

NTPIR: Efficient Silent-Preprocessing PIR using NTRU

Zhihao Li
Ant Group
Beijing, China
lzh458070@antgroup.com

Cheng Hong*
Ant Group
Beijing, China
vince.hc@antgroup.com

Abstract

Private Information Retrieval (PIR) enables a client to query a database without revealing the requested index. In the silent preprocessing model, existing protocols typically adopt a two-layer pipeline: a first-layer retrieval based on LWE, followed by an RLWE ring-packing stage and a second-layer homomorphic column retrieval, with ring packing being the primary performance bottleneck. We observe that NTRU supports the homomorphic operations required by this pipeline while representing each ciphertext with a single ring element, suggesting its potential to replace RLWE for a more compact intermediate representation. However, NTRU lacks RLWE's public randomness component, which is required in the offline/online separation of most existing PIR schemes.

We introduce NTPIR to address this challenge through three techniques: (1) Pack-then-switch: we perform ring packing in the RLWE domain and transform the result to NTRU via a single key switch, preserving the offline preprocessing structure while yielding compact, single-element packed intermediates. (2) Split-point picking: this method integrates second-layer column selection with response packing, reducing automorphism complexity from linear to square-root complexity in the number of packed second-layer inputs. (3) Optimized arithmetic: we employ FFT-based polynomial multiplication and approximate gadget decomposition to improve the computational efficiency of the underlying homomorphic operations, with concrete correctness error at most 2^{-40} .

On databases ranging from 1 to 8 GB, NTPIR achieves up to a $1.81\times$ server-side online speedup over InsPIRe⁽²⁾ (S&P'26). At 8 GB, preprocessing step is $7\text{--}41\times$ faster than InsPIRe⁽²⁾. In addition, at the highest-throughput configuration, NTPIR sustains 8,609 MB/s on an 8 GB database, corresponding to a $1.38\times$ speedup over InsPIRe⁽²⁾, $1.8\times$ over SimpleYPIR, and $2.0\times$ over HintlessPIR.

CCS Concepts

• Security and privacy → Cryptography.

Keywords

Private Information Retrieval, Silent Preprocessing, NTRU, Split-point Picking

*Corresponding author.



This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

CCS '26, The Hague, Netherlands

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2871-6/2026/11

<https://doi.org/10.1145/3830454.3846680>

ACM Reference Format:

Zhihao Li and Cheng Hong. 2026. NTPIR: Efficient Silent-Preprocessing PIR using NTRU. In *Proceedings of the 2026 ACM SIGSAC Conference on Computer and Communications Security (CCS '26)*, November 15–19, 2026, The Hague, Netherlands. ACM, New York, NY, USA, 22 pages. <https://doi.org/10.1145/3830454.3846680>

1 Introduction

Private Information Retrieval (PIR) enables a client to fetch an entry from a server-held database without revealing the queried index. Since its introduction by Chor et al. [10], PIR has evolved from a theoretical curiosity into a practical cryptographic primitive with applications spanning private contact discovery [18], password breach checking [24], certificate transparency auditing [21], private advertising delivery [4], and private media consumption [14]. In each of these scenarios, the server must remain oblivious to the client's query index while still returning the correct database entry.

PIR has been explored in both multi-server settings (e.g., [10, 13]) and the single-server setting [20]. Although multi-server PIR schemes can be more efficient, they rely on non-collusion assumptions that are difficult to enforce in practice. Among single-server PIR schemes, we focus on the *client-stateless and server-stateless* setting. To place this setting in context, we follow the taxonomy in [3] and classify these schemes according to whether state is maintained by the client or the server and whether it is shared across clients. This gives the following four categories.

Client-stateless schemes require no database-dependent state at the client. The server performs all database preprocessing and handles subsequent updates. These schemes further differ in whether the server retains client-specific state:

- *Server-stateful* (e.g., SealPIR [2], MulPIR [1], Spiral [30], Respire [8]): during setup, each client uploads client-specific cryptographic material, typically FHE evaluation keys and key-switching hints. The server stores this material and uses it for per-query homomorphic operations over the database. It therefore maintains a separate key set for every active client and must associate each query with the corresponding keys.
- *Server-stateless* (e.g., HintlessPIR [23], YPIR [31], InsPIRe [28], KSPIR [26]): the server stores no client-specific material. All clients use the same preprocessed database, while each query carries the necessary key-switching keys. This allows the public, message-independent components of the cryptographic operations to be precomputed offline, independently of the client and query.

Client-stateful schemes require the client to obtain and store a database-dependent *hint* before issuing a query. The hint reduces online server computation or communication. Any database update invalidates it and requires it to be refreshed. These schemes differ

according to whether the hint is common to all clients or generated separately for each client:

- *Client-independent hint* (e.g., SimplePIR, DoublePIR [16]): the server computes a database-dependent hint that is common to all clients and distributes it through a one-way offline transfer. The server computes this hint once, and each client can reuse it for multiple queries until the database changes. The client stores the hint locally and uses it during decryption to remove the public component of the server response. In SimplePIR, its size grows proportionally to the square root of the database size. DoublePIR applies a second recursive retrieval step over the hint, making client storage independent of the number of database records.
- *Client-specific hint* (e.g., Piano [39], Plinko [17]): each client derives a separate hint by streaming the database and compressing it into a compact local structure. During the online phase, this hint allows the server to access only a sublinear fraction of the database entries. This is the only category considered here with sublinear online server computation.

This setting is also known as silent preprocessing. Our choice of this setting is motivated by three practical considerations. First, many real-world PIR applications involve frequently updated databases (e.g., Certificate Transparency logs update daily, password breach databases update continuously). Both client-independent and client-specific hints must be refreshed after database updates, whereas server-side preprocessing avoids distributing updated state to each client. Second, applications serving large user bases cannot afford per-client state on the server. Third, privacy-preserving systems that compose PIR with anonymous channels (e.g., Tor) require query unlinkability across sessions, which is incompatible with server-stateful designs.

The Double PIR paradigm. The efficient protocols in this setting, notably HintlessPIR [23], YPIR [31], and InsPIRe⁽²⁾ [28], share a common *Double PIR* structure: the N -entry database is organized as an $n_{\text{row}} \times n_{\text{col}}$ matrix, and two layers of homomorphic retrieval are performed with a packing step in between to compress the intermediate representation. The workflow proceeds in three stages. Consider $\mathbf{D} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ and suppose that the requested entry is 6. The underlying row and column selections satisfy

$$\mathbf{D}^\top (0, 1)^\top = (4, 5, 6)^\top, \quad (4, 5, 6)(0, 0, 1)^\top = 6.$$

- (1) **First-layer retrieval and packing.** The client sends LWE encryptions of an n_{row} -dimensional indicator vector. The server computes a homomorphic matrix-vector multiplication, producing n_{col} LWE ciphertexts that encrypt the target row. In the example, an encryption of $(0, 1)^\top$ produces three LWE ciphertexts encrypting 4, 5, and 6. These LWE ciphertexts are then *packed* into a small number of RLWE ciphertexts, compressing the representation for the second layer.
- (2) **Second-layer retrieval and response packing.** The packed ring ciphertexts are represented as plaintext entries of the second-layer database. An encryption of $(0, 0, 1)^\top$ selects the representation of the first-layer ciphertext encrypting 6, and the result is packed into a response ciphertext.
- (3) **Response extraction.** The client reconstructs the selected first-layer ciphertext from the response and then decrypts it to recover 6.

1.1 Our Contributions

We identify three independent sources of inefficiency in the existing design and address each with a dedicated technique. The three optimizations are largely orthogonal and their benefits compound across the pipeline.

- **Pack-then-switch.** We replace RLWE with NTRU as the packed intermediate format, reducing each packed ciphertext from a pair of ring elements to a single polynomial. We propose a *pack-then-switch* paradigm to preserve the CRS-based precomputation advantages of InsPIRe [28]. In this design, we utilize the RLWE packing infrastructure to perform initial data processing, followed by a lightweight key-switching operation that converts RLWE outputs to NTRU ciphertexts, thereby enabling efficient LWE-to-NTRU packing.
- **Fused second-layer retrieval via split-point picking.** The second layer needs to merge multiple intermediate ciphertexts via homomorphic automorphisms. We observe that each intermediate is the product of a database-dependent plaintext coefficient and a single shared encrypted query, so each automorphism decomposes into a free plaintext operation and a ciphertext rotation that is computed once and reused. Building on the split-point picking framework [15], this method reduces the online automorphism count from linear to square-root complexity in the number of packed second-layer inputs.
- **Optimized ring arithmetic.** Prior schemes require a prime ciphertext modulus to ensure a certain algebraic inverse in the packing embedding is well-defined, which ties them to NTT-based polynomial arithmetic. We remove this restriction through modulus switching and modulus raising, allowing the use of a power-of-two modulus and FFT-based arithmetic. Independently, we adopt approximate gadget decomposition [9], which reduces the number of decomposition levels by introducing a bounded decomposition residual.

Summary of results. We implement NTPIR in Rust with AVX-512 SIMD and evaluate it on different database sizes. We use InsPIRe⁽²⁾, the state-of-the-art Double PIR construction, as the primary baseline. Across the four shared operating points on a 1 GB database, NTPIR achieves up to a 1.81× speedup in server-side online computation over InsPIRe⁽²⁾. At $N/t=32768$, it reaches 6,204 MB/s, corresponding to a 1.76× speedup over InsPIRe⁽²⁾ and a 3.4× speedup over SimpleYPIR. In addition, for 8 GB database, NTPIR outperforms InsPIRe⁽²⁾ by 1.38× to 1.80×. At $N/t=65536$, it reaches 8,609 MB/s, corresponding to a 1.38× speedup over InsPIRe⁽²⁾, a 1.8× speedup over SimpleYPIR, and a 2.0× speedup over HintlessPIR. At 8 GB, total communication is 1.1–1.6× lower than SimpleYPIR (1,354 KB) and 3.5–4.9× lower than HintlessPIR (4,188 KB). The response size is a fixed 112 KB. Finally, offline preprocessing completes in under 7 s for 1 GB and under 56 s for 8 GB. At 8 GB, it is 7–41× faster than InsPIRe⁽²⁾. At the highest-throughput operating point for each database size, the offline preprocessing of NTPIR is more than two orders of magnitude faster than that of HintlessPIR.

1.2 Technical Overview

We present the key ideas behind each of our three optimizations.

Insight 1: switching the packed format from RLWE to NTRU.

An NTRU ciphertext consists of one ring element rather than an RLWE pair, making it a natural candidate for compressing the intermediate representation in the Double PIR pipeline. Existing schemes such as InspiRe and YPIR pack the LWE ciphertexts into RLWE ciphertexts. NTRU ciphertexts support the homomorphic operations required by the subsequent column selection and response packing. The difficulty is that RLWE ciphertexts have the form (a, b) , where a is drawn from a public common reference string (CRS) fixed across all queries. This structure allows the server to precompute the a -dependent operations before receiving a query, leaving only the b -dependent operations online. Because an NTRU ciphertext has no public component, directly replacing RLWE with NTRU would move the entire packing procedure to the online phase. We therefore use InspiRING, the LWE-to-RLWE sparse packing algorithm of InspiRe⁽²⁾, and apply an RLWE-to-NTRU key switch to its output. This *pack-then-switch* approach preserves the offline/online separation of InspiRING while producing a single-element NTRU ciphertext, as detailed in Section 3.2.

Insight 2: fusing column selection with response packing.

Using NTRU ciphertexts as the packed intermediates changes the second retrieval layer. In InspiRe, the a -side of packing over the intermediate ciphertexts is computed offline, while only the b -side is computed for each query. This separation is unavailable for NTRU intermediates, and directly applying the same packing procedure would place its full automorphism cost on the online phase. We adapt PackRLWEs [15] to fuse column selection with response packing. Each leaf of the second-layer packing tree is the product of a database-dependent plaintext polynomial and the encrypted column-selection query shared across the leaves. Since ring automorphisms distribute over these products, the automorphisms of the plaintext polynomials can be computed offline, while each automorphism of the encrypted query is computed once and reused. Using split-point picking [15], the lower tree levels are reduced to offline plaintext operations together with a shared set of online ciphertext automorphisms, while PackRLWEs completes the upper levels. This reduces the number of online automorphisms from linear to square-root complexity in the number of packed second-layer inputs. Section 3.3 gives the construction.

Insight 3: optimized ring arithmetic. The InspiRING embedding requires a multiplicative inverse modulo the ciphertext modulus, which forces InspiRe to use a prime modulus and NTT-based polynomial arithmetic. We remove this restriction through modulus switching and modulus raising. The partial trace used during aggregation supplies the factor removed by modulus switching, allowing a power-of-two ciphertext modulus and FFT-based polynomial arithmetic. We also adopt approximate gadget decomposition [9] to reduce the decomposition length while maintaining comparable noise growth. These techniques are detailed in Sections 3.2.1 and 3.2.2.

1.3 Organization

Section 2 reviews background. Sections 3 and 4 present our packing techniques and full protocol construction, respectively. Section 5 evaluates performance and Section 6 concludes this paper. Finally, we provide a comprehensive survey of related work in Appendix A.

2 Preliminaries

For a positive integer n , we write $[n] = \{0, 1, \dots, n-1\}$. We use $\mathbb{Z}_q$ to represent integers modulo q , centered in $(-q/2, q/2]$. The *cyclotomic ring* of degree d (a power of two) is $R = \mathbb{Z}[X]/(X^d + 1)$, and $R_q = R/qR$. For a vector $\mathbf{v}$, the ℓ_∞ -norm is $\|\mathbf{v}\|_\infty = \max_i |v_i|$. For a polynomial $f \in R$, $\|f\|_\infty$ denotes the ℓ_∞ -norm of its coefficient vector.

2.1 Private Information Retrieval

We consider single-server PIR in the Common Reference String (CRS) model with server-side preprocessing [16, 28]. A PIR scheme consists of four algorithms:

- $\text{Setup}(1^\lambda, \text{DB}) \rightarrow (\text{pp}, \text{dbp})$: the server, given security parameter 1^λ and database $\text{DB} \in (\{0, 1\}^L)^N$ of N records of L bits each, outputs public parameters pp (shared with all clients) and preprocessed database dbp .
- $\text{Query}(\text{pp}, \text{idx}) \rightarrow (\text{st}, \text{qry})$: the client, given index $\text{idx} \in [N]$, outputs secret state st and query qry .
- $\text{Respond}(\text{dbp}, \text{qry}) \rightarrow \text{resp}$: the server computes a response from the preprocessed database and query.
- $\text{Extract}(\text{st}, \text{resp}) \rightarrow \text{DB}[\text{idx}]$: the client recovers the target record.

Correctness. A PIR scheme has correctness error δ if for all databases and indices: $\Pr[\text{Extract}(\text{st}, \text{resp}) = \text{DB}[\text{idx}]] \geq 1 - \delta$, where the probability is over all randomness.

Query privacy. For all efficient adversaries $\mathcal{A}$ and indices $\text{idx}_0, \text{idx}_1$: $|\Pr[\mathcal{A}(\text{qry}_0) = 1] - \Pr[\mathcal{A}(\text{qry}_1) = 1]| \leq \text{negl}(\lambda)$.

CRS model and preprocessing. We work in the CRS model, where all parties share a globally generated random string. For lattice-based PIR, the CRS fixes the random components of ciphertexts (e.g., the a vector in LWE) across all queries, while each client samples a fresh secret key per query. This separation enables *server-side preprocessing*: the server precomputes all operations that depend only on the CRS and the database content before any query arrives, confining the online phase to query-dependent computation.

2.2 Lattice-Based Encryption

We review the gadget decomposition primitive and the three lattice-based encryption schemes used in our construction. Detailed correctness proofs and noise analysis are deferred to Appendix B.

2.2.1 Gadget Decomposition. Gadget decomposition is central to key switching in both RLWE and NTRU. For modulus q and base B , let $\ell = \lceil \log_B q \rceil$. The gadget vector is $\mathbf{g} = (g_0, g_1, \dots, g_{\ell-1}) = (1, B, B^2, \dots, B^{\ell-1})$. The inverse operation $\mathbf{g}^{-1} : \mathbb{Z}_q \rightarrow [-B/2, B/2]^\ell$ decomposes each integer into signed base- B digits. This extends coefficient-wise to polynomials: $\mathbf{g}^{-1} : R_q \rightarrow R^\ell$. With $\ell = \lceil \log_B q \rceil$, the decomposition is *exact*: $\langle \mathbf{g}^{-1}(a), \mathbf{g} \rangle = a \pmod{q}$.

2.2.2 Learning with Errors (LWE). *Decisional LWE assumption.* Let χ_s and χ_e be distributions over $\mathbb{Z}$. For $\mathbf{s} \leftarrow \chi_s^n$, let $\mathcal{A}_{\mathbf{s}, \chi_e}$ denote the distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by sampling $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ uniformly and $e \leftarrow \chi_e$ independently, and outputting $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$. The decisional $\text{LWE}_{n, q, \chi_s, \chi_e}$ assumption states that polynomially many independent samples from $\mathcal{A}_{\mathbf{s}, \chi_e}$, generated using the same secret $\mathbf{s}$, are computationally indistinguishable from independent uniform samples in $\mathbb{Z}_q^n \times \mathbb{Z}_q$ [34, 38].

LWE ciphertext. We adopt the secret-key LWE ciphertext from Regev-type lattice encryption [34]. To encrypt $m \in \mathbb{Z}_p$ under $s \in \mathbb{Z}_q^n$, sample $a \leftarrow \mathbb{Z}_q^n$ uniformly and $e \leftarrow \chi_e$, and output $ct = (a, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, where

$$b = \langle a, s \rangle + e + \Delta m \pmod{q}, \quad (1)$$

and $\Delta = \lfloor q/p \rfloor$. Decryption computes $b - \langle a, s \rangle = \Delta m + e$ and rounds to the nearest multiple of Δ .

2.2.3 Ring Learning with Errors (RLWE). Decisional RLWE assumption. Let χ_s^R and χ_e^R be distributions over R . For $s \leftarrow \chi_s^R$, let $\mathcal{A}_{s, \chi_e^R}$ denote the distribution over R_q^2 obtained by sampling $a \leftarrow R_q$ uniformly and $e \leftarrow \chi_e^R$ independently, and outputting $(a, as + e)$. The decisional $\text{RLWE}_{R_q, \chi_s^R, \chi_e^R}$ assumption states that polynomially many independent samples from $\mathcal{A}_{s, \chi_e^R}$, generated using the same secret s , are computationally indistinguishable from independent uniform samples in R_q^2 [27, 38].

RLWE ciphertext. To encrypt $\mu \in R_p$ under $s \in R_q$, sample $a \leftarrow R_q$ uniformly and $e \leftarrow \chi_e^R$, and output $ct = (a, b) \in R_q^2$, where

$$b = as + e + \Delta \mu \pmod{q}, \quad (2)$$

and $\Delta = \lfloor q/p \rfloor$. Decryption computes $b - as = \Delta \mu + e$ and rounds coefficient-wise to the nearest multiple of Δ [12]. We write $ct \in \text{RLWE}_{s, q}(\mu)$.

Homomorphic operations. RLWE ciphertexts support the following key operations (noise bounds in Appendix B):

- **Key switching.** Given $ct \in \text{RLWE}_{s_1, q}(\mu)$, and let $\text{KSK}_{s_1 \rightarrow s_2}$ denote the vector of ℓ RLWE evaluation-key samples from s_1 to s_2 :

$$\text{KSK}_{s_1 \rightarrow s_2} = \{(w_j, s_2 \cdot w_j + g_j \cdot s_1 + e_{\text{KSK}, j})\}_{j=0}^{\ell-1} \in (R_q^2)^\ell, \quad (3)$$

where $w_j \leftarrow R_q$ is uniform and $e_{\text{KSK}, j}$ is sampled from the key-switching error distribution. RLWE key switching outputs

$$ct' = (a', b') = (0, b) - \sum_{j=0}^{\ell-1} d_j \cdot \text{KSK}_{s_1 \rightarrow s_2}[j] \in \text{RLWE}_{s_2, q}(\mu). \quad (4)$$

We write $\text{KS.Setup}(s_1, s_2)$ as shorthand for generating $\text{KSK}_{s_1 \rightarrow s_2}$.

- **Automorphism.** For $k \in \mathbb{Z}_{2d}^*$, the ring automorphism $\tau_k : a(X) \mapsto a(X^k)$ permutes the coefficient slots of the plaintext polynomial. Homomorphic evaluation applies τ_k to both ciphertext components, then key-switches from $\tau_k(s)$ back to s as follows:

$$\text{HomAut}_k(ct) = \text{KeySwitch}(\tau_k(ct), \text{KSK}_{\tau_k(s) \rightarrow s}).$$

2.2.4 NTRU and NGS. Decisional NTRU assumption. Let χ_f^N and χ_e^N be distributions over R . For $f \leftarrow \chi_f^N$ conditioned on $f \in R_q^\times$, and independently sampled $e \leftarrow \chi_e^N$, the decisional $\text{NTRU}_{R_q, \chi_f^N, \chi_e^N}$ assumption states that the distribution of $ef^{-1} \in R_q$ is computationally indistinguishable from the uniform distribution over R_q [38]. **NTRU ciphertext.** To encrypt $\mu \in R_p$ under an invertible secret key $f \in R_q^\times$, sample $e \leftarrow \chi_e^N$ and output the ciphertext

$$ct = \text{NTRU}_{f, q}(\mu) = \frac{e + \Delta \mu}{f} \in R_q. \quad (5)$$

Here $\Delta = \lfloor q/p \rfloor$. Decryption computes ctf and rounds each coefficient to the nearest multiple of Δ .

NGS ciphertext. Following the NTRU-based GSW-like construction [38], for $m \in R_q$, sample $e_{\text{NGS}, 0}, \dots, e_{\text{NGS}, \ell-1} \leftarrow \chi_e^N$ independently and define

$$\begin{aligned} \text{NGS}_{f, q}(m) &= (ct_0, \dots, ct_{\ell-1}) \in R_q^\ell, \\ ct_j &= \frac{e_{\text{NGS}, j} + g_j m}{f}, \quad j = 0, \dots, \ell-1. \end{aligned} \quad (6)$$

Here $g_j = B^j$ denotes the j -th component of the gadget vector $\mathbf{g}$. **Homomorphic operations.** NTRU ciphertexts support the following key operations (correctness and noise bounds are present in Appendix B):

- **External product.** Given any $u \in R_q$ and $CT = \text{NGS}_{f, q}(m)$, the external product computes and outputs

$$u \boxtimes CT = \langle g^{-1}(u), CT \rangle \in \text{NTRU}_{f, q}(um). \quad (7)$$

- **Key switching.** Given $ct \in \text{NTRU}_{f_1, q}(\mu)$ and a key-switching key $\text{KSK} = \text{NGS}_{f_2, q}(f_1)$, compute:

$$ct' = \langle g^{-1}(ct), \text{KSK} \rangle \in \text{NTRU}_{f_2, q}(\mu). \quad (8)$$

- **Automorphism.** For $k \in \mathbb{Z}_{2d}^*$, apply τ_k to the ciphertext, then key-switch from $\tau_k(f)$ to f :

$$\text{HomAut}_k(ct) = \langle g^{-1}(\tau_k(ct)), \text{NGS}_{f, q}(\tau_k(f)) \rangle.$$

Automorphisms permute the coefficient slots of the plaintext and are used for ring packing and slot rotation.

- **Scalar multiplication.** RLWE and NTRU ciphertexts support plaintext-ciphertext multiplication. Multiplying each ciphertext component by $c \in R_q$ yields a ciphertext encrypting $c\mu$ with noise ce .

3 Ring Packing

3.1 Revisiting InspiRING Packing

The InspiRING sparse packing algorithm [28], used by NTPiR, transforms γ LWE ciphertexts into a single RLWE ciphertext in three stages: embedding, aggregation, and collapse.

3.1.1 Sparse Packing. For $\gamma = 2^k \leq d/2$, sparse packing uses the cyclic subgroup generated by

$$g_\gamma = \frac{2d}{\gamma} + 1 \in \mathbb{Z}_{2d}^*, \quad \text{ord}(g_\gamma) = \gamma.$$

Then, the partial trace is defined as

$$\text{Tr}_\gamma(p) = \sum_{j=0}^{\gamma-1} \tau_{g_\gamma^j}(p). \quad (9)$$

For $p = \sum_{i=0}^{d-1} c_i X^i$, the partial trace satisfies

$$\text{Tr}_\gamma(p) = \gamma \sum_{\substack{0 \leq i < d \\ \gamma | i}} c_i X^i. \quad (10)$$

Thus, sparse packing requires a single key-switching key KSK_{g_γ} .

3.1.2 Stage 1: from LWE to Intermediate Ciphertexts. Each LWE ciphertext $(a, b) \in \mathbb{Z}_q^d \times \mathbb{Z}_q$ is embedded as

$$\tilde{a} = \gamma^{-1} \sum_{i=0}^{d-1} a[i] \cdot X^{-i} \pmod{q}, \quad \tilde{b} = b. \quad (11)$$

Here $\tilde{s} = \sum_{i=0}^{d-1} s[i]X^i$ is the polynomial embedding of the LWE secret. Let $A = \sum_i a[i]X^{-i}$, $S = \tilde{s}$, and $M = b - AS$, the constant coefficient of M is $M[0] = \Delta m + e$. Applying the partial trace in Eq. (9) then yields the intermediate ciphertext $\text{IRCTx}(\tilde{M}) = (\hat{a}, \tilde{b}) \in R_q^\gamma \times R_q$, with components $\hat{a}[j] = \tau_{g_\gamma}^j(\tilde{a})$ for $j = 0, \dots, \gamma - 1$. The corresponding intermediate secret key $\hat{s} \in R_q^\gamma$ inherits the correlated structure $\hat{s}[j] = \tau_{g_\gamma}^j(\tilde{s})$. Therefore, the intermediate ciphertext satisfies

$$\begin{aligned} \tilde{b} - \langle \hat{a}, \hat{s} \rangle &= \gamma^{-1} \text{Tr}_\gamma(b - AS) \\ &= \sum_{\substack{0 \leq i < d \\ \gamma | i}} M[i]X^i =: \hat{M} \pmod{q}, \end{aligned} \quad (12)$$

where $\hat{M}[0] = \Delta m + e$, and $\hat{M}[i] = 0$ for $i \not\equiv 0 \pmod{\gamma}$.

3.1.3 Stage 2: Aggregation of Intermediate Ciphertexts. The γ intermediate ciphertexts are combined by shifting each message into a distinct coefficient position:

$$(\hat{a}_{\text{agg}}, \tilde{b}_{\text{agg}}) = \sum_{k=0}^{\gamma-1} \text{IRCTx}(\hat{M}_k)X^k.$$

By Eq. (12), the first γ coefficients satisfy

$$(\tilde{b}_{\text{agg}} - \langle \hat{a}_{\text{agg}}, \hat{s} \rangle)[k] = \Delta m_k + e_k, \quad k = 0, \dots, \gamma - 1.$$

Note that for each k , the coefficients of $X^k \hat{M}_k$ occur only at indices $k + j\gamma$. Hence its only contribution among the first γ coefficients is $(\Delta m_k + e_k)X^k$. The subsequent collapse preserves the plaintext and adds the key-switching noise.

3.1.4 Stage 3: Conversion to RLWE Ciphertext. The aggregated intermediate ciphertext $\text{IRCTx}(\hat{M}_{\text{agg}}) = (\hat{a}_{\text{agg}}, \tilde{b}_{\text{agg}})$ comprises $\gamma + 1$ ring elements and is converted to an RLWE ciphertext under $\tilde{s}$. The components of $\hat{s}$ satisfy

$$\hat{s}[j+1] = \tau_{g_\gamma}(\hat{s}[j]), \quad j = 0, \dots, \gamma - 2.$$

Consequently, the collapse reuses the single key-switching key

$$\text{KSK}_{g_\gamma} = \text{KS.Setup}(\tau_{g_\gamma}(\tilde{s}), \tilde{s}).$$

Let $\text{KSK}_{g_\gamma} = (\mathbf{w}_{g_\gamma}, \mathbf{y}_{g_\gamma}) \in R_q^\ell \times R_q^\ell$ be the key-switching key defined in Section 2.2.3. The collapse iterates from $j = \gamma - 2$ down to $j = 0$:

$$\begin{aligned} \mathbf{d}_j &\leftarrow \mathbf{g}^{-1}(\hat{a}_{\text{agg}}[j+1]), \\ \hat{a}_{\text{agg}}[j] &\leftarrow \hat{a}_{\text{agg}}[j] - \langle \mathbf{d}_j, \tau_{g_\gamma}^j(\mathbf{w}_{g_\gamma}) \rangle, \\ \tilde{b}_{\text{agg}} &\leftarrow \tilde{b}_{\text{agg}} - \langle \mathbf{d}_j, \tau_{g_\gamma}^j(\mathbf{y}_{g_\gamma}) \rangle. \end{aligned} \quad (13)$$

The same decomposition vector $\mathbf{d}_j$ is used in both updates. After $\gamma - 1$ steps, only $\hat{a}_{\text{agg}}[0]$ remains on the a -side, and the output satisfies

$$b_{\text{fin}} - a_{\text{fin}}\tilde{s} = \hat{M}_{\text{agg}} + E_{\text{coll}} \pmod{q},$$

where $a_{\text{fin}} = \hat{a}_{\text{agg}}[0]$ and E_{coll} is the key-switching noise accumulated during the collapse. For $k = 0, \dots, \gamma - 1$, let $e_{\text{pack}}[k] = e_k + E_{\text{coll}}[k]$. Then

$$(b_{\text{fin}} - a_{\text{fin}}\tilde{s})[k] = \Delta m_k + e_{\text{pack}}[k], \quad k = 0, \dots, \gamma - 1.$$

3.1.5 Offline/Online Separation in CRS Model. InspiRING is designed for the CRS model, in which the random components $\{\mathbf{a}_k\}$ and the uniformly random $\mathbf{w}_{g_\gamma}$ are fixed prior to any client query. For every query, the client samples a fresh $\mathbf{s}$ and generates the corresponding $\mathbf{y}_{g_\gamma}$. This admits a clean offline/online separation:

- **Offline:** All operations depending solely on $\{\mathbf{a}_k\}$ and $\mathbf{w}_{g_\gamma}$, namely ring embedding (Stage 1), aggregation (Stage 2), and the a -fold of the collapse (Stage 3), are precomputed and cached.
- **Online:** Only the assembly of $\tilde{b}_{\text{agg}}$ from the query-dependent scalars $\{b_k\}$ and the corresponding b -fold collapse are executed at query time.

For sparse packing with $\gamma \leq d/2$, InspiRING achieves $O(\gamma^2 d + \ell \gamma d \log d)$ offline time and $O(\ell \gamma d)$ online time, where $\ell = \lceil \log_B q \rceil$ is the gadget decomposition. The complete procedure is presented in lines 1–13 of Algorithm 1.

3.2 Pack-then-Switch: LWE to NTRU

In this section, we show how to transform γ LWE ciphertexts into a single NTRU ciphertext. Our construction combines an improved InspiRING packing with an RLWE-to-NTRU key-switching step:

$$\text{LWE} \xrightarrow{\text{improved InspiRING}} \text{RLWE} \xrightarrow{\text{key switching}} \text{NTRU}.$$

The improvements to InspiRING are twofold. First, we remove the multiplicative inverse $\gamma^{-1} \pmod{q_R}$ in the embedding step through modulus switching and modulus raising, enabling the use of a power-of-two modulus and FFT-based polynomial arithmetic (Section 3.2.1). Second, we adopt approximate gadget decomposition to reduce the decomposition length (Section 3.2.2). The RLWE-to-NTRU conversion is described in Section 3.2.3.

3.2.1 FFT-Friendly Arithmetic. The embedding step in InspiRING needs to compute the inverse element γ^{-1} in Equation (11). Since $\gamma^{-1} \pmod{q_R}$ exists only when $\gcd(\gamma, q_R) = 1$, this forces InspiRING to use a prime modulus with NTT-based arithmetic.

Modulus switching. Instead of computing $\gamma^{-1} \cdot a[i] \pmod{q_R}$, we apply modulus switching: $a'[i] = \left\lfloor \frac{a[i]}{\gamma} \right\rfloor \in \mathbb{Z}_{q_R/\gamma}$. Thus, $a[i] = \gamma \cdot a'[i] + r_i$ with remainder $r_i \in [0, \gamma)$. The resulting rounding error is $E_{\text{emb}} = \langle \mathbf{r}, \mathbf{s} \rangle$. The switched components are then embedded into the ring *without* the γ^{-1} factor: $\tilde{a}' = \sum_{i=0}^{d-1} a'[i] \cdot X^{-i}$, with coefficients in $\mathbb{Z}_{q_R/\gamma}$.

Modulus raising via partial trace. When the partial trace Tr_γ is performed on the original modulus q_R , it supplies the missing factor γ , similarly to [36]:

$$[\text{Tr}_\gamma(\tilde{a}'\tilde{s})]_0 = \gamma \langle \mathbf{a}', \mathbf{s} \rangle \pmod{q_R},$$

where $\tilde{s} = \iota(\mathbf{s})$ is the coefficient embedding of the LWE secret. Since $\gamma \cdot a'[i] = a[i] - r_i$, the product $\gamma \cdot \langle \mathbf{a}', \mathbf{s} \rangle$ recovers the original inner product $\langle \mathbf{a}, \mathbf{s} \rangle$ up to the rounding residual, and the standard InspiRING packing relation is preserved:

$$b - \gamma \cdot \langle \mathbf{a}', \mathbf{s} \rangle = \Delta m + e + E_{\text{emb}}.$$

In effect, modulus switching maps q_R to q_R/γ , and multiplication by γ inside the partial trace raises the modulus back to q_R . With γ^{-1} eliminated, the ciphertext modulus q_R is no longer constrained to be coprime with γ and may be set to a power of two. This enables FFT-based polynomial multiplication using native 64-bit integer arithmetic, avoiding the prime-modulus constraint that forces InspiRe to use NTT with per-coefficient modular reductions.

3.2.2 Approximate Gadget Decomposition. Recall from Section 2.2.1 that exact gadget decomposition with modulus $q = 2^Q$ and power-of-two base B requires $\ell_{\text{exact}} = \lceil Q/\log_2 B \rceil$ to represent all Q bits of each coefficient, yielding ℓ_{exact} polynomial multiplications per key-switching call. We adopt an *approximate* gadget decomposition [9] that reduces the decomposition length while maintaining comparable noise growth.

Definition 1 (Approximate Gadget Decomposition). Let $q = 2^Q$, $B = 2^b$, and $\ell b \leq Q$. Set $\eta = Q - \ell b$ and define

$$\mathbf{g}_\eta = (g_{\eta,0}, \dots, g_{\eta,\ell-1}) = 2^\eta (1, B, \dots, B^{\ell-1}).$$

For $a \in \mathbb{Z}_q$, represented in $[-q/2, q/2)$, let

$$a_\eta = \left\lfloor \frac{a}{2^\eta} \right\rfloor, \quad \varepsilon = a - 2^\eta a_\eta,$$

and define $\mathbf{g}_\eta^{-1}(a)$ as the ℓ -digit signed base- B decomposition of a_η . Then

$$\langle \mathbf{g}_\eta^{-1}(a), \mathbf{g}_\eta \rangle = a - \varepsilon \pmod{q}, \quad \varepsilon \in [-2^{\eta-1}, 2^{\eta-1}). \quad (14)$$

When $\eta = 0$, $\mathbf{g}_\eta$ reduces to the exact gadget of Section 2.2.1.

Noise trade-off. For RLWE key switching, the noise under approximate decomposition has two independent sources [9]:

$$V_{\text{ks}} \leq \ell \cdot d \cdot \frac{B^2}{12} \cdot \sigma_{\text{LWE}}^2 + \mathbb{E}[\|s_1\|_2^2] \cdot \frac{2^{2\eta}}{12}. \quad (15)$$

The first part grows linearly with ℓ and originates from the encryption noise in the key-switching key, and the second part (approximation residual) decreases exponentially as ℓ increases. This trade-off allows us to use a smaller ℓ while maintaining comparable noise growth. The approximate gadgets used for RLWE key switching and NTRU/NGS key switching are denoted by $(B_R, \ell_R, \eta_R, \mathbf{g}_R)$ and $(B_N, \ell_N, \eta_N, \mathbf{g}_N)$, respectively.

3.2.3 RLWE-to-NTRU Key Switching. After the improved InspiRING step, we obtain a standard RLWE ciphertext $(a, b) \in R_{q_R}^2$ satisfying $b - a\tilde{s} = \Delta\mu + e$. Before the RLWE-to-NTRU conversion, we apply modulus switching from q_R to q_N :

$$\bar{a} = \left\lfloor \frac{q_N}{q_R} a \right\rfloor, \quad \bar{b} = \left\lfloor \frac{q_N}{q_R} b \right\rfloor.$$

Thus $(\bar{a}, \bar{b}) \in R_{q_N}^2$ satisfies $\bar{b} - \bar{a}\tilde{s} = \lfloor q_N/p \rfloor \mu + \bar{e}$. We then convert this two-component ciphertext into a single NTRU ciphertext under the same modulus q_N .

Definition 2 (RLWE-to-NTRU Conversion Keys). Let $\tilde{s} = \iota(\mathbf{s}) \in R_{q_R}$ be the RLWE secret and $f \in R_{q_N}^\times$ the NTRU secret. The conversion key pair consists of two NGS ciphertexts under f :

$$\text{KSK}_0 = \text{NGS}_{f,q_N}(1), \quad \text{KSK}_1 = \text{NGS}_{f,q_N}(\tilde{s}). \quad (16)$$

That is, KSK_0 encodes the identity, while KSK_1 encodes the RLWE secret $\tilde{s}$ in NGS form under f .

Theorem 2 (RLWE-to-NTRU Conversion). Let $(a, b) \in R_{q_R}^2$ be an RLWE ciphertext satisfying

$$b - a\tilde{s} = \left\lfloor \frac{q_R}{p} \right\rfloor \mu + e.$$

and $(\bar{a}, \bar{b}) \in R_{q_N}^2$ denote the ciphertext obtained by modulus switching (a, b) from q_R to q_N . Given the conversion keys $(\text{KSK}_0, \text{KSK}_1)$ of Def. 2, the conversion computes

$$\text{ct}' := \langle \mathbf{g}_N^{-1}(\bar{b}), \text{KSK}_0 \rangle - \langle \mathbf{g}_N^{-1}(\bar{a}), \text{KSK}_1 \rangle. \quad (17)$$

Then $\text{ct}' \in \text{NTRU}_{f,q_N}(\mu)$, and its noise e_{out} satisfies

$$\begin{aligned} \text{Var}(e_{\text{out}}) &\leq \left(\frac{q_N}{q_R} \right)^2 \sigma^2 + \frac{1 + d\sigma_s^2}{12} \\ &\quad + (1 + d\sigma_s^2) \frac{2^{2\eta_N}}{12} + 2\ell_N d \frac{B_N^2}{12} \sigma_{\text{NTRU}}^2. \end{aligned} \quad (18)$$

PROOF. By Lemma 3, we have

$$\bar{b} - \bar{a}\tilde{s} = \left\lfloor \frac{q_N}{p} \right\rfloor \mu + \bar{e}, \quad \text{Var}(\bar{e}) \leq \left(\frac{q_N}{q_R} \right)^2 \sigma^2 + \frac{1 + d\sigma_s^2}{12}.$$

Then, Equation (6) gives

$$\text{KSK}_0[j]f = e_j^{(0)} + g_{N,j}, \quad \text{KSK}_1[j]f = e_j^{(1)} + g_{N,j}\tilde{s}.$$

By substitution into Eq. (17), we can obtain

$$\begin{aligned} \text{ct}' \cdot f &= (\bar{b} - \varepsilon_b + e_{\text{ks}}^{(0)}) - \tilde{s} \cdot (\bar{a} - \varepsilon_a) - e_{\text{ks}}^{(1)} \\ &= (\bar{b} - \bar{a} \cdot \tilde{s}) - (\varepsilon_b - \tilde{s} \cdot \varepsilon_a) + (e_{\text{ks}}^{(0)} - e_{\text{ks}}^{(1)}) \\ &= \left\lfloor \frac{q_N}{p} \right\rfloor \mu + \bar{e} + e_{\text{conv}}, \end{aligned} \quad (19)$$

where

$$e_{\text{conv}} = (e_{\text{ks}}^{(0)} - e_{\text{ks}}^{(1)}) - (\varepsilon_b - \tilde{s}\varepsilon_a).$$

This step outputs

$$\text{Var}(e_{\text{conv}}) \leq (1 + d\sigma_s^2) \frac{2^{2\eta_N}}{12} + 2\ell_N d \frac{B_N^2}{12} \sigma_{\text{NTRU}}^2$$

as shown in Lemma 8. Finally, combining the two bounds proves Eq. (18). $\square$

Algorithm 1 presents the complete Pack-then-Switch procedure, which follows the InspiRING offline/online decomposition and adds a single RLWE-to-NTRU key-switching call ($2\ell_N$ polynomial multiplications) to the online phase, a cost negligible compared with the $(\gamma - 1)\ell_R$ multiplications of the collapse. The offline complexity remains $O(\gamma^2 d + \ell_R \gamma d \log d)$, while the online complexity is $O((\ell_R \gamma + \ell_N)d)$, as shown in Section 3.1.5.

3.3 Integrated Second-Layer Retrieval and Packing

Recall that the first layer produces the NTRU ciphertext per column group of the database matrix. After modulus switching and digit decomposition (detailed in Section 4), the server arranges the resulting scalars as the *second-layer matrix* $\mathbf{W} \in \mathbb{Z}_{p_{\text{SPP}}}^{r \times c}$ with $r = d\tau$ rows and $c = \lceil n_{\text{col}}/\gamma \rceil$ columns. The task of the second layer is to select a single column from $\mathbf{W}$ and compress the result into compact response ciphertexts. The client provides an encrypted

Algorithm 1 Pack-then-Switch from LWE to NTRU. Gray shading identifies the CRS-dependent operations performed offline. The query-dependent b -side aggregation, b -fold, and RLWE-to-NTRU conversion are performed online.

Input: LWE ciphertexts $\{(a_i, b_i)\}_{i=0}^{\gamma-1}$ under secret s .
Input: InspiRING key $\text{KSK}_{g_Y} = (\mathbf{w}_{g_Y}, \mathbf{y}_{g_Y}) \in R_{q_R}^{\ell_R} \times R_{q_R}^{\ell_R}$, with $\text{KSK}_{g_Y} = \text{KS.Setup}(\tau_{g_Y}(\tilde{s}), \tilde{s})$.
Input: Conversion keys $\text{KSK}_0 = \text{NGS}_{f, q_N}(1)$, $\text{KSK}_1 = \text{NGS}_{f, q_N}(\tilde{s})$.
Output: $\text{ct}_{\text{out}} \in \text{NTRU}_{f, q_N}(\mu_{\text{out}})$ with $\mu_{\text{out}}[i] = m_i$ for $i = 0, \dots, \gamma - 1$.

```

1: // Stages 1–2: embedding and aggregation (CRS-dependent, offline)
2: for  $i = 0$  to  $\gamma - 1$  do
3:    $\mathbf{a}'_i \leftarrow \lfloor \mathbf{a}_i / \gamma \rfloor$ 
4:    $\tilde{\mathbf{a}}'_i \leftarrow \sum_{k=0}^{d-1} \mathbf{a}'_i[k] \cdot X^{-k}$ 
5:   for  $j = 0$  to  $\gamma - 1$  do
6:      $\hat{\mathbf{a}}_{\text{agg}}[j] \leftarrow \sum_{i=0}^{\gamma-1} \tau_{g_Y}^j(\tilde{\mathbf{a}}'_i) \cdot X^i$ 
7:    $\tilde{\mathbf{b}}_{\text{agg}} \leftarrow \sum_{i=0}^{\gamma-1} b_i \cdot X^i$  // online: aggregate b-side
8: // Stage 3: iterative collapse
9: for  $j = \gamma - 2$  down to  $0$  do
10:   $\mathbf{d}_j \leftarrow \mathbf{g}_R^{-1}(\hat{\mathbf{a}}_{\text{agg}}[j+1])$  // shared decomposition
11:   $\hat{\mathbf{a}}_{\text{agg}}[j] \leftarrow \hat{\mathbf{a}}_{\text{agg}}[j] - \langle \mathbf{d}_j, \tau_{g_Y}^j(\mathbf{w}_{g_Y}) \rangle$  // a-fold
12:   $\tilde{\mathbf{b}}_{\text{agg}} \leftarrow \tilde{\mathbf{b}}_{\text{agg}} - \langle \mathbf{d}_j, \tau_{g_Y}^j(\mathbf{y}_{g_Y}) \rangle$  // b-fold
13:   $(\mathbf{a}_{\text{rlwe}}, \mathbf{b}_{\text{rlwe}}) \leftarrow (\hat{\mathbf{a}}_{\text{agg}}[0], \tilde{\mathbf{b}}_{\text{agg}})$  // standard  $b - a\tilde{s}$  convention
14: // Stage 4: RLWE-to-NTRU key switch
15:  $\bar{\mathbf{a}} \leftarrow \lfloor (q_N/q_R)\mathbf{a}_{\text{rlwe}} \rfloor$ ,  $\bar{\mathbf{b}} \leftarrow \lfloor (q_N/q_R)\mathbf{b}_{\text{rlwe}} \rfloor$ 
16:  $\text{ct}_{\text{out}} \leftarrow \langle \mathbf{g}_N^{-1}(\bar{\mathbf{b}}), \text{KSK}_0 \rangle - \langle \mathbf{g}_N^{-1}(\bar{\mathbf{a}}), \text{KSK}_1 \rangle$ 
17: return  $\text{ct}_{\text{out}} \in \text{NTRU}_{f, q_N}(\mu_{\text{out}})$ 

```

column-selection vector $\text{ct}_v \in \text{NTRU}_{f, q_N}(\hat{\mathbf{v}})$ with plaintext modulus p_{SPP} , in which $\hat{\mathbf{v}} = \sum_{j=0}^{c-1} v_j X^j$ encodes the indicator vector $\mathbf{v} \in \{0, 1\}^c$ via coefficient embedding. Thus

$$\text{ct}_v f = \left\lfloor \frac{q_N}{p_{\text{SPP}}} \right\rfloor \cdot \hat{\mathbf{v}} + \mathbf{e}_v \pmod{q_N}.$$

When $r > d$, the matrix is partitioned into $n_b = \lceil r/d \rceil$ blocks of at most d rows each, with the last block zero-padded when necessary. We describe a full block of d rows, where all blocks share the same ciphertexts obtained by applying automorphisms to ct_v .

Recall that each row $\mathbf{w}_i = (w_{i,0}, \dots, w_{i,c-1})$ is encoded as a polynomial whose powers of X^{-1} mirror the powers of X used in $\hat{\mathbf{v}}$:

$$\hat{\mathbf{w}}_i(X) = w_{i,0} + w_{i,1}X^{-1} + \dots + w_{i,c-1}X^{-(c-1)}. \quad (20)$$

Then, the constant term of the product $\hat{\mathbf{w}}_i \cdot \hat{\mathbf{v}}$ equals the inner product $\langle \mathbf{w}_i, \mathbf{v} \rangle$, achieving column selection through a single polynomial multiplication. Since each encoded row occupies c coefficient slots, d/c rows can be packed into a single polynomial by shifting them into non-overlapping positions. For each block $b \in [n_b]$, we define

$n_p = c$ input-packed polynomials:

$$y_i^{(b)} = \sum_{j=0}^{d/c-1} X^{jc} \cdot \hat{\mathbf{w}}_{bd+i+jc}, \quad i \in [n_p], \quad (21)$$

where $y_i^{(b)}$ encodes rows $bd+i+jc$ for $j \in [d/c]$. Thus, when $r = d\tau$, the τ blocks contain $\tau n_p = \tau c = rc/d$ input-packed polynomials in total. For each block, the second layer computes $\{y_i^{(b)} \cdot \text{ct}_v\}_{i \in [n_p]}$ and merges them into one response ciphertext whose coefficient slots hold the corresponding d inner products. The protocol therefore returns one response ciphertext per block.

3.3.1 Standard Packing via PackRLWEs. The PackRLWEs procedure was introduced for RLWE ciphertexts [15]. Since its operations are also supported by NTRU, we apply the same binary-tree merge to NTRU ciphertexts. Specifically, it merges $n = 2^k$ ciphertexts, each containing 2^h regularly spaced payload coefficients, into one through a binary tree of k levels. At local level $l \in \{1, \dots, k\}$, the number of live ciphertexts is halved. The corresponding absolute tree level is $v = h + l$, with monomial shift $s_v = d/2^v$ and automorphism $\tau_v = \tau_{2^v+1}$. For each pair $(\text{ct}_j, \text{ct}_{j+n/2^l})$, the merge step computes

$$\text{ct}_j \leftarrow (\text{ct}_j + X^{s_v} \cdot \text{ct}_{j+n/2^l}) + \text{HomAut}_{\tau_v}(\text{ct}_j - X^{s_v} \cdot \text{ct}_{j+n/2^l}). \quad (22)$$

A merge step doubles the packed plaintext coefficients and the input noise. Each merge step requires exactly one HomAut call, for a total of $n - 1$ calls across all $\log_2 n$ levels. Applying PackRLWEs independently to the $n_p = c$ leaf ciphertexts of each block would require $c - 1$ HomAut calls per block, that is $n_b(c - 1)$ calls in total. The complete binary merge and its noise recurrence are given in Appendix B.2.1.

3.3.2 Split-Point Picking. We apply the split-point picking (SPP) technique [15] to the lower levels of the packing tree. The key observation is that every leaf of the packing tree has the form $y_i \cdot \text{ct}_v$, in which y_i is a precomputed plaintext polynomial and ct_v is the shared encrypted query. Since automorphisms distribute over plaintext-ciphertext products, $\tau(y_i \cdot \text{ct}_v) = \tau(y_i) \cdot \tau(\text{ct}_v)$, the plaintext $\tau(y_i)$ can be precomputed offline. The ciphertext $\tau(\text{ct}_v)$ then needs to be evaluated only *once* and can be reused across all branches at the same tree level. The HomAut calls in the lower levels of the packing tree therefore reduce to offline plaintext operations together with a shared set of online ciphertext automorphisms.

Recall that the packing tree over the d row results has $\log_2 d$ levels. The first $h_p = \log_2(d/c)$ levels correspond to the within-polynomial coefficient structure already arranged by Eq. (21). The remaining $L = \log_2 d - h_p = \log_2 c = \log_2 n_p$ levels merge the $n_p = c$ input-packed polynomials of one block. We choose a *split point* that partitions these L levels into a lower half of depth L_{lo} and an upper half of depth $L_{hi} = L - L_{lo}$, with $L_{lo} \approx L/2$. Define $\alpha = 2^{L_{lo}}$ to be the number of ciphertexts computed in Phase 1 and $\beta = n_p/\alpha = 2^{L_{hi}}$ to be the number of intermediate ciphertexts per block after Phase 2. After the z -tree is constructed, every z -polynomial in the root node is multiplied by α^{-1} in $R_{p_{\text{SPP}}}^{-1}$. This removes the factor α introduced by the lower L_{lo} merge levels and preserves the intended plaintext.

¹We require α to be invertible modulo p_{SPP} .

The normalization depends only on the database and is performed offline. The Galois subgroup used in the lower half is

$$\mathcal{G} = \langle \tau_{2^s+1} \rangle_{s=h_p+1}^{h_p+L_{l_0}}, \quad |\mathcal{G}| = \alpha. \quad (23)$$

The automorphism indices start at $h_p + 1$ rather than 1 because the first h_p levels are absorbed by the input packing.

Specifically, the algorithm proceeds in three phases. Phase 1 applies the shared automorphisms to the encrypted query. Phase 2 absorbs the lower tree levels into offline plaintext operations. Phase 3 completes the upper tree levels using PackRLWEs with per-level modulus switching.

Phase 1 (shared automorphisms). Starting from $\text{ct}_{v,\text{id}} = \text{ct}_v$, the server applies the automorphisms level by level to obtain α ciphertexts. At lower level $l = 0, \dots, L_{l_0} - 1$, let $\tau_l = \tau_{2^{h_p+1+l}+1}$. For every ciphertext $\text{ct}_{v,\sigma}$ available at the beginning of that level, the server computes

$$\text{ct}_{v,\sigma\tau_l} = \text{HomAut}_{\tau_l}(\text{ct}_{v,\sigma}, \text{KSK}_{\tau_l}).$$

At level $l = 0$, the subgroup generated by the preceding levels contains only id . The number of available ciphertexts doubles at every level, yielding one ciphertext for each element of $\mathcal{G}$ after L_{l_0} levels. Thus Phase 1 requires one automorphism key per level and $1+2+\dots+\alpha/2 = \alpha-1$ HomAut calls. The resulting α ciphertexts are shared across all β intermediate ciphertexts and all n_b row blocks.

Phase 2 (z-tree and online accumulation). The lower L_{l_0} levels of the packing tree are folded into plaintext preprocessing. Concretely, each intermediate ciphertext is expressed as a linear combination of the Phase 1 ciphertexts,

$$\text{ct}_{i_1}^{(b)} = \sum_{\tau \in \mathcal{G}} z_{i_1,\tau}^{(b)} \cdot \text{ct}_{v,\tau}, \quad (24)$$

in which the z -polynomials $\{z_{i_1,\tau}^{(b)}\}_{\tau \in \mathcal{G}}$ are plaintext coefficients that can be precomputed offline. For each block b and index $i_1 \in [\beta]$, the corresponding intermediate ciphertext requires α such z -polynomials (one per element of $\mathcal{G}$), and at query time the server evaluates Equation (24) via α plaintext-ciphertext multiplications without HomAut calls.

Then, the z -polynomials are constructed via a binary merge tree (the z -tree) that mirrors the PackRLWEs structure of Equation (22) but operates entirely on plaintexts. Each node in the z -tree is a map from elements of $\mathcal{G}$ to plaintext polynomials. The map records which ciphertext $\text{ct}_{v,\tau}$ each polynomial will multiply in Equation (24). For a fixed block b , we initialize α leaf nodes, each containing a map with one entry: $z_{i_0}^{(b)} = \{\text{id} \mapsto y_{i_0\beta+i_1}^{(b)}\}$ for $i_0 \in [\alpha]$. At each level $l = 0, \dots, L_{l_0} - 1$, two adjacent nodes (denoted “left” and “right”) are merged into one node with twice as many map entries. For every group element σ present in the left node:

$$\begin{aligned} z_{\sigma}^{(l+1)} &\leftarrow z_{\sigma,\text{left}}^{(l)} + X^{s_l} \cdot z_{\sigma,\text{right}}^{(l)}, \\ z_{\sigma \cdot \tau_l}^{(l+1)} &\leftarrow \tau_l(z_{\sigma,\text{left}}^{(l)} - X^{s_l} \cdot z_{\sigma,\text{right}}^{(l)}), \end{aligned} \quad (25)$$

in which the shift and automorphism at local level l correspond to absolute tree level $l' = h_p + 1 + l$, i.e., $s_l = d/2^{l'}$ and $\tau_l = \tau_{2^{l'}+1}$. The first line inherits the existing group element σ , and the second line creates a new entry at $\sigma \cdot \tau_l$, doubling the map size at each

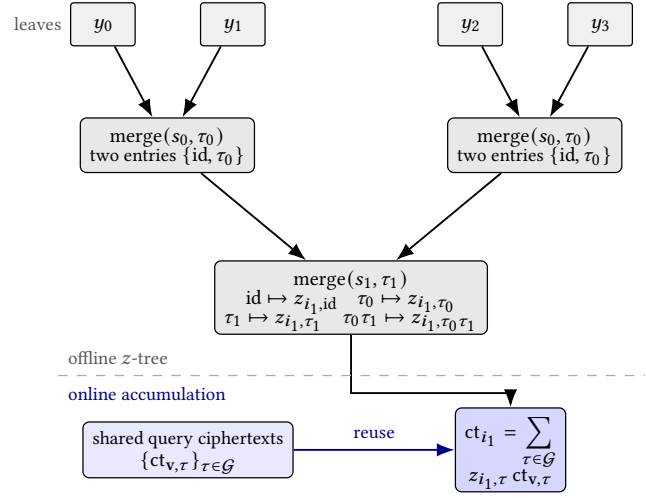


Figure 1: Z-tree functionality for $\alpha = 4$. The concrete polynomial updates are given in Eq. (25) and illustrated in Remark 1.

level. After L_{l_0} levels, a single root node remains containing the α z -polynomials $\{z_{i_1,\tau}^{(b)}\}_{\tau \in \mathcal{G}}$. We multiply each of these z -polynomials by α^{-1} in R_{pspp} and continue to use $z_{i_1,\tau}^{(b)}$ for the resulting polynomial. These depend only on the database and can be precomputed offline (and pre-transformed to the FFT domain). In Equation (24), each z -polynomial is represented by its coefficient-wise centered lift from R_{pspp} to R .

REMARK 1 (Z-TREE WALKTHROUGH FOR $\alpha = 4$). We show a specific construction in Figure 1. Fix a row block b and an intermediate index i_1 , and abbreviate $y_{i_0\beta+i_1}^{(b)}$ as y_{i_0} . Let $L_{l_0} = 2$ with automorphisms $\tau_0 = \tau_{2^{h_p+1}+1}$ and $\tau_1 = \tau_{2^{h_p+2}+1}$. The four leaves are $z_{i_0}^{(0)} = \{\text{id} \mapsto y_{i_0}\}$ for $i_0 = 0, 1, 2, 3$. For the level $l = 0$, the pairs $(0, 1)$ and $(2, 3)$ merge. For example, we have the first pair as

$$\text{id} \mapsto y_0 + X^{s_0} y_1, \quad \tau_0 \mapsto \tau_0(y_0 - X^{s_0} y_1).$$

Then, the two surviving nodes merge in level $l = 1$. For each existing entry $\sigma \in \{\text{id}, \tau_0\}$, the merge produces σ and $\sigma \cdot \tau_1$, yielding the full group $\mathcal{G} = \{\text{id}, \tau_0, \tau_1, \tau_0\tau_1\}$.

Phase 3 (PackRLWEs for the upper levels). For each block, the β intermediate ciphertexts from Phase 2 are merged into one response using the per-level modulus-switching technique of MS-PackLWEs [22]. At absolute tree level l' , let $s_{l'} = d/2^{l'}$ and define $\text{ModRaise}_{q_N/2 \rightarrow q_N}$ as the coefficient-wise centered lift from $R_{q_N/2}$ to R_{q_N} .

$$\tilde{\text{ct}}_j \leftarrow \text{ModRaise}_{q_N/2 \rightarrow q_N} \left(\text{ModSwitch}_{q_N \rightarrow q_N/2} \left(\text{ct}_j - X^{s_{l'}} \text{ct}_{j+\beta/2^{l'}} \right) \right), \quad (26)$$

$$\text{ct}_j \leftarrow \tilde{\text{ct}}_j + \text{HomAut}_{\tau_{l'}}(\tilde{\text{ct}}_j) + X^{s_{l'}} \text{ct}_{j+\beta/2^{l'}}. \quad (27)$$

The modulus switching in Equation (26) halves the input noise before the automorphism. Consequently, the merge preserves the plaintext scale instead of doubling it at each upper level. This step still requires exactly $\beta - 1$ HomAut calls. Its correctness and noise

Algorithm 2 Second-Layer Retrieval with SPP

Input: Second-layer matrix $\mathbf{W} \in \mathbb{Z}_{p_{\text{SPP}}}^{r \times c}$, encrypted query $\text{ct}_v \in \text{NTRU}_{f, q_N}(\hat{\mathbf{v}})$ with plaintext modulus p_{SPP} , split parameters α and β , automorphism keys $\{\text{KSK}_\tau\}$.

Output: Response ciphertexts $\{\text{ct}_{\text{resp}}^{(b)}\}$.

- 1: *// Input packing and z-tree construction*
- 2: Partition $\mathbf{W}$ into $n_b = \lceil r/d \rceil$ row blocks and form $n_p = c$ input-packed polynomials $\{y_i^{(b)}\}_{i \in [n_p]}$ per block via Eqs. (20)–(21).
- 3: **for** each block $b = 0, \dots, n_b - 1$ **do**
- 4: **for** $i_1 = 0$ **to** $\beta - 1$ **do**
- 5: Build the z-tree using Eq. (25) to obtain $\{z_{i_1, \tau}^{(b)}\}_{\tau \in \mathcal{G}}$
- 6: $z_{i_1, \tau}^{(b)} \leftarrow \alpha^{-1} z_{i_1, \tau}^{(b)}$ in R_{SPP} for each $\tau \in \mathcal{G}$
- 7: *// Online: query-dependent computation*
- 8: $\text{ct}_{v, \text{id}} \leftarrow \text{ct}_v$
- 9: **for** $l = 0$ **to** $L_{\text{lo}} - 1$ **do**
- 10: $\tau_l \leftarrow \tau_{2^{h_p+1+l}+1}$
- 11: **for** each $\sigma \in \langle \tau_0, \dots, \tau_{l-1} \rangle$ **do**
- 12: $\text{ct}_{v, \sigma \tau_l} \leftarrow \text{HomAut}_{\tau_l}(\text{ct}_{v, \sigma}, \text{KSK}_{\tau_l})$
- 13: **for** each block b **do**
- 14: **for** $i_1 = 0$ **to** $\beta - 1$ **do**
- 15: $\text{ct}_{i_1}^{(b)} \leftarrow \sum_{\tau \in \mathcal{G}} z_{i_1, \tau}^{(b)} \cdot \text{ct}_{v, \tau}$
- 16: *// merge $\text{ct}_0^{(b)}, \dots, \text{ct}_{\beta-1}^{(b)}$ via Eqs. (26)–(27):*
- 17: **for** $l = 1$ **to** $\log_2 \beta$ **do** *// level $l' = h_p + L_{\text{lo}} + l$*
- 18: **for** $j = 0$ **to** $\beta/2^l - 1$ **do**
- 19: $\text{ct}_j^{(b)} \leftarrow \text{ct}_j^{(b)} - X^{s_{l'}} \text{ct}_{j+\beta/2^l}^{(b)}$
- 20: $\tilde{\text{ct}}_j^{(b)} \leftarrow \text{ModRaise}_{q_N/2 \rightarrow q_N} \left(\text{ModSwitch}_{q_N \rightarrow q_N/2} \left(\text{ct}_j^{(b)} \right) \right)$
- 21: $\text{ct}_j^{(b)} \leftarrow \tilde{\text{ct}}_j^{(b)} + \text{HomAut}_{\tau_{l'}}(\tilde{\text{ct}}_j^{(b)}, \text{KSK}_{\tau_{l'}}) + X^{s_{l'}} \text{ct}_{j+\beta/2^l}^{(b)}$
- 22: $\text{ct}_{\text{resp}}^{(b)} \leftarrow \text{ct}_0^{(b)}$
- 23: **return** $\{\text{ct}_{\text{resp}}^{(b)}\}$

bound are established in Lemma 11. Finally, algorithm 2 summarizes the full second-layer procedure.

3.3.3 Complexity Analysis. For a single block, the online HomAut count is $(\alpha - 1) + (\beta - 1) = \alpha + \beta - 2$. Because the $\alpha - 1$ Phase 1 calls are shared across all n_b blocks, the total count is

$$(\alpha - 1) + n_b(\beta - 1), \quad \alpha\beta = n_p = c.$$

For one block, this expression is minimized at $\alpha = \beta = \sqrt{c}$ and gives $O(\sqrt{c})$ HomAut calls. Across multiple blocks, the continuous optimum is $\alpha = \sqrt{n_b c}$, subject to the power-of-two and $\alpha \leq c$ constraints. Phase 2 involves only plaintext-ciphertext multiplications and adds no HomAut calls, and the z-tree construction is performed entirely offline. When multiple output blocks share the same Phase 1 ciphertexts, the amortized per-block cost is dominated by the $\beta - 1$ calls of Phase 3, and the optimal split shifts toward a larger α to further amortize Phase 1 across blocks (Appendix C.4).

4 Full Protocol Construction

This section presents the complete NTPIR protocol, which integrates the building blocks of Section 3. As in InsPIRe⁽²⁾ [28], the

database consists of N records, each an element of $\mathbb{Z}_p^{\gamma_0}$. The database $\text{DB} \in (\mathbb{Z}_p^{\gamma_0})^N$ is reshaped as a matrix $\mathbf{D} \in \mathbb{Z}_p^{N/t \times t\gamma_0}$, where t controls the trade-off between first-layer work (proportional to N/t) and second-layer work (proportional to $t\gamma_0$). Specifically, for $i \in [N/t]$, $i_1 \in [t]$, and $k \in [\gamma_0]$, $\mathbf{D}[i, i_1\gamma_0 + k] = \text{DB}[it + i_1][k]$. The packing parameter γ denotes the number of consecutive LWE ciphertexts processed by one Pack-then-Switch invocation. We use $\gamma_0 \mid \gamma$ and $\gamma \mid t\gamma_0$, so each packed NTRU ciphertext contains γ/γ_0 complete records. This instantiates the second-layer setting of Section 3.3 with $n_{\text{col}} = t\gamma_0$, $r = d\tau$, and $c = t\gamma_0/\gamma$, where c is a power of two and $c \mid d$. The protocol is summarized in Algorithms 3 and 4.

4.1 Protocol Description

Setup. The global public parameters contain a uniformly sampled matrix $\mathbf{A}_0 \in \mathbb{Z}_{q_R}^{N/t \times d}$ for the first-layer LWE encryption and the uniformly random component $\mathbf{w}_{g_y}$ of the RLWE packing key. Using only $(\mathbf{D}, \mathbf{A}_0, \mathbf{w}_{g_y})$, the server precomputes the first-layer mask $\mathbf{a}_j = \sum_i \mathbf{D}[i][j] \mathbf{A}_0[i]$ for every column j and the a -fold of Pack-then-Switch. This preprocessing requires no client secret key. The b -fold collapse is performed online using $\mathbf{y}_{g_y}$ sent with the query.

Query. For every query, the client independently samples a ternary LWE secret $\mathbf{s} \leftarrow \chi_s^d$ and an NTRU secret $f \leftarrow D_{\mathbb{Z}_s \sigma_f}^d$ conditioned on $f \in R_{q_N}^\times$. The coefficient embedding $\tilde{s} = \iota(\mathbf{s})$ is used only by the LWE/RLWE ciphertext, while f corresponds to the NTRU/NGS ciphertext. As in InsPIRe [28], the random component $\mathbf{w}_{g_y}$ of the packing key is fixed in the CRS. The client generates $\mathbf{y}_{g_y}$ from $(\mathbf{w}_{g_y}, \tilde{s})$. The conversion keys are $\text{KSK}_0 = \text{NGS}_{f, q_N}(1)$ and $\text{KSK}_1 = \text{NGS}_{f, q_N}(\tilde{s})$. Finally, it generates $\text{KSK}_\tau = \text{NGS}_{f, q_N}(\tau(f))$ for every $\tau \in \mathcal{T}_{\text{spp}}$, where $\mathcal{T}_{\text{spp}}$ contains one automorphism for each of the $\log_2 \alpha$ lower levels and the $\log_2 \beta$ upper levels required by SPP. Since $\alpha\beta = c$, $|\mathcal{T}_{\text{spp}}| = \log_2 c$. The client decomposes the target index idx into a row index $i_0 = \lfloor \text{idx}/t \rfloor$ and a record index $i_1 = \text{idx} \bmod t$ within that row. The record index is further split into a column-group index $g^* = \lfloor i_1/(\gamma/\gamma_0) \rfloor$ and an intra-group offset $o^* = (i_1 \bmod (\gamma/\gamma_0))\gamma_0$. The client then forms two query components. The first is a vector of N/t LWE b -components modulo q_R under the shared CRS $\mathbf{A}_0$ that encrypts a one-hot indicator at row i_0 . The second is an NTRU ciphertext $\text{ct}_{\text{col}} \in \text{NTRU}_{f, q_N}(\hat{\mathbf{v}})$ with plaintext modulus p_{SPP} that selects the target column group. Following Section 3.3, the indicator vector $\mathbf{v} \in \{0, 1\}^c$ with $c = \lceil t\gamma_0/\gamma \rceil$ is embedded as $\hat{\mathbf{v}} = \sum_{j=0}^{c-1} v_j X^j$. Since $\mathbf{v}$ is one-hot at position g^* , this expression collapses to the monomial $\hat{\mathbf{v}} = X^{g^*}$. The query contains these two ciphertext components and the per-query evaluation material. The client retains (f, g^*, o^*) for extraction and may erase $\mathbf{s}$ after forming the query.

Respond. The server processes the query in four stages.

Stage 1 (first-layer retrieval). The matrix-vector product $\mathbf{D}^\top \cdot \mathbf{b}_0$ produces $t\gamma_0$ LWE ciphertexts, each encrypting one element of the target row of $\mathbf{D}$.

Stage 2 (LWE-to-NTRU packing). Every group of γ consecutive LWE ciphertexts is converted into a single NTRU ciphertext via the Pack-then-Switch procedure of Section 3.2, producing $\lceil t\gamma_0/\gamma \rceil$ NTRU intermediates. The precomputed $\hat{\mathbf{a}}_{\text{agg}}$ from Setup allows the a -fold to be performed offline.

Stage 3 (modulus switching and digit decomposition). The server first applies modulus switching to each NTRU intermediate from q_N to q' . Every coefficient of the resulting polynomial is then expressed in $\tau = \lceil \log_{p_{\text{SPP}}} q' \rceil$ base- p_{SPP} digits. More precisely, for the canonical representative $\hat{c}_g[k] \in \{0, \dots, q' - 1\}$, the digits $w_{\delta,g}[k] \in \{0, \dots, p_{\text{SPP}} - 1\}$ satisfy

$$\hat{c}_g[k] = \sum_{\delta=0}^{\tau-1} p_{\text{SPP}}^\delta w_{\delta,g}[k].$$

This representation is exact because $p_{\text{SPP}}^\tau \geq q'$. Arranging these digits column by column yields the second-layer plaintext matrix $\mathbf{W} \in \mathbb{Z}_{p_{\text{SPP}}}^{r \times c}$ with $r = d\tau$ rows and $c = \lceil t\gamma_0/\gamma \rceil$ columns. This matches the setting of Section 3.3 with $n_{\text{col}} = t\gamma_0$ and $c = \lceil n_{\text{col}}/\gamma \rceil = \lceil t\gamma_0/\gamma \rceil$. We emphasize that the digit decomposition is applied to the NTRU ciphertext, which lets SPP operate on plaintext digits in Stage 4, at the cost of one additional decryption layer in Extract that reconstructs the NTRU intermediate from the digits before the final NTRU decoding.

Stage 4 (second-layer retrieval). Algorithm 2 applies SPP to $\mathbf{W}$ and the column-selection ciphertext ct_{col} , producing τ response ciphertexts for the selected column of $\mathbf{W}$.

Extract. Extraction proceeds in two layers, reflecting the digit decomposition performed in Stage 3.

Layer 1 (SPP response decoding). Each SPP response $\text{ct}_{\text{resp}}^{(\delta)}$ is an NTRU ciphertext at modulus q_N that carries the δ -th base- p_{SPP} digit polynomial $w_\delta \in R_{p_{\text{SPP}}}$ of the selected intermediate. Operating on digits encrypted at q_N while decomposing the intermediate at q' keeps the number of digits at $\tau = \lceil \log_{p_{\text{SPP}}} q' \rceil$ instead of $\lceil \log_{p_{\text{SPP}}} q_N \rceil$. The client decodes each response as

$$\begin{aligned} w_\delta &= \text{NTRU.Dec}_{f, q_N \rightarrow p_{\text{SPP}}}(\text{ct}_{\text{resp}}^{(\delta)}) \\ &:= \left\lfloor \frac{\text{ct}_{\text{resp}}^{(\delta)} f \bmod q_N}{\lfloor q_N/p_{\text{SPP}} \rfloor} \right\rfloor \bmod p_{\text{SPP}}. \end{aligned}$$

This recovers the τ digit polynomials $w_0, \dots, w_{\tau-1} \in R_{p_{\text{SPP}}}$.

Layer 2 (NTRU intermediate decoding). The client reconstructs the NTRU intermediate at modulus q' as $\hat{\text{ct}} = \sum_{\delta=0}^{\tau-1} p_{\text{SPP}}^\delta \cdot w_\delta \in R_{q'}$ and decodes it at modulus q' :

$$\mu = \text{NTRU.Dec}_{f, q' \rightarrow p}(\hat{\text{ct}}) := \left\lfloor \frac{p}{q'} (\hat{\text{ct}} \cdot f \bmod q') \right\rfloor \in R_p.$$

The packed plaintext polynomial μ satisfies $(\mu[o^*], \dots, \mu[o^* + \gamma_0 - 1]) = \text{DB}[\text{idx}]$.

4.2 Correctness and Security

Theorem 3 (Correctness). *For the parameter configurations in Tables 1, 4, 5, and 7, under the subgaussian noise and independence model stated in Appendix B, NTPIR satisfies $\Pr[\text{Extract}(\text{st}, \text{resp}) = \text{DB}[\text{idx}]] \geq 1 - 2^{-40}$.*

PROOF. Note that failure can occur in either of the two decoding layers of Extract. Layer 1 decodes the digit polynomial w_δ from each SPP response. By Lemma 13, this decoding is correct if

$$\|e_{\text{SPP}}\|_\infty + 15 < \frac{1}{2} \left\lfloor \frac{q_N}{p_{\text{SPP}}} \right\rfloor.$$

Then, Lemma 12 bounds the per-coefficient variance of e_{SPP} in this condition. The recovered digit polynomials satisfy $\hat{\text{ct}} = \sum_{\delta=0}^{\tau-1} p_{\text{SPP}}^\delta w_\delta$

Algorithm 3 NTPIR Protocol — Setup and Client-side Query

Input: Security parameter 1^λ , database $\text{DB} \in (\mathbb{Z}_p^{\gamma_0})^N$, query index $\text{idx} \in [N]$.

Output: Public parameters pp , preprocessed database dbp , per-query client state st , and query qry .

```

1:
2: procedure Setup( $1^\lambda, \text{DB}$ ):
3:    $\mathbf{D} \leftarrow \text{Reshape}(\text{DB}, N/t \times t\gamma_0)$ 
4:    $\mathbf{A}_0 \leftarrow \mathbb{Z}_{q_R}^{N/t \times d}$ ;  $\mathbf{w}_{g_Y} \leftarrow R_{q_R}^{\text{fr}}$  // global CRS randomness
5:   for  $j = 0$  to  $t\gamma_0 - 1$  do
6:      $\mathbf{a}_j \leftarrow \sum_{i=0}^{N/t-1} \mathbf{D}[i][j] \mathbf{A}_0[i]$ 
7:   for  $g = 0$  to  $\lceil t\gamma_0/\gamma \rceil - 1$  do
8:      $\text{pre}_g \leftarrow \text{Pack.Preprocess}(\{\mathbf{a}_{g_Y+k}\}_{k=0}^{\gamma-1}, \mathbf{w}_{g_Y})$  // includes
       the a-side collapse
9:    $\text{pp} \leftarrow (\mathbf{A}_0, \mathbf{w}_{g_Y}, q_R, q_N, q', p, p_{\text{SPP}}, d, t, \gamma_0, \gamma, \alpha, \beta)$ ;  $\text{dbp} \leftarrow$ 
        $(\mathbf{D}, \{\mathbf{a}_j\}_j, \{\text{pre}_g\}_g)$ 
10:  return ( $\text{pp}, \text{dbp}$ )
11:
12: procedure Query( $\text{pp}, \text{idx}$ ):
13:   // Independent keys for this query
14:    $\mathbf{s} \leftarrow \chi_s^d$ ,  $\tilde{\mathbf{s}} \leftarrow \iota(\mathbf{s})$ 
15:    $f \leftarrow D_{\mathbb{Z}, sf}^d$  conditioned on  $f \in R_{q_N}^\times$ 
16:    $\mathbf{y}_{g_Y} \leftarrow \text{GenKSMatY}(\tilde{\mathbf{s}}, \mathbf{w}_{g_Y}, \tau_{g_Y})$  // completes  $\text{KSK}_{g_Y} =$ 
        $(\mathbf{w}_{g_Y}, \mathbf{y}_{g_Y})$ 
17:    $\text{KSK}_0 \leftarrow \text{NGS}_{f, q_N}(1)$ ,  $\text{KSK}_1 \leftarrow \text{NGS}_{f, q_N}(\tilde{\mathbf{s}} \bmod q_N)$ 
18:    $\{\text{KSK}_\tau\}_{\tau \in \mathcal{T}_{\text{SPP}}} \leftarrow \{\text{NGS}_{f, q_N}(\tau(f))\}_{\tau \in \mathcal{T}_{\text{SPP}}}$ 
19:    $\text{evk} \leftarrow (\mathbf{y}_{g_Y}, \text{KSK}_0, \text{KSK}_1, \{\text{KSK}_\tau\}_{\tau \in \mathcal{T}_{\text{SPP}}})$ 
20:    $(i_0, i_1) \leftarrow (\lfloor \text{idx}/t \rfloor, \text{idx} \bmod t)$ 
21:    $(g^*, o^*) \leftarrow (\lfloor i_1/(\gamma/\gamma_0) \rfloor, (i_1 \bmod (\gamma/\gamma_0))\gamma_0)$ 
22:   for  $i = 0$  to  $N/t - 1$  do
23:      $e_i \leftarrow \chi_e$ ,  $b_i \leftarrow \langle \mathbf{A}_0[i], \mathbf{s} \rangle + e_i + \Delta \cdot [i = i_0] \pmod{q_R}$ 
24:    $\mathbf{b}_0 \leftarrow (b_0, \dots, b_{N/t-1})$ 
25:    $\hat{\mathbf{v}} \leftarrow X^{g^*}$ ,  $\text{ct}_{\text{col}} \leftarrow \text{NTRU}_{f, q_N}(\hat{\mathbf{v}})$  // plaintext modulus  $p_{\text{SPP}}$ 
26:  return ( $\text{st} = (f, g^*, o^*)$ ,  $\text{qry} = (\mathbf{b}_0, \text{ct}_{\text{col}}, \text{evk})$ )

```

over the integers by Lemma 14. Hence, they determine $\hat{\text{ct}}$ exactly without introducing additional noise. Lemma 14 also bounds the per-coefficient variance of e_{pack} , so Layer 2 is correct if $\|e_{\text{pack}}\|_\infty < \frac{q'}{2p}$. Consequently,

$$\begin{aligned} \Pr[\text{failure}] &\leq \Pr \left[\|e_{\text{SPP}}\|_\infty + 15 \geq \frac{1}{2} \left\lfloor \frac{q_N}{p_{\text{SPP}}} \right\rfloor \right] \\ &\quad + \Pr \left[\|e_{\text{pack}}\|_\infty \geq \frac{q'}{2p} \right]. \end{aligned}$$

The concrete calculation in Appendix B.4 applies the subgaussian tail bound to these events and a union bound over the τd coefficients decoded in Layer 1 and the γ_0 target coefficients decoded in Layer 2 to obtain

$$\Pr[\text{failure}] < 2^{-40}.$$

□

Theorem 4 (Query Privacy). *Under the decisional LWE and NTRU assumptions and the circular/KDM-security assumption, NTPIR provides computational query privacy in the CRS model.*

Algorithm 4 NTPIR Protocol — Server-side Respond and Client-side Extract

```

1: procedure Respond(pp, dbp, qry):
2:    $(\mathbf{b}_0, \text{ct}_{\text{col}}, \text{evk}) \leftarrow \text{qry}$ 
3:    $(\mathbf{y}_{g_Y}, \text{KSK}_0, \text{KSK}_1, \{\text{KSK}_\tau\}_\tau) \leftarrow \text{evk}; \quad \text{KSK}_{g_Y} \leftarrow (\mathbf{w}_{g_Y}, \mathbf{y}_{g_Y})$ 
4:   // Stage 1: first-layer matrix-vector product
5:   for  $j = 0$  to  $t\gamma_0 - 1$  do
6:      $\text{ct}_j^{\text{lwe}} \cdot \mathbf{a} \leftarrow \mathbf{a}_j$  // read from dbp
7:      $\text{ct}_j^{\text{lwe}} \cdot \mathbf{b} \leftarrow \sum_i \mathbf{D}[i][j] \cdot b_i$ 
8:     // Stage 2: LWE-to-NTRU packing
9:     for  $g = 0$  to  $\lceil t\gamma_0/\gamma \rceil - 1$  do
10:       $\text{ct}_g^{\text{ntru}} \leftarrow \text{PackThenSwitch}(\{\text{ct}_{g_Y+k}^{\text{lwe}}\}_{k=0}^{\gamma-1}, \text{pre}_g, \text{evk})$ 
11:      // Stage 3: modulus switching and base- $p_{\text{SPP}}$  digit decomposition
12:      for  $g = 0$  to  $\lceil t\gamma_0/\gamma \rceil - 1$  do
13:         $\hat{\text{ct}}_g \leftarrow \lfloor \frac{q}{q_N} \text{ct}_g^{\text{ntru}} \rfloor$  // modulus switching to  $q'$ 
14:        for  $\delta = 0$  to  $\tau - 1$  do
15:           $w_{\delta,g} \leftarrow \lfloor \hat{\text{ct}}_g / p_{\text{SPP}}^\delta \rfloor \bmod p_{\text{SPP}}$  // coefficient-wise
16:          Arrange  $\{w_{\delta,g}\}$  as  $\mathbf{W} \in \mathbb{Z}_{p_{\text{SPP}}}^{r \times c}$  with  $r = d\tau$ ,  $c = \lceil t\gamma_0/\gamma \rceil$ , where
          row block  $b$  contains the digits for  $\delta = b$ .
17:          // Stage 4: second-layer retrieval with SPP
18:           $\{\text{ct}_{\text{resp}}^{(\delta)}\}_{\delta \in [\tau]} \leftarrow \text{IntegratedSPP}(\mathbf{W}, \text{ct}_{\text{col}}, \alpha, \beta, \{\text{KSK}_\tau\}_\tau)$ 
19:          return  $\{\text{ct}_{\text{resp}}^{(\delta)}\}_{\delta \in [\tau]}$ 
20:
21: procedure Extract(st,  $\{\text{ct}_{\text{resp}}^{(\delta)}\}$ ):
22:    $(f, g^*, o^*) \leftarrow \text{st}$ 
23:   // Layer 1: decode each SPP response from  $R_{q_N}$  down to  $R_{p_{\text{SPP}}}$ 
24:   for  $\delta = 0$  to  $\tau - 1$  do
25:      $w_\delta \leftarrow \left\lfloor \frac{\text{ct}_{\text{resp}}^{(\delta)} f \bmod q_N}{\lfloor q_N/p_{\text{SPP}} \rfloor} \right\rfloor \bmod p_{\text{SPP}}$ 
26:     // Layer 2: rebuild NTRU intermediate at  $q'$  and decode
27:      $\hat{\text{ct}} \leftarrow \sum_{\delta=0}^{\tau-1} p_{\text{SPP}}^\delta \cdot w_\delta \in R_{q'}$ 
28:      $\mu \leftarrow \lfloor \frac{p}{q'} (\hat{\text{ct}} \cdot f \bmod q') \rfloor$ 
29:     return  $(\mu[o^*], \dots, \mu[o^* + \gamma_0 - 1])$ 

```

PROOF. Fix an index idx and write its decomposition as (i_0, i_1) and (g^*, o^*) , as in Query. For each query, the client independently samples $\mathbf{s}$ and f . The two ciphertext components of the query are

$$b_i = \langle \mathbf{A}_0[i], \mathbf{s} \rangle + e_i + \Delta[i = i_0] \pmod{q_R}, \quad 0 \leq i < N/t,$$

and, by Equation (5),

$$\text{ct}_{\text{col}} = \frac{e + \lfloor q_N/p_{\text{SPP}} \rfloor X^{g^*}}{f} \pmod{q_N}.$$

Here the errors $e_i \leftarrow \chi_e$ and $e \leftarrow \chi_N$ are sampled independently. The group offset o^* is retained in the client state and is not included in the query. The remaining query component is

$$\text{evk} = (\mathbf{y}_{g_Y}, \text{KSK}_0, \text{KSK}_1, \{\text{KSK}_\tau\}_{\tau \in \mathcal{T}_{\text{SPP}}}).$$

It is generated from these keys and independently sampled errors and does not depend on idx . The component $\mathbf{y}_{g_Y}$ is generated from $(\mathbf{w}_{g_Y}, \tilde{\mathbf{s}})$. The key KSK_0 encrypts 1 under f , KSK_1 encrypts $\tilde{\mathbf{s}}$ under f , and each KSK_τ encrypts $\tau(f)$ under f . Thus, the evaluation material

contains encryptions of functions of the secret keys, to which the circular/KDM-security assumption [7] applies.

We prove query privacy by a hybrid argument. First, this assumption allows evk to be replaced by independent uniform elements in the joint query distribution. Next, replace $\mathbf{b}_0 = (b_0, \dots, b_{N/t-1})$ by a uniformly random element of $\mathbb{Z}_{q_R}^{N/t}$. For a fixed i_0 , subtracting the public terms $\Delta[i = i_0]$ from the entries of $\mathbf{b}_0$ gives the LWE samples $\langle \mathbf{A}_0[i], \mathbf{s} \rangle + e_i$. The decisional LWE assumption therefore implies that this replacement is computationally indistinguishable. Adding the fixed public terms to a uniform vector preserves uniformity. Finally, Equation (5), together with the decisional NTRU assumption, allows ct_{col} to be replaced by a uniformly random element of R_{q_N} .

After these replacements, $\mathbf{b}_0$, ct_{col} , and evk are independent uniform elements, and their joint distribution is independent of (i_0, g^*, o^*) . Hence the query distributions for any two indices idx_0 and idx_1 are computationally indistinguishable, which proves the query privacy definition of Section 2.1. $\square$

4.3 Communication and Computation Complexity

Communication. Let w_T denote the number of bits used to serialize one ring coefficient. Per-query communication has three components. A side-by-side comparison with SimpleYPIR, KSPIR, HintlessPIR, and InsPIRe⁽²⁾ is given in Section 5.

- **Key material.** The global CRS already contains the uniformly random component $\mathbf{w}_{g_Y}$ of the RLWE packing key. The per-query component $\mathbf{y}_{g_Y}$ contains $\ell_R d$ words. Each RLWE-to-NTRU conversion or SPP automorphism key is an NGS ciphertext containing $\ell_N d$ words. There are two conversion keys and $|\mathcal{T}_{\text{SPP}}| = \log_2 \alpha + \log_2 \beta = \log_2 c$ automorphism keys. Consequently, the per-query evaluation material occupies $(\ell_R + (2 + |\mathcal{T}_{\text{SPP}}|)\ell_N)dw_T$ bits and is transmitted with every query, not once in global Setup.
- **Query.** Excluding the evaluation material counted above, the ciphertext query contains N/t scalar b -components of $\log q_R$ bits and one serialized NTRU ciphertext of dw_T bits, for a total of $(N/t) \log q_R + dw_T$ bits.
- **Response.** The server returns τ NTRU ciphertexts of τdw_T bits in total.

Computation. Server-side work breaks down into the four stages of Algorithm 4. The first-layer matrix-vector product has $O(N\gamma_0 d)$ offline cost and $O(N\gamma_0)$ online cost. Pack-then-Switch processes $\lceil t\gamma_0/\gamma \rceil$ groups, each requiring $O((\ell_R \gamma + \ell_N)d)$ ring operations for the InspiRING collapse and the RLWE-to-NTRU conversion, for a total of $O(\lceil t\gamma_0/\gamma \rceil (\ell_R \gamma + \ell_N)d)$. Modulus switching and digit decomposition cost $O(t\gamma_0 d)$ scalar operations and are negligible compared with the ring multiplications in Stages 2 and 4. For Stage 4, the SPP Phase 1 automorphisms ($\alpha - 1$ HomAut calls) are shared across all τ row blocks, while Phase 2 (α plaintext-ciphertext multiplications per output group) and Phase 3 ($\beta - 1$ HomAut calls) are paid per block. The total online cost is therefore $((\alpha - 1) + \tau(\beta - 1)) \cdot O(\ell_N d)$ for HomAut and $\tau\beta\alpha \cdot O(d)$ for plaintext-ciphertext products. Since $r = d\tau$ and $\alpha\beta = c$, choosing $\alpha \approx \sqrt{rc} = \sqrt{rc/d}$ amortizes Phase 1 across the row blocks and balances the two HomAut terms (Appendix C.4). Compared with InsPIRe [28], each NTRU intermediate contributes one ciphertext polynomial rather than an RLWE pair.

Table 1: Lattice parameters for NTPIR.

	d	Plaintext modulus	$\log_2 q$	Key dist.	Error dist.	ℓ	B	Sec.
RLWE/LWE	2048	2^{16}	64	ternary	Gauss., $\sigma=4.1 \times 10^3$	2	2^{16}	128 bits
NTRU	2048	2^{16} (Stage 2) 31 (SPP)	45	Gauss., $\sigma_f=23$	Gauss., $\sigma=23$	3	2^{12}	128 bits

Stage 4 therefore processes one component per intermediate, reducing the ring-multiplication count by a factor of two.

5 Experimental Evaluation

We compare NTPIR with four state-of-the-art single-server PIR schemes that support server-side preprocessing: InsPIRe⁽²⁾ [28], SimpleYPIR [31], KSPIR [26], and HintlessPIR [23]. We exclude schemes that require client-specific *server state* [1, 2, 8, 29, 30, 32, 33], database-dependent client hints [16], or client-specific hints [35, 39], because they operate under fundamentally different state or communication models. The baseline comparisons use 64-byte entries and report server online computation time and total per-query communication, measured as the sum of per-query evaluation keys, query ciphertexts, and response.

5.1 Implementation and Parameters

Our Rust implementation of NTPIR runs single-threaded on an Intel Xeon 6982P-C (3.0 GHz, 128 GB DDR5, Ubuntu 22.04). Reported runtimes are averaged over 500 independent executions. Table 1 lists the lattice parameters. For NTRU, the secret $f \leftarrow D_{\mathbb{Z}, \sigma_f}^d$ is conditioned on being invertible in R_{q_N} , with $\sigma_f = \sigma_{\text{NTRU}} = 23$. The smaller NTRU modulus reflects the tighter modulus-to-noise constraint imposed by the NTRU assumption at a given ring degree. We evaluated both parameter sets with the lattice estimator², and each achieves 128-bit security. Note that for the NTRU instance, we evaluated the Gaussian secret and error distributions using the estimator’s `primal_dsd` method rather than applying the $\sigma = 1$ criterion for overstretched NTRU instances [19]. The concrete correctness error is at most 2^{-40} , as verified in Appendix B.4. For $q'=2^{30}$, $\tau = \lceil \log_{p_{\text{SPP}}} q' \rceil = 7$, producing seven response ciphertexts of fixed total size 112 KB.

5.2 Microbenchmarks

We isolate the two building blocks that drive the performance gains of NTPIR, namely first-layer ring packing (Section 3.2) and second-layer retrieval via SPP (Section 3.3).

First-layer ring packing. Table 2 compares InsPIRe’s InspiRING implementation with the Pack-then-Switch pipeline of NTPIR on 2,048 LWE ciphertexts at packing depths $\gamma \in \{64, 256, 1024\}$. The power-of-two modulus removes Barrett reductions, while reducing the RLWE decomposition length from 4 to 2. The final RLWE-to-NTRU conversion uses $\ell_N = 3$, as specified in Table 1. Together, these changes produce a $1.6\times$ to $2.4\times$ online speedup and a $7.7\times$ to $8.6\times$ offline speedup. The offline gap is wider because the pre-computed a -fold performs $\gamma - 1$ automorphism-and-key-switching iterations per group. Pack-then-Switch also produces 16 KB NTRU

Table 2: First-layer ring packing with 2,048 LWE ciphertexts.

	γ	InsPIRe	NTPIR	Speedup
Key (KB)	N/A	42	128	N/A
Packed (KB)	N/A	28	16	1.8×
Offline (ms)	64	764.5	99.8	7.7×
	256	1828.2	219.7	8.3×
	1024	5992.7	699.7	8.6×
Online (ms)	64	10.9	6.7	1.6×
	256	9.9	5.2	1.9×
	1024	11.5	4.8	2.4×

Table 3: Second-layer per-query cost (ms) versus column count c at $q'=2^{30}$ (two base- 2^{16} digits for InsPIRe and seven base- p_{SPP} digits for NTPIR).

		Column count c		
		64	256	2048
InsPIRe	a -fold	186.0	186.5	186.2
	b -fold	10.1	10.1	10.4
	Total	196.1	196.7	196.5
NTPIR (SPP)	Phases 1–2	1.6	4.3	31.5
	Phase 3	1.4	3.3	14.5
	Total	3.0	7.5	45.9
Speedup (Total)		65.0×	26.2×	4.3×

ciphertexts rather than 28 KB RLWE pairs, reducing the packed intermediate size by $1.8\times$.

Second-layer retrieval (SPP). Table 3 compares SPP with the second-layer packing of InsPIRe across three values of the second-layer column count c (Section 4). Both schemes process the same c first-layer packed intermediates at $q'=2^{30}$. InsPIRe decomposes each coefficient into two base- 2^{16} digits and uses the second-layer packing factor $\gamma_{12}=64$, whereas NTPIR decomposes each coefficient into seven base- p_{SPP} digits. SPP fuses column selection and response packing into a single pass (Section 3.3). The corresponding split choices for $c = 64, 256$, and 2048 are $(\alpha, \beta) = (16, 4), (32, 8)$, and $(64, 32)$, respectively. Since $p_{\text{SPP}} = 31$, the corresponding values of α^{-1} are 2, 1, and 16 modulo p_{SPP} . As described in Section 3.3.2, the z -polynomials are pre-transformed to the FFT domain. At query time, each Phase 1 ciphertext is transformed once and reused across all intermediate ciphertexts. For each intermediate ciphertext, the α products are accumulated in the FFT domain. This evaluates Equation (24) without repeating the same transforms for every plaintext-ciphertext product.

Across $c \in \{64, 256, 2048\}$, SPP achieves $65.0\times$, $26.2\times$, and $4.3\times$ speedups, respectively. As c grows toward d , the number of Phase 2 plaintext-ciphertext products increases and the advantage narrows. SPP is therefore most effective when $c \ll d$, which is precisely the regime that arises under the parameter choices of NTPIR (Table 11). The corresponding response sizes are reported in Tables 4 and 5.

²<https://github.com/malb/lattice-estimator>

Table 4: PIR performance at 1 GB ($2^{24} \times 64$ B, single-threaded).

N/t	SimpleYPIR	KSPIR	HintlessPIR	InsPIRe ⁽²⁾				NTPIR			
	N/A	N/A	N/A	4096	8192	16384	32768	4096	8192	16384	32768
Offline (s)	112	13	201	69	44	31	65	6.7	2.7	1.3	0.7
Keys (KB)	462	2352	360	80	80	80	80	656	560	560	512
Query (KB)	112	14	128	53	66	113	215	48	80	144	272
Response (KB)	228	224	1748	52	52	52	52	112	112	112	112
Total Comm. (KB)	802	2590	2236	185	198	244	347	816	752	816	896
Server (ms)	566	731	713	1072	632	443	291	799	350	253	165
Throughput (MB/s)	1809	1401	1436	955	1620	2312	3519	1281	2925	4050	6204

Table 5: PIR performance at 8 GB ($2^{27} \times 64$ B, single-threaded).

N/t	SimpleYPIR	KSPIR	HintlessPIR	InsPIRe ⁽²⁾				NTPIR			
	N/A	N/A	N/A	8192	16384	32768	65536	8192	16384	32768	65536
Offline (s)	267	111	1415	379	222	148	111	55.3	17.9	6.6	2.7
Keys (KB)	462	2688	360	80	80	80	80	656	656	608	560
Query (KB)	448	14	512	106	132	225	431	80	144	272	528
Response (KB)	444	224	3316	52	52	52	52	112	112	112	112
Total Comm. (KB)	1354	2926	4188	238	264	357	563	848	912	992	1200
Server (ms)	1747	5575	1918	4415	2519	1745	1311	2454	1707	1164	952
Throughput (MB/s)	4689	1470	4271	1855	3252	4694	6247	3338	4800	7036	8609

5.3 End-to-End Comparison

Tables 4 and 5 present end-to-end comparisons at 1 GB ($2^{24} \times 64$ B) and 8 GB ($2^{27} \times 64$ B), respectively. The column N/t in each table denotes the number of LWE rows in the first layer, equivalently the row dimension of the reshaped database matrix D (Section 4). Both InsPIRe⁽²⁾ and NTPIR are evaluated at the same four operating points at each database size, allowing a direct comparison. Server throughput is reported as database size divided by server-side online time. For the Total Comm. row, we report the sum Keys + Query + Response. The Keys row is a genuine per-query cost in NTPIR because every query uses freshly sampled client keys and freshly generated secret-dependent evaluation material.

Table 6 reports peak server memory usage. For InsPIRe⁽²⁾ and NTPIR, these measurements use $N/t = 4096$. At 1 GB, NTPIR uses 5.5 GB, compared with 7.9 GB for InsPIRe⁽²⁾. At 8 GB, NTPIR uses 26.1 GB, the lowest among the evaluated schemes, compared with 36.9 GB for InsPIRe⁽²⁾.

5.3.1 Protocol-level Parameter Selection. As in InsPIRe⁽²⁾, each 64-byte record comprises $\gamma_0 = 32$ plaintext elements at $p = 2^{16}$. The packing factor γ controls the trade-off between first-layer packing cost and second-layer matrix width. A larger γ groups more LWE ciphertexts together, which reduces the column count $c = \lceil t\gamma_0/\gamma \rceil$ of W and therefore the SPP cost, while lengthening the per-group collapse chain. Table 11 quantifies this trade-off for the 1 GB database at $N/t=4096$: online server time decreases monotonically with γ , from 799 ms at $\gamma=64$ to 479 ms at $\gamma=1024$. The concrete correctness analysis in Appendix B.4 includes $\gamma = 1024$ and therefore covers every value used below. We select γ separately at each operating point according to the protocol-level trade-off quantified in Table 11, rather

than fixing $\gamma = 1024$ across all configurations. For the 1 GB experiments, we use $\gamma \in \{64, 128, 64, 64\}$, giving $c \in \{2048, 512, 512, 256\}$ for the four values of N/t in Table 4. For the 8 GB experiments, we use $\gamma \in \{256, 128, 128, 128\}$, giving $c \in \{2048, 2048, 1024, 512\}$ for the four values of N/t in Table 5. These choices keep $c \leq d$ while preserving the database layouts used in the reported measurements. For these end-to-end configurations, we set $\alpha = 32$ and $\beta = c/32$.

Larger records. Table 7 evaluates larger records for a fixed 1 GB database at $N/t = 4096$ and $N/t = 32768$. At $p = 2^{16}$, the 64-byte, 256-byte, and 2-KB records contain $\gamma_0 = 32, 128$, and 1024 plaintext elements, respectively. We use $\gamma = 64, 256$, and 1024. The corresponding split choices at $N/t = 4096$ are $(\alpha, \beta) = (32, 64), (32, 16)$, and $(32, 4)$ for the 64-byte, 256-byte, and 2-KB records, respectively. At $N/t = 32768$, they are $(32, 8), (16, 4)$, and $(16, 1)$, respectively. Increasing γ reduces the number $c = \lceil t\gamma_0/\gamma \rceil$ of NTRU intermediates, so server time and total per-query communication decrease, while offline preprocessing increases. At $N/t = 4096$, server time decreases from 799 ms to 477 ms, while offline preprocessing increases from 6.7 s to 44.4 s. The same trade-off is observed at $N/t = 32768$. The 64-byte rows reproduce the corresponding entries of Table 4. The Keys and Total Comm. columns follow the communication accounting of Section 4.3.

Table 6: Peak server memory usage (GB).

Database	SimpleYPIR	KSPIR	HintlessPIR	InsPIRe ⁽²⁾	NTPIR
1 GB	9.3	4.5	5.5	7.9	5.5
8 GB	46.7	34.6	36.8	36.9	26.1

Table 7: PIR performance for larger records at 1 GB (single-threaded).

Record	γ	Offline (s)	Server (ms)	Throughput (MB/s)	Keys (KB)	Total Comm. (KB)
$N/t = 4096$						
64 B	64	6.7	799	1281	656	816
256 B	256	14.0	539	1900	560	720
2 KB	1024	44.4	477	2148	464	624
$N/t = 32768$						
64 B	64	0.7	165	6204	512	896
256 B	256	1.8	131	7816	416	800
2 KB	1024	5.6	123	8325	320	704

5.3.2 Discussion. Online and offline speedup. At $\gamma=64$, ring packing accounts for 80% of online server time, so the speedups in Table 2 yield a 1.34× to 1.81× end-to-end online improvement. The first-layer matrix-vector product is common to every Double-PIR scheme and accounts for the residual gap. With no matrix-vector product on the offline path, aggregate offline speedup reaches 41× at $N/t=65536$ (8 GB). A breakdown of these factors is provided in Appendix D.2.

Results at 8 GB (Table 5). At all four shared operating points, NTPIR achieves a 1.38× to 1.80× server-time speedup over InsPIRe⁽²⁾. At the highest-throughput configuration, NTPIR reaches 8,609 MB/s at $N/t=65536$, corresponding to a 1.38× speedup over InsPIRe⁽²⁾, a 1.8× speedup over SimpleYPIR, and a 2.0× speedup over HintlessPIR. Total per-query communication (848 to 1,200 KB) is 1.1× to 1.6× lower than SimpleYPIR (1,354 KB) and 3.5× to 4.9× lower than HintlessPIR (4,188 KB), while remaining within 2.1× to 3.6× of InsPIRe⁽²⁾ (238 to 563 KB). Offline preprocessing remains 7× to 41× faster than InsPIRe⁽²⁾ at 8 GB. A per-step timing breakdown is provided in Table 10 of Appendix C.2.

Key size trade-off. The key material of NTPIR ranges from 512 to 656 KB, which is 6.4× to 8.2× larger than that of InsPIRe⁽²⁾ (80 KB). The overhead comes from the NGS-format automorphism keys that are inherent to the NTRU-based design, since each such key is a vector of ℓ_N NTRU ciphertexts and the SPP second layer requires $|\mathcal{T}_{\text{spp}}| = \log_2 c$ such keys. Because every query uses fresh client keys, the corresponding key material is included in every Total Comm. entry and is not amortized across queries. The CRS and database preprocessing remain reusable because they contain only the random component $\mathbf{w}_{g_\gamma}$ of the packing key and other secret-independent material.

6 Conclusion

We presented NTPIR, an efficient single-server PIR protocol for the silent-preprocessing setting. The design combines pack-then-switch, fused second-layer retrieval via split-point picking [15],

and optimized ring arithmetic. On databases ranging from 1 to 8 GB, NTPIR achieves up to a 1.81× server-side online speedup over InsPIRe⁽²⁾ across shared operating points. At the highest-throughput configuration, NTPIR reaches 8,609 MB/s on 8 GB, corresponding to a 1.38× speedup over InsPIRe⁽²⁾. At 8 GB, offline preprocessing is 7–41× faster than InsPIRe⁽²⁾.

References

- [1] Asra Ali, Tancrède Lepoint, Sarvar Patel, Mariana Raykova, Philipp Schoppmann, Karn Seth, and Kevin Yeo. 2021. Communication–Computation Trade-offs in PIR. In *Proceedings of the 30th USENIX Security Symposium (USENIX Security 21)*. 1811–1828.
- [2] Sebastian Angel, Hao Chen, Kim Laine, and Srinath Setty. 2018. PIR with compressed queries and amortized query processing. In *2018 IEEE symposium on security and privacy (SP)*. IEEE, 962–979.
- [3] Ali Atiia and Keewoo Lee. 2026. Sharded PIR Design for the Ethereum State. Ethereum Research. <https://ethresear.ch/t/sharded-pir-design-for-the-ethereum-state/24552>
- [4] Michael Backes, Aniket Kate, Matteo Maffei, and Kim Pecina. 2012. Obliviad: Provably secure and practical online behavioral advertising. In *2012 IEEE Symposium on Security and Privacy*. IEEE, 257–271.
- [5] Alexander Bienstock, Sarvar Patel, Joon Young Seo, and Kevin Yeo. 2024. Batch {PIR} and labeled {PSI} with oblivious ciphertext compression. In *33rd USENIX Security Symposium (USENIX Security 24)*. 5949–5966.
- [6] Elette Boyle, Niv Gilboa, and Yuval Ishai. 2016. Function secret sharing: Improvements and extensions. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 1292–1303.
- [7] Zvika Brakerski and Vinod Vaikuntanathan. 2011. Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages. In *Advances in Cryptology – CRYPTO 2011 (Lecture Notes in Computer Science, Vol. 6841)*. Springer, 505–524. doi:10.1007/978-3-642-22792-9_29
- [8] Alexander Burton, Samir Jordan Menon, and David J Wu. 2024. Respire: High-rate PIR for databases with small records. In *Proceedings of the 2024 on ACM SIGSAC conference on computer and communications security*. 1463–1477.
- [9] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. 2020. TFHE: Fast Fully Homomorphic Encryption Over the Torus. I. Chillotti et al. *Journal of Cryptology* 33, 1 (2020), 34–91.
- [10] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. 1998. Private information retrieval. *Journal of the ACM (JACM)* 45, 6 (1998), 965–981.
- [11] Henry Corrigan-Gibbs and Dmitry Kogan. 2020. Private Information Retrieval with Sublinear Online Time. In *Advances in Cryptology – EUROCRYPT 2020*. Springer, 44–75.
- [12] Junfeng Fan and Frederik Vercauteren. 2012. Somewhat Practical Fully Homomorphic Encryption. *Cryptology ePrint Archive*, Paper 2012/144. <https://eprint.iacr.org/2012/144>
- [13] Niv Gilboa and Yuval Ishai. 2014. Distributed point functions and their applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 640–658.
- [14] Trinabh Gupta, Natacha Crooks, Whitney Mulhern, Srinath Setty, Lorenzo Alvisi, and Michael Walfish. 2016. Scalable and private media consumption with popcorn. In *13th USENIX symposium on networked systems design and implementation (NSDI 16)*. 91–107.
- [15] Jiaxing He, Kang Yang, Guofeng Tang, Zhangjie Huang, Li Lin, Changzheng Wei, Ying Yan, and Wei Wang. 2024. Rhombus: Fast homomorphic matrix-vector multiplication for secure two-party inference. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*. 2490–2504.
- [16] Alexandra Henzinger, Matthew M. Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn, and Vinod Vaikuntanathan. 2023. One Server for the Price of Two: Simple and Fast Single-Server Private Information Retrieval. In *Proceedings of the 32nd USENIX Security Symposium (USENIX Security 23)*. 2009–2026.
- [17] Alexander Hoover, Sarvar Patel, Giuseppe Persiano, and Kevin Yeo. 2025. Plinko: single-server PIR with efficient updates via invertible PRFs. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 3–33.

- [18] Daniel Kales, Christian Rechberger, Thomas Schneider, Matthias Senker, and Christian Weinert. 2019. Mobile private contact discovery at scale. In *28th USENIX Security Symposium (USENIX Security 19)*. 1447–1464.
- [19] Paul Kirchner and Pierre-Alain Fouque. 2017. Revisiting lattice attacks on over-stretched NTRU parameters. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 3–26.
- [20] Eyal Kushilevitz and Rafail Ostrovsky. 1997. Replication is not needed: Single database, computationally-private information retrieval. In *Proceedings 38th annual symposium on foundations of computer science*. IEEE, 364–373.
- [21] Ben Laurie. 2014. Certificate transparency. *Commun. ACM* 57, 10 (2014), 40–46.
- [22] Kang Hoon Lee and Ji Won Yoon. 2026. Homomorphic Field Trace Revisited: Breaking the Cubic Noise Barrier. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2026, 1 (2026), 105–134.
- [23] Baiyu Li, Daniele Micciancio, Mariana Raykova, and Mark Schultz-Wu. 2024. Hintless single-server private information retrieval. In *Annual International Cryptology Conference*. Springer, 183–217.
- [24] Lucy Li, Bijeta Pal, Junade Ali, Nick Sullivan, Rahul Chatterjee, and Thomas Ristenpart. 2019. Protocols for checking compromised credentials. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 1387–1403.
- [25] Yuliang Lin, Baosheng Wang, Yi Wang, and Rongmao Chen. 2025. \textsc{Npir}: High-Rate PIR for Databases with Moderate-Size Records. *Cryptology ePrint Archive* (2025).
- [26] Ming Luo, Feng-Hao Liu, and Han Wang. 2024. Faster fhe-based single-server private information retrieval. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*. 1405–1419.
- [27] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. 2010. On ideal lattices and learning with errors over rings. In *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 1–23.
- [28] Rasoul Akhavan Mahdavi, Sarvar Patel, Joon Young Seo, and Kevin Yeo. 2026. InsPIRe: Communication-Efficient PIR with Server-Side Preprocessing. In *2026 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1463–1482. doi:10.1109/SP63933.2026.00061
- [29] Carlos Aguilar Melchor, Joris Barrier, Laurent Fousse, and Marc-Olivier Killijian. 2016. XPIR: Private information retrieval for everyone. *Proceedings on Privacy Enhancing Technologies* (2016), 155–174.
- [30] Samir Jordan Menon and David J Wu. 2022. Spiral: Fast, high-rate single-server PIR via FHE composition. In *2022 IEEE symposium on security and privacy (SP)*. IEEE, 930–947.
- [31] Samir Jordan Menon and David J Wu. 2024. {YPIR}:{High-Throughput}{Single-Server}{PIR} with silent preprocessing. In *33rd USENIX Security Symposium (USENIX Security 24)*. 5985–6002.
- [32] Muhammad Haris Mughees, Hao Chen, and Ling Ren. 2021. OnionPIR: Response efficient single-server PIR. In *Proceedings of the 2021 ACM SIGSAC conference on computer and communications security*. 2292–2306.
- [33] Muhammad Haris Mughees and Ling Ren. 2023. Vectorized batch private information retrieval. In *2023 IEEE symposium on security and privacy (SP)*. IEEE, 437–452.
- [34] Oded Regev. 2009. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)* 56, 6 (2009), 1–40.
- [35] Ling Ren, Muhammad Haris Mughees, and I Sun. 2024. Simple and practical amortized sublinear private information retrieval using dummy subsets. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*. 1420–1433.
- [36] Ruida Wang, Jincheol Ha, Xuan Shen, Xianhui Lu, Chunling Chen, Kunpeng Wang, and Jooyoung Lee. 2025. Refined TFHE leveled homomorphic evaluation and its application. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*. 2309–2323.
- [37] Han Xia and Mingsheng Wang. 2024. Single-Server PIR via NTRU-Based FHE: Simpler, Smaller, and Faster. In *2024 IEEE 9th European Symposium on Security and Privacy (EuroS&P)*. IEEE, 293–310.
- [38] Binwu Xiang, Jiang Zhang, Yi Deng, Yiran Dai, and Dengguo Feng. 2023. Fast Blind Rotation for Bootstrapping FHEs. In *Advances in Cryptology-CRYPTO 2023, Part IV (Lecture Notes in Computer Science, Vol. 14084)*. Springer, 3–36. doi:10.1007/978-3-031-38551-3_1
- [39] Mingxun Zhou, Andrew Park, Wenting Zheng, and Elaine Shi. 2024. Piano: extremely simple, single-server PIR with sublinear server computation. In *2024 IEEE symposium on security and privacy (SP)*. IEEE, 4296–4314.

Ethical Considerations.

This work presents NTPIR, a cryptographic protocol for single-server private information retrieval (PIR) in the silent-preprocessing model. PIR enables a client to retrieve a record from a server-hosted database without revealing which record is requested, thereby protecting query privacy. The research is purely theoretical and

systems-oriented: it designs and evaluates a new lattice-based protocol and does not involve human subjects, user data, deception, or real-world vulnerability analysis. As a privacy-enhancing technology, NTPIR is designed to strengthen user privacy in database access patterns. It does not introduce new attack surfaces, surveillance capabilities, or dual-use concerns beyond those inherent to general-purpose public-key encryption. We are not aware of any negative societal impact that could arise from the publication of this work.

Open Science.

We enumerate the artifacts required to evaluate the core contributions of this paper and describe how they can be accessed during review.

- **Source code.** The repository contains the full Rust implementation of NTPIR, including all packing algorithms (InspIRING, PackRLWEs, and SPP), the end-to-end PIR protocol, and the AVX-512 instructions. The repository is available at:

<https://github.com/antcplab/openntpir>

- **Benchmarking binaries.** The repository provides the correctness tests and benchmarking code used in our evaluation.
- **Parameter configurations.** All cryptographic parameters (ring degree, modulus, gadget base, decomposition length, key distributions) are fully specified in Section 5.1, Table 1, and Appendix B. No additional configuration files are needed beyond the constants defined in the source code.
- **Datasets.** No external or proprietary datasets are required. All experiments use synthetic databases with the sizes and record lengths specified in Section 5 and Appendix C. The databases are generated deterministically by the benchmarking code.

A Related Work

A.1 Single-Server PIR

Kushilevitz and Ostrovsky [20] established the feasibility of single-server PIR under computational assumptions.

FHE-based approaches. A line of work reduces communication by leveraging the ciphertext expansion capability of fully or somewhat homomorphic encryption (FHE/SHE). XPIR [29] first applied ring-LWE-based FHE to PIR, compressing queries via ciphertext packing but incurring high server computation. SealPIR [2] introduced query compression through ciphertext expansion and Galois automorphisms, reducing query size to a single ciphertext while amortizing computation across multiple queries. MulPIR [1] extended this approach by exploiting the multiplicative homomorphism of the BFV scheme to reduce server computation, demonstrating that deeper homomorphic circuits can trade modest noise growth for fewer ciphertext–plaintext multiplications. OnionPIR [32] further reduced response size by composing multiple layers of homomorphic encryption, using a combination of SHE and leveled FHE to compress each response element. Spiral [30] unified these ideas through a composition of Regev (matrix-based) and GSW encryption, introducing ciphertext translation techniques that enable new trade-offs between communication and computation. Spiral achieves the best balance among server-stateful schemes, offering low communication with moderate server cost. Respire [8]

addressed databases with small records (below 256 bytes), achieving high rate by combining RLWE packing with modulus-switching compression.

Offline/online separation and preprocessing. Corrigan-Gibbs and Kogan [11] initiated the study of PIR with preprocessing, showing that a client-side offline phase can reduce online server time to sublinear. Henzinger et al. [16] proposed SimplePIR and DoublePIR, which exploit the LWE structure to achieve extremely fast online computation via a single matrix–vector product over the database. SimplePIR yields the highest raw throughput among all known schemes but requires the client to store a hint proportional to $\sqrt{N}$. DoublePIR adds a second recursive retrieval to collapse client storage to a fixed size.

Silent-preprocessing schemes. Li et al. [23] introduced HintlessPIR, the first scheme to eliminate both per-client server state and client-side hints: the client sends key-switching material as part of each query, and the server preprocesses only database-dependent components independently of any client. YPIR [31] introduced silent preprocessing via puncturable pseudorandom functions, enabling the server to generate CRS-dependent material from a short seed and substantially reducing offline time. KSPIR [26] replaced the ciphertext expansion step in Spiral with dimension folding, a simpler homomorphic procedure that reduces key-switching overhead and improves server throughput. InsPIRe [28] unified these advances into a cohesive Double PIR framework whose core insight is the InspiRING packing algorithm: a tree of automorphisms and key switches that packs LWE ciphertexts into RLWE format entirely over a public CRS, enabling offline precomputation and achieving the best known communication–computation trade-off among server-stateless schemes.

Other models. Piano [39] and Plinko [17] operate in the client-stateful model with client-specific hints, where the client streams the database during an offline phase to build a compact local structure that enables sublinear online server computation. However, hint acquisition requires communication proportional to the database size and must be repeated upon every update.

A.2 NTRU-Based PIR

NTRU ciphertexts consist of a single ring element, half the size of an RLWE ciphertext pair, making them attractive for PIR constructions where response compactness is critical.

Xia and Wang [37] proposed NTRU-PIR, the first PIR protocol built on NTRU-based FHE. Their construction directly replaces RLWE with NTRU inside the Spiral framework [30], inheriting its server-stateful architecture. By substituting NTRU for RLWE, NTRU-PIR halves the response size relative to Spiral while maintaining comparable computation, though the underlying pipeline—query expansion, homomorphic absorption, and response extraction—remains structurally identical to Spiral.

Lin et al. [25] proposed NPIR, which introduces an NTRU packing technique that compresses the constant terms of multiple NTRU encodings into a single encoding, targeting databases with moderate-size records (tens of kilobytes). Like NTRU-PIR, NPIR operates in the server-stateful model.

Both NTRU-PIR and NPIR belong to the server-stateful Spiral lineage and apply NTRU as a drop-in replacement without altering

the algorithmic structure. In contrast, NTPIR targets the server-stateless Double PIR setting and introduces pipeline-level redesigns around NTRU, yielding structural improvements beyond a format substitution (see Section 1.1).

A.3 Multi-Server PIR and Batch PIR

Multi-server PIR. Chor et al. [10] introduced PIR in the multi-server, information-theoretic setting. Gilboa and Ishai [13] proposed distributed point functions (DPFs), which enable two-server PIR with polylogarithmic communication. Boyle, Gilboa, and Ishai [6] extended DPFs to function secret sharing (FSS), yielding efficient two-server PIR constructions that remain the basis of most practical multi-server schemes [17, 39]. While multi-server PIR can achieve lower computation per query, the non-collusion assumption between servers is difficult to enforce in practice, motivating the single-server setting.

Batch PIR. Batch PIR amortizes the cost of retrieving multiple records in a single round. SealPIR [2] introduced probabilistic batch codes based on cuckoo hashing to distribute batch queries across independent PIR instances. Mughees and Ren [33] further improved batch efficiency by packing multiple small entries into a single ciphertext via vectorization. Bienstock et al. [5] proposed oblivious ciphertext compression to reduce response size in batch PIR, combining it with labeled PSI applications. Our work focuses on the single-query setting. Combining NTPIR with batch techniques is an orthogonal direction for future work.

B Noise Analysis for Homomorphic Operations

This appendix derives noise variance bounds for the homomorphic operations used in Sections 3 and 4. Throughout this appendix, a lowercase e denotes the noise already carried by an input ciphertext, whereas E_{op} denotes the additive noise introduced by an operation op . Thus, an output noise is written as $e_{\text{out}} = e_{\text{in}} + E_{\text{op}}$ whenever the operation preserves the input noise without rescaling. We use V_{op} for the variance upper bound on E_{op} , and σ_{out}^2 for the accumulated noise variance of an output ciphertext. The RLWE and NTRU gadgets $\mathbf{g}_R$ and $\mathbf{g}_N$ are defined in Section 3.2.2. Following InsPIRe [28], we analyze these noise terms under the independence heuristic, which treats the error terms in intermediate computations as independent and adds their variances.

B.1 RLWE Operations

Lemma 1 (RLWE Key Switching). *Let $\text{ct} \in \text{RLWE}_{s_1, q_R}(\mu)$ with noise e and $\text{KSK}_{s_1 \rightarrow s_2}$ be a key-switching key whose errors variance is σ_{LWE}^2 . Key switching produces $\text{ct}' \in \text{RLWE}_{s_2, q_R}(\mu)$ whose noise satisfies $e' = e + E_{\text{ks}}^{\text{RLWE}}$. The variance of $E_{\text{ks}}^{\text{RLWE}}$ satisfies*

$$V_{\text{ks}}^{\text{RLWE}} \leq \mathbb{E}[\|\mathbf{s}_1\|_2^2] \cdot \frac{2^{2q_R}}{12} + \ell_R \cdot d \cdot \frac{B_R^2}{12} \cdot \sigma_{\text{LWE}}^2. \quad (28)$$

The first term is the approximation noise from the gadget decomposition residual and the second term is the encryption noise from the key-switching key.

PROOF. Instantiating Eq. (3) with $\mathbf{g}_R$ gives

$$\text{KSK}[j] = (w_j, s_2 w_j + g_{R, j s_1} + e_{\text{KSK}, j}).$$

For $(d_0, \dots, d_{\ell_R-1}) = \mathbf{g}_R^{-1}(a)$, the key-switching output is

$$\mathbf{ct}' = (0, b) - \sum_{j=0}^{\ell_R-1} d_j \cdot \mathbf{KSK}[j].$$

Consequently,

$$a' = - \sum_j d_j w_j, \quad b' = b - \sum_j d_j (s_2 w_j + g_{R,j} s_1 + e_{\mathbf{KSK},j}).$$

Decrypting under s_2 yields $b' - a' \cdot s_2 = \Delta\mu + e + \varepsilon \cdot s_1 - \sum_j d_j \cdot e_{\mathbf{KSK},j}$, where $\varepsilon = a - \sum_j d_j \cdot g_{R,j}$ is the decomposition residual with coefficients in $[-2^{\eta_R-1}, 2^{\eta_R-1}]$. For uniformly distributed coefficients of a , the residual coefficients are uniform in this interval, and $\text{Var}(\varepsilon[i]) \leq 2^{2\eta_R}/12$. The approximation noise $\varepsilon \cdot s_1$ therefore has variance at most $\|\mathbf{s}_1\|_2^2 \cdot 2^{2\eta_R}/12$ by the independence of the $\varepsilon[i]$. In addition, the signed decomposition digits satisfy $\mathbb{E}[d_j[i]^2] \leq B_R^2/12$. Hence the encryption noise $\sum_j d_j \cdot e_{\mathbf{KSK},j}$ has variance at most $\ell_R d B_R^2 \sigma_{\text{LWE}}^2/12$. $\square$

Lemma 2 (RLWE Automorphism). *Let $\mathbf{ct} \in \text{RLWE}_{s,q_R}(\mu)$, and the homomorphic automorphism $\text{HomAut}_k(\mathbf{ct})$ applies the Galois automorphism τ_k , which permutes coefficients without noise increase, followed by a single key switching. The noise growth equals that of one key-switching operation: $V_{\text{auto}}^{\text{RLWE}} = V_{\text{ks}}^{\text{RLWE}}$.*

Lemma 3 (RLWE Modulus Switching). *Let $(a, b) \in R_{q_R}^2$ satisfy $b - a\tilde{s} = \left\lfloor \frac{q_R}{p} \right\rfloor \mu + e$. Define*

$$\bar{a} = \left\lfloor \frac{q_N}{q_R} a \right\rfloor, \quad \bar{b} = \left\lfloor \frac{q_N}{q_R} b \right\rfloor.$$

The resulting ciphertext satisfies

$$\bar{b} - \bar{a}\tilde{s} = \left\lfloor \frac{q_N}{p} \right\rfloor \mu + \bar{e}.$$

If $\text{Var}(e) \leq \sigma^2$, then the variance of $\bar{e}$ satisfies

$$\text{Var}(\bar{e}) \leq \left(\frac{q_N}{q_R} \right)^2 \sigma^2 + \frac{1 + d\sigma_s^2}{12}. \quad (29)$$

PROOF. Let $r_a = \bar{a} - (q_N/q_R)a$ and $r_b = \bar{b} - (q_N/q_R)b$ denote the coefficient-wise rounding errors. The decryption relation gives

$$\bar{e} = \frac{q_N}{q_R} e + r_b - r_a \tilde{s} + \left(\frac{q_N}{q_R} \left\lfloor \frac{q_R}{p} \right\rfloor - \left\lfloor \frac{q_N}{p} \right\rfloor \right) \mu.$$

Because q_R , q_N , and p are powers of two with $p \mid q_N \mid q_R$, the last term is zero. The centered rounding errors have variance at most $1/12$. Moreover, $\mathbb{E}[\|\tilde{s}\|_2^2] = d\sigma_s^2$. Under the independence assumption stated at the beginning of this appendix, the three remaining terms therefore give Eq. (29). $\square$

Lemma 4 (Embedding Noise). *Let $(\mathbf{a}_i, b_i)$ be an LWE sample under keys $\mathbf{s}$. The embedding of Section 3.2.1 computes $\mathbf{a}'_i = \lfloor \mathbf{a}_i/\gamma \rfloor$ by modulus switching. The residual $\mathbf{r}_i = \mathbf{a}_i - \gamma \mathbf{a}'_i \in [0, \gamma)^d$ contributes the embedding error $E_{\text{emb}} = \langle \mathbf{r}_i, \mathbf{s} \rangle$, whose variance satisfies*

$$V_{\text{emb}} := \text{Var}(E_{\text{emb}}) \leq \frac{\mathbb{E}[\|\mathbf{s}\|^2] (\gamma-1)(2\gamma-1)}{6} < \frac{\mathbb{E}[\|\mathbf{s}\|^2] \gamma^2}{3}, \quad (30)$$

and this term is added once per packing group.

PROOF. For the power-of-two parameters used by NTPIR, γ divides the LWE modulus, so each $r_i[k]$ is uniform on $\{0, \dots, \gamma-1\}$ and

$$\mathbb{E}[r_i[k]^2] = \frac{(\gamma-1)(2\gamma-1)}{6}.$$

Since the secret coefficients are centered and independent of the residual, E_{emb} is centered. Independence therefore gives

$$\text{Var}(E_{\text{emb}}) = \sum_k \mathbb{E}[r_i[k]^2] \mathbb{E}[s[k]^2] \leq \mathbb{E}[\|\mathbf{s}\|_2^2] \frac{(\gamma-1)(2\gamma-1)}{6},$$

which proves Eq. (30). Thus, a deterministic bound is $|E_{\text{emb}}| \leq (\gamma-1)\|\mathbf{s}\|_1$. $\square$

B.2 NTRU Operations

Lemma 5 (Plaintext–Ciphertext Multiplication). *Let $\mathbf{ct} \in \text{NTRU}_{f,q_N}(\mu)$ have independent subgaussian noise coefficients with variance as σ^2 , and let $z \in R$ be a plaintext polynomial. Then $z\mathbf{ct} \in \text{NTRU}_{f,q_N}(z\mu)$ has noise ze with*

$$\text{Var}(ze)[k] \mid z \leq \|z\|_2^2 \sigma^2. \quad (31)$$

PROOF. Multiplying the NTRU decryption by z gives $z\mathbf{ct}f = \Delta z\mu + ze$. Each coefficient of ze is a signed linear combination of the coefficients of e , with squared coefficient norm $\|z\|_2^2$. Equation (31) follows from independence and subgaussian closure under linear combinations. $\square$

Lemma 6 (Modulus Switching). *Let $\mathbf{ct} \in R_{q_N}$ satisfy $\mathbf{ct}f = \left\lfloor \frac{q_N}{p} \right\rfloor \mu + e$, where $\mathbb{E}[\|f\|_2^2] \leq d\sigma_f^2$. Apply modulus switching and define $\hat{\mathbf{ct}} = \left\lfloor \frac{q'}{q_N} \mathbf{ct} \right\rfloor \in R_{q'}$. If $p \mid q'$ and $\text{Var}(e) \leq \sigma^2$, then $\hat{\mathbf{ct}}f = \left\lfloor \frac{q'}{p} \right\rfloor \mu + e_{\text{ms}}$, where the variance satisfies*

$$\text{Var}(e_{\text{ms}}) \leq \left(\frac{q'}{q_N} \right)^2 \sigma^2 + \frac{d\sigma_f^2}{12}. \quad (32)$$

PROOF. Let $r_{\text{ct}} = \hat{\mathbf{ct}} - (q'/q_N)\mathbf{ct}$ denote the coefficient-wise rounding error. Expanding the decryption relation gives

$$\hat{\mathbf{ct}}f = \frac{q'}{q_N} \left\lfloor \frac{q_N}{p} \right\rfloor \mu + \frac{q'}{q_N} e + r_{\text{ct}}f.$$

The plaintext term equals $\lfloor q'/p \rfloor \mu$ because $p \mid q' \mid q_N$. Each coefficient of r_{ct} has variance at most $1/12$. Under the independence assumption, the rounding term $r_{\text{ct}}f$ has variance at most $d\sigma_f^2/12$, which proves Eq. (32). $\square$

Lemma 7 (External Product). *Let $u \in R_{q_N}$ be an arbitrary input polynomial and $\mathbf{CT} = \text{NGS}_{f,q_N}(m)$ have noise variance σ_{NTRU}^2 . For $(d_0, \dots, d_{\ell_N-1}) = \mathbf{g}_N^{-1}(u)$, write $u = \sum_j d_j g_{N,j} + \varepsilon$. Then the external product $c' = \sum_j d_j \mathbf{CT}[j]$ satisfies*

$$c'f = mu + E_{\text{ext}}, \quad E_{\text{ext}} = -m\varepsilon + \sum_j d_j e_{\text{NGS},j}, \quad (33)$$

and, for every coefficient k ,

$$\text{Var}(E_{\text{ext}}[k] \mid m) \leq \|m\|_2^2 \frac{2^{2\eta_N}}{12} + \ell_N d \frac{B_N^2}{12} \sigma_{\text{NTRU}}^2. \quad (34)$$

PROOF. Each NGS component satisfies $\text{CT}[j]f = e_{\text{NGS},j} + g_{\text{N},j}m$. Therefore $c'f = \sum_j d_j(e_{\text{NGS},j} + g_{\text{N},j}m) = m(u - \varepsilon) + \sum_j d_j e_{\text{NGS},j}$, which proves Eq. (33). Independence of the residual coefficients gives $\text{Var}((m\varepsilon)[k] \mid m) \leq \|m\|_2^2 2^{2\eta_{\text{N}}}/12$. Hence the digit products contribute at most $\ell_{\text{N}} d B_{\text{N}}^2 \sigma_{\text{NTRU}}^2/12$. $\square$

Lemma 8 (RLWE-to-NTRU Conversion Noise). *Let $(\bar{a}, \bar{b}) \in R_{q_{\text{N}}}^2$ satisfy*

$$\bar{b} - \bar{a}\bar{s} = \left\lfloor \frac{q_{\text{N}}}{p} \right\rfloor \mu + \bar{e},$$

where the coefficients of $\bar{s}$ are independent, centered, and have second moment σ_s^2 . Let $\text{KSK}_0 = \text{NGS}_{f, q_{\text{N}}}(1)$ and $\text{KSK}_1 = \text{NGS}_{f, q_{\text{N}}}(\bar{s} \bmod q_{\text{N}})$ have independent component errors of variance σ_{NTRU}^2 . Then the conversion

$$\text{ct}' = \langle g_{\text{N}}^{-1}(\bar{b}), \text{KSK}_0 \rangle - \langle g_{\text{N}}^{-1}(\bar{a}), \text{KSK}_1 \rangle$$

satisfies

$$\text{ct}'f = \left\lfloor \frac{q_{\text{N}}}{p} \right\rfloor \mu + \bar{e} + e_{\text{conv}},$$

where

$$\text{Var}(e_{\text{conv}}) \leq (1 + d\sigma_s^2) \frac{2^{2\eta_{\text{N}}}}{12} + 2\ell_{\text{N}} d \frac{B_{\text{N}}^2}{12} \sigma_{\text{NTRU}}^2. \quad (35)$$

Consequently,

$$\text{Var}(\bar{e} + e_{\text{conv}}) \leq \text{Var}(\bar{e}) + (1 + d\sigma_s^2) \frac{2^{2\eta_{\text{N}}}}{12} + 2\ell_{\text{N}} d \frac{B_{\text{N}}^2}{12} \sigma_{\text{NTRU}}^2. \quad (36)$$

PROOF. Apply Lemma 7 first with $u = \bar{b}$ and $m = 1$, and then with $u = \bar{a}$ and $m = \bar{s}$. The first external product contributes at most $\frac{2^{2\eta_{\text{N}}}}{12} + \ell_{\text{N}} d \frac{B_{\text{N}}^2}{12} \sigma_{\text{NTRU}}^2$, whereas the second contributes at most $d\sigma_s^2 \frac{2^{2\eta_{\text{N}}}}{12} + \ell_{\text{N}} d \frac{B_{\text{N}}^2}{12} \sigma_{\text{NTRU}}^2$, because $\mathbb{E}[\|\bar{s}\|_2^2] = d\sigma_s^2$. Subtracting the two external products reproduces $\bar{b} - \bar{a}\bar{s}$ and adds their two noise terms with opposite signs. Their variances add under the independence assumption, which proves Eq. (35). Adding the independent input noise $\bar{e}$ gives Eq. (36). $\square$

Lemma 9 (Key Switching with Gaussian Secrets). *Let $\text{ct} \in \text{NTRU}_{f_1, q_{\text{N}}}(\mu)$ have noise e , and let $\text{KSK} = \text{NGS}_{f_2, q_{\text{N}}}(f_1)$ have noise variance σ_{NTRU}^2 . Key switching produces $\text{ct}' \in \text{NTRU}_{f_2, q_{\text{N}}}(\mu)$ with $e' = e + E_{\text{ks}}^{\text{NTRU}}$, where, conditional on f_1 ,*

$$\text{Var}(E_{\text{ks}}^{\text{NTRU}}[k] \mid f_1) \leq \|f_1\|_2^2 \frac{2^{2\eta_{\text{N}}}}{12} + \ell_{\text{N}} d \frac{B_{\text{N}}^2}{12} \sigma_{\text{NTRU}}^2. \quad (37)$$

For the discrete Gaussian distribution used by NTPIR, we let σ_f^2 denote the coefficient second moment after conditioning on invertibility, so that $\mathbb{E}[\|f_1\|_2^2] = d\sigma_f^2$. Hence the expected increment is at most

$$V_{\text{ks}}^{\text{NTRU}} \leq d\sigma_f^2 \frac{2^{2\eta_{\text{N}}}}{12} + \ell_{\text{N}} d \frac{B_{\text{N}}^2}{12} \sigma_{\text{NTRU}}^2. \quad (38)$$

PROOF. Apply Lemma 7 with $u = \text{ct}$ and $m = f_1$. Since $\text{ct}f_1 = \Delta\mu + e$, Eq. (33) gives

$$\text{ct}'f_2 = f_1(\text{ct} - \varepsilon) + E_{\text{NGS}} = \Delta\mu + e - f_1\varepsilon + E_{\text{NGS}},$$

where $E_{\text{NGS}} = \sum_j d_j e_{\text{NGS},j}$. Thus $E_{\text{ks}}^{\text{NTRU}} = -f_1\varepsilon + E_{\text{NGS}}$, and Eq. (37) follows directly from Eq. (34). Taking expectation over the Gaussian key yields Eq. (38). $\square$

Lemma 10 (NTRU Automorphism). *The homomorphic automorphism $\text{HomAut}_k(\text{ct}) = \langle g_{\text{N}}^{-1}(\tau_k(\text{ct})), \text{NGS}_{f, q_{\text{N}}}(\tau_k(f)) \rangle$ permutes coefficients via τ_k and then applies the key-switching operation of Lemma 9 with $f_1 = \tau_k(f)$. Because automorphisms preserve the coefficient norm, $V_{\text{auto}}^{\text{NTRU}} = V_{\text{ks}}^{\text{NTRU}}$.*

B.2.1 PackRLWEs over NTRU Ciphertexts. Algorithm 5 gives the binary merge of Eq. (22), with $v_0 = h$ in the notation of Section 3.3.1.

Algorithm 5 PackRLWEs Merge over NTRU Ciphertexts

Input: NTRU ciphertexts $\text{ct}_0, \dots, \text{ct}_{n-1}$ ($n = 2^k$) and v_0 such that $v_0 + k \leq \log_2 d$.

Input: Automorphism keys $\{\text{KSK}_{\tau_v}\}_{v=v_0+1}^{v_0+k}$.

Output: Merged ciphertext ct_{out} .

```

1: for  $l = 1$  to  $k$  do
2:    $v \leftarrow v_0 + l$ 
3:    $s_v \leftarrow d/2^v$ 
4:    $n_l \leftarrow n/2^l$ 
5:   for  $j = 0$  to  $n_l - 1$  do
6:      $\text{ct}_+ \leftarrow \text{ct}_j + X^{s_v} \text{ct}_{j+n_l}$ 
7:      $\text{ct}_- \leftarrow \text{ct}_j - X^{s_v} \text{ct}_{j+n_l}$ 
8:      $\text{ct}_j \leftarrow \text{ct}_+ + \text{HomAut}_{\tau_v}(\text{ct}_-, \text{KSK}_{\tau_v})$ 
9: return  $\text{ct}_0$ 
```

The procedure uses $n - 1$ HomAut calls. Let V_l denote the maximum noise variance after level l , and suppose that the input variance is at most V_0 . The output noise of each merge contains two contributions from each input and the noise added by HomAut. The variance bounds above give

$$V_l \leq 4V_{l-1} + V_{\text{auto}}^{\text{NTRU}}, \quad V_k \leq 4^k V_0 + \frac{4^k - 1}{3} V_{\text{auto}}^{\text{NTRU}}. \quad (39)$$

Phase 3 of NTPIR applies the per-level modulus-switching technique of Section 3.3.2. The following lemma gives the resulting noise bound.

Lemma 11 (PackRLWEs with Per-Level Modulus Switching). *Let $\text{ct}_0, \dots, \text{ct}_{\beta-1}$, where $\beta = 2^h$, be NTRU ciphertexts at the same plaintext scale. Suppose that these ciphertexts are intermediate nodes after packing level v_0 , so their payload coefficients occupy*

$$\mathcal{I}_{v_0} = \left\{ j \frac{d}{2^{v_0}} : 0 \leq j < 2^{v_0} \right\}.$$

For each subsequent level $v = v_0 + 1, \dots, v_0 + h$, set $s_v = d/2^v$ and let τ_v be the automorphism $X \mapsto X^{2^{v_0+1}}$. Replace each ordinary PackRLWEs merge by

$$\tilde{\text{ct}} = \text{ModRaise}_{q_{\text{N}}/2 \rightarrow q_{\text{N}}}(\text{ModSwitch}_{q_{\text{N}} \rightarrow q_{\text{N}}/2}(\text{ct}_L - X^{s_v} \text{ct}_R)), \quad (40)$$

$$\text{ct}_{\text{merge}} = \tilde{\text{ct}} + \text{HomAut}_{\tau_v}(\tilde{\text{ct}}) + X^{s_v} \text{ct}_R. \quad (41)$$

After h levels, the output contains the packed payloads in $\mathcal{I}_{v_0+h}$ at the same plaintext scale as the inputs. If every input has noise variance at most σ_{in}^2 , then the output noise satisfies

$$\sigma_{\text{out}}^2 \leq \sigma_{\text{in}}^2 + h \left(V_{\text{auto}}^{\text{NTRU}} + \frac{d\sigma_f^2}{3} \right). \quad (42)$$

PROOF. Consider one merge at level v , and write $\phi_L = \text{ct}_L f$ and $\phi_R = \text{ct}_R f$. Modulus switching to $q_N/2$ followed by raising to q_N gives

$$\tilde{\text{ct}}_f = \frac{1}{2} (\phi_L - X^{s_v} \phi_R) + \frac{q_N}{2} U + E_{\text{MS}}, \quad (43)$$

for some polynomial U with binary coefficients. The rounding contribution is $E_{\text{MS}} = rf$. Since each centered rounding coefficient has variance at most $1/12$,

$$V_{\text{MS}}^{\text{NTRU}} \leq \frac{d\sigma_f^2}{12}. \quad (44)$$

Applying HomAut_{τ_v} to Eq. (43) applies τ_v to its phase and adds noise of variance at most $V_{\text{auto}}^{\text{NTRU}}$ by Lemma 10.

It remains to verify the payload coefficients. Since $s_v = d/2^v$, the indices in $\mathcal{I}_{v-1}$ are the even multiples of s_v , and

$$\tau_v(X^k) = X^k \quad \text{for } k \in \mathcal{I}_{v-1}, \quad \tau_v(X^k) = -X^k \quad \text{for } k \in s_v + \mathcal{I}_{v-1}.$$

Substituting Eq. (43) into Eq. (41) therefore shows that the coefficients indexed by $\mathcal{I}_{v-1}$ retain the payload of ct_L , whereas those indexed by $s_v + \mathcal{I}_{v-1}$ retain the shifted payload of ct_R . Moreover, the term $q_N(U + \tau_v(U))/2$ vanishes modulo q_N at both sets of indices. Hence the new payload positions are

$$\mathcal{I}_v = \mathcal{I}_{v-1} \cup (s_v + \mathcal{I}_{v-1}),$$

and their plaintext scale is unchanged.

At a position in $\mathcal{I}_{v-1}$, the variance contributed by the input is at most V_{v-1} , and the remaining contributions are $4V_{\text{MS}}^{\text{NTRU}}$ and $V_{\text{auto}}^{\text{NTRU}}$. At a position in $s_v + \mathcal{I}_{v-1}$, the variance contributed by the input is also at most V_{v-1} , and the only additional contribution is $V_{\text{auto}}^{\text{NTRU}}$. Thus the maximum variance V_v over the payload positions obeys

$$V_v \leq V_{v-1} + 4V_{\text{MS}}^{\text{NTRU}} + V_{\text{auto}}^{\text{NTRU}} \leq V_{v-1} + \frac{d\sigma_f^2}{3} + V_{\text{auto}}^{\text{NTRU}},$$

where the second inequality follows from Eq. (44). Iterating this recurrence over the h upper levels proves Eq. (42). $\square$

Lemma 12 (Noise through SPP). *Let*

$$\text{ct}_v f = \left\lfloor \frac{q_N}{p_{\text{SPP}}} \right\rfloor \hat{v} + e_v \pmod{q_N},$$

where e_v with variance $\alpha^2 \sigma_{\text{NTRU}}^2$. Let $\tilde{z}_{i_1, \tau}^{(b)} \in R$ denote the coefficient-wise centered lift of $z_{i_1, \tau}^{(b)}$, and define

$$Z^2 := \max_{b, i_1} \sum_{\tau \in \mathcal{G}} \|\tilde{z}_{i_1, \tau}^{(b)}\|_2^2.$$

If $\alpha = 2^{L_{\text{lo}}}$ and $\beta = 2^h$, the noise e_{SPP} of the Phase 3 response has variance at most

$$\sigma_{\text{SPP}}^2 \leq Z^2 \left(\alpha \sigma_{\text{NTRU}}^2 + L_{\text{lo}} V_{\text{auto}}^{\text{NTRU}} \right) + h \left(V_{\text{auto}}^{\text{NTRU}} + \frac{d\sigma_f^2}{3} \right). \quad (45)$$

Furthermore,

$$Z^2 \leq \alpha d \left(\frac{p_{\text{SPP}} - 1}{2} \right)^2. \quad (46)$$

PROOF. Phase 1 computes the ciphertexts level by level as described in Section 3.3.2. Each path from $\text{ct}_{v, \text{id}}$ to a ciphertext $\text{ct}_{v, \tau}$ contains at most L_{lo} applications of Lemma 10. Hence

$$\text{ct}_{v, \tau} f = \left\lfloor \frac{q_N}{p_{\text{SPP}}} \right\rfloor \tau(\hat{v}) + \tau(e_v) + E_\tau \pmod{q_N},$$

where $E_{\text{id}} = 0$ and every E_τ has variance at most $L_{\text{lo}} V_{\text{auto}}^{\text{NTRU}}$. Equation (24) therefore shows that the evaluation noise of a Phase 2 intermediate is

$$\sum_{\tau \in \mathcal{G}} \tilde{z}_{i_1, \tau}^{(b)} \tau(e_v) + \sum_{\tau \in \mathcal{G} \setminus \{\text{id}\}} \tilde{z}_{i_1, \tau}^{(b)} E_\tau.$$

The terms $\tau(e_v)$ are not independent because they are obtained from the same error polynomial. Since automorphisms permute coefficients without changing their norm, Cauchy–Schwarz gives

$$\text{Var} \left(\sum_{\tau \in \mathcal{G}} \tilde{z}_{i_1, \tau}^{(b)} \tau(e_v) \right) \leq \alpha \sigma_{\text{NTRU}}^2 \sum_{\tau \in \mathcal{G}} \|\tilde{z}_{i_1, \tau}^{(b)}\|_2^2 \leq \alpha Z^2 \sigma_{\text{NTRU}}^2.$$

Lemma 5 bounds the variance of the second sum by $Z^2 L_{\text{lo}} V_{\text{auto}}^{\text{NTRU}}$. Thus every Phase 2 intermediate has noise variance at most $Z^2 (\alpha \sigma_{\text{NTRU}}^2 + L_{\text{lo}} V_{\text{auto}}^{\text{NTRU}})$. Applying Lemma 11 over the $h = \log_2 \beta$ upper levels gives Eq. (45).

Finally, each centered z-polynomial has d coefficients of absolute value at most $(p_{\text{SPP}} - 1)/2$. Summing the squared norms of the z -polynomials proves Eq. (46). $\square$

Lemma 13 (SPP Encoding). *Let $w_\delta \in R_{p_{\text{SPP}}}$ be a selected base- p_{SPP} digit polynomial, and let $\tilde{w}_\delta \in R$ denote its coefficient-wise centered lift. Let $M_\delta \in R$ denote the integer representative of the plaintext on the corresponding payload positions after Phase 2, using the centered lifts of the z -polynomials in Equation (24). For $q_N = 2^{45}$ and $p_{\text{SPP}} = 31$, there exists $K_\delta \in R$ such that*

$$\|K_\delta\|_\infty \leq \left\lfloor \frac{(\alpha + 1)(p_{\text{SPP}} - 1)}{2p_{\text{SPP}}} \right\rfloor$$

and $M_\delta = \tilde{w}_\delta + p_{\text{SPP}} K_\delta$. For the parameters in Tables 4, 5, and 7, $\alpha \leq 32$, and hence $\|K_\delta\|_\infty \leq 15$. If e_{SPP} denotes the noise of the Phase 3 response, then

$$\text{ct}_{\text{resp}}^{(\delta)} f = \left\lfloor \frac{q_N}{p_{\text{SPP}}} \right\rfloor \cdot \tilde{w}_\delta + (e_{\text{SPP}} - K_\delta) \pmod{q_N} \quad (47)$$

on every payload position. Hence Layer 1 recovers w_δ whenever

$$\|e_{\text{SPP}}\|_\infty + 15 < \frac{1}{2} \left\lfloor \frac{q_N}{p_{\text{SPP}}} \right\rfloor. \quad (48)$$

PROOF. The z -tree implements the lower L_{lo} packing levels in $R_{p_{\text{SPP}}}$. The z -polynomials in the root node are multiplied by α^{-1} in $R_{p_{\text{SPP}}}$ before Equation (24) is evaluated. Consequently, on the payload positions,

$$M_\delta \equiv \alpha^{-1} \alpha w_\delta \equiv w_\delta \pmod{p_{\text{SPP}}},$$

and thus $K_\delta = (M_\delta - \tilde{w}_\delta)/p_{\text{SPP}}$ belongs to R . Each centered z -polynomial coefficient has absolute value at most $(p_{\text{SPP}} - 1)/2$. Since applying any automorphism to the one-hot query plaintext gives a monomial with coefficient in $\{-1, 1\}$, Equation (24) gives

$$\|M_\delta\|_\infty \leq \frac{\alpha(p_{\text{SPP}} - 1)}{2}.$$

Since $\|\tilde{w}_\delta\|_\infty \leq (p_{\text{SPP}} - 1)/2$ and K_δ has integer coefficients,

$$\|K_\delta\|_\infty \leq \left\lfloor \frac{(\alpha + 1)(p_{\text{SPP}} - 1)}{2p_{\text{SPP}}} \right\rfloor.$$

For $p_{\text{SPP}} = 31$ and $\alpha \leq 32$, this bound is at most 15. Lemma 11 preserves these payload values through Phase 3. Since $q_N - 1$ is divisible by p_{SPP} , we have

$$\left\lfloor \frac{q_N}{p_{\text{SPP}}} \right\rfloor M_\delta = \left\lfloor \frac{q_N}{p_{\text{SPP}}} \right\rfloor \tilde{w}_\delta + (q_N - 1)K_\delta \equiv \left\lfloor \frac{q_N}{p_{\text{SPP}}} \right\rfloor \tilde{w}_\delta - K_\delta \pmod{q_N}.$$

This proves Eq. (47). Under Eq. (48),

$$\|e_{\text{SPP}} - K_\delta\|_\infty < \frac{1}{2} \left\lfloor \frac{q_N}{p_{\text{SPP}}} \right\rfloor.$$

Rounding therefore recovers $\tilde{w}_\delta$, whose reduction modulo p_{SPP} is w_δ . $\square$

B.3 End-to-End Noise Composition

The following lemma composes the bounds above through the first three stages of NTPIR. It is stated independently of any concrete parameter choice.

Lemma 14 (Noise through Stage 3). *Let $n_{\text{row}} = N/t$, and let $\mathbf{D}_{:,j} \in \mathbb{Z}_p^{n_{\text{row}}}$ denote the j -th database column, represented by fixed integer lifts. Suppose that the input LWE errors have variance σ_{LWE}^2 , and let σ_s^2 denote the coefficient second moment of the LWE/RLWE secret. Define*

$$\sigma_1^2 := \max_{j \in [ty_0]} \|\mathbf{D}_{:,j}\|_2^2 \sigma_{\text{LWE}}^2. \quad (49)$$

The RLWE ciphertext obtained by packing γ Stage 1 outputs has noise variance at most

$$\sigma_R^2 := \sigma_1^2 + V_{\text{emb}} + (\gamma - 1)V_{\text{ks}}^{\text{RLWE}}, \quad V_{\text{emb}} \leq d\sigma_s^2 \frac{(\gamma - 1)(2\gamma - 1)}{6}. \quad (50)$$

After modulus switching to q_N and RLWE-to-NTRU conversion, the Stage 2 output has noise variance at most

$$\begin{aligned} \sigma_2^2 &:= \left(\frac{q_N}{q_R} \right)^2 \sigma_R^2 + \frac{1 + d\sigma_s^2}{12} \\ &\quad + (1 + d\sigma_s^2) \frac{2^{2\eta_N}}{12} + 2\ell_N d \frac{B_N^2}{12} \sigma_{\text{NTRU}}^2. \end{aligned} \quad (51)$$

Finally, after modulus switching the NTRU ciphertext from q_N to q' , the Stage 3 intermediate has noise variance at most

$$\sigma_{\text{pack}}^2 := \left(\frac{q'}{q_N} \right)^2 \sigma_2^2 + \frac{d\sigma_f^2}{12}. \quad (52)$$

For the canonical representative $\hat{\text{ct}}[k] \in \{0, \dots, q' - 1\}$, the subsequent base- p_{SPP} decomposition satisfies

$$\hat{\text{ct}}[k] = \sum_{\delta=0}^{\tau-1} p_{\text{SPP}}^\delta w_\delta[k] \quad \text{over the integers,}$$

because $p_{\text{SPP}}^\tau \geq q'$. It is therefore exact and introduces no additional noise.

PROOF. For each Stage 1 output, the noise is $\sum_{i=0}^{n_{\text{row}}-1} \mathbf{D}[i, j]e_i$. Independence of the e_i gives Eq. (49). The packing step preserves the selected input noise and adds one embedding error from Lemma 4 and $\gamma - 1$ key-switching errors from Lemma 1, which gives Eq. (50). Equation (51) follows by applying Lemma 3 and then Lemma 8. Applying Lemma 6 to the resulting NTRU ciphertext gives Eq. (52). $\square$

B.4 Concrete Noise Budget

We instantiate the lemmas of Sections B.1 and B.2 with the parameters of Table 1 and the configurations in Tables 4, 5, and 7 to verify the correctness condition of Theorem 3. Following the noise-tracking style of InsPIRe [28], we report the resulting noise standard deviation at the end of each protocol stage and check it against the corresponding decryption threshold. The protocol has two decryption events with different thresholds: the packed NTRU intermediate (Stage 3 output) is decrypted under the smaller modulus $q' = 2^{30}$, requiring $\|e_{\text{pack}}\|_\infty < q'/(2p) = 2^{13}$. The SPP response (Stage 4 output) is decrypted under the NTRU modulus $q_N = 2^{45}$ with plaintext modulus $p_{\text{SPP}} = 31$, and its sufficient decoding condition is Eq. (48).

To bound these two events, we instantiate Eqs. (49)–(52) and Lemma 12, using $\|\mathbf{D}_{:,j}\|_2^2 \leq (N/t)p^2$ for Stage 1 and $Z^2 \leq \alpha d((p_{\text{SPP}} - 1)/2)^2$ for Stage 4. In this calculation, the base- p_{SPP} digit extraction is exact by Lemma 14. At Stage 3, the prior Stage 2 variance in R_{q_N} is rescaled by $(q'/q_N)^2 = 2^{-30}$, and the noise includes the modulus-switching rounding term $\langle r, f \rangle$. The Stage 4 bound then accounts for the multiplication of the query and automorphism errors by the centered lifts of the z -polynomials, followed by the upper packing levels of Lemma 11. The maximum Stage 1 bound uses $N/t = 65536$. The Stage 2 and Stage 3 bounds use $\gamma = 1024$, and the Stage 4 bound uses $(\alpha, \beta) = (32, 64)$, which gives its largest value among the configurations in Tables 4, 5, and 7. These choices therefore upper-bound all reported configurations. Accordingly, Table 8 summarizes the maximum noise at each stage, expressed as $\log_2 \sigma$ in bits. Stage 3 retains a margin of at least 4.6 bits, and Stage 4 retains a margin of 4.9 bits. Thus, the Layer 1 and Layer 2 decoding conditions hold strictly for the parameters of Section 5.1.

For the Stage 4 configuration $(\alpha, \beta) = (32, 64)$, Lemma 13 gives $\|K_\delta\|_\infty \leq 15$ in the Layer 1 decoding condition. Combining the two decoding events with a union bound gives

$$\begin{aligned} \Pr[\text{failure}] &\leq 2\gamma_0 \exp\left(-\frac{(q'/(2p))^2}{8\sigma_{\text{pack}}^2}\right) \\ &\quad + 2\tau d \exp\left(-\frac{\left(\frac{1}{2} \left\lfloor \frac{q_N}{p_{\text{SPP}}} \right\rfloor - 15\right)^2}{8\sigma_{\text{SPP}}^2}\right). \end{aligned}$$

The first term corresponds to Layer 2 and applies a union bound over the γ_0 coefficients of the target record. Layer 1 must recover all d coefficients of each of the τ digit polynomials before the NTRU intermediate at modulus q' can be reconstructed exactly. The second term therefore applies a union bound over τd coefficients. Substituting these parameters into the bound above gives

$$\Pr[\text{failure}] < 2^{-40}$$

for all reported configurations, including $\gamma_0 = 1024$ in Table 7.

C Extended Evaluation

This appendix supplements Section 5.3 with scaling data, per-step timing breakdowns, and the SPP split-point parameter sweep.

Table 8: Maximum noise standard deviation ($\log_2 \sigma$, in bits) and decryption thresholds over the reported configurations.

Stage	Maximum	Threshold
1. First-layer retrieval	36.0	—
2. LWE-to-NTRU packing	22.0	—
3. Modulus switch and digit decomposition	8.3	13
4. Second-layer retrieval	34.1	39.0

Table 9: NTPIR scaling across database sizes ($N/t=32768$, single-threaded, Xeon 3.0 GHz).

DB	Layout ($n_{\text{row}} \times n_{\text{col}}$)	MatMul (ms)	Pack+KS (ms)	Online (ms)	Thru. (GB/s)	Comm. (KB)	Offline (s)
256 MB	32768×4096	18.4	20.3	42.9	6.0	800	0.3
512 MB	32768×8192	37.1	40.0	84.2	6.1	848	0.5
1 GB	32768×16384	74.0	78.8	165.0	6.2	896	0.7
2 GB	32768×32768	147.9	159.6	331.2	6.2	944	1.8
4 GB	32768×65536	296.4	321.6	664.6	6.2	992	4.9
8 GB	32768×131072	594.8	648.7	1338.8	6.1	1040	9.8

C.1 Scaling across Database Sizes

Table 9 reports scaling results from 256 MB to 8 GB using the same configuration, with $N/t=32768$, the layout $32768 \times n_{\text{col}}$, and $\gamma=64$.

As the database size doubles, both the matrix-vector product and packing time approximately double, while throughput remains between 6.0 and 6.2 GB/s. The query and response sizes remain fixed at 272 KB and 112 KB, respectively. The key size increases by 48 KB whenever c doubles because SPP requires $\log_2 c$ automorphism keys, giving total communication from 800 to 1,040 KB. Offline preprocessing follows the same trend, increasing from 0.3 to 9.8 s.

C.2 Per-Step Timing Breakdown

To examine the contribution of each processing step, Table 10 decomposes the server time for each N/t configuration reported in Tables 4 and 5, using the packing factors specified in Section 5.3.1. The first-layer matrix-vector product cost is proportional to the database size and approximately constant across the four N/t configurations within each database-size block. First-layer packing dominates at small N/t and decreases as N/t grows, since larger N/t implies smaller t and therefore fewer packing groups. The decomposition and SPP costs follow the same pattern because both scale with the number of groups.

Table 10: Per-step timing (ms) for NTPIR at 1 GB and 8 GB (single-threaded, Xeon 3.0 GHz).

N/t	1 GB			
	4096	8192	16384	32768
γ	64	128	64	64
c	2048	512	512	256
L1 MatMul	65.9	66.6	69.0	74.0
L1 Pack+KS	641.1	259.5	160.0	78.8
Decompose	28.2	6.7	6.4	2.6
L2 SPP	64.2	17.3	17.4	9.6
Total	799	350	253	165

N/t	8 GB			
	8192	16384	32768	65536
γ	256	128	128	128
c	2048	2048	1024	512
L1 MatMul	540.7	564.6	597.8	664.8
L1 Pack+KS	1820.3	1049.9	519.9	261.9
Decompose	28.2	27.8	13.3	7.3
L2 SPP	64.4	64.4	33.2	17.4
Total	2454	1707	1164	952

C.3 Packing Factor

To further examine the packing cost, Table 11 reports the effect of γ on online server time for the 1 GB database at $N/t=4096$.

Table 11: Effect of γ on online server time (1 GB, $N/t=4096$, single-threaded).

γ	Groups	MatMul	Pack+KS	Decompose	SPP	Online (ms)
64	2048	65.9	641.1	28.2	64.2	799.5
128	1024	67.1	519.8	14.8	33.0	634.7
256	512	67.0	451.8	6.6	17.4	542.8
512	256	67.1	420.2	1.7	9.1	498.1
1024	128	66.7	405.9	0.7	5.3	478.6

C.4 SPP Split-Point Parameter Sweep

We next examine the split-point parameter used by SPP. Let $u = h_p + L_{l_0}$ denote the split point in the packing tree. The lower plaintext z -tree has $L_{l_0} = u - h_p$ levels, while the upper ciphertext PackRLWEs merge has $L_{hi} = \log_2 d - u = \log_2 c - L_{l_0}$ levels. Table 12 sweeps u for the concrete second-layer configuration ($r=d\tau=14336$, $c=256$, $d=2048$, seven row blocks), reporting the number of HomAut calls contributed by each phase.

Table 12: Split-point sweep for second-layer SPP ($r=14336$, $c=256$, $d=2048$, seven row blocks).

u	α	Phase 1	Phase 3	HomAut calls
4	2	1	889	890
5	4	3	441	444
6	8	7	217	224
7	16	15	105	120
8	32	31	49	80 $\leftarrow u^*$
9	64	63	21	84
10	128	127	7	134
11	256	255	0	255

The total count is $(\alpha - 1) + 7(\beta - 1)$ with $\alpha = 2^{u-h_p}$, $\beta = c/\alpha$, and $h_p = \log_2(d/c) = 3$. It is minimized at $u^* = 8$, for which $(\alpha, \beta) = (32, 8)$. The 31 Phase 1 HomAut results are shared across all seven row blocks, while Phase 3 uses seven HomAut calls per block. The resulting 80 HomAut calls are $22.3\times$ fewer than the 1,785 calls required by applying a full c -leaf PackRLWEs tree independently to the seven row blocks.

D Detailed Performance Analysis

This appendix provides additional details on the online and offline speedups reported in Section 5.3.

D.1 Comparison with InsPIRe Variants

InsPIRe [28] reports three constructions: InsPIRe₀, InsPIRe⁽²⁾, and InsPIRe. Since InsPIRe₀ is evaluated only for 1-bit records, Table 13 compares NTPIR with the two variants evaluated for 64-byte records. For each construction and database size, the table reports the configuration with the highest throughput. The NTPIR rows are taken from Tables 4 and 5. They use $(N/t, \gamma) = (32768, 64)$ at 1 GB and $(N/t, \gamma) = (65536, 128)$ at 8 GB.

Table 13: Comparison with the InsPIRe variants evaluated for 64-byte records. Each row reports the configuration with the highest throughput for that construction and database size.

DB	Construction	Offline (s)	Total Comm. (KB)	Server (ms)	Throughput (MB/s)
1 GB	InsPIRe ⁽²⁾	33.0	347	302	3424
	InsPIRe	93.4	628	198	5141
	NTPIR	0.7	896	165	6204
8 GB	InsPIRe ⁽²⁾	88.7	986	1085	7579
	InsPIRe	440.6	2056	962	8533
	NTPIR	2.7	1200	952	8609

D.2 Speedup Sources

Section 5.2 quantifies the two main arithmetic factors. FFT arithmetic over $\mathbb{Z}_{2^{64}}$ removes Barrett or Montgomery reductions, and approximate gadget decomposition reduces the RLWE decomposition length from 4 to $\ell_R = 2$, halving the number of polynomial multiplications per key-switching call.

In each of the $\gamma - 1$ collapse steps, the offline a -fold computes $\mathbf{d}_j$ and its inner product with $\tau_{g_\gamma}^j(\mathbf{w}_{g_\gamma})$. The online b -fold reuses $\mathbf{d}_j$ and computes its inner product with $\tau_{g_\gamma}^j(\mathbf{y}_{g_\gamma})$. Each inner product requires ℓ_R polynomial multiplications, giving $2\ell_R$ for the two folds together. The online phase also performs the RLWE-to-NTRU conversion, which requires $2\ell_N$ polynomial multiplications per packed output. The measured online and offline costs are reported in Table 2.