

# NOMOS: Secure Non-Interactive $k$ NN under CKKS

Shihao Li\*, Wenhao Wang\*, Xiaomei Tang\*, Jian Liu\*, Rongmao Chen\*,

Lu Li\*, Zhigang Chen<sup>†</sup>, Guangfu Sun\*

\*National University of Defense Technology

<sup>†</sup>Ningbo University of Finance and Economics

**Abstract**—The  $k$ -nearest neighbor ( $k$ NN) algorithm is a core primitive for similarity search over sensitive data, but evaluating  $k$ NN under fully homomorphic encryption remains expensive because it requires distance computation, encrypted ranking, and top- $k$  extraction. CKKS is attractive for this setting because it natively supports packed approximate arithmetic, yet existing CKKS-based systems such as Engorgio (USENIX Security 2025) incur high overhead when used for reranking candidate sets.

We present NOMOS, an encrypted  $k$ NN protocol for candidate sets that targets the ranking and top- $k$  extraction bottleneck. NOMOS builds on Mazzone et al.’s matrix-based ranking method (USENIX Security 2025), but introduces a gap-amplified sign approximation that focuses precision on near-zero distance gaps, where  $k$ NN rank decisions are most sensitive. NOMOS integrates this primitive into a complete encrypted  $k$ NN pipeline, using slot-index alignment and ReLU-based top- $k$  extraction to avoid full sorting. For large databases, offline  $k$ -means preprocessing forms fixed-capacity candidate sets, allowing NOMOS to avoid global top- $k$  extraction and reduce sign-approximation calls from quadratic global growth to near-linear clustered scaling. With these techniques, NOMOS outperforms state-of-the-art encrypted ranking and top- $k$  baselines. Across the Mazzone-style ranking benchmark and candidate-set workloads compared with Engorgio, NOMOS reduces latency by nearly  $8\times$  and up to  $828\times$ , respectively. On the real-world SIFT dataset, NOMOS achieves 100% top-1 and top-8 retrieval accuracy on evaluated candidate sets, showing consistent retrieval quality on real data.

## 1. Introduction

Modern data-driven services increasingly rely on  $k$ -nearest neighbor ( $k$ NN) search over sensitive feature vectors. Medical analytics, fraud detection, biometric search, recommendation, and retrieval-augmented applications all compare a query against a private database and return the  $k$  nearest records. We focus on the single-owner outsourced setting, where the same owner-client holds both the database and the query and wishes to keep both confidential from the server. In this setting, both the database and the query may be encrypted, so the server must evaluate the  $k$ NN pipeline without learning the records, the query, distances, ranks, or selected outputs.

Fully homomorphic encryption (FHE) provides a non-interactive foundation for this setting, but encrypted  $k$ NN remains expensive because it combines arithmetic distance computation with order-based selection. CKKS handles packed real-valued arithmetic efficiently, making it natural for encrypted feature vectors. Ranking is harder because the server must compare encrypted distances, derive ranks, and extract the top- $k$  records, all without data-dependent branching or intermediate decryption.

Recent encrypted database and vector-search systems have made outsourced queries more expressive, but they leave a gap for CKKS-native real-valued  $k$ NN reranking. TFHE-based methods [1], [2], [3], [4] support exact or bit-level comparisons, but rely on quantized representations and expensive gate-level evaluation. Engorgio [6] supports top- $k$  queries through quantized CKKS ordering, but its general database path pays precision-management and layout costs that are unnecessary for real-valued reranking. CKKS ranking and sorting systems [5], [7], [8], [9] reduce encrypted ordering cost, yet they mainly operate on already encoded values and often compute more ordering information than a  $k$ NN query needs.

To address this gap, we present NOMOS, a CKKS-based encrypted  $k$ NN protocol family for secure non-interactive nearest-neighbor search. NOMOS specializes encrypted ordering to the  $k$ NN reranking pipeline by computing distances natively in CKKS, ranking encrypted distance values with a comparison primitive tuned to near-zero distance gaps, and extracting only the requested top- $k$  outputs rather than materializing a full ordering. The key observation is that  $k$ NN rank decisions are most fragile when two distances are close; therefore, the sign approximation used for ranking should prioritize the near-zero distance-gap region instead of optimizing uniformly over the full comparison domain.

The base protocol, Direct NOMOS, operates on one encrypted candidate set, which can be the full database in the unclustered setting or a fixed-capacity block inside a larger clustered layout. For larger databases, Clustered NOMOS (C-NOMOS) wraps Direct NOMOS with owner-side capacity-bounded clustering: the owner-client prepares encrypted blocks and representatives offline, and online evaluation selects clusters, evaluates blocks, and merges winners under encryption. Throughout the paper, NOMOS denotes the protocol family, Direct NOMOS denotes the base protocol, and C-NOMOS denotes the clustered scaling

TABLE 1: Qualitative comparison of encrypted  $k$ NN and CKKS ordering systems

| Features                    | Zuber<br><i>et al.</i> [1] | TFHE-rs<br>[2] | Cong<br><i>et al.</i> [3] | Azogagh<br><i>et al.</i> [4] | Hong<br><i>et al.</i> [5] | Engorgio<br>[6]   | Mazzone<br><i>et al.</i> [7] | Kim<br><i>et al.</i> [8] | Rovida<br><i>et al.</i> [9] | NOMOS<br>(Ours) |
|-----------------------------|----------------------------|----------------|---------------------------|------------------------------|---------------------------|-------------------|------------------------------|--------------------------|-----------------------------|-----------------|
| End-to-End $k$ NN Pipeline  | ●                          | ○              | ●                         | ●                            | ○                         | ●                 | ○                            | ○                        | ○                           | ●               |
| $k$ -Independent Extraction | ○                          | ○              | ○                         | ○                            | ○                         | ○                 | ○                            | ○                        | ○                           | ●               |
| Gap-Amplified Sign Approx.  | ○                          | ○              | ○                         | ○                            | ○                         | ○                 | ○                            | ○                        | ○                           | ●               |
| Quantization-Free           | ○                          | ○              | ○                         | ○                            | ●                         | ○                 | ●                            | ●                        | ●                           | ●               |
| Scheme                      | TFHE                       | TFHE           | TFHE                      | TFHE                         | CKKS                      | CKKS              | CKKS                         | CKKS                     | CKKS                        | CKKS            |
| Min. Reported Depth         | –                          | –              | –                         | –                            | 40                        | High <sup>†</sup> | 65                           | 30                       | 20                          | 18              |

<sup>†</sup>Engorgio uses repeated quantized comparison/bit-extraction stages, so we denote its depth as High. For NOMOS,  $k$ -independent extraction applies only to Direct NOMOS terminal extraction.

wrapper.

## 1.1. Our Contribution

This paper makes four contributions:

- **Gap-amplified sign approximation.** We introduce a CKKS sign approximation that amplifies near-zero distance gaps before applying the outer sigmoid approximation, focusing precision where  $k$ NN rank decisions are most sensitive. This reduces encrypted comparison cost while matching the protocol’s normalized distance-difference domain.
- **Efficient encrypted  $k$ NN pipeline.** Direct NOMOS combines CKKS-native distance computation, distance-difference encoding, encrypted ranking, slot-index alignment, and ReLU-based top- $k$  extraction. It identifies top- $k$  candidates directly from ranks and slot positions, avoiding full sorting and an additional high-cost sign-approximation stage.
- **C-NOMOS for clustered encrypted search.** C-NOMOS organizes the database into fixed-capacity clustered blocks, selects representative clusters under encryption, runs Direct NOMOS inside cluster blocks, and merges selected block winners. Under fixed block capacity and selected-cluster parameters, this changes the dominant sign-approximation calls from quadratic global growth to near-linear clustered scaling.
- **Implementation and evaluation.** Our OpenFHE implementation shows that Direct NOMOS is  $2\times$ – $8\times$  faster than the Mazzone-style ranking benchmark [7], reaches up to  $828\times$  speedup over Engorgio in the smaller direct database regime with the largest-speedup point accuracy-matched, and achieves 100% top-1/top-8 accuracy in the reported Direct NOMOS SIFT runs. C-NOMOS achieves 100% precision and recall in the reported clustered SIFT top-8 runs and reduces ciphertext-level gap-sign calls from 2080 to 152 at  $M = 16384$  (excluding terminal ReLU extraction). In database-size scaling, C-NOMOS reaches  $46\times$  speedup at  $M = 131072$ , where the Direct NOMOS timing reference no longer preserves full top-8 recall.

## 1.2. Related Work

Table 1 summarizes the qualitative landscape; the remainder of this subsection discusses each line of work.

**1.2.1. Quantized FHE Search.** Quantization-based methods convert high-precision data into low-precision or segmented representations to accommodate FHE constraints. TFHE-based approaches [1], [2], [3], [4] leverage functional bootstrapping for exact comparisons but incur high per-gate latency and require input quantization. Zuber et al. [1] propose a non-interactive secure  $k$ NN classifier under TFHE, but serial bootstrapping drives execution times beyond 70 minutes for moderate databases. Engorgio [6] is a state-of-the-art FHE database system that uses quantized CKKS ordering to support hybrid relational and vector queries over large encrypted databases. Its quantized ordering path targets expressive database workloads; for CKKS-native real-valued  $k$ NN reranking, however, this path introduces repeated comparison, precision-management, and layout costs that Direct NOMOS avoids by operating directly on encrypted feature vectors.

**1.2.2. CKKS-based  $k$ NN and Sorting.** CKKS [10] supports real-valued arithmetic without quantization. Hong et al. [5] propose  $k$ -way sorting networks that reduce comparison depth but require bootstrapping, resulting in execution times exceeding one hour. Mazzone et al. [7] introduce an efficient matrix-based ranking method with constant-depth comparisons and a 65-level ordering circuit. Kim et al. [8] and Rovida et al. [9] further optimize CKKS ordering through improved polynomials, reducing depth to 30 and 20 levels respectively. These works primarily target encrypted ranking or sorting over already encoded values, not an end-to-end  $k$ NN pipeline from feature vectors to top- $k$  outputs. We adopt Mazzone et al. as the primary ranking baseline because its released benchmark matches the matrix-style ranking stage in NOMOS; other CKKS ordering systems serve as qualitative context.

**1.2.3. Sign Approximation.** Sign function approximation underpins comparison operations in encrypted ranking and sorting. The most common approach uses Chebyshev interpolation [11], which offers numerical stability but requires

polynomial degrees exceeding 1000 for adequate accuracy. Engorgio [6], Mazzone et al. [7], and Rovida et al. [9] use high-degree Chebyshev-style sign approximations in their comparison paths, resulting in substantial computational overhead.

Composite approximation offers an alternative: it factorizes the target as  $f(x) \approx g_2(g_1(x))$  with lower-degree components. Cheon et al. [12] introduce a general-purpose composite scheme, while Mazzone et al. [7] and Kim et al. [8] apply this to achieve high precision, consuming 15 multiplicative levels for sign approximation alone. Lee et al. [13] propose a theoretically optimal multi-interval scheme, but the complex coefficient computation hinders practical adoption. Composite methods reduce evaluation cost through low-degree polynomials but optimize global sign quality; NOMOS targets the near-zero threshold governing encrypted  $k$ NN ordering.

## 2. Preliminaries

This section fixes notation and reviews CKKS, function approximation, and order-based primitives used in encrypted ranking.

### 2.1. Notations

Table 2 summarizes the key notation.

TABLE 2: Key notation

| Symbol                     | Description                                 |
|----------------------------|---------------------------------------------|
| $\mathcal{D}$              | Encrypted database or block                 |
| $\mathbf{v}_i, \mathbf{q}$ | Record vector and query                     |
| $M$                        | Database size                               |
| $m$                        | Invocation/block size                       |
| $n$                        | Ranking pool size                           |
| $k$                        | Requested neighbors                         |
| $N$                        | CKKS ring dimension                         |
| $\delta_{i,j}$             | Normalized distance gap                     |
| $G_\ell$                   | Stacked $\ell$ -gap function                |
| $\varepsilon_s$            | Sign saturation error                       |
| $\tau_f$                   | Nominal dead-zone half-width                |
| $B$                        | Block capacity                              |
| $c, r$                     | Clusters and selected clusters              |
| $L$                        | Blocks per cluster, $L = \lceil U/B \rceil$ |
| $E_\ell[j]$                | Rank- $\ell$ cluster selector               |
| $A_t, D_t$                 | Prefix and rank- $t$ masks                  |
| $W[j, b, t]$               | Encrypted local winner record               |
| $S[\ell, b, t]$            | Repacked selected winner                    |

### 2.2. CKKS Homomorphic Encryption

CKKS [10] supports approximate SIMD arithmetic over encrypted real or complex slots; a ciphertext with ring dimension  $N$  provides  $N/2$  slots. NOMOS uses RNS-CKKS [14] through OpenFHE [15] and relies on slot-wise addition/subtraction, ciphertext-ciphertext multiplication `Mult`, plaintext-ciphertext multiplication `PMult`, and cyclic rotation `Rot`. Each `Mult` consumes one level and is followed by rescaling; our implementation uses improved rescaling [16] and double-hoisting hybrid key switching [17], [18].

### 2.3. Function Approximation in FHE

Since FHE schemes support only polynomial operations, non-polynomial functions such as sign and ReLU must be approximated.

*Chebyshev interpolation* expresses an approximating polynomial in the Chebyshev basis, improving numerical stability for high-degree approximations and often approaching minimax error [19]. It is widely used in CKKS because Taylor expansion converges poorly over wide intervals, while Remez-style minimax and multi-interval methods can require costly coefficient recomputation [11], [20].

*Composite function approximations* factorize a target as  $f(x) \approx g_2(g_1(x))$  so that several lower-degree polynomials replace one very high-degree polynomial. Cheon et al.’s  $f/g$  composite sign method [12] is the standard example: iterated composition shrinks the transition region and improves global sign approximation while reducing depth via Paterson–Stockmeyer evaluation [21]. Composite approximations may target either uniform accuracy over a broad interval or sharper resolution around a task-specific transition region.

### 2.4. Order-Based Operations

Ranking and sorting rely on the sign function for pairwise comparison. We define:

$$\text{sign}(x) = \begin{cases} 1, & x > 0 \\ 0.5, & x = 0 \\ 0, & x < 0 \end{cases} \quad (1)$$

Given values  $\{d_1, \dots, d_n\}$ , the rank of  $d_i$  is computed as:

$$\text{rank}(d_i) = 0.5 + \sum_{j \neq i} \text{sign}(d_i - d_j). \quad (2)$$

Matrix-based approaches [7] encode pairwise comparisons with constant comparison depth. Expansion encoding (`epa`) repeats each distance across  $n$  consecutive slots, while replication encoding (`rep`) repeats the whole distance vector  $n$  times. This permutation layout [9] expands the slot representation from  $n$  to  $n^2$  entries, but amortizes pairwise comparisons into SIMD slots and avoids  $O(n^2)$  separate ciphertext-level comparison invocations. Figure 1 illustrates the method with distance vector  $\{1, 6, 0, 2\}$ .

General sorting extends ranking with permutation masks and an equal function to fully reorder elements; we refer readers to [7] for details. In  $k$ NN, only the indices of the  $k$  nearest neighbors are needed rather than a complete ordering.

## 3. Framework Overview

This section presents the NOMOS workflow, system architecture, and leakage boundary.

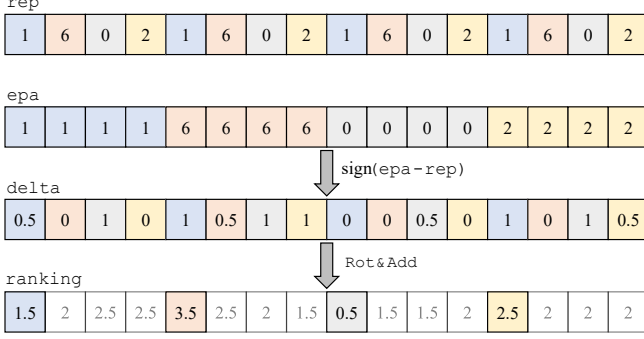


Figure 1: Ranking example for distance vector  $\{1, 6, 0, 2\}$  using expansion and replication encoding.

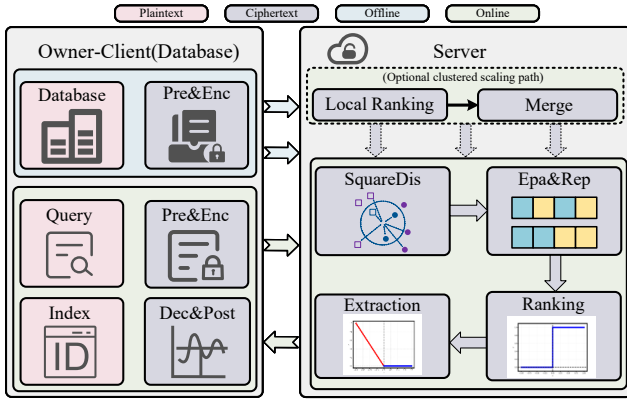


Figure 2: NOMOS protocol workflow.

### 3.1. System Architecture

Figure 2 shows the NOMOS protocol workflow. The owner-client encodes and encrypts the database block and query, while the server evaluates the fixed homomorphic trace for distance computation, ranking, and top- $k$  extraction. The server returns only encrypted outputs, and the owner-client decrypts the final result locally. Dashed components indicate the optional C-NOMOS scaling path, whose owner-side setup prepares encrypted blocks and representatives before online evaluation:

- **Direct NOMOS.** The owner-client encrypts a database block and a query. The server runs the NOMOS pipeline homomorphically to compute encrypted distances, encrypted ranks, and encrypted top- $k$  outputs. The owner-client decrypts only the final result.
- **C-NOMOS.** The owner-client organizes a large database into encrypted blocks and representatives before upload. Online, C-NOMOS selects representative clusters, evaluates blocks, and merges encrypted winner records into the final top- $k$  output.

In NOMOS’s single-owner model, C-NOMOS clustering is an offline owner-side setup in which the owner-client runs capacity-bounded  $k$ -means on the plaintext database, forms representatives, pads clusters to public fixed capacities, and encrypts the clustered layout before upload. For multi-owner

or vertically partitioned databases, a privacy-preserving clustering protocol [22] could replace this step; that integration is outside NOMOS’s current scope.

### 3.2. Threat Model and Security Analysis

Following the outsourced FHE model used by systems such as Engorgio [6], NOMOS considers a two-party single-owner setting. The owner-client owns both the database and the query, holds the CKKS secret key, and outsources encrypted computation to a semi-honest server that follows the protocol but may attempt to infer information from ciphertexts, public parameters, and the fixed execution trace.

**Direct NOMOS.** For Direct NOMOS, the online protocol is a single non-interactive FHE evaluation over an encrypted database block. The server may learn:

- public parameters, including the block size, feature dimension, CKKS parameters, and  $k$ ;
- ciphertext sizes and the fixed evaluation trace.

The server does not learn plaintext records, the query, distances, ranks, selected indices, or final top- $k$  outputs.

**C-NOMOS.** C-NOMOS inherits the Direct NOMOS online security model, but assumes the owner-side clustered setup has already been completed before encryption and upload. This setup is not part of the C-NOMOS online protocol or online latency. In addition to the Direct NOMOS public information, the server may learn:

- setup-visible clustered-layout metadata, including  $c$ ,  $r$ ,  $\alpha$ ,  $U$ ,  $L$ , block capacity  $B$ , and ciphertext/layout dimensions;
- public shape parameters of the encrypted owner-side layout.

The server observes public cluster indices and layout dimensions, but these indices are layout labels rather than plaintext semantic labels. It does not learn representative distances, selected top- $r$  clusters, active cluster sizes beyond the public capacity-bound layout, local winners, selected block winners, invalid padding-entry positions beyond setup-visible metadata, or final top- $k$  outputs.

Encrypted validity indicators propagate through local Direct NOMOS evaluations, selected-cluster composition, and the final merge, so online control flow does not reveal active valid counts or padding-entry positions beyond setup-visible metadata.

**Security argument.** NOMOS adds no cryptographic assumption beyond CKKS semantic security under RLWE [10]. Let  $\mathcal{L}_D$  contain the public Direct NOMOS parameters, ciphertext dimensions, and fixed trace, and let  $\mathcal{L}_C$  additionally contain the setup-visible C-NOMOS layout metadata. Given this leakage, the online computation is a fixed sequence of homomorphic operations, public masks, and public polynomial evaluations, with no data-dependent branching or intermediate decryption. A semi-honest server’s online view can be simulated from  $\mathcal{L}_D$  or  $\mathcal{L}_C$  by encrypting zeros with the same public dimensions and replaying the fixed trace; Appendix A states this fixed-trace simulation formally. The online view thus reveals nothing

beyond the stated leakage. We do not claim malicious security, output integrity, or circuit privacy.

The attack of Li et al. [23] requires access to released decrypted approximate values. In NOMOS, all decryptions are performed locally by the owner-client after online evaluation; the server receives no decrypted values and therefore does not obtain the decryption-side interface required by that attack model.

## 4. Gap-Amplified Sign Approximation

This section presents the gap-amplified sign approximation. Section 4.1 motivates the rank-resolution requirement, Section 4.2 introduces the gap-sigmoid construction, and Section 4.3 compares with prior strategies.

### 4.1. Challenges and Observations

NOMOS does not seek a uniformly optimal sign approximation. Its goal is to minimize the rank-resolution threshold for encrypted  $k$ NN, where near-zero distance gaps determine ordering—an objective distinct from general sorting, which demands high-quality sign approximation over the full input interval.

Let  $P$  be a public upper bound on the squared Euclidean distance between any database record and any query in the allowed domain. Since nearest-neighbor retrieval depends only on the relative order of distances, multiplying all squared distances by a positive public constant preserves the result. NOMOS therefore reserves a factor-two comparator-domain margin and normalizes valid squared distances as

$$\tilde{d}_i = \frac{D^2(\mathbf{v}_i, \mathbf{q})}{2P}, \quad \tilde{d}_i \in [0, 1/2]. \quad (3)$$

The comparator input for ranking is then

$$\delta_{ij} = \tilde{d}_i - \tilde{d}_j \in [-1/2, 1/2]. \quad (4)$$

This compact domain eases the approximation burden on the gap function while preserving a steep response near zero.

Ranking aggregates  $n - 1$  pairwise sign values per candidate (Equation (2)). If each sign approximation incurs saturation error  $\varepsilon_s$ , the worst-case accumulated rank error reaches  $(n-1)\varepsilon_s$  when all errors align. A conservative target for distinguishing adjacent ranks is therefore

$$\varepsilon_s < \frac{1}{2(n-1)}. \quad (5)$$

For Direct NOMOS,  $n = M$ ; inside C-NOMOS,  $n$  is the public block or merge-pool size. Equation (5) bounds accumulated saturation error outside the unresolved near-zero region; pairs inside the dead zone may be misordered. NOMOS preserves the plaintext top- $k$  set when the normalized top- $k$  boundary gap exceeds the dead-zone width plus CKKS numerical error, and accumulated saturation error stays below half a rank unit. For a sign approximation  $f$ , we define the nominal dead-zone half-width as the first

point where the approximation reaches the target saturation threshold:

$$\tau_f(\varepsilon_s) = \inf\{x > 0 : f(x) \geq 1 - \varepsilon_s \text{ and } f(-x) \leq \varepsilon_s\}. \quad (6)$$

This metric captures the near-zero rank-resolution threshold rather than global uniform sign-approximation error.

### 4.2. Gap-Amplified Sign Design

The gap function is a local amplifier, not a standalone sign approximation. The design target is to approximate  $\text{sign}(x)$  through a transformed input  $H(x)$  that moves small nonzero distance gaps away from zero before the outer sigmoid approximation is applied. We choose the inner transform according to four design criteria:

- **Direction-preserving amplification.** The transform should be odd, so positive and negative distance gaps keep their comparison direction after amplification.
- **Large near-zero slope.** The transform should have a steep response at the origin, pushing small nonzero gaps away from the outer sigmoid’s hardest transition region.
- **Low-degree approximability.** The transform should be smooth on the calibrated comparison domain, allowing a low-degree Chebyshev approximation with stable coefficients.
- **Controlled output range.** The transform should remain bounded, so stacked gap layers stay within the intended approximation interval and residual off-origin error can be absorbed by sigmoid saturation.

These criteria lead NOMOS to use the rational map

$$G(x) = \frac{30x}{1 + 300x^2}. \quad (7)$$

as a design target. It is odd, analytic on the real line, has origin slope  $G'(0) = 30$ , and reaches a bounded magnitude of about 0.87 before decreasing smoothly. The implementation evaluates its degree-13 Chebyshev approximation over  $[-1/2, 1/2]$  homomorphically. Near the origin, small distance gaps are rapidly moved out of the transition region of the outer sigmoid. This decouples two objectives conflated in a direct sign approximation: the gap layer enlarges the distance gaps that determine  $k$ NN ranks, while the outer sigmoid supplies a smooth saturation map. Consequently, the gap polynomial need not approximate sign uniformly; once its output enters the sigmoid’s saturated region, moderate off-origin error is absorbed.

NOMOS stacks gap layers by inserting a plaintext factor 0.5 between layers:

$$G_1(x) = G(x), \quad G_\ell(x) = G(0.5 \cdot G_{\ell-1}(x)), \quad \ell \geq 2. \quad (8)$$

This factor keeps each subsequent gap-layer input within the approximation range without adding multiplicative depth. The  $\ell$ -gap sign target is

$$F_\ell(x) = \text{sigmoid}_{70}(G_\ell(x)), \quad (9)$$

$$\text{sigmoid}_{70}(x) = \frac{1}{1 + e^{-70x}}. \quad (10)$$

We call  $F_1$ ,  $F_2$ , and  $F_3$  the single-gap, double-gap, and triple-gap variants, respectively.

In the implementation, each gap layer is evaluated as a degree-13 Chebyshev polynomial consuming 4 multiplicative levels, and  $\text{sigmoid}_{70}$  is replaced by a degree-119 Chebyshev approximation consuming 7 levels. Hence the sign-approximation depth with  $\ell$  gap layers is

$$D_{\text{sign}}(\ell) = 4\ell + 7. \quad (11)$$

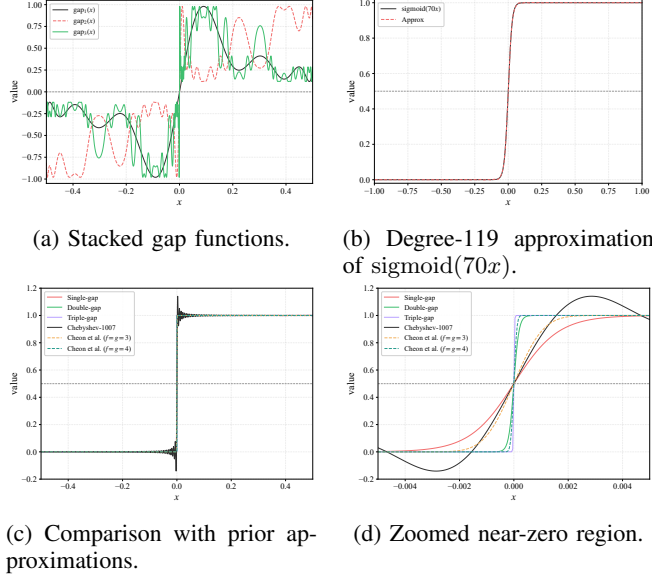


Figure 3: Gap-amplified sign approximation and near-zero behavior.

Figure 3a shows the role of the stacked gap functions: they do not approximate sign directly, but move near-zero inputs into a range where the outer approximation can saturate. Figure 3b shows the degree-119 approximation used for this outer sigmoid. Errors of the gap approximation away from the origin are less important because the final sigmoid absorbs them once the transformed value lies in its saturated region.

#### 4.3. Comparison with Prior Sign Approximations

Direct Chebyshev approximation targets the discontinuous sign function over a broad interval and therefore typically requires high degree [6], [7], [9]. As shown in Figure 3c, Chebyshev approximation also exhibits visible Gibbs-style oscillations, so a first crossing of the saturation threshold does not by itself imply stable comparison behavior. Cheon et al.’s composite approximation [12] follows a different strategy by improving global sign approximation through repeated transition-region compression.

NOMOS optimizes a different objective. Rather than minimizing uniform error over the full interval, it targets the near-zero region that determines encrypted  $k$ NN ranks. The gap layers steepen the response around zero, and the

#### Algorithm 1 Direct NOMOS Workflow

---

**Input:** Database block  $\mathcal{D} = \{\mathbf{v}_1, \dots, \mathbf{v}_m\}$ , Query  $\mathbf{q}$ , Parameter  $k$

**Output:** Indices of  $k$  nearest neighbors

- 1: **Owner-Client: Pre&Enc** ▷ Data encoding
- 2:  $\mathbf{ct}_{\text{db}}, \mathbf{ct}_{\text{q}} \leftarrow \text{Encrypt}(\text{Encode}(\mathcal{D}, \mathbf{q}))$
- 3: **Server-Side Phase 1: SquareDis** ▷ Distance computation
- 4:  $\mathbf{ct}_{\text{dist}} \leftarrow \text{SquareDis}(\mathbf{ct}_{\text{db}}, \mathbf{ct}_{\text{q}})$
- 5: **Server-Side Phase 2: Epa&Rep** ▷ Difference encoding
- 6:  $\mathbf{ct}_{\text{epa}}, \mathbf{ct}_{\text{rep}} \leftarrow \text{Epa\&Rep}(\mathbf{ct}_{\text{dist}})$
- 7:  $\mathbf{ct}_{\text{delta}} \leftarrow \text{Sub}(\mathbf{ct}_{\text{epa}}, \mathbf{ct}_{\text{rep}})$
- 8: **Server-Side Phase 3: Ranking** ▷ Encrypted ranking
- 9:  $\mathbf{ct}_{\text{sign}} \leftarrow \text{SignApprox}(\mathbf{ct}_{\text{delta}})$
- 10:  $\mathbf{ct}_{\text{rank}} \leftarrow \text{Aggregate}(\mathbf{ct}_{\text{sign}})$
- 11: **Server-Side Phase 4: Extraction** ▷ Top- $k$  extraction
- 12:  $\mathbf{ct}_{\text{out}} \leftarrow \text{ReLUApprox}(\mathbf{ct}_{\text{rank}}, k)$
- 13: **Owner-Client: Result Return**
- 14:  $\text{result} \leftarrow \text{Decrypt}(\mathbf{ct}_{\text{out}})$
- 15:  $\mathcal{I} \leftarrow \text{ThresholdDetect}(\text{result})$
- 16: **return**  $\mathcal{I}$

---

outer sigmoid converts the amplified signal into a saturated comparison value. Off-origin deviations of the gap polynomial are therefore not the primary bottleneck. As shown in Figure 3d, increasing the number of gap layers progressively narrows the rank-resolution threshold, at the cost of additional multiplicative depth.

## 5. NOMOS Protocols

This section details the NOMOS protocols. Section 5.1 presents Direct NOMOS, including its base workflow and multi-ciphertext extension. Section 5.2 presents C-NOMOS. Section 5.3 analyzes sign-approximation costs and overall latency.

### 5.1. Direct NOMOS

**5.1.1. Base Workflow.** Algorithm 1 presents Direct NOMOS over an encrypted database or block. Unlike prior CKKS ranking systems [7], [9], which assume pre-encoded distances, Direct NOMOS starts from encrypted feature vectors and computes the top- $k$  output end to end. The workflow has four server-side phases: distance computation, difference encoding, encrypted ranking, and ReLU-based extraction. Here  $m$  denotes the current Direct NOMOS invocation size. Standalone Direct NOMOS has  $m = M$ , while C-NOMOS invokes the same primitive on public fixed-size pools such as representatives and padded blocks.

Data encoding and packing. The owner-client encodes the current database block and query into CKKS ciphertexts before transmission. In the single-ciphertext base case, the block occupies  $m^2$  slots, with vector  $\mathbf{v}_i$  stored in the  $m$ -slot group starting at position  $(i - 1)m$ . The query

$\mathbf{q}$  is zero-padded and replicated to align with all candidate vectors. Slot positions identify candidate indices throughout ranking and extraction, avoiding permutation networks for output recovery.

Distance computation and difference encoding. Given the encrypted block  $\mathbf{ct}_{db}$  and query  $\mathbf{ct}_q$ , the server performs two operations to construct all pairwise distance differences in a single ciphertext.

**SquareDis.** Squared Euclidean distances are computed through element-wise subtraction, squaring, and tree-based reduction:

$$\begin{aligned} \mathbf{ct}_{diff} &= \text{Sub}(\mathbf{ct}_{db}, \mathbf{ct}_q) \\ \mathbf{ct}_{sq} &= \text{Mult}(\mathbf{ct}_{diff}, \mathbf{ct}_{diff}) \\ \mathbf{ct}_{dist}^{(0)} &= \mathbf{ct}_{sq} \\ \mathbf{ct}_{dist}^{(j+1)} &= \text{Add}\left(\mathbf{ct}_{dist}^{(j)}, \text{Rot}(\mathbf{ct}_{dist}^{(j)}, 2^j)\right) \\ \mathbf{ct}_{dist} &= \mathbf{ct}_{dist}^{(\lceil \log_2 m \rceil)} \end{aligned} \quad (12)$$

where  $j = 0, \dots, \lceil \log_2 m \rceil - 1$  in the recursive reduction. The tree-based reduction aggregates squared differences across the padded coordinate group of length  $m$ ; only the feature-coordinate slots are nonzero. After reduction, the first slot of every  $m$ -slot group contains a valid squared distance value. This phase consumes 1 multiplicative level and  $O(\log m)$  rotations.

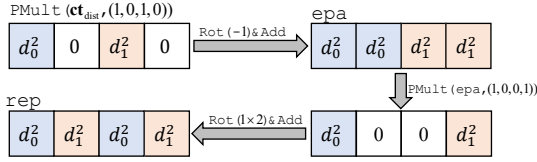


Figure 4: Epa&Rep encoding for distance vector  $\{d_0^2, d_1^2\}$ .

**Epa&Rep.** For pairwise distance comparisons, we construct two complementary encodings. The expansion encoding  $\mathbf{ct}_{epa}$  replicates each normalized distance  $\tilde{d}_i$  across  $m$  consecutive slots. The replication encoding  $\mathbf{ct}_{rep}$  repeats the entire normalized distance vector  $m$  times. Figure 4 illustrates this construction.

To construct  $\mathbf{ct}_{epa}$ , we apply a scaling mask at positions  $\{0, m, 2m, \dots, (m-1)m\}$  and zeros elsewhere. The mask value is  $1/(2P)$  for the public bound in Equation (3); for features normalized to  $[0, 1]^d$ , this specializes to  $P = d$ . The mask clears invalid intermediate values while normalizing distances to  $[0, 1/2]$ . Subsequent rotation and accumulation operations replicate each normalized distance across its  $m$ -slot group.

To construct  $\mathbf{ct}_{rep}$ , we apply a diagonal mask selecting positions  $\{0, m+1, 2m+2, \dots, (m-1)m+(m-1)\}$ , then replicate through rotation and accumulation. Our approach reduces the number of rotations by half compared to [7] by trading one additional multiplicative level, a favorable tradeoff given the high cost of rotation operations [24].

The difference  $\mathbf{ct}_{delta} = \text{Sub}(\mathbf{ct}_{epa}, \mathbf{ct}_{rep})$  yields all normalized pairwise distance differences  $\tilde{d}_i - \tilde{d}_j$  in a single ciphertext. This direction matches the rank definition in

Equation (2): smaller distances receive smaller rank scores. This phase consumes 2 multiplicative levels.

Ranking via gap-amplified sign approximation. The ranking phase transforms normalized pairwise distance differences into encrypted rank values using the gap-amplified sign approximation from Section 4. We define two operations used in Algorithm 1:

**SignApprox.** This operation applies the gap-amplified sign approximation  $F_\ell(x) = \text{sigmoid}_{70}(G_\ell(x))$  from Section 4 to  $\mathbf{ct}_{delta}$  through sequential polynomial evaluations. The sign variant is selected by the number of stacked gap layers  $\ell$ . Each gap layer consumes 4 multiplicative levels, and the degree-119 approximation of  $\text{sigmoid}_{70}$  consumes 7 levels, giving sign depth  $D_{\text{sign}}(\ell) = 4\ell + 7$ . After approximation, each slot contains a value near 1 if  $\tilde{d}_i > \tilde{d}_j$ , near 0 if  $\tilde{d}_i < \tilde{d}_j$ , and near 0.5 if the two normalized distances are equal.

**Aggregate.** This operation computes rank scores through tree-based reduction. The score of element  $i$  follows Equation (2): it counts how many encrypted distances are smaller than  $\tilde{d}_i$ , with the self-comparison contributing approximately 0.5. Comparison results for element  $i$  occupy slots  $\{(i-1)m, \dots, im-1\}$ . The aggregation uses  $\lceil \log_2 m \rceil$  rotation-and-add operations:

$$\begin{aligned} \mathbf{ct}_{rank}^{(0)} &\leftarrow \mathbf{ct}_{sign}, \\ \mathbf{ct}_{rank}^{(j+1)} &\leftarrow \text{Add}\left(\mathbf{ct}_{rank}^{(j)}, \text{Rot}(\mathbf{ct}_{rank}^{(j)}, 2^j)\right), \\ \mathbf{ct}_{rank} &\leftarrow \mathbf{ct}_{rank}^{(\lceil \log_2 m \rceil)} \end{aligned} \quad (13)$$

for  $j = 0, \dots, \lceil \log_2 m \rceil - 1$ . After aggregation, the first slot of each  $m$ -slot group contains a rank score in  $[0.5, m - 0.5]$  for distinct distances, where smaller scores correspond to nearer candidates. We normalize this score to the ReLU input domain by applying Sub with the midpoint  $(m+1)/2$  and PMult with scaling factor  $2/m$ .

**ReLU-based top- $k$  extraction.** Conventionally, sign approximation distinguishes top- $k$  from non-target rank scores after normalization, requiring 10+ additional levels. Slot-index alignment makes explicit index computation unnecessary: binary discrimination suffices, and ReLU provides it more efficiently.

We define  $\text{ReLUApprox}(\mathbf{ct}_{rank}, k)$  as the operation that applies the shifted ReLU approximation polynomial to the normalized rank ciphertext with parameter  $k$ . This operation identifies elements whose ranks fall within the top- $k$  range.

**Shifted ReLU Design.** We replace sign with shifted ReLU:

$$\text{ReLU}_{\text{shift}}(x) = \begin{cases} -s \cdot (x + 1 - \frac{2k}{m}), & x < -1 + \frac{2k}{m} \\ 0, & x \geq -1 + \frac{2k}{m} \end{cases} \quad (14)$$

The shift  $-1 + 2k/m$  places the kink at the first non-target rank score after normalization. Top- $k$  candidates lie to the left of this kink, and the slope  $s = m/2$  maps the nearest target rank point, at distance  $2/m$  from the kink, to unit scale.

ReLU is better suited than sign for extraction because sign's zero-order jump discontinuity causes slow Chebyshev

error decay and Gibbs oscillations requiring degree-1000+ polynomials, whereas ReLU’s first-order kink discontinuity admits much faster decay and accurate approximation at substantially lower degree.

**Depth-Stable  $k$  Adjustment.** Varying  $k$  only changes the ReLU bias term. The approximation polynomial and multiplicative depth remain unchanged for a fixed input size, so Direct NOMOS avoids the additional encrypted selection stages required by sorting-based top- $k$  extraction.

**Polynomial Approximation.** A key structural property simplifies degree selection: the ReLU kink at  $x_0 = -1 + 2k/m$  aligns with the first non-target normalized rank score. The nearest target point lies at deterministic distance  $2/m$  to the left of the kink.

We use the standard decay heuristic for Chebyshev approximation of first-order kink functions as a calibrated design rule: the approximation error decreases with degree  $d_R$  and distance  $\Delta_x$  from the kink, roughly as  $C_0/(d_R\Delta_x)$ . At the critical target point  $x_{\text{left}} = x_0 - 2/m$ , we set  $\Delta_x = 2/m$  and choose  $d_R$  so this error stays below the detection threshold  $\tau_{\text{det}} = 0.2$ :

$$\frac{C_0 \cdot m}{2d_R} < \tau_{\text{det}}. \quad (15)$$

Folding the fixed threshold factor into the calibrated constant gives the rule

$$d_R > C \cdot m \quad (16)$$

with  $C \approx 0.1$ . This rule is a calibrated design heuristic, not a uniform approximation theorem; for each configured pool size, we validate the chosen degree on the discrete normalized rank grid, especially the two points adjacent to the kink. It yields degree-13 (4 levels) for  $m \leq 128$ , degree-27 (5 levels) for  $m = 256$ , degree-55 (6 levels) for  $m = 512$ , and degree-119 (7 levels) for  $m = 1024$ .

**Owner-Client Post-Processing.** The server returns encrypted ReLU outputs. After decryption, the owner-client classifies slots with values  $\geq \tau_{\text{det}} = 0.2$  as targets. This threshold is 20% of the nearest target’s ideal ReLU value and provides a conservative margin accommodating approximation errors. Target slot positions directly yield  $k$  nearest neighbor indices.

**5.1.2. Direct Multi-Ciphertext Extension.** The base workflow operates when  $m^2 \leq N/2$ , which supports single-ciphertext processing up to  $m = 128$  under the evaluated  $N = 2^{16}$  configuration. This subsection describes the Direct NOMOS extension for handling larger unclustered blocks.

**Multi-Ciphertext Decomposition.** For inputs exceeding single-ciphertext capacity,  $m$  denotes the total invocation size and  $m'$  denotes the per-ciphertext partition size. We partition data across  $u = \lceil m/m' \rceil$  ciphertexts where  $m' = 2^{\lfloor \log_2(\sqrt{N/2}) \rfloor}$  denotes the maximum partition size. Each partition independently executes distance computation and local ranking. Cross-partition comparisons establish global rankings through  $u(u-1)/2$  pairwise operations. For each partition pair  $(i, j)$ , we apply the same gap-amplified sign approximation to cross-partition distance differences

to determine how elements in partition  $i$  rank relative to partition  $j$ .

**TransRanking.** Cross-partition comparisons require one gap-amplified sign evaluation for each partition pair. Rather than evaluating the complementary direction again, TransRanking derives it from the existing sign outputs using rotations, masking, and replication (Algorithm 4, Appendix C). This trades  $O(\log m')$  rotations and one multiplicative level for avoiding another sign-approximation evaluation.

**Global Ranking Assembly.** Each partition adjusts local rankings by incorporating cross-partition corrections. Let  $F_{i \leftarrow j}$  denote the encrypted count of elements in partition  $j$  that are farther than a given element in partition  $i$ . Since the global rank counts closer elements, partition  $j$  contributes  $m' - F_{i \leftarrow j}$ . The global rank score for elements in partition  $i$  is therefore

$$\begin{aligned} \text{global\_rank}_i &= \text{local\_rank}_i + \sum_{j \neq i} (m' - F_{i \leftarrow j}) \\ &= \text{local\_rank}_i - \sum_{j \neq i} F_{i \leftarrow j} + (u-1)m'. \end{aligned} \quad (17)$$

Here  $\text{local\_rank}_i$  is the within-partition rank score, and  $F_{i \leftarrow j}$  is obtained from the  $(i, j)$  sign evaluation and the TransRanking-derived complementary direction. Shifted ReLU approximation then extracts top- $k$  elements from each partition based on global rankings. This extraction uses the total invocation size  $m$ , not the per-ciphertext partition size  $m'$ , so global rank scores in  $[0.5, m - 0.5]$  use the same offset and scale as in the Direct NOMOS base workflow.

## 5.2. C-NOMOS: Clustered Protocol

C-NOMOS is a fixed-trace clustered scheduling wrapper around Direct NOMOS. It reorganizes a large database into encrypted fixed-capacity blocks and merges selected block winners, avoiding one global ranking operation over all  $M$  records while reusing the same ranking primitive.

Figure 5 gives the workflow. Owner-side  $k$ -means setup produces blocks and representatives before encryption and upload. Online evaluation selects representative clusters, evaluates all blocks under the same public trace, repacks selected local winner records, and runs the final merge under encryption.

Algorithm 2 gives the trace-level protocol. The owner-side clustered setup is not counted as online latency. Direct NOMOS calls over representatives and cluster blocks reuse SquareDis, Epa&Rep, ranking, and extraction without intermediate decryption; block mode uses packed rank-threshold extraction (Section 5.2.2) to produce mergeable encrypted winner records.

**5.2.1. Capacity-Bounded Preprocessing and Layout.** Before online encrypted query evaluation, the owner-client partitions the plaintext database and produces representatives as

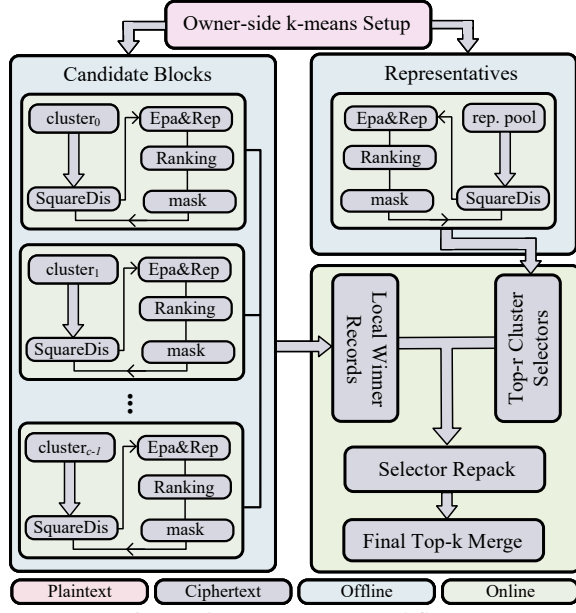


Figure 5: C-NOMOS workflow.

an offline setup step. C-NOMOS uses a capacity-bounded assignment so every public cluster obeys

$$|C_j| \leq U = \left\lceil \frac{\alpha M}{c} \right\rceil \quad \text{for all } j, \quad (18)$$

where  $M$  is the database size,  $c$  is the number of clusters, and  $\alpha$  is a public slack parameter. The owner-client pads each cluster to capacity  $U$ , encrypts the representatives and padded blocks, and uploads the fixed layout.

Each cluster is split into

$$L = \left\lceil \frac{U}{B} \right\rceil \quad (19)$$

fixed-capacity blocks, where  $B$  is the maximum number of records handled by one Direct NOMOS ranking pool. Invalid padding entries remain in encrypted slots, so the server follows the same public trace for every cluster. Each uploaded entry carries encrypted feature, validity/weight, and metadata channels; score channels are formed online after query-dependent distances are computed. For mergeable encrypted records, the score channel is

$$\text{score}_i = w_i \left( \frac{1}{2} - \tilde{d}_i \right), \quad (20)$$

where  $w_i$  is an encrypted validity/weight channel and  $1/2$  is the normalized distance bound from Equation (3). A valid record stores  $1/2 - \tilde{d}_i$ , so nearer records receive larger scores; an invalid padding entry has  $w_i = 0$  and zeroed metadata. Merge pools pack up to  $B$  encrypted records into the Direct NOMOS comparison layout, skip distance computation because records already carry scalar scores, and rank scores in descending order. For two encrypted records, the comparison input is

$$\Delta(i, j) = \text{score}_j - \text{score}_i. \quad (21)$$

## Algorithm 2 C-NOMOS Clustered Workflow

**Input:** Database  $\mathcal{D}$ , Query  $q$ , Parameters  $k, r, c, \alpha, B$

**Output:** Indices of  $k$  nearest neighbors

- 1: **Owner-Side Setup: Capacity-Bounded Clustering**
- 2: Set  $M \leftarrow |\mathcal{D}|$
- 3: Generate clusters  $C_0, \dots, C_{c-1}$  and representatives  $\mu_0, \dots, \mu_{c-1}$
- 4: Set  $U \leftarrow \lceil \alpha M / c \rceil$  and  $L \leftarrow \lceil U / B \rceil$
- 5: **Owner-Client: Layout and Encryption**
- 6: Construct fixed-capacity clustered layout with encrypted validity indicators
- 7: Encrypt cluster blocks, representatives, and query
- 8: **Server-Side Phase 1: Representative Selection**
- 9: Run Direct NOMOS over encrypted representatives to obtain  $E_\ell[j]$  for  $\ell = 0, \dots, r-1$
- 10: **Server-Side Phase 2: Local Candidate Selection**
- 11: **for** each cluster  $j$  and block  $b$  under the public fixed trace **do**
- 12: Run Direct NOMOS over the block to obtain  $W[j, b, t]$  for  $t = 1, \dots, k$
- 13: **end for**
- 14: **Server-Side Phase 3: Selector Repacking**
- 15:  $S[\ell, b, t] \leftarrow \sum_{j=0}^{c-1} E_\ell[j] \cdot W[j, b, t]$
- 16: **Server-Side Phase 4: Selected and Final Merge**
- 17: Merge selected block winners and run final top- $k$  merge
- 18: **Owner-Client: Result Return**
- 19: Decrypt final encrypted outputs and apply threshold detection
- 20: **return**  $\mathcal{I}$

Since  $\tilde{d}_i \in [0, 1/2]$ , the comparison input remains in the same calibrated interval  $[-1/2, 1/2]$  used by the gap-amplified sign approximation. Invalid padding records may tie with the worst-bound valid score but never receive a positive score advantage. The evaluated workloads do not place valid records exactly at the public worst-distance boundary; if such a boundary tie occurs, fewer than  $k$  positive-weight records may be returned. The weight channel is propagated through local selection, selected-cluster composition, and the final merge, so the owner-client can discard any zero-weight output after decryption.

**5.2.2. Two-Level Encrypted Selection.** Online evaluation has two encrypted selection levels. First, the server evaluates Direct NOMOS over encrypted representatives. This stage does not reveal the selected cluster identifiers; it outputs near-binary rank-specific encrypted selectors

$$E_\ell[j] \in [0, 1], \quad \ell = 0, \dots, r-1, \quad j = 0, \dots, c-1, \quad (22)$$

where  $E_\ell[j]$  approximately indicates whether cluster  $j$  is selected at representative rank  $\ell$ . These selectors are produced by the same packed threshold-extraction routine used below: representative prefix indicators are differenced across adjacent ranks and squared, yielding near-binary selectors before encrypted gather.

---

**Algorithm 3** Encrypted Winner Repacking

---

**Input:** Prefix indicators  $A_t$ , score channel  $\text{score}_i$ , weight channel  $w_i$ , metadata channels  $\text{meta}_i$ , selectors  $E_\ell[j]$

**Output:** Local records  $W[j, b, t]$  and selected records  $S[\ell, b, t]$

```
1: for  $t = 1$  to  $k$  do
2:   if  $t = 1$  then
3:      $\tilde{D}_t \leftarrow A_t$ 
4:   else
5:      $\tilde{D}_t \leftarrow A_t - A_{t-1}$ 
6:   end if
7:    $D_t \leftarrow \tilde{D}_t^2$ 
8:    $\text{score}_t \leftarrow \sum_i D_t[i] \cdot \text{score}_i$ 
9:    $w_t \leftarrow \sum_i D_t[i] \cdot w_i$ 
10:   $\text{meta}_t \leftarrow \sum_i D_t[i] \cdot \text{meta}_i$ 
11:   $W[j, b, t] \leftarrow (\text{score}_t, w_t, \text{meta}_t)$ 
12: end for
13: for  $\ell = 0$  to  $r - 1$  do
14:   for each block  $b$  and local rank  $t$  do
15:      $S[\ell, b, t] \leftarrow \sum_{j=0}^{c-1} E_\ell[j] \cdot W[j, b, t]$ 
16:   end for
17: end for
18: return  $W, S$ 
```

---

Second, the server evaluates Direct NOMOS over all fixed-capacity cluster blocks. For each cluster  $j$  and block  $b$ , block-mode extraction outputs top- $k$  encrypted winner records

$$W[j, b, t], \quad b = 1, \dots, L, \quad t = 1, \dots, k. \quad (23)$$

Each  $W[j, b, t]$  is an encrypted record—not a mask—containing score, weight, and metadata channels; local rank-derived masks are used only during extraction. For a ranking pool of public size  $n$ , C-NOMOS forms packed prefix indicators  $A_t$  with the same gap-amplified sign approximation  $F_\ell$  from Section 4; Appendix B states the calibrated selector margin. The rank- $t$  selector is obtained by adjacent prefix differencing:

$$D_1 = A_1, \quad D_t = A_t - A_{t-1} \quad (t > 1), \quad (24)$$

followed by the cleanup  $D_t \leftarrow D_t^2$ . Then  $D_t$  is an approximate encrypted one-hot selector for the rank- $t$  local winner. It gates the score, weight, and metadata channels to form  $W[j, b, t]$ , as summarized in Algorithm 3. The fixed trace hides local winners and selected clusters from the server.

**5.2.3. Encrypted Composition and Final Merge.** C-NOMOS composes representative-level and local-level outputs without decrypting selected cluster identifiers or routing selected clusters in plaintext. For each representative rank  $\ell$ , the server applies the encrypted selector vector  $E_\ell$  to the block-level records of all clusters:

$$S[\ell, b, t] = \sum_{j=0}^{c-1} E_\ell[j] \cdot W[j, b, t], \quad \ell = 0, \dots, r - 1. \quad (25)$$

This gather/repack step forms encrypted selected records while hiding selected clusters. The same encrypted composition is applied to score, weight, and metadata channels. Correctness requires non-selected selector residuals to remain small relative to the selected-record score gap; Appendix B states the calibrated sufficient condition. The gather/repack stage is dominated by linear ciphertext operations, rotations, and plaintext masks; it does not introduce additional sign-approximation calls.

For each representative rank  $\ell$ , C-NOMOS reduces the encrypted records  $\{S[\ell, b, t]\}_{b=1, \dots, L; t=1, \dots, k}$  into one cluster-level top- $k$  list. The merge schedule is public and fixed by  $L$ ,  $k$ , and  $B$ . If a single ranking pool can hold all  $Lk$  block winners, one merge is sufficient. Otherwise, C-NOMOS applies a hierarchical block-winner merge, whose pool count is denoted by  $H(L, k, B)$  in Section 5.3.

After selected block-winner merging, C-NOMOS has at most  $rk$  encrypted candidates from the  $r$  representative-selected clusters. A final Direct NOMOS merge extracts the global encrypted top- $k$  result from these selected candidates. Its terminal ReLU masks gate the score, weight, and metadata channels of the final merge pool, and the owner-client decrypts the gated records. The metadata channel carries the encrypted record identifier; after decryption, the owner-client thresholds the weight/output mask, normalizes surviving metadata by that mask, and decodes the identifiers. Invalid padding entries keep zero validity throughout the local selections, selected composition, and final merge. If fewer than  $k$  valid records are present, zero-weight outputs may appear, but the owner-client ignores them after decryption.

### 5.3. Complexity Analysis

**Sign-evaluation cost.** Let  $C^{\text{sign}}$  count ciphertext-level gap-sign evaluations, including pairwise ranking signs and packed threshold-selector signs, but excluding terminal ReLU extraction. We write  $\text{RankPool}_t(n)$  for the number of fixed-capacity pairwise ranking pools required to select top- $t$  from  $n$  encrypted candidates. Rotations, masking, packing, bootstrapping, selector repacking, and terminal ReLU evaluation are excluded from  $C^{\text{sign}}$  but included in measured online latency. Direct NOMOS does not require bootstrapping in the evaluated direct configurations; C-NOMOS uses CKKS bootstrapping as a black-box refresh when multi-stage evaluation exhausts the level budget.

**Direct NOMOS.** For Direct NOMOS over an unclustered database of size  $M$ , let

$$n_B = \left\lceil \frac{M}{B} \right\rceil \quad (26)$$

be the number of fixed-capacity ranking blocks. When  $M > B$ , exact global ranking requires all symmetric block pairs, giving

$$C_{\text{Direct}}^{\text{sign}}(M) = \frac{n_B(n_B + 1)}{2} = \Theta(M^2) \quad (27)$$

for fixed  $B$ . This quadratic growth is the main cost that C-NOMOS avoids applying to the full database.

**C-NOMOS.** C-NOMOS starts from fixed-capacity clusters with public capacity

$$U = \left\lceil \frac{\alpha M}{c} \right\rceil \quad (28)$$

and is split into

$$L = \left\lceil \frac{U}{B} \right\rceil \quad (29)$$

fixed-capacity blocks. The cluster-block branch processes all  $cL$  blocks, while the representative branch selects  $r$  clusters and repacks only their encrypted winner records for downstream merging. The high-level sign-evaluation count is

$$C_{\text{C-NOMOS}}^{\text{sign}} = cL + kL + 2 + C_{\text{merge}}^{\text{sign}} + \text{RankPool}_k(rk) + C_{\text{inter}}^{\text{thr}}. \quad (30)$$

Here  $cL$  counts local pairwise ranking,  $kL$  counts packed local threshold selectors, the constant 2 counts representative ranking and packed representative threshold extraction,  $C_{\text{merge}}^{\text{sign}}$  counts selected block-winner merge signs, and the last two terms count the final hierarchical merge before terminal ReLU extraction. The representative path is a fixed-trace scheduling choice that reduces downstream merge volume; Appendix E gives the detailed term accounting.

TABLE 3: Ciphertext-level gap-sign evaluations under  $M = 16384$ ,  $B = 256$ ,  $k = 8$ ,  $\alpha = 2$ ,  $c = 64$ ,  $r = 48$ , and  $U = 512$ , excluding terminal ReLU extraction. These parameters give  $n_B = 64$ ,  $L = 2$ , and  $H(L, k, B) = 1$ .

|              | Equation                                                                                 | Number | Reduction     |
|--------------|------------------------------------------------------------------------------------------|--------|---------------|
| Direct NOMOS | $n_B(n_B + 1)/2$                                                                         | 2080   | $1 \times$    |
| C-NOMOS      | $cL + kL + 2 + C_{\text{merge}}^{\text{sign}} + R_k(rk) + C_{\text{inter}}^{\text{thr}}$ | 152    | $13.7 \times$ |

In Table 3,  $R_t(n)$  abbreviates  $\text{RankPool}_t(n)$ . At  $M = 16384$ , C-NOMOS reduces the Direct NOMOS gap-sign count from 2080 to 152 ( $13.7 \times$ ), while full measured latency also includes gather/repack, rotations, packing, bootstrapping, terminal ReLU extraction, and other ciphertext-layout costs.

## 6. Evaluation

This section evaluates NOMOS through microbenchmarks, system-level comparison with a state-of-the-art FHE database system supporting top- $k$  queries, and real-world validation on SIFT.

### 6.1. Experimental Setup

Experiments run on one Intel Xeon Platinum 8473C CPU with 52 cores, 104 hardware threads, and 512GB RAM. Our OpenFHE [15] implementation uses OpenMP and 128-bit CKKS parameters [25] with  $N = 2^{16}$  where

depth and slot budgets suffice and  $N = 2^{17}$  for larger workloads that need higher depth or a 256-record single-ciphertext layout. Online latency covers encrypted input processing through encrypted output generation and owner-client result detection; offline key/context generation is excluded. Synthetic workloads use 16-dimensional normalized vectors with a fixed sampled query per configuration, checked against plaintext  $k$ NN on the same sampled database. For the SIFT dataset [26], plaintext ground truth is computed over the same sampled database subset and query/layout used by the encrypted run.

Baselines use the same machine and multithreading. The Mazzone-style ranking benchmark uses the same  $N = 2^{16}$  as our ranking microbenchmark; its larger- $M$  points use the released  $f/g$  setting, which lowers latency but keeps a deeper comparison circuit. Engorgio uses its released single-database implementation, the same minimum distance gap, and online top- $k$  latency. This is an online trace-to-trace comparison under released runtime-oriented configurations, without offline setup or a separate accuracy-only Engorgio tuning point. Ground truth is plaintext  $k$ NN and memory is peak RSS. Throughout,  $M$  denotes the unclustered database size for Direct NOMOS or the pre-clustering total for C-NOMOS.

### 6.2. Microbenchmarks

**Dead-zone half-width.** Table 4 reports the nominal dead-zone half-width defined in Equation (6). For each  $M$ , the saturation target follows from  $\varepsilon_s < 1/(2(M - 1))$ , and the reported value is the smallest positive input at which the approximation reaches  $1 - \varepsilon_s$  (with its negative side reaching  $\varepsilon_s$ ); smaller values indicate finer rank resolution near zero. Columns are ordered by their value at  $M = 128$ , from larger to smaller half-width.

The selection rule is to choose the lowest-depth gap variant whose dead-zone half-width falls below the workload’s minimum distance gap. Single-gap suffices for the Mazzone-style ranking benchmark with gap 0.01, while the Engorgio comparison uses triple-gap for gap 0.0001. This targets near-zero ranking resolution rather than global uniform approximation error. Cheb1007 has fewer nominal levels but far more plaintext-ciphertext multiplications, and stronger Cheon-style variants require more levels.

**Ranking benchmark.** Following the resolution setting used by the Mazzone-style benchmark, we use a minimum distance gap of 0.01 and evaluate the ranking stage with single-gap. This is a ranking-to-ranking microbenchmark: both systems start from encoded distance values and measure encrypted ranking, not full  $k$ NN with distance computation and extraction. Figure 6 shows that Direct NOMOS reduces ranking latency by  $2 \times - 8 \times$  across the evaluated sizes, with the largest speedup at  $M = 512$ .

### 6.3. NOMOS Performance

**6.3.1. Comparison with a State-of-the-Art FHE Top- $k$  System.** Figure 7 compares Direct NOMOS with Engorgio

TABLE 4: Nominal dead-zone half-width under the rank-accuracy target. Smaller is better. The suffix after “/” gives sign-approximation levels.

| $M$   | $f = 3, g = 2/15$     | Single-gap/11         | $f = 3, g = 3/18$     | Cheb1007 <sup>†</sup> /10 | $f = 4, g = 3/21$     | Double-gap/15         | $f = 4, g = 4/24$     | Triple-gap/19         |
|-------|-----------------------|-----------------------|-----------------------|---------------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| 128   | $4.90 \times 10^{-3}$ | $4.17 \times 10^{-3}$ | $2.27 \times 10^{-3}$ | $1.54 \times 10^{-3}$     | $5.06 \times 10^{-4}$ | $4.76 \times 10^{-4}$ | $2.32 \times 10^{-4}$ | $5.44 \times 10^{-5}$ |
| 256   | $5.31 \times 10^{-3}$ | $4.41 \times 10^{-3}$ | $2.46 \times 10^{-3}$ | $1.55 \times 10^{-3}$     | $5.50 \times 10^{-4}$ | $5.04 \times 10^{-4}$ | $2.52 \times 10^{-4}$ | $5.75 \times 10^{-5}$ |
| 512   | $5.70 \times 10^{-3}$ | $4.68 \times 10^{-3}$ | $2.65 \times 10^{-3}$ | $1.55 \times 10^{-3}$     | $5.91 \times 10^{-4}$ | $5.34 \times 10^{-4}$ | $2.71 \times 10^{-4}$ | $6.10 \times 10^{-5}$ |
| 1024  | $6.06 \times 10^{-3}$ | $6.68 \times 10^{-3}$ | $2.82 \times 10^{-3}$ | $1.56 \times 10^{-3}$     | $6.30 \times 10^{-4}$ | $7.62 \times 10^{-4}$ | $2.89 \times 10^{-4}$ | $8.70 \times 10^{-5}$ |
| 2048  | $6.40 \times 10^{-3}$ | $6.73 \times 10^{-3}$ | $2.99 \times 10^{-3}$ | $1.56 \times 10^{-3}$     | $6.67 \times 10^{-4}$ | $7.69 \times 10^{-4}$ | $3.06 \times 10^{-4}$ | $8.78 \times 10^{-5}$ |
| 4096  | $6.73 \times 10^{-3}$ | $6.76 \times 10^{-3}$ | $3.15 \times 10^{-3}$ | $1.56 \times 10^{-3}$     | $7.02 \times 10^{-4}$ | $7.72 \times 10^{-4}$ | $3.23 \times 10^{-4}$ | $8.81 \times 10^{-5}$ |
| 8192  | $7.04 \times 10^{-3}$ | $6.77 \times 10^{-3}$ | $3.30 \times 10^{-3}$ | $1.56 \times 10^{-3}$     | $7.36 \times 10^{-4}$ | $7.73 \times 10^{-4}$ | $3.38 \times 10^{-4}$ | $8.83 \times 10^{-5}$ |
| 16384 | $7.33 \times 10^{-3}$ | $6.78 \times 10^{-3}$ | $3.45 \times 10^{-3}$ | $1.56 \times 10^{-3}$     | $7.69 \times 10^{-4}$ | $7.74 \times 10^{-4}$ | $3.54 \times 10^{-4}$ | $8.84 \times 10^{-5}$ |

The  $f, g$  columns are Cheon-style composite sign approximations. <sup>†</sup>Cheb1007 exhibits strong post-threshold oscillations.

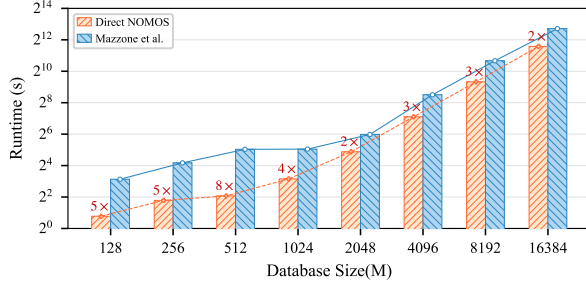
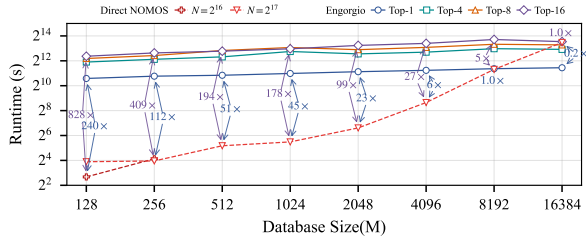
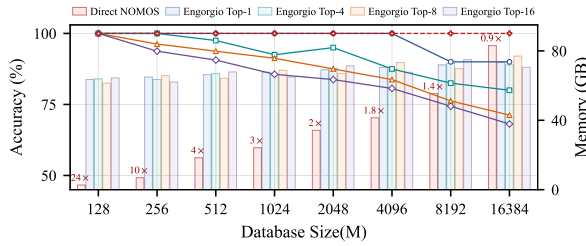


Figure 6: Ranking-only microbenchmark over encoded distances.



(a) Online runtime.



(b) Accuracy (lines) and memory reduction (bars).

Figure 7: Direct NOMOS vs. Engorgio on unclustered databases. Runtime annotations reference Top-16/Top-1 baselines; memory bars reference Top-1.

on unclustered databases, using the faster Direct NOMOS ring configuration at each  $M$ . Engorgio’s top- $k$  path scales as  $O(M \log_2 k)$ ; Direct NOMOS changes only the ReLU bias when  $k$  varies, leaving the ranking trace and polynomial degree unchanged for fixed  $M$ .

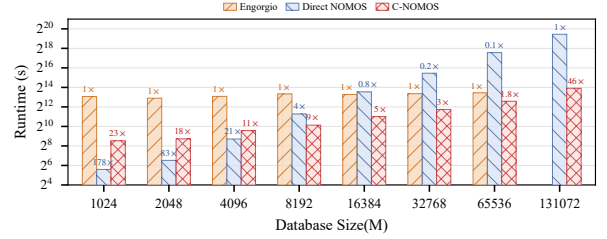


Figure 8: Top-8 online scaling; C-NOMOS reports complete online time, and the rightmost speedup is vs. Direct NOMOS.

The largest measured speedup is at  $M = 128$ :  $828\times$  over Engorgio Top-16 and  $240\times$  over Top-1. At this largest-speedup point, the released Engorgio configuration also matches plaintext top- $k$  accuracy in Figure 7b. The advantage narrows with  $M$  as the unclustered global ranking stage grows quadratically; ranking already accounts for 74.8%, 98.2%, and 99.2% of online time at  $M = 2048$ , 8192, and 16384. At  $M = 16384$  Engorgio top-1 is faster, motivating C-NOMOS.

Figure 7b reports accuracy and peak memory. Direct NOMOS matches plaintext top- $k$  results in all evaluated Engorgio-comparison settings. Engorgio uses its released runtime-oriented configuration; the plotted accuracy is reported at the same runtime point, and higher precision settings would define a different latency point. We therefore treat the Engorgio curve as an online baseline at its released configuration, not as an iso-accuracy comparison; the largest reported speedup point is accuracy-matched. Memory trends mirror runtime: Direct NOMOS is efficient in the direct regime, while larger unclustered sizes increase ranking-layout pressure.

**6.3.2. Direct NOMOS and C-NOMOS Scaling.** Figure 8 evaluates top-8 database-size scaling under the same gap 0.0001 and triple-gap setting as the Engorgio comparison. C-NOMOS time is complete online runtime for representative top- $r$ , block-local top- $k$ , selector repacking, selected-winner grouping, compaction, and terminal top- $k$ /ReLU extraction. Engorgio is evaluated up to  $M = 65536$ , the largest size supported by its released single-trace implementation in our setup, so the final speedup is relative to Direct NOMOS.

TABLE 5: SIFT validation for NOMOS. Direct NOMOS time entries report top-1/top-8; C-NOMOS reports clustered top-8. Prec./Rec. values are percentages.

| Method  | $M$    | $c$ | $r$ | Time (s)        | Prec./Rec. |
|---------|--------|-----|-----|-----------------|------------|
| Direct  | 128    | –   | –   | 7.9/8.0         | 100/100    |
| Direct  | 256    | –   | –   | 17.4/17.3       | 100/100    |
| Direct  | 512    | –   | –   | 45.4/45.6       | 100/100    |
| Direct  | 1024   | –   | –   | 54.9/54.6       | 100/100    |
| Direct  | 2048   | –   | –   | 117.4/121.9     | 100/100    |
| Direct  | 4096   | –   | –   | 584.2/657.6     | 100/100    |
| Direct  | 8192   | –   | –   | 2520.2/2508.5   | 100/100    |
| Direct  | 16384  | –   | –   | 11432.5/11371.6 | 100/100    |
| C-NOMOS | 1024   | 8   | 8   | 222.7           | 100/100    |
| C-NOMOS | 2048   | 16  | 16  | 265.5           | 100/100    |
| C-NOMOS | 4096   | 32  | 32  | 551.3           | 100/100    |
| C-NOMOS | 8192   | 64  | 32  | 933.4           | 100/100    |
| C-NOMOS | 16384  | 64  | 48  | 1941.6          | 100/100    |
| C-NOMOS | 32768  | 64  | 48  | 3401.1          | 100/100    |
| C-NOMOS | 65536  | 128 | 64  | 6142.1          | 100/100    |
| C-NOMOS | 131072 | 256 | 96  | 15178.1         | 100/100    |

At  $M = 131072$ , C-NOMOS reaches  $46\times$  speedup over the Direct NOMOS timing reference, which no longer preserves full top-8 recall under this configuration.

The public C-NOMOS configurations appear in Appendix D. Direct NOMOS is effective while a single global ranking stage remains competitive; beyond that point, C-NOMOS invokes it inside fixed-capacity cluster blocks and merges selected winner records. A 100-trial representative-count sweep guides the public choice of  $r$ , which is an accuracy-latency tuning parameter, not a universal constant.

C-NOMOS achieves 100% precision and recall across the reported clustered validation configurations, with public  $r$  chosen no smaller than the observed sweep maximum; at the smallest sizes this can select all clusters. Unclustered Direct NOMOS remains 100% accurate through  $M = 16384$ , then recovers 7/8, 6/8, and 4/8 top-8 results at  $M = 32768$ , 65536, and 131072. In the large direct-workload scaling runs, Direct NOMOS uses a degree-1007 extraction cap. The empirical boundary motivates C-NOMOS by tying extraction and merge costs to  $B$ ,  $r$ , and  $k$  rather than the global  $M$ .

#### 6.4. Real-World Validation on SIFT

We further evaluate NOMOS on SIFT real feature vectors. SIFT inputs use the same public scaling and distance-bound normalization as the encrypted run, and plaintext ground truth is computed after the same preprocessing. Table 5 reports Direct NOMOS top-1/top-8 retrieval over 128 to 16384 records and C-NOMOS clustered top-8 retrieval over 1024 to 131072 records. The reported Direct NOMOS and C-NOMOS SIFT runs achieve 100% precision and recall. The C-NOMOS online totals include configured bootstrap-refresh cost but exclude owner-side clustering, context setup, and key generation. We do not include a separate Engorgio SIFT tuning study, as higher Engorgio precision settings would define different runtime points.

**Discussion and limitations.** NOMOS targets a two-party single-owner outsourcing model; multi-owner clustering would require a separate privacy-preserving setup. C-NOMOS is a clustered scheduling wrapper, so recall is empirical and depends on the public  $r$  sweep, although the reported clustered SIFT top-8 runs achieve 100% precision and recall. Absolute costs remain high at the largest direct sizes, so near-term deployment is most practical for hundreds to a few thousand records; C-NOMOS addresses clustered scaling beyond this regime.

## 7. Conclusion

We presented NOMOS, a CKKS-based encrypted  $k$ NN protocol family for the ranking and top- $k$  extraction bottleneck in a two-party single-owner outsourcing model. Direct NOMOS combines CKKS-native distance computation, gap-amplified ranking, slot-index alignment, and ReLU-based terminal extraction to evaluate encrypted  $k$ NN without full sorting, while C-NOMOS wraps the same primitive with owner-side capacity-bounded clustering and encrypted selector repacking for larger databases. The resulting design reduces the dominant encrypted sign-evaluation cost without hiding the remaining layout, bootstrapping, and merge costs. Our OpenFHE implementation is  $2\times$ – $8\times$  faster than the Mazzone-style ranking benchmark and reaches up to  $828\times$  speedup over Engorgio in the smaller direct regime, where the largest-speedup point is accuracy-matched. In database-size scaling, C-NOMOS reaches  $46\times$  speedup at  $M = 131072$  where the Direct NOMOS timing reference no longer preserves full top-8 recall; on SIFT, Direct NOMOS achieves 100% top-1/top-8 accuracy in the reported direct runs, and C-NOMOS achieves 100% precision and recall in the reported clustered top-8 runs.

## References

- [1] M. Zuber and R. Sirdey, “Efficient homomorphic evaluation of  $k$ -NN classifiers,” *Proceedings on Privacy Enhancing Technologies*, vol. 2021, no. 2, pp. 111–129, 2021.
- [2] Zama, “TFHE-rs: A pure Rust implementation of the TFHE scheme for boolean and integer arithmetics over encrypted data,” 2022, <https://github.com/zama-ai/tfhe-rs>.
- [3] K. Cong, R. Geelen, J. Kang, and J. Park, “Revisiting oblivious top- $k$  selection with applications to secure  $k$ -NN classification,” in *Selected Areas in Cryptography (SAC ’24)*, ser. Lecture Notes in Computer Science, vol. 15516. Springer, 2024, pp. 3–25.
- [4] S. Azogagh, M.-O. Killijian, and F. Larose-Gervais, “A non-comparison oblivious sort and its application to private  $k$ -NN,” *IACR Cryptology ePrint Archive*, no. 1894, 2024.
- [5] S. Hong, S. Kim, J. Choi, Y. Lee, and J. H. Cheon, “Efficient sorting of homomorphic encrypted data with  $k$ -way sorting network,” *IEEE Transactions on Information Forensics and Security (TIFS 2021)*, vol. 16, pp. 4389–4404, 2021.
- [6] S. Bian, H. Pan, J. Hu, Z. Zhang, Y. Fu, J. Hua, Y. Chen, B. Zhang, Y. Jin, J. Dong, and Z. Guan, “Engorgio: An arbitrary-precision unbounded-size hybrid encrypted database via quantized fully homomorphic encryption,” in *34th USENIX Security Symposium (USENIX Security ’25)*. USENIX Association, 2025, pp. 8441–8460.

- [7] F. Mazzone, M. H. Everts, F. Hahn, and A. Peter, “Efficient ranking, order statistics, and sorting under CKKS,” in *34th USENIX Security Symposium (USENIX Security ’25)*. USENIX Association, 2025, pp. 8541–8558.
- [8] S. Kim, E. Ünay, A. Yilmazer-Metin, and H. T. Lee, “Optimized rank sort for encrypted real numbers,” *IACR Cryptology ePrint Archive*, no. 1170, 2025.
- [9] L. Rovidia, A. Leporati, and S. Basile, “Lightweight sorting in approximate homomorphic encryption,” *IACR Cryptology ePrint Archive*, no. 1150, 2025.
- [10] J. H. Cheon, A. Kim, M. Kim, and Y. S. Song, “Homomorphic encryption for arithmetic of approximate numbers,” in *Advances in Cryptology (ASIACRYPT ’17)*, ser. Lecture Notes in Computer Science, vol. 10624. Springer, 2017, pp. 409–437.
- [11] H. Chen, I. Chillotti, and Y. Song, “Improved bootstrapping for approximate homomorphic encryption,” in *Advances in Cryptology (EUROCRYPT ’19)*, ser. Lecture Notes in Computer Science, vol. 11477. Springer, 2019, pp. 34–54.
- [12] J. H. Cheon, D. Kim, and D. Kim, “Efficient homomorphic comparison methods with optimal complexity,” in *Advances in Cryptology (ASIACRYPT ’20)*, ser. Lecture Notes in Computer Science, vol. 12492. Springer, 2020, pp. 221–256.
- [13] E. Lee, J.-W. Lee, J.-S. No, and Y.-S. Kim, “Minimax approximation of sign function by composite polynomial for homomorphic comparison,” *IEEE Transactions on Dependable and Secure Computing (TDSC 2022)*, vol. 19, no. 6, pp. 3711–3727, 2022.
- [14] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, “A full RNS variant of approximate homomorphic encryption,” in *Selected Areas in Cryptography (SAC ’18)*, ser. Lecture Notes in Computer Science, vol. 11349. Springer, 2018, pp. 347–368.
- [15] A. A. Badawi, J. Bates, F. Bergamaschi, D. B. Cousins, S. Erabelli, N. Genise, S. Halevi, H. Hunt, A. Kim, Y. Lee, Z. Liu, D. Micciancio, I. Quah, Y. Polyakov, S. R.V., K. Rohloff, J. Saylor, D. Sponitsky, M. Triplett, V. Vaikuntanathan, and V. Zucca, “OpenFHE: Open-source fully homomorphic encryption library,” in *10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography (WAHC ’22)*. Association for Computing Machinery, 2022, pp. 53–63.
- [16] A. Kim, A. Papadimitriou, and Y. Polyakov, “Approximate homomorphic encryption with reduced approximation error,” in *Topics in Cryptology (CT-RSA ’22)*, ser. Lecture Notes in Computer Science, vol. 13161. Springer, 2022, pp. 120–144.
- [17] K. Han and D. Ki, “Better bootstrapping for approximate homomorphic encryption,” in *Topics in Cryptology (CT-RSA ’20)*, ser. Lecture Notes in Computer Science, vol. 12006. Springer, 2020, pp. 364–390.
- [18] J.-P. Bossuat, C. Mouchet, J. R. Troncoso-Pastoriza, and J.-P. Hubaux, “Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys,” in *Advances in Cryptology (EUROCRYPT ’21)*, ser. Lecture Notes in Computer Science, vol. 12696. Springer, 2021, pp. 587–617.
- [19] L. N. Trefethen, *Approximation theory and approximation practice, extended edition*. SIAM, 2019.
- [20] J.-W. Lee, E. Lee, Y. Lee, Y.-S. Kim, and J.-S. No, “High-precision bootstrapping of RNS-CKKS homomorphic encryption using optimal minimax polynomial approximation and inverse sine function,” in *Advances in Cryptology (EUROCRYPT ’21)*, ser. Lecture Notes in Computer Science, vol. 12696. Springer, 2021, pp. 618–647.
- [21] M. Paterson and L. J. Stockmeyer, “On the number of nonscalar multiplications necessary to evaluate polynomials,” *SIAM Journal on Computing*, vol. 2, no. 1, pp. 60–66, 1973.
- [22] F. Mazzone, T. Brown, F. Kerschbaum, K. H. Wilson, M. Everts, F. Hahn, and A. Peter, “Privacy-preserving vertical k-means clustering,” *arXiv preprint arXiv:2504.07578*, 2025.
- [23] B. Li and D. Micciancio, “On the security of homomorphic encryption on approximate numbers,” in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2021, pp. 648–677.
- [24] S. Halevi and V. Shoup, “Algorithms in HELib,” in *Advances in Cryptology (CRYPTO ’14)*, ser. Lecture Notes in Computer Science, vol. 8616. Springer, 2014, pp. 554–571.
- [25] M. R. Albrecht, M. Chase, H. Chen, J. Ding, S. Goldwasser, S. Gorbunov, S. Halevi, J. Hoffstein, K. Laine, K. E. Lauter, S. Lokam, D. Micciancio, D. Moody, T. Morrison, A. Sahai, and V. Vaikuntanathan, “Homomorphic encryption standard,” *IACR Cryptology ePrint Archive*, no. 939, 2019.
- [26] H. Jegou, M. Douze, and C. Schmid, “Product quantization for nearest neighbor search,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 1, pp. 117–128, 2010.

## Appendix A. Fixed-Trace Security Argument

This appendix formalizes the simulation statement used in Section 3.2. The statement is intentionally scoped to the semi-honest online view and to the leakage already listed in the main text.

**Proposition 1** (Fixed-trace online privacy). *Assume CKKS encryption is semantically secure. For Direct NOMOS, let  $\mathcal{L}_D$  contain the public parameters, ciphertext dimensions, and fixed online trace. For C-NOMOS, let  $\mathcal{L}_C$  additionally contain the setup-visible clustered-layout metadata, including owner-chosen public shape parameters. Then the online view of any semi-honest server executing NOMOS is computationally simulatable from  $\mathcal{L}_D$  or  $\mathcal{L}_C$ .*

*Proof.* The simulator creates encryptions of zero with the same ciphertext dimensions as the real encrypted database, query, representatives, validity channels, and intermediate records implied by the leakage. It then replays the public sequence of homomorphic additions, multiplications, rotations, plaintext masks, polynomial evaluations, and bootstrapping calls in the fixed trace. Because NOMOS has no data-dependent branching or intermediate decryption in the online protocol, the simulated trace has the same public shape and operation schedule as the real trace. A standard hybrid argument replaces each real input ciphertext by an encryption of zero; semantic security makes adjacent hybrids indistinguishable, and deterministic homomorphic evaluation of public circuits preserves indistinguishability. Thus the server’s online view reveals no information beyond the stated leakage. The proposition does not claim malicious security, output integrity, circuit privacy, or hiding of the owner-chosen public setup parameters included in  $\mathcal{L}_C$ .  $\square$

## Appendix B. Selector-Repacking Error Condition

This appendix spells out the calibrated condition behind the C-NOMOS selector-repacking step. It is a sufficient condition for the evaluated fixed-trace construction, not a distribution-independent recall guarantee.

For a ranking pool of public size  $n$ , C-NOMOS first maps the raw rank to the descending grid

$$\rho_i = \frac{2}{n}(n - \text{rank}[i]).$$

For local position  $t$ , the public threshold is  $\theta_t = 2(n-t)/n$ , and the packed prefix indicator is

$$A_t[i] = F_\ell(0.25(\rho_i - \theta_t)),$$

where  $F_\ell$  is the same gap-amplified sign approximation used for ranking. The factor 0.25 maps the rank-grid difference back into the calibrated comparator domain. The ideal scaled threshold half-margin is  $1/(4n)$ ; for the evaluated pools with  $n \leq B = 256$ , this is at least  $1/(4B) \approx 9.8 \times 10^{-4}$ , above the triple-gap dead-zone scale reported in Table 4.

**Lemma 1** (Sufficient selector-repacking margin). *Consider a C-NOMOS gather/repack step*

$$S = \sum_{j=0}^{c-1} E[j] \cdot W[j],$$

where one cluster  $j^*$  is selected. Suppose each scalar channel of every encrypted winner record is bounded by  $R$ , the selected selector satisfies  $|E[j^*] - 1| \leq \eta$ , and every non-selected selector satisfies  $|E[j]| \leq \eta$ . Then the scalar-channel perturbation introduced by gather/repack is at most  $\eta cR$ . If the local winner extraction for each  $W[j]$  has channel perturbation at most  $\xi$  and CKKS numerical error contributes at most  $\nu$ , then any downstream merge decision with plaintext score gap

$$\Gamma > 2(\xi + \eta cR + \nu)$$

is preserved after selector repacking.

*Proof.* For one scalar channel, the ideal selected value is  $W[j^*]$ . The gathered value differs from it by

$$(E[j^*] - 1)W[j^*] + \sum_{j \neq j^*} E[j]W[j],$$

whose magnitude is at most  $\eta R + (c-1)\eta R \leq \eta cR$ . Adding the local extraction perturbation  $\xi$  and CKKS numerical error  $\nu$  gives a total per-record score perturbation budget of  $\xi + \eta cR + \nu$ . A pairwise merge comparison changes the difference between two scores by at most twice this budget, so a plaintext gap larger than  $2(\xi + \eta cR + \nu)$  keeps the comparison sign unchanged. Applying this condition to every boundary comparison that can affect top- $k$  selection preserves the selected top- $k$  records for that calibrated configuration.  $\square$

## Appendix C. TransRanking Details

Algorithm 4 derives complementary rankings through rotation-based transformations, avoiding redundant polynomial evaluations in multi-ciphertext mode.

---

### Algorithm 4 TransRanking

---

**Input:** Sign output  $\text{ct}_{\text{sign}_{ij}}$ , partition size  $m'$

**Output:** Complementary ranking  $\text{ct}_{\text{rank}_{ji}}$

```

1:  $\text{ct}_{\text{rank}_{ji}} \leftarrow \text{ct}_{\text{sign}_{ij}}$ 
2: for  $i = 0$  to  $\lceil \log_2(m') \rceil - 1$  do
3:    $\text{ct}_{\text{rank}_{ji}} \leftarrow \text{Add}(\text{ct}_{\text{rank}_{ji}}, \text{Rot}(\text{ct}_{\text{rank}_{ji}}, 2^i))$ 
4: end for
5: Initialize extraction mask  $\text{pt}_{\text{mask}} \leftarrow 0$ 
6: for  $i = 0$  to  $m' - 1$  do
7:    $\text{pt}_{\text{mask}}[i \times m'] \leftarrow 1$ 
8: end for
9:  $\text{ct}_{\text{rank}_{ji}} \leftarrow \text{PMult}(\text{ct}_{\text{rank}_{ji}}, \text{pt}_{\text{mask}})$ 
10: for  $i = 1$  to  $\lceil \log_2(m') \rceil$  do
11:    $\text{ct}_{\text{rank}_{ji}} \leftarrow \text{Add}(\text{ct}_{\text{rank}_{ji}}, \text{Rot}(\text{ct}_{\text{rank}_{ji}}, m' \cdot (m' - 1)/2^i))$ 
12: end for
13: return  $\text{ct}_{\text{rank}_{ji}}$ 

```

---

## Appendix D. C-NOMOS Evaluation Configurations

Table 6 lists the public C-NOMOS configuration parameters used in Figure 8. Here  $M$  is the total database size before clustered preprocessing,  $c$  is the number of clusters,  $r$  is the number of selected representative clusters,  $B$  is the block capacity,  $\alpha$  is the capacity slack parameter, and  $U = \lceil \alpha M/c \rceil$  is the fixed cluster capacity.

TABLE 6: C-NOMOS configurations for the scaling experiment.

| $M$    | $c$ | $r$ | $B$ | $\alpha$ | $U$  |
|--------|-----|-----|-----|----------|------|
| 1024   | 8   | 8   | 256 | 2        | 256  |
| 2048   | 16  | 16  | 256 | 2        | 256  |
| 4096   | 32  | 32  | 256 | 2        | 256  |
| 8192   | 64  | 32  | 256 | 2        | 256  |
| 16384  | 64  | 48  | 256 | 2        | 512  |
| 32768  | 64  | 48  | 256 | 2        | 1024 |
| 65536  | 128 | 64  | 256 | 2        | 1024 |
| 131072 | 256 | 96  | 256 | 2        | 1024 |

## Appendix E. C-NOMOS Sign-Count Details

This appendix expands the sign-count terms used in Section 5.3. The cluster-block branch produces local winner records  $W[j, b, t]$  for every public cluster block. Local pairwise ranking contributes  $cL$  gap-sign evaluations. Local winner extraction contributes  $kL$  packed threshold-selector signs: for each block layer and threshold, records from all  $c$  clusters are packed into one ciphertext-level batched sign evaluation.

The representative branch contributes two signs when the representative pool and its packed thresholds fit the ciphertext layout: one pairwise ranking sign over representatives and one packed threshold-selector sign for  $A_1, \dots, A_r$ .

This is the case for the reported configurations with  $c \leq B$  and packed threshold slots within the slot budget. Selector repacking then forms  $S[\ell, b, t]$  from  $E_\ell[j]$  and  $W[j, b, t]$  using rotations and masks. Its layout cost is  $G_{\text{select}} = \Theta(rkcL) = \Theta(rkM/B)$  and is included in measured latency, but it is not counted in  $C^{\text{sign}}$ .

Let  $H(L, k, B)$  denote the selected block-winner merge pool count. No selected merge is needed when  $L = 1$ ; if  $Lk \leq B$  and  $L > 1$ , then  $H(L, k, B) = 1$ . The term  $C_{\text{merge}}^{\text{sign}}$  counts the pairwise and threshold-selector signs for these pools after segmented packing.

Finally,  $\text{RankPool}_k(rk)$  counts only the pairwise ranking pools in the final hierarchical merge. Non-terminal final-merge pools must repack encrypted winner records for another merge layer, so their packed threshold-selector signs are counted by  $C_{\text{inter}}^{\text{thr}}$ . Terminal ReLU masks are excluded from  $C^{\text{sign}}$ .

For Table 3, the terms are local ranking  $64 \cdot 2 = 128$ , local extraction  $8 \cdot 2 = 16$ , representative branch 2, selected block-winner merging 2,  $\text{RankPool}_8(384) = 3$ , and one non-terminal final-merge selector, for  $128 + 16 + 2 + 2 + 3 + 1 = 152$ .

The slot budget fits the evaluated  $N = 2^{17}$  layout with 65536 CKKS slots. Local threshold extraction packs  $64 \cdot 256 = 16384$  slots per threshold, so  $k = 8$  thresholds over  $L = 2$  layers give 16 packed threshold signs. Selected block-winner merging packs  $r(Lk)^2 = 48 \cdot 16^2 = 12288$  comparison slots, so one pairwise sign and one packed threshold-selector sign give  $C_{\text{merge}}^{\text{sign}} = 2$ . The final merge has  $rk = 384$  records and uses a hierarchical schedule with two first-layer ranking pools and one terminal ranking pool, so  $\text{RankPool}_8(384) = 3$ . The two first-layer pools are non-terminal and their threshold selectors are packed into one ciphertext-level threshold sign, giving  $C_{\text{inter}}^{\text{thr}} = 1$ ; the terminal pool exposes final masks through ReLU, which stays outside  $C^{\text{sign}}$ .