \psi_{\mathcal{M}}$ *is a product distribution, and let* $\|\psi_{\mathcal{M}}\| := \sqrt{\sum_{m \in M} \psi_M(m)^2}$*. Let* $\mathbf{H}_1 : \mathcal{M} \to \mathcal{M}$ *be a random oracle. If* DPKE *is* δ*-average-case-correct, then* PKE $:=$ ACWC[DPKE, $\boldsymbol{H}_1$] *is* δ'*-worst-case-correct, where* ACWC *is defined in Figure 7, and*

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

Proof. The proof follows Lemma 3.6 in [15]. The two lemmas are identical, with the only difference being that Lemma 3.6 uses a PKE scheme as the underlying component, while we use a DPKE scheme here. Since the δ-average-case-correct is well-defined for both PKE and DPKE, the only difference is that the DPKE scheme does not involve randomness. By treating the random value in the PKE scheme from Lemma 3.6 as a constant, we naturally obtain the result for DPKE.

For END-512, $\delta_{\text{ac}} < 2^{-130}$ from Table 3 gives $\delta_{\text{wc}} < 2^{-124.3}$ through Lemma 5; for END-1024, the corresponding value is essentially 2^{-157}.

Lemma 6 ([15] DPKE OW-CPA $\overset{\text{QROM}}{\Rightarrow}$ ACWC[DPKE, $\mathbf{H}_1$] OW-CPA). *Let* $\mathbf{H}_1 : \mathcal{M} \to \mathcal{M}$ *be a random oracle. Let* $\mathcal{A}$ *be a* OW-CPA *adversary against* PKE $:=$ ACWC[DPKE, $\boldsymbol{H}_1$] *making* q_1 *quantum random oracle queries to* $\mathbf{H}_1$*, where* ACWC *is defined in Figure 7. Then we can construct a* OW-CPA *adversary* $\mathcal{B}$ *against* DPKE*, such that:*

$$\mathrm{Adv}^{\text{OW-CPA}}_{\text{PKE}}(\mathcal{A}) \leq (2q_1 + 1)^2 \mathrm{Adv}^{\text{OW-CPA}}_{\text{DPKE}}(\mathcal{B}).$$

Proof. Lemma 6 can be viewed as a special case of Theorem 3.10 in [15], where the underlying component uses a DPKE scheme instead of a general PKE scheme, and the generalized one-time pad (GOTP) algorithm uses the $\oplus$ operation. We now explain why the case in which the underlying component is a DPKE scheme can be regarded as a special case of one using a PKE scheme. Specifically, the ACWC transformation does not interact with the randomness used by the encryption algorithm of the underlying PKE when invoking it. Thus, the encryption randomness is independent of the rest of the transformation. As a result, a DPKE scheme can be seen as a special case of a PKE scheme where the randomness space is empty. This distinction does not affect the reduction used in the proof of the lemma.

Lemma 7 ([26]PKE OW-CPA $\overset{\text{QROM}}{\Rightarrow}$ FO$^{\not\perp}$[PKE, $\mathbf{H}_2$, $\mathbf{H}_3$] IND-CCA). *Let* PKE *be a PKE scheme that is* δ'*-worst-case-correct with message space* $\mathcal{M}$*. Let* $\mathbf{H}_2 : \{0,1\}^* \to \mathcal{R}$*,* $\mathbf{H}_3 : \{0,1\}^* \to \mathcal{K}$ *be random oracles. Let* $\mathcal{A}$ *be an* IND-CCA *adversary against* KEM $=$ FO$^{\not\perp}$[PKE, $\mathbf{H}_2$, $\mathbf{H}_3$] *making* q_2 *quantum queries to* $\mathbf{H}_2$*,* q_3 *quantum queries to* $\mathbf{H}_3$*, and* q_D *classical queries to the decapsulation oracles* Decaps*, where* FO$^{\not\perp}$ *is defined in Figure 8. Then we can construct a* OW-CPA *adversary* $\mathcal{B}$ *against* PKE*, such that:*

$$\mathrm{Adv}^{\text{IND-CCA}}_{\text{KEM}}(\mathcal{A}) \leq 2(q_2 + q_3)\sqrt{\mathrm{Adv}^{\text{OW-CPA}}_{\text{PKE}}(\mathcal{B})} + \frac{2q_3}{\sqrt{|\mathcal{M}|}} + 4q_3\sqrt{\delta'}.$$

Fixing $q_1 = q_2 = q_3 = 2^{16}$, the current END-512 reduction-level IND-CCA bound is about $2^{-43.9}$. Under the same query limit, the corresponding bounds are about $2^{-34.4}$ for Kyber-512, $2^{-50.2}$ for NEV-512, and $2^{-46.5}$ for DAWN-α-512. These are reduction-level bounds and are distinct from the concrete lattice-security estimates in Table 4. A tighter FOAC$'$ reduction [10] uses average-case DFR directly. After adding the simple check required to match END to FOAC$'$, the corresponding END-512 bound becomes about $2^{-44.6}$.