MESS: FAST AND PRIVATE SEMANTIC SEARCH ON MULTI-GRAPH HNSW

Anonymous authors
Paper under double-blind review

ABSTRACT

Semantic search systems map data to a high-dimensional vector space and support retrieval of similar data via approximate nearest neighbor search. When the system is hosted by untrusted cloud servers, plaintext data and queries receive no privacy protection. Our goal is to design a system with three properties: privacy, accuracy, and efficiency. Existing works adopt either homomorphic encryption (HE), oblivious RAM (ORAM), or a differential privacy (DP) approach. They fall short of achieving all three properties.

In this paper, we present MESS, a system that realizes our goal. It maps the original vectors into binary codes, applies locality-sensitive hashing (LSH) and randomized response, and distributes a multi-graph Hierarchical Navigable Small World (HNSW) index over multiple non-colluding servers, one graph shard per server. Randomized authenticated encryption protects payload contents, while extended differential privacy (XDP) gives explicit leakage bounds separately for each server's local view of stored embeddings and query transcripts. A two-phase query perturbation mechanism further bounds repeated-query leakage, without claiming oblivious access or perfect query unlinkability. We make no joint-view privacy claim across servers. The multi-graph index mitigates the impact of perturbation on result quality, thereby achieving accuracy. MESS is efficient because each server searches its perturbed codes locally and in parallel, without the overhead of homomorphic encryption or ORAM. We provide a formal analysis of the per-server privacy guarantees and an extensive evaluation of its performance. The results show that MESS achieves up to $15.08\times$ lower latency than state-of-the-art baselines.

1 INTRODUCTION

Semantic search systems support a wide range of applications, from classic information retrieval, image matching Lowe (2004); Sivic & Zisserman (2003), and recommendation Sarwar et al. (2001) to pattern recognition Cover & Hart (1967); Weinberger & Saul (2009) and recent retrieval-augmented generation (RAG) applications. A typical semantic search system maps data and queries to high-dimensional vectors (or embeddings). Its main operation is approximate nearest neighbor (ANN) search, which finds the closest vectors to a given search vector using a distance metric. Recent RAG applications, driven by the success of large language models (LLMs), have led to renewed interest in ANN algorithms, indices, and vector database systems that enable efficient and scalable semantic search Hu et al. (2025); Lao et al. (2026); Wang et al. (2025). As data and query volumes grow, semantic search systems are migrating to the cloud Google Cloud (2026); Amazon Web Services (2026); Weaviate (2026). Specifically, cloud-based semantic search services are attractive because they can be efficient, accurate, cost-effective, and scalable. However, they offer no privacy guarantees, as they operate on raw (or plaintext) embeddings and user queries, which can reveal sensitive information about the data and the user. As a consequence, they do not support applications that process sensitive data, such as private search over personal photos, documents, and personal AI agents Kaviani et al. (2026).

Our goal is to design a semantic search system with three properties: security, accuracy, and efficiency. Security here means confidentiality of encrypted payloads together with quantified leakage bounds for perturbed stored codes and query transcripts, including access- and search-pattern observations; it does not mean ORAM-style obliviousness. Accuracy means that the search results

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

have high recall, while efficiency means low search overhead. The main challenge is to satisfy all three properties simultaneously. Existing works on private semantic search adopt one of the following three approaches. First, the homomorphic encryption and secure-computation approach, for example, Chen et al. (2020); Riazi et al. (2019); Liu et al. (2025); Li et al. (2025), computes vector distance directly on encrypted vectors. This approach achieves privacy but is inefficient due to the overhead of encrypted computation with high-dimensional vectors. Even when using clustering to reduce the search space, the search cost is still significant because of the large number of candidate vectors. Second, the ORAM approach, for example Zhu et al. (2025); Ahmad et al. (2021), achieves privacy by hiding the vectors being accessed during search. In particular, Compass Zhu et al. (2025) combines an efficient vector index, ORAM, and product quantization to protect graph-access patterns Zhu et al. (2025). However, it still fails to achieve efficiency because adaptive graph traversal requires multiple ORAM operations, increasing communication overhead. Third, the differential privacy (DP) approach, for example Fernandes et al. (2021); Aumüller et al. (2020); Chen et al. (2018), trades privacy for efficiency by reducing strict privacy to quantified leakage and allowing distance computation on perturbed vectors. However, existing works do not consider any index, thus suffering from high search overhead. They fail to achieve accuracy due to recall degradation caused by vector perturbations.

We present MESS as a semantic search system with M non-colluding servers. It addresses the above challenge via a combination of differential privacy and a multi-graph index. Each server maintains one hierarchical navigable small world (HNSW) graph for ANN search. The client applies locality-sensitive hashing (LSH) followed by LSH randomized response before distributing perturbed codes to the selected servers. Randomized authenticated encryption provides payload confidentiality, while the randomized-response mechanisms provide quantified leakage bounds for each server-local view of stored codes and query transcript. MESS overcomes the accuracy challenge of the DP-based approach by searching multiple graphs and aggregating their candidates at the client, which reduces rank distortion and candidate set expansion. A two-phase query-perturbation process bounds leakage from repeated-query search patterns. We next summarize the design and contributions of MESS.

- We propose an efficient HNSW search over perturbed binary embeddings. The client maps data and query vectors to binary codes, then applies an LSH randomized response mechanism to them. Each of M servers builds one HNSW graph over its local perturbed codes and performs Hamming-distance search. The client aggregates the encrypted candidates returned in one parallel request–response round.
- We propose a multi-graph design that addresses search quality degradation caused by perturbations, and a two-stage randomization process to protect repeated queries against averaging and direct linkage attacks. In particular, each query is sent to all M servers and their results are aggregated by the client, whereas each database record is independently routed to only t servers. Furthermore, the client applies a permanent and an instantaneous randomization to each query.
- We analyze payload confidentiality and quantified leakage bounds separately for each non-colluding server's stored index, query access transcript, and repeated-query search pattern; we neither combine server views nor make a joint-view privacy claim across servers.
- We evaluate MESS on standard datasets and compare it against non-private, homomorphic-encryption, and ORAM baselines. The results show that MESS achieves up to $15.08\times$ lower latency and $35.28\times$ lower communication overhead than Compass Zhu et al. (2025). On the SIFT100M dataset, MESS achieves 52.53 ms/query, demonstrating its scalability.

2 Related Work

Vector Databases and ANN Search. Vector databases use approximate nearest-neighbor (ANN) indexes to support large-scale, high-dimensional embedding search. HNSW provides low-latency and high-recall graph traversal, while FAISS and SPANN study GPU acceleration and memory–disk hybrid indexing at billion-vector scale Malkov & Yashunin (2018); Johnson et al. (2021); Chen et al. (2021). Recent systems also consider database-level requirements: HAKES supports high-recall search under concurrent updates, while VBase integrates vector similarity search with relational query processing Hu et al. (2025); Zhang et al. (2023). These systems target the performance and scalability of plaintext vector search, whereas MESS targets privacy. While indexes other than

HNSW can be used in MESS, our current security analysis assumes the use of graph indexes. Extending MESS to support other indexes is left as future work.

Encrypted Semantic Retrieval. Existing works reduce secure candidate-generation costs through clustering, LSH, tree indexes, or stronger deployment assumptions. Tiptoe and Wally use clustering, with Wally additionally combining epochs and differential privacy Henzinger et al. (2023); Asi et al. (2024). Preco uses LSH with two non-colluding servers, while VCC24 combines tree indexing and k-means under information-theoretic security Servan-Schreiber et al. (2022); Vithana et al. (2024). SANNS uses DORAM and garbled circuits for secure Top-K selection Chen et al. (2020), whereas trusted-hardware approaches introduce additional trust assumptions Amjad et al. (2019); Vo et al. (2021). Other works adopt Private Information Retrieval (PIR). In particular, Preco notes that replacing its non-colluding servers with single-server PIR would increase overhead, while Panther combines PIR2A with secret sharing for private access and secure Top-K selection Servan-Schreiber et al. (2022); Li et al. (2025).

Cryptographic solutions use order- or property-preserving encryption for distance comparison, but the ciphertext may leak metric relations Wong et al. (2009); Huang et al. (2019); Liu et al. (2025). Compass uses ORAM to hide HNSW accesses Zhu et al. (2025), while PACMANN combines PIR with a secure neighborhood graph Zhou et al. (2024). These solutions provide stronger obliviousness but suffer significant overhead from cryptographic operations. In contrast, MESS searches server-visible HNSW shards over perturbed binary embeddings, with well-defined stored-data, access-pattern, and search-pattern leakage under the stated non-colluding-server view.

3 Problem Formulation and Threat Model

This section defines the search task, threat model, and privacy goals; Section 4 gives the construction, and Appendix A gives quantitative bounds and complete proofs.

3.1 Search Task and System Setting

Let $\mathcal{D} = \{(\mathsf{id}_i, \boldsymbol{x}_i)\}_{i=1}^{N}$, with $\boldsymbol{x}_i \in \mathcal{X}$, be an embedding database under application distance $d_{\mathcal{X}}$. A trusted client outsources search to M servers $\mathcal{S}_1, \ldots, \mathcal{S}_M$. Server $\mathcal{S}_s$ maintains an HNSW shard $\mathcal{G}^{(s)}$ containing perturbed index codes and randomized authenticated-encryption payloads for its assigned records; no server receives plaintext identifiers or embeddings. The client owns $\mathcal{D}$ during setup and retains the payload keys and routing, deletion, hashing, and query-memoization state.

Given $\boldsymbol{q} \in \mathcal{X}$ and K, the target is the global Top-K under $d_{\mathcal{X}}$, with deterministic tie-breaking. Each server returns encrypted local candidates, which the client aggregates, decrypts, deduplicates, and exactly reranks under $d_{\mathcal{X}}$. Because generation uses perturbed codes and a finite budget, recovery is approximate and evaluated in Section 5.

At setup, for every record the client independently samples $\mathcal{R}_i \xleftarrow{\$} \binom{[M]}{t}$ uniformly without using its embedding. Thus, a fixed server receives it with probability t/M, whereas every query is submitted to all M servers.

3.2 Threat Model

We consider M semi-honest, non-colluding servers. Each follows the protocol but may analyze its own execution using arbitrary auxiliary information. Let View_s denote server $\mathcal{S}_s$'s complete observation: public parameters; local routing membership, perturbed codes, labels, ciphertexts, and graph topology; and observed query reports, HNSW traces, candidates, response sizes, and timings. A server obtains no other server's state or communications.

Formal XDP guarantees apply to a specified projection of one fixed server's View_s; they do not combine server views, compose server-indexed parameters, or cover collusion. Ciphertext payloads are protected separately by randomized authenticated encryption and are not XDP outputs. Malicious-server security, record-membership DP, access-pattern obliviousness, perfect query unlinkability, and a global network observer are outside scope.

3.3 Privacy Definitions and Goals

Definition 1 (Extended Differential Privacy)**.** Let $\mathcal{U}$ be an input domain, let $\xi : \mathcal{U} \times \mathcal{U} \to \mathbb{R}_{\geq 0}$, and let $\delta : \mathcal{U} \times \mathcal{U} \to [0, 1]$. A randomized mechanism $\mathcal{M}$ satisfies (ξ, δ)-extended differential privacy (XDP) if, for every $u, u' \in \mathcal{U}$ and measurable event E,

$$\Pr[\mathcal{M}(u) \in E] \leq e^{\xi(u,u')} \Pr[\mathcal{M}(u') \in E] + \delta(u, u').$$

For these guarantees, each server-specific mapping $H_s : \mathcal{X} \to \{0,1\}^{\kappa_s}$ is fixed before database instantiation; its training data are public, disjoint from that database, or excluded from adjacency. The stored-embedding and access-privacy bounds use the induced Hamming pseudometric

$$d_s(\boldsymbol{x}, \boldsymbol{x}') = d_H(H_s(\boldsymbol{x}), H_s(\boldsymbol{x}'))\,.$$

Stored-data neighbors have identical identifiers and differ only in one fixed identifier's embedding; record membership is not protected.

Definition 2 (Non-Colluding Multi-Server XDP and Privacy Goals)**.** For a privacy target $g \in \{D, \mathrm{AP}, \mathrm{SP}\}$, let $\mathsf{V}_{g,s} = \pi_{g,s}(\mathsf{View}_s)$ be the protected projection of server $\mathcal{S}_s$'s complete observation. Let $\omega \sim_g \omega'$ denote the corresponding adjacent inputs. The protocol satisfies the server-indexed XDP profiles $\boldsymbol{\xi}_g = (\xi_{g,1}, \ldots, \xi_{g,M})$ and $\boldsymbol{\delta}_g = (\delta_{g,1}, \ldots, \delta_{g,M})$ if, for every fixed $s \in [M]$, adjacent ω, ω', and measurable event E_s,

$$\Pr[\mathsf{V}_{g,s}(\omega) \in E_s] \leq e^{\xi_{g,s}(\omega,\omega')} \Pr[\mathsf{V}_{g,s}(\omega') \in E_s] + \delta_{g,s}(\omega, \omega').$$

The three targets instantiate this condition as follows.

(i) **Stored-embedding privacy.** The adjacent inputs are databases $\mathcal{D}, \mathcal{D}'$ under the identifier-fixed substitution above. The projection $\mathsf{V}_{D,s}$ contains routing membership, server-local labels, perturbed stored codes, the resulting local graph, and their server-local post-processing under fixed query inputs and schedules.

(ii) **Access privacy.** The adjacent inputs are query values $\boldsymbol{q}, \boldsymbol{q}'$. The projection $\mathsf{V}_{\mathrm{AP},s}$ contains the perturbed reports and local search traces for every invocation of one logical query. Adjacency replaces its value in every invocation while fixing the client, schedule, reuse grouping, local index, and search configuration.

(iii) **Search privacy.** The adjacent inputs are the single reuse/split workloads W_0, W_1 of Appendix A. For $R \geq 2$, W_0 reuses one permanent randomized-response (PRR) report across all R submissions of $\boldsymbol{q}$; W_1 uses one report for the first $R - 1$ and an independent PRR report for the final $\boldsymbol{q}'$. Submission count and timing, contacted servers, and search configuration are fixed; $\boldsymbol{q}' = \boldsymbol{q}$ changes only the final reuse decision. The projection $\mathsf{V}_{\mathrm{SP},s}$ contains the reports and local traces; arbitrary changes to reuse relations are excluded.

The vector notation collects separate fixed-server inequalities, not a joint-view guarantee.

For storage-side randomized response with flip probability p_D, let $\delta_u^{\mathrm{RR}}(\varepsilon)$ be the exact profile for codes at Hamming distance u (Appendix A), and let $\delta_{D,s}^{\mathrm{nc}}$ be server s's actual stored-view profile. Independent uniform routing gives, for every fixed server,

$$\delta_{D,s}^{\mathrm{nc}}(\varepsilon; \boldsymbol{x}, \boldsymbol{x}') \leq \frac{t}{M}\, \delta_{d_s(\boldsymbol{x},\boldsymbol{x}')}^{\mathrm{RR}}(\varepsilon).$$

This fixed-ε profile bound does not multiply a pure XDP parameter by t/M. Since every query reaches every server, no t/M factor applies to access or search privacy. Appendix A defines the mechanisms and parameters and proves all three bounds.

Beyond these XDP targets, we pursue the following system goals:

- **Payload confidentiality.** Equal-length identifiers and embeddings are computationally protected by randomized authenticated encryption, separately from XDP.
- **Search utility.** Hashing and perturbation should preserve the original-distance ordering sufficiently for exact candidate reranking; utility uses each workload's standard search metric.
- **Efficiency.** Sharded approximate graph search aims to reduce node visits, while bounded candidate returns control communication; both are evaluated empirically in Section 5.

4 DESIGN OF MESS

In this section, we describe the design of MESS, which balances quantified privacy, accuracy, and efficiency. MESS adopts differential privacy, allowing it to bound privacy leakage and avoid the overheads of other cryptographic approaches. However, searching over perturbed codes degrades accuracy because the distances between codes do not match those between the original embeddings. The system can improve accuracy by returning larger candidate sets to the client, but doing so increases communication and computational overhead. Below, we elaborate on the two challenges related to accuracy and discuss our approach.

Challenge 1: Candidate-set expansion under perturbation. High bit-flip probabilities provide strong privacy, but distort nearest-neighbor rankings. Our experiments confirm this effect: with a single-graph baseline, SIFT achieves only 56.65% recall using 1,500 candidates, while LAION achieves only 69.6% recall using 700 candidates. To improve recall, each server can return a larger candidate set for client-side reranking, but this incurs higher communication and computational overhead. Reducing the bit-flip probability could also improve recall, but the codes expose more semantic information, thereby reducing privacy.

Our approach. MESS employs M servers, each holding one index graph (or shard), to reduce competition within each shard. Randomized response may cause non-target points to overtake a true neighbor in the server's candidate order (perturbed Hamming distance followed by a unique shard-local tie label). Let $N(d)$ be the number of points at distance d, and let $P_{\mathrm{overtake}}(d)$ be the probability that such a point precedes the true neighbor after perturbation. The expected overtaking count in a single-graph baseline is $\mu_{\mathrm{single}} = \sum_{d=0}^{\kappa} N(d) P_{\mathrm{overtake}}(d)$. When the codes are replicated uniformly across t out of M servers, $\mu_{\mathrm{shard}} = (t/M)\mu_{\mathrm{single}}$. Each shard uses a small candidate set, and aggregation over all the shards gives the true neighbors multiple opportunities to be recovered.

Challenge 2: Search instability under perturbation. Random bit flipping may severely displace the true neighbors of certain embeddings. A single-graph baseline may perform well for most queries, but require a prohibitively large candidate set for queries with severely displaced neighbors.

Our approach. MESS builds one shard at each server, using independent IsoHash mappings and independently sampled randomized-response coins. A true neighbor that is poorly positioned in one shard may remain close to the query in another. Let $P_{\mathrm{hit}}(b)$ be the recovery probability in one shard with local candidate budget b, under an approximate independence assumption, the recovery probability from at least one of the t shards is $P_{\mathrm{multi}}(b) \approx 1 - (1 - P_{\mathrm{hit}}(b))^t$. The client aggregates the server-local candidate sets, decrypts their payloads, removes duplicates, and exactly reranks the remaining candidates. The actual utility improvement is evaluated experimentally.

4.1 MULTI-SERVER GRAPH CONSTRUCTION

As defined in Section 3, server $\mathcal{S}_s$ maintains the HNSW graph shard $\mathcal{G}^{(s)}$. We now detail how the client constructs these server-local shards from the outsourced database.

The complete multi-server graph-construction protocol is provided in Appendix A (Figure 4). For each embedding i, the client independently samples a routing set $\mathcal{R}_i \xleftarrow{\$} \binom{[M]}{t}$ uniformly and independently of the embedding, using private client-side routing randomness. Thus, for every fixed server s, $\Pr[s \in \mathcal{R}_i] = t/M$. For every $s \in \mathcal{R}_i$, the client computes the server-specific binary embedding $\boldsymbol{v}_{i,s} \leftarrow \Pi_{\mathrm{IsoHash}}.\mathsf{Hash}(\boldsymbol{x}_i, \boldsymbol{\mu}_s, \mathbf{W}_s, \mathbf{Q}_s)$, then applies LSHRR to obtain $\widetilde{\boldsymbol{v}}_{i,s} \leftarrow \Pi_{\mathrm{LSHRR}}(\boldsymbol{v}_{i,s}; p)$. It encrypts the payload $m_i = \mathsf{id}_i \| \boldsymbol{x}_i$ with the client-held payload key associated with server $\mathcal{S}_s$ to obtain $\mathsf{ct}_{i,s} \leftarrow \Pi_{\mathrm{Enc}}.\mathsf{Enc}_{sk_s}(m_i)$. The client also samples a unique exchangeable server-local label $\ell_{i,s}$, used as the deterministic tie-break key and later for deletion. The resulting tuple $(\widetilde{\boldsymbol{v}}_{i,s}, \ell_{i,s}, \mathsf{ct}_{i,s})$ is sent only to $\mathcal{S}_s$, which inserts it into its local graph $\mathcal{G}^{(s)}$. The scope of server-local privacy and the cross-server linkage mitigations are discussed in Appendix A.1.

4.2 PROCESSING USER QUERIES

Repeated queries may reveal the underlying value if each query is independently perturbed. By collecting many versions of the same query, an individual server can average out the randomness

and infer its server-specific binary embedding. To mitigate this averaging attack, MESS adopts a two-phase LSHRR mechanism, inspired by the memoization technique proposed by RAPPOR Erlingsson et al. (2014).

Permanent LSHRR. For a query $\boldsymbol{q}$, the client computes a server-specific binary embedding $\boldsymbol{v}_{\boldsymbol{q}}^{(s)} = H_s(\boldsymbol{q})$ for each $s \in [M]$. Let p_{PRR} denote the permanent flip probability. The client associates repeated invocations of the same logical query with a local handle qid, which is never transmitted, and indexes its cache by the pair (qid, s). If that pair has not appeared before, the client samples

$$\widetilde{\boldsymbol{v}}_{\boldsymbol{q},s}^{(\mathrm{PRR})} \leftarrow \Pi_{\mathrm{LSHRR}}(\boldsymbol{v}_{\boldsymbol{q}}^{(s)}, p_{\mathrm{PRR}}),$$

and writes it to $\mathsf{Cache}[\mathsf{qid}, s]$; otherwise, it reuses the cached value. Thus, repeated submissions do not expose independently perturbed versions of the original server-specific binary embedding.

Instantaneous LSHRR. Before contacting server $\mathcal{S}_s$, the client applies an instantaneous perturbation

$$\widetilde{\boldsymbol{v}}_{\boldsymbol{q},s}^{(\mathrm{IRR})} \leftarrow \Pi_{\mathrm{LSHRR}}\big(\widetilde{\boldsymbol{v}}_{\boldsymbol{q},s}^{(\mathrm{PRR})}, p_{\mathrm{IRR}}\big),$$

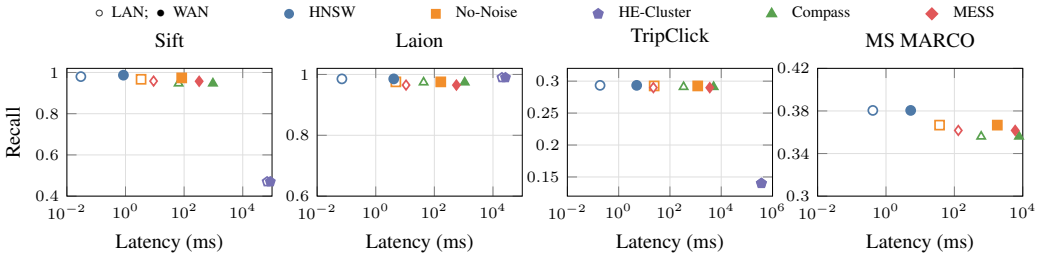
where p_{IRR} is the instantaneous flip probability, and sends only $\widetilde{\boldsymbol{v}}_{\boldsymbol{q},s}^{(\mathrm{IRR})}$ to $\mathcal{S}_s$. Server $\mathcal{S}_s$ uses this report to search $\mathcal{G}^{(s)}$. Fresh IRR coins mean that equality of transmitted reports is not a deterministic test for repeated logical queries. This does not guarantee perfect unlinkability: the remaining leakage from reports and their HNSW search traces is bounded by the search-pattern XDP analysis in Appendix A.

4.3 MESS Protocol

We now combine the graph construction and query protocol above into the complete protocol. MESS consists of four key operations: initialization, secure query generation, search, and decryption and reranking. It also supports update operations, i.e., insertion and deletion.

1. *Initialization.* Given $\mathcal{D} = \{(\mathsf{id}_i, \boldsymbol{x}_i)\}_{i=1}^{N}$, the client trains server-specific hashing parameters on $\mathcal{D}_{\mathrm{train}}$ and generates the client-held per-server payload keys, while each $\mathcal{S}_s$ initializes its local graph $\mathcal{G}^{(s)}$. Each item is assigned to $\mathcal{R}_i \xleftarrow{\$} \binom{[M]}{t}$. For every $s \in \mathcal{R}_i$, the client computes and perturbs the server-specific binary embedding, encrypts $m_i = \mathsf{id}_i \| \boldsymbol{x}_i$ under the corresponding sk_s, and sends the resulting entry only to $\mathcal{S}_s$. That server inserts the entry into $\mathcal{G}^{(s)}$. The resulting states are $\mathsf{st}_{\mathcal{C}} = (\mathsf{st}_{\mathrm{hash}}, \mathcal{K}, \{\mathsf{id}_i \mapsto \{(s, \ell_{i,s})\}_{s \in \mathcal{R}_i}\}_{i=1}^{N})$ and $\{\mathsf{st}_{\mathcal{S}_s} = \mathcal{G}^{(s)}\}_{s=1}^{M}$.
2. *Secure query generation.* Given a query embedding $\boldsymbol{q}$, the client derives one query binary embedding for each server. For each pair (qid, s), it reuses the cached permanent perturbation and then applies fresh instantaneous perturbation. It sends the resulting report for server s only to $\mathcal{S}_s$.
3. *Search.* Each server $\mathcal{S}_s$ independently searches its local HNSW graph $\mathcal{G}^{(s)}$ using Hamming distance between the perturbed query and index nodes, then returns its local encrypted candidate payloads directly to the client.
4. *Aggregation, decryption, and reranking.* The client aggregates the server-local candidate sets, decrypts their payloads, removes duplicates using the identifiers, and reranks the remaining candidates using the original query and data embeddings. It then outputs the final top-K results.
5. *Insertion.* To insert a new tuple $(\mathsf{id}_i, \boldsymbol{x}_i)$, the client reuses the existing hashing parameters and per-server payload keys. It selects $\mathcal{R}_i$, generates independently perturbed binary embeddings and encrypted payloads for the selected servers, and samples a unique exchangeable server-local label $\ell_{i,s}$, which is used as the deterministic tie-break key. Server $\mathcal{S}_s$ inserts its entry into $\mathcal{G}^{(s)}$, while the client stores $\{(s, \ell_{i,s})\}_{s \in \mathcal{R}_i}$ for future deletion. Insertion does not require rebuilding any unaffected index.
6. *Deletion.* The client retrieves the server-local labels associated with id_i and sends each deletion request only to the corresponding server. Each affected server marks its local index node as inactive and excludes it from search, and periodically rebuilds its local index to remove inactive nodes.

Figure 1: Search quality versus latency. Hollow/filled markers denote LAN/WAN; TripClick and MS MARCO use MRR@10. Compass-reported HE-Cluster points may not be quality-matched.



5 Evaluation

In this section, we evaluate the performance of MESS. We structure the results around two research questions:

1. How does MESS compare with the baselines in search quality, index construction cost, latency, and communication cost?
2. How does MESS scale to support a hundred-million-vector index?

Baselines. We compare MESS against four baselines.

- **Plaintext-HNSW.** This baseline performs non-private semantic search. It builds an HNSW index over the original embeddings using `hnswlib` and answers queries directly in plaintext. We use this to evaluate the overhead of MESS's protection mechanisms.
- **Compass.** This baseline combines HNSW search with ORAM Zhu et al. (2025). It hides the logical graph nodes accessed during adaptive search, but requires multiple encrypted block transfers and client–server interactions. We use the Compass configurations corresponding to the evaluated datasets and apply the same network settings to its reported communication cost.
- **HE-Cluster.** This baseline adopts homomorphic encryption for semantic search. It is a variant of Tiptoe Henzinger et al. (2023) and is also used as a baseline in Compass Zhu et al. (2025). It partitions N items into approximately $\sqrt{N}$ padded semantic clusters, computes encrypted similarities within the selected cluster, and returns encrypted scores for client-side ranking. Its main costs are homomorphic computation and ciphertext transmission.
- **No-Noise.** This baseline is the same as MESS but without storage- and query-side LSHRR. We use this for ablation: the gap between Plaintext-HNSW and No-Noise captures the effect of binary multi-graph retrieval, while the gap between No-Noise and MESS captures the impact of randomized perturbation.

Datasets and metrics. We evaluate SIFT, LAION, TripClick, and MS MARCO, covering dense-vector, multimodal, and text-retrieval workloads. For SIFT and LAION, we use Recall@K, which measures the overlap between the returned and ground-truth Top-K results. For TripClick and MS MARCO, we use MRR@10, which emphasizes highly ranked relevant results. Communication is measured by total request and response bytes and the number of messages. Dataset sizes and dimensions are in Appendix Table 2; dataset-specific parameters, query counts, and candidate budgets are reported in Appendix B.1.

Experiment settings. All experiments run on a dual-socket machine with Hygon C86 7375 32-core processors at 2.0 GHz and 512 GB memory. We emulate the M servers as local workers on this machine, one graph shard each. Thus, reported server-side times measure this local implementation; we do not deploy networked servers or claim measured distributed or network-parallel speedup. Linux traffic control emulates a 3 Gbps/1 ms-RTT LAN and a 400 Mbps/80 ms-RTT WAN.

5.1 Performance Against Baselines

Figure 1 compares the search quality of MESS and the baselines. Appendix Table 2 breaks down the latency and communication cost. For graph-based methods, we report configurations selected

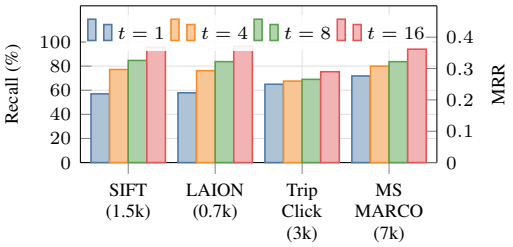


Figure 2: Effect of the number of selected servers t (one shard per server) on search quality with $M = 64$ and a fixed candidate pool for each dataset. SIFT and LAION are evaluated using Recall, while TripClick and MS MARCO are evaluated using MRR.

Impact of t. Figure 2 evaluates the impact of t on search quality, when M is fixed at 64. It can be seen that search quality improves as records are replicated to more server-local shards. We also note that larger t increases the number of shard-local entries and the storage cost. Increasing t from 1 to 16 raises Recall from 57.0% to 95.73% on SIFT and from 57.9% to 96.5% on LAION. TripClick MRR increases from 0.250 to 0.290, while MS MARCO MRR increases from 0.276 to 0.362. This is because a larger t places each vector in more independently perturbed graph views, increasing the probability that it retains a favorable rank in at least one shard.

The gains are more pronounced for the Recall metric than for the MRR metric, because the latter considers the vector's final position after exact reranking.

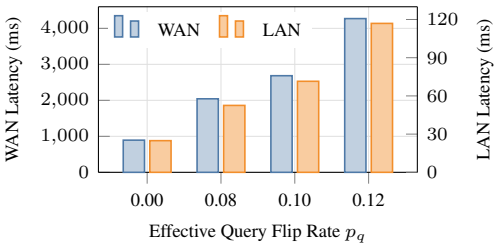
for high accuracy. The results for HE-Cluster are based on the configurations used in Compass Zhu et al. (2025). The gains are more pronounced for the Recall metric than for the MRR metric, because the latter considers the vector's final position after exact reranking.

Search quality. Figure 1 shows that MESS remains close to Plaintext-HNSW: the Recall gap is below 0.05, and the MRR gap is below 0.03. We can explain this gap using No-Noise's result. Specifically, although No-Noise removes randomized perturbation, it still searches finite-length binary embeddings in Hamming space, distributes items over several graph shards, aggregates candidates, transfers encrypted payloads, removes duplicates, and reranks on the client. Its gap to Plaintext-HNSW represents the effect of searching binary multi-graph without privacy noise. Its gap to MESS is attributable to LSHRR and the larger candidate set needed to compensate for randomized bit flips. We can see that adding LSHRR leads to a quality change of at most 0.011. This small quality change comes at the cost of efficiency, because MESS must retrieve more candidates to maintain quality, increasing communication cost.

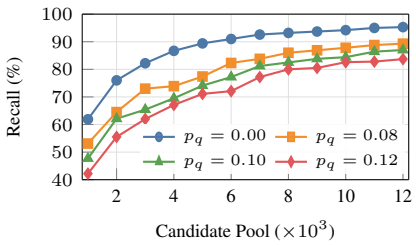
Comparison with cryptographic baselines. Compass achieves accuracy close to that of non-private search, but ORAM-protected adaptive traversal requires repeated accesses to encrypted blocks. Appendix Table 2 shows that MESS is 4.0–$15.1\times$ faster under LAN and 1.3–$2.9\times$ faster under WAN. The largest LAN improvement appears on TripClick, where the cost of interactive traversal is especially high. The WAN advantage is smaller because transfer time dominates in both systems as network speed decreases. HE-Cluster shows a different bottleneck. On the three datasets for which Compass reports results, encrypted similarity evaluation and encrypted score transfer require approximately 20–373 seconds and 209–783 MB per query. Although its quality is competitive on LAION, the results indicate significant computational and communication overhead that is orders of magnitude higher than MESS. In summary, the two baselines that adopt cryptographic approaches suffer from high overhead: HE-Cluster incurs significant overhead in vector computation and produces large ciphertexts, and Compass incurs large communication overhead.

Latency breakdown. For MESS under LAN, the combined local-server search stage remains below 7 ms and client decryption and reranking below 3 ms across the four datasets. Therefore, both Hamming graph traversal and local cryptographic processing are efficient. The remaining cost is encrypted candidate transfer, which becomes significant under WAN: on MS MARCO, this transfer accounts for 5997.02 ms of the 6002.67 ms end-to-end latency. Compass is also network-bound under WAN, but incurs high computation cost due to ORAM operations. These results suggest that optimizing candidate set and payload size are interesting future extensions for MESS.

Offline cost. The offline cost of MESS includes shard-specific hashing, independent perturbation, payload encryption, and construction of multiple HNSW shards across the M emulated server workers. It is therefore notably higher than constructing one plaintext index, as shown in Appendix Table 2. However, this cost is incurred once and can be amortized over subsequent queries.



(a) Total search latency.



(b) Recall under different candidate-pool sizes.

Figure 3: SIFT100M performance under different effective query flip rates p_q: (a) total search latency under LAN and WAN settings; and (b) Recall@10 under different candidate-pool sizes.

Insertion and deletion. MESS supports incremental updates without a full rebuild. For insertion, the client computes and perturbs the new item's shard-specific binary embeddings, encrypts its payload, and sends it to the $t = 16$ selected servers, which insert it into their local shards. For lazy deletion, the client identifies the shard-local labels, and these servers mark the nodes inactive to exclude them from future searches. On MS MARCO, insertion takes 96.16 ms over LAN and 351.29 ms over WAN, while deletion takes 1.33 ms and 80.35 ms, respectively. Their communication costs are 3.09 KB and 0.09 KB.

5.2 Scalability

We evaluate the performance of MESS at scale, using a large dataset (SIFT100M) created by selecting the first 100 million vectors from SIFT1B. We note that neither Compass Zhu et al. (2025) nor Tiptoe Henzinger et al. (2023) report results at this scale. We focus only on the online-phase performance and note that the offline phase takes 26.19 hours.

Figure 3 and Table 1 vary the effective query flip rate $p_q = p_{\mathrm{PRR}}(1 - p_{\mathrm{IRR}}) + p_{\mathrm{IRR}}(1 - p_{\mathrm{PRR}})$, namely the crossover probability between a clean query code and the report received by a server. It is not either component probability alone. Appendix A uses p_{PRR} and p_{IRR} separately for transcript privacy; $p_q = 0$ is only the no-query-perturbation performance ablation. With 12,000 candidates, increasing p_q from 0 to 0.12 reduces Recall@10 from 95.3% to 83.7%. It can be seen that the gap decreases with larger candidate sets. However, as the candidate sets grow from 4,000 to 20,000 candidates, the communication cost increases 5.0×. These results demonstrate the accuracy and efficiency trade-off. The latency breakdown in Table 1 shows that the costs of graph search and client reranking remain small compared to that of encrypted candidate transfer. In particular, at $p_q = 0.08$, MESS processes SIFT100M in 52.53 ms/query under LAN; even at $p_q = 0.12$, LAN latency remains below 120 ms/query. Under WAN, the latency increases from 893.73 ms to 4271.17 ms due to large payloads. These results confirm that MESS is practical for large-scale datasets. They highlight the fundamental trade-off: stronger query perturbation increases rank uncertainty, therefore requires larger candidate sets and incurs higher communication overhead. The results indicate that reducing encrypted payload size or improving shard-level candidate filtering is more beneficial than optimizing the relatively small graph-search and reranking costs.

6 Conclusion

We presented MESS, a multi-server system for semantic search with privacy, accuracy, and efficiency. Each non-colluding server searches one HNSW graph shard over perturbed binary codes, and the client aggregates and exactly reranks the encrypted candidates. Randomized authenticated encryption protects payload contents, while our XDP analysis quantifies stored-embedding and query-transcript leakage separately for each server's local view, including repeated-query leakage. We make no joint-view privacy claim across servers. The multi-graph design compensates for the accuracy loss from perturbation. The evaluation results show that MESS outperforms cryptographic baselines, reducing communication costs by up to 35.28× compared to Compass. The system remains practical at scale, achieving 52.53 ms/query on a dataset of 100M vectors.

AI USE STATEMENT

Generative AI was used to assist with consistency checks of mathematical arguments, and manuscript editing, formatting, and English-language polishing. It was not used to generate experimental data. The authors reviewed and verified all AI-assisted material, and take full responsibility for the final methods, analysis, claims, and text.

REPRODUCIBILITY

The anonymized repository is available at anonymous.4open.science/r/MESS-anonymization.

REFERENCES

Ishtiyaque Ahmad, Laboni Sarker, Divyakant Agrawal, Amr El Abbadi, and Trinabh Gupta. Coeus: A system for oblivious document ranking and retrieval. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*, pp. 672–690, 2021.

Amazon Web Services. Amazon OpenSearch Service. `https://aws.amazon.com/opensearch-service/serverless-vector-database/`, 2026. Accessed: 2026.

Ghous Amjad, Seny Kamara, and Tarik Moataz. Forward and backward private searchable encryption with sgx. In *Proceedings of the 12th European Workshop on Systems Security*, pp. 1–6, 2019.

Hilal Asi, Fabian Boemer, Nicholas Genise, Muhammad Haris Mughees, Tabitha Ogilvie, Rehan Rishi, Kunal Talwar, Karl Tarbe, Akshay Wadia, Ruiyu Zhu, et al. Scalable private search with wally. *arXiv preprint arXiv:2406.06761*, 2024.

Martin Aumüller, Anders Bourgeat, and Jana Schmurr. Differentially private sketches for jaccard similarity estimation. In *Similarity Search and Applications - 13th International Conference, SISAP 2020*, pp. 18–32. Springer, 2020. doi: 10.1007/978-3-030-60936-8_2.

Guoxing Chen, Ten-Hwang Lai, Michael K Reiter, and Yinqian Zhang. Differentially private access patterns for searchable symmetric encryption. In *IEEE INFOCOM 2018-IEEE conference on computer communications*, pp. 810–818. IEEE, 2018.

Hao Chen, Ilaria Chillotti, Yihe Dong, Oxana Poburinnaya, Ilya Razenshteyn, and M Sadegh Riazi. {SANNS}: Scaling up secure approximate {k-Nearest} neighbors search. In *29th USENIX Security Symposium (USENIX Security 20)*, pp. 2111–2128, 2020.

Qi Chen, Bing Zhao, Haidong Wang, Mingqin Li, Chuanjie Liu, Zengzhong Li, Mao Yang, and Jingdong Wang. Spann: Highly-efficient billion-scale approximate nearest neighbor search. In *Advances in Neural Information Processing Systems*, volume 34, pp. 5199–5212, 2021.

Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology – ASIACRYPT 2017*, volume 10624 of *Lecture Notes in Computer Science*, pp. 409–437. Springer, 2017. doi: 10.1007/978-3-319-70694-8_15. URL `https://eprint.iacr.org/2016/421`.

Tianji Cong et al. Revisiting oblivious top-k selection with applications to secure k-nn classification. In *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*, 2024. URL `https://sacworkshop.org/SAC24/preproceedings/CongGeelenKangPark.pdf`.

Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1054–1067. ACM, 2014. doi: 10.1145/2660267.2660348.

Natasha Fernandes, Yusuke Kawamoto, and Takao Murakami. Locality sensitive hashing with extended differential privacy. In *Computer Security – ESORICS 2021, Part II*, volume 12973 of *Lecture Notes in Computer Science*, pp. 563–583. Springer, 2021. doi: 10.1007/978-3-030-88428-4_28.

Donniell E. Fishkind, Sancar Adali, Heather G. Patsolic, Lingyao Meng, Digvijay Singh, Vince Lyzinski, and Carey E. Priebe. Seeded graph matching. *Pattern Recognition*, 87:203–215, 2019. doi: 10.1016/j.patcog.2018.09.014.

Google Cloud. Vector Search. https://docs.cloud.google.com/gemini-enterprise-agent-platform/build/vector-search/overview, 2026. Accessed: 2026.

Alexandra Henzinger, Emma Dauterman, Henry Corrigan-Gibbs, and Nickolai Zeldovich. Private web search with tiptoe. In *Proceedings of the 29th symposium on operating systems principles*, pp. 396–416, 2023.

Guoyu Hu, Shaofeng Cai, Tien Tuan Anh Dinh, Zhongle Xie, Cong Yue, Gang Chen, and Beng Chin Ooi. Hakes: Scalable vector database for embedding search service. *Proceedings of the VLDB Endowment*, 18(9):3049–3062, 2025. doi: 10.14778/3746405.3746427.

Zhengbai Huang, Meng Zhang, and Yi Zhang. Toward efficient encrypted image retrieval in cloud environment. *IEEE Access*, 7:174541–174550, 2019. doi: 10.1109/ACCESS.2019.2957497.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547, 2021. doi: 10.1109/TBDATA.2019.2921572.

Darya Kaviani, Alp Eren Ozdarendeli, Jinhao Zhu, Yu Ding, and Raluca Ada Popa. Opal: Private memory for personal ai. *arXiv preprint arXiv:2604.02522*, 2026.

Ehsan Kazemi, Seyed Hamed Hassani, and Matthias Grossglauser. Growing a graph matching from a handful of seeds. *Proceedings of the VLDB Endowment*, 8(10):1010–1021, 2015. doi: 10.14778/2794367.2794371.

Jiale Lao, Andreas Zimmerer, Olga Ovcharenko, Tianji Cong, Matthew Russo, Gerardo Vitagliano, Michael Cochez, Fatma Özcan, Gautam Gupta, Thibaud Hottelier, H. V. Jagadish, Kris Kissel, Sebastian Schelter, Andreas Kipf, and Immanuel Trummer. Sembench: A benchmark for semantic query processing engines. *Proc. VLDB Endow.*, 19(8):1754–1767, July 2026. ISSN 2150-8097. doi: 10.14778/3811243.3811249. URL https://doi.org/10.14778/3811243.3811249.

Jingyu Li, Zhicong Huang, Min Zhang, Cheng Hong, Jian Liu, Tao Wei, and Wenguang Chen. Panther: Private approximate nearest neighbor search in the single server setting. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pp. 365–379, 2025.

Hang Liu, Anna Scaglione, and Hoi-To Wai. Blind graph matching using graph signals. *IEEE Transactions on Signal Processing*, 72:1766–1781, 2024. doi: 10.1109/TSP.2024.3382840.

Yingfan Liu, Yandi Zhang, Jiadong Xie, Hui Li, Jeffrey Xu Yu, and Jiangtao Cui. Privacy-preserving approximate nearest neighbor search on high-dimensional data. In *Proceedings of the IEEE 41st International Conference on Data Engineering (ICDE 2025)*, pp. 3017–3029, 2025. doi: 10.1109/ICDE65448.2025.00226.

David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

Yu A Malkov and Dmitry A Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836, 2018.

Matteo Mazzone et al. Efficient ranking, order statistics, and sorting under ckks. In *34th USENIX Security Symposium (USENIX Security 25)*, 2025. URL https://www.usenix.org/conference/usenixsecurity25/presentation/mazzone.

Arvind Narayanan and Vitaly Shmatikov. De-anonymizing social networks. In *2009 30th IEEE Symposium on Security and Privacy*, pp. 173–187. IEEE, 2009. doi: 10.1109/SP.2009.22.

M. Sadegh Riazi, Beidi Chen, Anshumali Shrivastava, Dan Wallach, and Farinaz Koushanfar. Sublinear privacy-preserving near-neighbor search. Cryptology ePrint Archive, Paper 2019/1222, 2019. URL https://eprint.iacr.org/2019/1222.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pp. 285–295, 2001.

Sacha Servan-Schreiber, Simon Langowski, and Srinivas Devadas. Private approximate nearest neighbor search with sublinear communication. In *2022 IEEE Symposium on Security and Privacy (SP)*, pp. 911–929. IEEE, 2022.

Sivic and Zisserman. Video google: A text retrieval approach to object matching in videos. In *Proceedings ninth IEEE international conference on computer vision*, pp. 1470–1477. IEEE, 2003.

Paul Swoboda, Dagmar Kainmüller, Ashkan Mokarian, Christian Theobalt, and Florian Bernard. A convex relaxation for multi-graph matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11156–11165, 2019. doi: 10.1109/CVPR.2019.01141.

Warren S Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17(4):401–419, 1952.

Sajani Vithana, Martina Cardone, and Flavio P Calmon. Private approximate nearest neighbor search for vector database querying. In *2024 IEEE International Symposium on Information Theory (ISIT)*, pp. 3666–3671. IEEE, 2024.

Viet Vo, Shangqi Lai, Xingliang Yuan, Surya Nepal, and Joseph K Liu. Towards efficient and strong backward private searchable encryption with secure enclaves. In *International Conference on Applied Cryptography and Network Security*, pp. 50–75. Springer, 2021.

Yichuan Wang, Zhifei Li, Shu Liu, Yongji Wu, Ziming Mao, Yilong Zhao, Xiao Yan, Zhiying Xu, Yang Zhou, Ion Stoica, et al. Leann: A low-storage vector index. *arXiv preprint arXiv:2506.08276*, 2025.

Weaviate. AI Database that developers love. https://weaviate.io/, 2026. Accessed: 2026.

Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of machine learning research*, 10(2), 2009.

Wai Kit Wong, David Wai-lok Cheung, Ben Kao, and Nikos Mamoulis. Secure knn computation on encrypted databases. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pp. 139–152, 2009.

Qianxi Zhang, Shuotao Xu, Qi Chen, Guoxin Sui, Jiadong Xie, Zhizhen Cai, Yaoqi Chen, Yinxuan He, Yuqing Yang, Fan Yang, Mao Yang, and Lidong Zhou. VBASE: Unifying online vector similarity search and relational queries via relaxed monotonicity. In *17th USENIX Symposium on Operating Systems Design and Implementation*, OSDI '23, pp. 377–395, 2023.

Mingxun Zhou, Elaine Shi, and Giulia Fanti. Pacmann: Efficient private approximate nearest neighbor search. In *The Thirteenth International Conference on Learning Representations*, 2024.

Jinhao Zhu, Liana Patel, Matei Zaharia, and Raluca Ada Popa. Compass: Encrypted semantic search with high accuracy. In *19th USENIX Symposium on Operating Systems Design and Implementation (OSDI 25)*, pp. 915–938, 2025.

A Theoretical Analysis

A.1 Security and Privacy under Non-Colluding Servers

There are M semi-honest, non-colluding servers $\mathcal{S}_1, \ldots, \mathcal{S}_M$, and server $\mathcal{S}_s$ maintains exactly one graph shard $G_s \equiv \mathcal{G}^{(s)}$. Each server observes only its own protocol state, perturbed index, query reports, and search traces. We state privacy guarantees separately for every server; we neither combine local views into a joint observation nor claim a privacy bound for a party that pools multiple servers' views. We protect stored embedding values, submitted query values, and the reuse relation among repeated queries, corresponding to stored-data, access-pattern, and search-pattern privacy.

For every record i, setup samples a route set $\mathcal{R}_i \xleftarrow{\$} \binom{[M]}{t}$, independently of its embedding and of all other records. The client retains $\mathsf{ID}_i \mapsto \{(s, \ell_{i,s})\}_{s \in \mathcal{R}_i}$ for deletion. For a fixed server s, write $B_{i,s} = \mathbf{1}\{s \in \mathcal{R}_i\}$. The three server-local views used below are defined directly by

$$\begin{aligned}
\mathsf{V}_{D,s} &= \left((B_{i,s})_i, \{(\widetilde{\boldsymbol{v}}_{i,s}, \ell_{i,s}) : B_{i,s} = 1\}, G_s, \mathrm{Post}_{D,s}(G_s)\right), \\
\mathsf{V}_{\mathrm{AP},s} &= \left(\widetilde{\boldsymbol{v}}_{q,s}^{(j)}, \mathrm{Trace}_s^{(j)}\right)_{j=1}^{R}, \\
\mathsf{V}_{\mathrm{SP},s} &= \left(\widetilde{\boldsymbol{v}}_{q_j,s}^{(j)}, \mathrm{Trace}_s^{(j)}\right)_{j=1}^{R}.
\end{aligned}$$

Here G_s is the local graph built from the perturbed storage codes, $\mathrm{Post}_{D,s}$ denotes any graph or result post-processing, and $\mathrm{Trace}_s^{(j)}$ contains the corresponding local search trajectory, visited nodes, candidates, response size, and timing fields. The access view uses the reports for a fixed logical query (with the repetition count shown by the index j); the search view records the whole repeated workload. Ciphertext payload contents are not XDP outputs in these views and are protected separately by randomized authenticated encryption. All probabilities in the stored-data result below are over the client's independent setup-time routing randomness and randomized-response coins.

The bold privacy profiles $\boldsymbol{\xi}_D^{\mathrm{nc}}$, $\boldsymbol{\xi}_{\mathrm{AP}}^{\mathrm{nc}}$, and $\boldsymbol{\xi}_{\mathrm{SP}}^{\mathrm{nc}}$ merely index the M local guarantees required by Definition 2; they are not a joint release of server views, and their components are never summed. These guarantees do not depend on the success of a concrete inference method.

Cross-server linkage mitigations. In addition to the per-server guarantees, MESS uses the following mechanisms to limit direct linkage across server-local copies:

- *Server-specific projection functions.* Each server uses different IsoHash parameters. The same original embedding $\boldsymbol{x}_i$ is therefore mapped to different server-specific binary embeddings, making bit coordinates across different servers not directly comparable.
- *Independent perturbation.* Randomized response uses fresh coins for every server. Even when the same embedding appears at multiple servers, its binary embeddings are independently perturbed, so equality of local codes is not a reliable cross-server linkage test.
- *Randomized server assignment.* Each embedding independently samples a uniformly random t-subset of the M servers. For every fixed server, the inclusion probability is $r = t/M$. This random routing amplifies the server-local stored-data privacy profile; it is not applied to the query-privacy profiles.
- *Server-local labels and encryption keys.* Labels are sampled as unique exchangeable random priorities independently for different servers; the same labels are used as local tie-break keys. Payloads are encrypted with server-specific keys and fresh randomness. Without common identifiers or equivalent metadata, label or ciphertext equality does not directly link copies across servers.

These mechanisms do not eliminate cross-server linkage when views are pooled and do not establish a joint-view privacy bound.

A.1.1 Scope and Fixed ISO-LSH Mappings

For stored data, we use substitution adjacency: two databases contain the same identifiers and differ only in the embedding associated with one identifier. Thus, the guarantee protects the embedding value rather than the presence of its identifier. Each server $\mathcal{S}_s$ and its shard G_s use a pretrained mapping $H_s : \mathcal{X} \to \{0,1\}^{\kappa_s}$ fixed before the protected database is instantiated. The mappings

may be correlated; the proofs require fresh randomized-response coins for each record, shard, and coordinate. If ISO-LSH is trained on the protected database, its training requires a separate privacy analysis. We therefore assume that the training data are public, disjoint from the protected database, or excluded from the neighboring relation.

A mechanism $\mathcal{M}$ satisfies (ξ, δ)-XDP if, for every pair x, x' and measurable event E,

$$\Pr[\mathcal{M}(x) \in E] \le e^{\xi(x,x')} \Pr[\mathcal{M}(x') \in E] + \delta(x, x').$$

Our guarantee uses the Hamming pseudometric induced by the fixed ISO-LSH mappings. It differs from angular-XDP averaged over freshly sampled random-LSH functions Fernandes et al. (2021); the collision distribution from that sampling is not applied to the pretrained mappings.

A.1.2 Stored Multi-Server Index

Let $p_D \in (0, 1/2)$ be the storage-side bit-flip probability and define

$$\eta_D = \ln \frac{1 - p_D}{p_D}, \qquad d_s(\boldsymbol{x}, \boldsymbol{x}') = d_H\big(H_s(\boldsymbol{x}), H_s(\boldsymbol{x}')\big), \qquad u = d_s(\boldsymbol{x}, \boldsymbol{x}').$$

For the u differing coordinates, the exact randomized-response hockey-stick profile is

$$\delta_u^{\mathrm{RR}}(\varepsilon) = \sum_{z=0}^{u} \binom{u}{z} (1 - p_D)^z p_D^{u-z} \left[1 - e^{\varepsilon - (2z-u)\eta_D}\right]_+, \qquad [a]_+ = \max\{a, 0\}.$$

Let $\delta_{D,s}^{\mathrm{nc}}(\varepsilon; \boldsymbol{x}, \boldsymbol{x}')$ denote the hockey-stick profile of the actual server-local stored-data view. Since a uniform random t-subset contains a fixed server with probability t/M,

$$\delta_{D,s}^{\mathrm{nc}}(\varepsilon; \boldsymbol{x}, \boldsymbol{x}') \le \frac{t}{M} \delta_{d_s(\boldsymbol{x},\boldsymbol{x}')}^{\mathrm{RR}}(\varepsilon).$$

For a target δ_0, define

$$\xi_{D,s}^{\mathrm{nc}}(\boldsymbol{x}, \boldsymbol{x}'; \delta_0) = \inf \left\{ \varepsilon \ge 0 : \frac{t}{M} \delta_{d_s(\boldsymbol{x},\boldsymbol{x}')}^{\mathrm{RR}}(\varepsilon) \le \delta_0 \right\}.$$

The vector $\boldsymbol{\xi}_D^{\mathrm{nc}}(\boldsymbol{x}, \boldsymbol{x}'; \delta_0) = (\xi_{D,s}^{\mathrm{nc}}(\boldsymbol{x}, \boldsymbol{x}'; \delta_0))_{s \in [M]}$ only indexes these M separate fixed-server guarantees.

Theorem A.1 (Non-Colluding Multi-Server Stored-Index Privacy)**.** *The M-server protocol satisfies the following non-colluding multi-server XDP profile: for every fixed server $s \in [M]$, every $\varepsilon \ge 0$, every measurable event E_s in its local stored-index view, and every pair of adjacent databases $\mathcal{D}, \mathcal{D}'$ that differ only in the embeddings $\boldsymbol{x}, \boldsymbol{x}'$ associated with one fixed identifier,*

$$\Pr[\mathsf{V}_{D,s}(\mathcal{D}) \in E_s] \le e^{\varepsilon} \Pr[\mathsf{V}_{D,s}(\mathcal{D}') \in E_s] + \delta_{D,s}^{\mathrm{nc}}(\varepsilon; \boldsymbol{x}, \boldsymbol{x}'),$$

with the symmetric inequality obtained by exchanging $\mathcal{D}$ and $\mathcal{D}'$ (and hence $\boldsymbol{x}$ and $\boldsymbol{x}'$). Equivalently, for any target δ_0, the protocol satisfies $(\boldsymbol{\xi}_D^{\mathrm{nc}}(\boldsymbol{x}, \boldsymbol{x}'; \delta_0), \delta_0 \mathbf{1})$-non-colluding multi-server XDP in the sense of Definition 2.

Proof. For stored-data adjacency, query inputs and submission schedules are fixed across the neighboring databases. Thus the server-local search traces are post-processing of the perturbed index and these fixed inputs. Fix s and the substituted record i, and let $B_s = \mathbf{1}\{s \in \mathcal{R}_i\}$. Data-independent uniform routing gives $\Pr[B_s = 1] = t/M$. If $B_s = 0$, the two neighboring databases induce identical local views at $\mathcal{S}_s$. If $B_s = 1$, the conditional randomized-response profile is $\delta_{d_s(\boldsymbol{x},\boldsymbol{x}')}^{\mathrm{RR}}(\varepsilon)$. Let $Y_s = \perp$ when $B_s = 0$ and let $Y_s = \mathrm{RR}_{p_D}(H_s(\boldsymbol{x}))$ when $B_s = 1$. For the membership-augmented primitive release $\mathsf{U}_{D,s} = (B_s, Y_s)$, the two branches are disjoint, so its hockey-stick divergence is exactly

$$\left(1 - \frac{t}{M}\right) \cdot 0 + \frac{t}{M} \delta_{d_s(\boldsymbol{x},\boldsymbol{x}')}^{\mathrm{RR}}(\varepsilon).$$

Together with all unchanged records, the actual server-local view is obtained from this primitive release by post-processing, including HNSW construction, traversal, and candidate output, so its profile can only decrease. Exchanging $\boldsymbol{x}$ and $\boldsymbol{x}'$ gives the reverse inequality. □

Thus, for every fixed non-colluding server, random t-out-of-M routing constitutes record subsampling with inclusion probability t/M. Consequently, the server-local stored-data privacy profile is amplified by a factor of t/M. Conditioned on the server receiving the record, the profile remains the original randomized-response profile; in particular, the result does not assert that a pure-XDP parameter is multiplied by t/M. The profile, parameter, and proof above form the complete stored-data accountant for the non-colluding server-local view.

Payload Confidentiality. Identifiers and original embeddings are stored in equal-length payloads using randomized authenticated encryption. Under IND-CPA security, fresh nonces and independent server keys prevent ciphertext equality from directly associating entries of the same record across servers, assuming no common identifier or equivalent metadata is exposed.

A.1.3 Access Privacy against Non-Colluding Servers

For query-value privacy, the released index, client identity, submission schedule, PRR-reuse pattern, and search configuration are fixed, while one logical query value is substituted. Every query in MESS is sent to the fixed set $\mathcal{S}_Q = [M]$. Let

$$a = p_{\mathrm{PRR}} \in (0, 1/2), \qquad b = p_{\mathrm{IRR}} \in (0, 1/2),$$

and, for a clean bit $v \in \{0, 1\}$, define the permanent- and instantaneous- randomized-response kernels

$$r_v(z) = (1-a)\mathbf{1}\{z = v\} + a\mathbf{1}\{z \neq v\}, \qquad i_z(y) = (1-b)\mathbf{1}\{y = z\} + b\mathbf{1}\{y \neq z\}.$$

For R submissions of the same logical query and a report sequence $\boldsymbol{y} = (y_1, \ldots, y_R) \in \{0,1\}^R$, the one-bit distribution is

$$P_v^{(R)}(\boldsymbol{y}) = \sum_{z \in \{0,1\}} r_v(z) \prod_{j=1}^{R} i_z(y_j).$$

Equivalently, with $n_v(\boldsymbol{y}) = \sum_{j=1}^{R} \mathbf{1}\{y_j = v\}$,

$$P_v^{(R)}(\boldsymbol{y}) = (1-a)(1-b)^{n_v(\boldsymbol{y})} b^{R - n_v(\boldsymbol{y})} + a\, b^{n_v(\boldsymbol{y})} (1-b)^{R - n_v(\boldsymbol{y})}.$$

Define the bidirectional one-bit likelihood parameter

$$\eta_R = \max_{v \in \{0,1\},\, \boldsymbol{y} \in \{0,1\}^R} \left| \ln \frac{P_v^{(R)}(\boldsymbol{y})}{P_{1-v}^{(R)}(\boldsymbol{y})} \right|.$$

For a fixed server, let

$$u_s(\boldsymbol{q}, \boldsymbol{q}') = d_H(H_s(\boldsymbol{q}), H_s(\boldsymbol{q}')), \qquad \xi_{\mathrm{AP},s}(\boldsymbol{q}, \boldsymbol{q}') = \eta_R u_s(\boldsymbol{q}, \boldsymbol{q}'),$$

and define the non-colluding access profile $\boldsymbol{\xi}_{\mathrm{AP}}^{\mathrm{nc}} = (\xi_{\mathrm{AP},s})_{s \in [M]}$.

Theorem A.2 (Access Privacy against Non-Colluding Servers)**.** For every $s \in [M]$ and every measurable local event E_s,

$$\Pr[\mathsf{V}_{\mathrm{AP},s}(\boldsymbol{q}) \in E_s] \leq e^{\xi_{\mathrm{AP},s}(\boldsymbol{q}, \boldsymbol{q}')} \Pr[\mathsf{V}_{\mathrm{AP},s}(\boldsymbol{q}') \in E_s],$$

and the reverse inequality holds after exchanging $\boldsymbol{q}$ and $\boldsymbol{q}'$. Thus MESS satisfies $(\boldsymbol{\xi}_{\mathrm{AP}}^{\mathrm{nc}}, \mathbf{0})$ in the sense of Definition 2. A uniform bound for any one non-colluding observer is $\max_{s \in [M]} \xi_{\mathrm{AP},s}$. This maximum is not a composition bound for a pooled view.

Proof. Fix s. Randomized-response coins are independent across coordinates within that server. A coordinate on which $H_s(\boldsymbol{q})$ and $H_s(\boldsymbol{q}')$ agree has likelihood ratio one. For each of the $u_s(\boldsymbol{q}, \boldsymbol{q}')$ differing coordinates, the definition of η_R bounds both likelihood-ratio directions by e^{η_R}. Multiplying the coordinatewise factors bounds the full report-sequence likelihood ratio in both directions by $e^{\eta_R u_s(\boldsymbol{q}, \boldsymbol{q}')}$. The local HNSW trajectory, visited nodes, candidate evolution, returned handles, response size, and timing fields are post-processing of this sequence, the fixed local index, public parameters, and query-independent randomness, so they do not increase the bound. Since s is arbitrary, this proves the family of M server-local inequalities. Every query is sent to every server, so no t/M factor is applied. □

The guarantee requires each trace field to depend only on the perturbed report, fixed local index, public parameters, and query-independent randomness; clean-query routing, clean-distance calculations, stable query identifiers, and external side channels are outside the theorem.

A.1.4 SEARCH PRIVACY AGAINST NON-COLLUDING SERVERS

We analyze only the single final-submission reuse/split adjacency from Definition 2. For $R \geq 2$, let $W_0 = (\boldsymbol{q}, \ldots, \boldsymbol{q})$ use one permanent report for all submissions and let $W_1 = (\boldsymbol{q}, \ldots, \boldsymbol{q}, \boldsymbol{q}')$ reuse it for the first $R-1$ submissions, with an independent permanent report for the final submission. The special case $\boldsymbol{q}' = \boldsymbol{q}$ changes only whether that final submission reuses the report. This adjacency does not cover arbitrary changes to the reuse relation. For one coordinate and $\boldsymbol{y} = (y_1, \ldots, y_R)$, define

$$P^{v}_{\text{reuse}}(\boldsymbol{y}) = \sum_{z \in \{0,1\}} r_v(z) \prod_{j=1}^{R} i_z(y_j),$$

$$P^{v,v'}_{\text{split}}(\boldsymbol{y}) = \left(\sum_{z \in \{0,1\}} r_v(z) \prod_{j=1}^{R-1} i_z(y_j) \right) \left(\sum_{z' \in \{0,1\}} r_{v'}(z') i_{z'}(y_R) \right).$$

The bidirectional one-bit reuse/split parameters are

$$\gamma_{=} = \max_{v, \boldsymbol{y}} \left| \ln \frac{P^{v}_{\text{reuse}}(\boldsymbol{y})}{P^{v,v}_{\text{split}}(\boldsymbol{y})} \right|, \qquad \gamma_{\neq} = \max_{v \neq v', \boldsymbol{y}} \left| \ln \frac{P^{v}_{\text{reuse}}(\boldsymbol{y})}{P^{v,v'}_{\text{split}}(\boldsymbol{y})} \right|.$$

Equivalently, for $c \in \{=, \neq\}$, γ_c is the maximum absolute log-likelihood ratio over the corresponding clean-bit case. For every server, set $u_s = d_H(H_s(\boldsymbol{q}), H_s(\boldsymbol{q}'))$ and

$$\xi_{\text{SP},s}(W_0, W_1) = (\kappa_s - u_s)\gamma_{=} + u_s \gamma_{\neq}.$$

Let $\boldsymbol{\xi}^{\text{nc}}_{\text{SP}} = (\xi_{\text{SP},s})_{s \in [M]}$.

Theorem A.3 (Search Privacy against Non-Colluding Servers)**.** For every $s \in [M]$ and every measurable local event E_s,

$$\Pr[\mathsf{V}_{\text{SP},s}(W_0) \in E_s] \leq e^{\xi_{\text{SP},s}(W_0, W_1)} \Pr[\mathsf{V}_{\text{SP},s}(W_1) \in E_s],$$

and the reverse inequality holds after exchanging W_0 and W_1. Hence MESS satisfies $(\boldsymbol{\xi}^{\text{nc}}_{\text{SP}}, \mathbf{0})$ in the sense of Definition 2 for this single reuse/split adjacency only. A uniform bound for any one non-colluding observer is $\max_{s \in [M]} \xi_{\text{SP},s}$, not a sum across servers. No probability law over $(\mathsf{V}_{\text{SP},1}, \ldots, \mathsf{V}_{\text{SP},M})$ is invoked.

Proof. Fix s. At a coordinate where $H_s(\boldsymbol{q})$ and $H_s(\boldsymbol{q}')$ agree, the reuse and split experiments fall in the equal-bit case and their likelihood ratio in either direction is at most $e^{\gamma_{=}}$. At each of the u_s differing coordinates, the corresponding bidirectional factor is at most $e^{\gamma_{\neq}}$. Independence of PRR and IRR coins across coordinates therefore gives

$$\max \left\{ \frac{\Pr[\mathsf{V}_{\text{SP},s}(W_0) = v_s]}{\Pr[\mathsf{V}_{\text{SP},s}(W_1) = v_s]}, \frac{\Pr[\mathsf{V}_{\text{SP},s}(W_1) = v_s]}{\Pr[\mathsf{V}_{\text{SP},s}(W_0) = v_s]} \right\} \leq e^{(\kappa_s - u_s)\gamma_{=} + u_s \gamma_{\neq}}$$

for every report-sequence output v_s. The associated local HNSW search traces and candidate outputs are post-processing of the reports and fixed local index, which proves both event inequalities. Since every workload submission contacts all M servers, no t/M factor is applied. □

The generally nonzero $(\kappa_s - u_s)\gamma_{=}$ term is required because this adjacency protects the reuse relation, not only query value. In the pure reuse-pattern case, $u_s = 0$ and the bound reduces to $\kappa_s \gamma_{=}$.

Algorithmic Primitives: Isotropic hashing Π_{IsoHash}, locality-sensitive hashing randomized response Π_{LSHRR}, symmetric encryption Π_{Enc}, HNSW indexing Π_{HNSW}.
Input: Plaintext embedding database $\mathcal{D} = \{(\mathsf{id}_i, \boldsymbol{x}_i)\}_{i=1}^{N}$ with $\boldsymbol{x}_i \in \mathcal{X}$, a public or independently collected training set $\mathcal{D}_{\mathrm{train}}$ disjoint from the protected records, server-specific binary embedding length κ, number of servers M, routing multiplicity t, randomized-response flip probability p, and HNSW parameters.
Output: Public configuration pp, client-side state $\mathsf{st}_{\mathcal{C}}$, and one local index state $\mathsf{st}_{\mathcal{S}_s}$ for every server $s \in [M]$.

[Phase 1: Parameter Setup]

1: The client trains server-specific IsoHash parameters only on $\mathcal{D}_{\mathrm{train}}$, $\mathsf{st}_{\mathrm{hash}} = \{(\boldsymbol{\mu}_s, \mathbf{W}_s, \mathbf{Q}_s)\}_{s=1}^{M}$. The public configuration is set as $\mathsf{pp} = (\kappa, M, t, p, \mathsf{params}_{\mathrm{HNSW}})$.

[Phase 2: Multi-Server Graph Initialization]

2: The client prepares and retains M independent symmetric payload-encryption keys, one for each target server, and forms $\mathcal{K} = \{sk_1, \ldots, sk_M\}$.

3: Each server $\mathcal{S}_s$ initializes one local HNSW graph $\mathcal{G}^{(s)} = \{G_0^{(s)}, \ldots, G_{L_s}^{(s)}\}$. We use $\mathbb{G} = \{\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(M)}\}$ only to denote the collection of these physically separated local indexes.

[Phase 3: Perturbation and Insertion]

4: For each data point $(\mathsf{id}_i, \boldsymbol{x}_i)$, the client independently samples a uniformly random t-server routing set $\mathcal{R}_i \xleftarrow{\$} \binom{[M]}{t}$ using private independent routing randomness.

5: For each selected server $s \in \mathcal{R}_i$, the client computes the server-specific binary embedding $\boldsymbol{v}_{i,s} \leftarrow \Pi_{\mathrm{IsoHash}}.\mathrm{Hash}(\boldsymbol{x}_i, \boldsymbol{\mu}_s, \mathbf{W}_s, \mathbf{Q}_s)$, and perturbs it as $\widetilde{\boldsymbol{v}}_{i,s} \leftarrow \Pi_{\mathrm{LSHRR}}(\boldsymbol{v}_{i,s}; p)$.

6: The client samples a unique exchangeable server-local label $\ell_{i,s}$, used as the deterministic tie-break key, and forms the encrypted payload $m_i \leftarrow \mathsf{id}_i \| \boldsymbol{x}_i$, where id_i is used for duplicate removal and result recovery, and $\boldsymbol{x}_i$ is used for client-side exact reranking. The client encrypts m_i under sk_s with fresh encryption randomness, yielding $\mathsf{ct}_{i,s} \leftarrow \Pi_{\mathrm{Enc}}.\mathrm{Enc}_{sk_s}(m_i)$.

7: The client sends $(\widetilde{\boldsymbol{v}}_{i,s}, \ell_{i,s}, \mathsf{ct}_{i,s})$ only to $\mathcal{S}_s$, which inserts it into its local HNSW graph by running $\mathcal{G}^{(s)} \leftarrow \Pi_{\mathrm{HNSW}}.\mathrm{Insert}(\mathcal{G}^{(s)}, \widetilde{\boldsymbol{v}}_{i,s}, \ell_{i,s}, \mathsf{ct}_{i,s})$.

[Phase 4: State Finalization]

8: The protocol outputs $\mathsf{st}_{\mathcal{C}} = (\mathsf{st}_{\mathrm{hash}}, \mathcal{K}, \{\mathsf{id}_i \mapsto \{(s, \ell_{i,s})\}_{s \in \mathcal{R}_i}\}_{i=1}^{N})$ and $\{\mathsf{st}_{\mathcal{S}_s} = \mathcal{G}^{(s)}\}_{s=1}^{M}$.

Figure 4: Multi-server index construction protocol ($\Pi_{\mathsf{GraphBuild}}$).

Public Parameters: dataset size N, vector dimension d, number of graph shards M, routing multiplicity t, shard hash lengths $\{\kappa_m\}_{m=1}^{M}$, storage-side flip probability p_D, query-side PRR flip probability p_{PRR}, query-side IRR flip probability p_{IRR}, per-shard candidate budget P, final result size K, and HNSW parameters.
Entities: Client $\mathcal{C}$ and M non-colluding, semi-honest cloud servers $\{\mathcal{S}_m\}_{m=1}^{M}$, each holding one graph shard.
Input: At setup, $\mathcal{C}$ holds a plaintext database $\mathcal{D} = \{(\boldsymbol{x}_i, \mathrm{ID}_i)\}_{i=1}^{N}$, a fixed mapping-training set $\mathcal{D}_{\mathrm{train}}$ that is public or disjoint from $\mathcal{D}$, shard seeds $\{\sigma_m\}_{m=1}^{M}$, private independent routing randomness, and shard encryption keys $\{sk_m\}_{m=1}^{M}$. At query time, $\mathcal{C}$ holds a query embedding $\boldsymbol{q}$ and a local memoization cache $\mathcal{C}_{\mathrm{memo}}$.
Server States: After setup, server $\mathcal{S}_m$ stores only its local graph $\mathcal{G}^{(m)}$ and attached encrypted payloads. The collection of server-local graphs is $\mathbb{G} = \{\mathcal{G}^{(m)}\}_{m=1}^{M}$.
Output: $\mathcal{C}$ outputs the identifiers and exact application-distance values of the Top-K reranked candidates.

[Initialization]

1: For each shard m, $\mathcal{C}$ invokes $\mathsf{IsoHash}$ on $\mathcal{D}_{\mathrm{train}}$ using shard seed σ_m to derive the shard-specific hash state $\mathsf{st}_{\mathrm{hash}}^{(m)}$. It obtains $\mathsf{st}_{\mathrm{hash}} = \{\mathsf{st}_{\mathrm{hash}}^{(m)}\}_{m=1}^{M}$.

2: For each database point $(\boldsymbol{x}_i, \mathrm{ID}_i)$, $\mathcal{C}$ independently samples its uniformly random t-server routing set $\mathcal{R}_i \xleftarrow{\$} \binom{[M]}{t}$. For each shard $m \in \mathcal{R}_i$, $\mathcal{C}$ computes $\boldsymbol{v}_{i,m} \leftarrow \mathsf{Hash}(\boldsymbol{x}_i; \mathsf{st}_{\mathrm{hash}}^{(m)})$, perturbs it as $\widetilde{\boldsymbol{v}}_{i,m} \leftarrow \mathsf{LSHRR}(\boldsymbol{v}_{i,m}, p_D)$, assigns a fresh unique exchangeable shard-local label $\ell_{i,m}$, used as the deterministic tie-break key, and samples fresh encryption randomness $\rho_{i,m}$, and computes $C_{i,m} \leftarrow \mathsf{Enc}_{sk_m}(\mathrm{ID}_i \| \boldsymbol{x}_i; \rho_{i,m})$. It uploads $(\ell_{i,m}, \widetilde{\boldsymbol{v}}_{i,m}, C_{i,m})$ to $\mathcal{S}_m$ and stores $\{(m, \ell_{i,m})\}_{m \in \mathcal{R}_i}$ locally for deletion.

3: Each server $\mathcal{S}_m$ constructs its local HNSW graph shard $\mathcal{G}^{(m)}$ over the received perturbed binary embeddings using Hamming distance. Each node is indexed by its shard-local label and stores the corresponding encrypted payload.

[Query Generation]

4: Given a query embedding $\boldsymbol{q}$ and a local logical-query handle qid that is never transmitted, $\mathcal{C}$ computes a shard-specific query binary embedding for each shard: $\boldsymbol{v}_q^{(m)} \leftarrow \mathsf{Hash}(\boldsymbol{q}; \mathsf{st}_{\mathrm{hash}}^{(m)})$ for $m \in [M]$.

5: For each shard m, $\mathcal{C}$ checks the local memoization cache. If a cached PRR representation exists, it retrieves $\overline{\boldsymbol{v}}_q^{(m)} \leftarrow \mathcal{C}_{\mathrm{memo}}[\mathsf{qid}, m]$. Otherwise, it samples $\overline{\boldsymbol{v}}_q^{(m)} \leftarrow \mathsf{LSHRR}(\boldsymbol{v}_q^{(m)}, p_{\mathrm{PRR}})$ and stores it as $\mathcal{C}_{\mathrm{memo}}[\mathsf{qid}, m]$.

6: Before transmission, $\mathcal{C}$ applies instantaneous perturbation to each memoized representation: $\widetilde{\boldsymbol{v}}_q^{(m)} \leftarrow \mathsf{LSHRR}(\overline{\boldsymbol{v}}_q^{(m)}, p_{\mathrm{IRR}})$. It sends each perturbed query binary embedding $\widetilde{\boldsymbol{v}}_q^{(m)}$ to the corresponding server $\mathcal{S}_m$.

[Server-Local Search]

7: Each server $\mathcal{S}_m$ performs HNSW search over its local graph $\mathcal{G}^{(m)}$ using $\widetilde{\boldsymbol{v}}_q^{(m)}$ as the query binary embedding and obtains a shard-level candidate-label set $\mathrm{Cand}^{(m)}$ with budget P.

8: Each server $\mathcal{S}_m$ sends its local encrypted candidates $\{(m, C_{i,m}) : \ell_{i,m} \in \mathrm{Cand}^{(m)}\}$ to $\mathcal{C}$. The client aggregates them as $\mathrm{Cand} = \bigcup_{m=1}^{M} \{(m, C_{i,m}) : \ell_{i,m} \in \mathrm{Cand}^{(m)}\}$. Each ciphertext is accompanied by its shard index for selecting the corresponding decryption key.

[Local Decryption and Exact Reranking]

9: For each pair $(m, C_{i,m})$ in the returned response, $\mathcal{C}$ decrypts the payload using the corresponding shard key: $(\mathrm{ID}_i, \boldsymbol{x}_i) \leftarrow \mathsf{Dec}_{sk_m}(C_{i,m})$.

10: $\mathcal{C}$ removes duplicate candidates caused by multi-graph routing, computes the exact application distance $d_{\mathcal{X}}(\boldsymbol{q}, \boldsymbol{x}_i)$ for each recovered embedding, and reranks the candidates according to these exact distances.

11: Finally, $\mathcal{C}$ outputs the Top-K reranked candidates together with their identifiers and exact distances.

Figure 6: The complete protocol of MESS, including initialization, PRR/IRR query generation, server-local HNSW search, and client-side aggregation, decryption, and reranking.

B ADDITIONAL EVALUATION RESULTS

Table 1: SIFT100M performance under different effective query flip rates $p_q = p_{\mathrm{PRR}}(1 - p_{\mathrm{IRR}}) + p_{\mathrm{IRR}}(1 - p_{\mathrm{PRR}})$.

Dataset	Build Time	p_q	Mode	Online Search Performance				Candidate	Communication
				Server-side Eval.	Trans.Lat.	Client Dec.&Rank	Total Time		
SIFT100M	26.19 h	0.00	WAN	3.21	888.89	1.63	893.73	4000	2123.54
			LAN	1.98	21.52	1.36	24.86	4000	2123.54
		0.08	WAN	8.61	2031.79	2.45	2042.85	9500	5069.04
			LAN	3.43	46.91	2.19	52.53	9500	5069.04
		0.10	WAN	14.33	2666.87	2.86	2684.06	12500	6678.79
			LAN	11.40	57.53	2.55	71.48	12500	6678.79
		0.12	WAN	41.78	4225.49	3.90	4271.17	20000	10686.04
			LAN	10.99	100.49	5.45	116.93	20000	10686.04

Table 2: Latency and communication cost. HNSW is non-private; No-Noise disables perturbation; HE-Cluster and Compass use HE and ORAM, respectively; Ours denotes MESS. Message counts report sequential messages on the critical path. For MESS, two messages denote one parallel fan-out/fan-in round across M servers (that is, $2M$ physical point-to-point transmissions), and communication totals include all server-local requests and responses.

Dataset	Scheme	Offline Setup Phase	Online Search Latency (ms/query)					Sequential Messages/Query	Comm. Overhead (KB/query)
		Build Time (s)	Mode	Server-side Eval.	Trans. Lat.	Client Eval.	Total Time		
SIFT (1M, 128d)	HNSW Malkov & Yashunin (2018)	44.42	WAN	0.02	0.83	–	0.85	2	0.64
			LAN	0.02	0.01	–	0.03		
	No-Noise	435.40	WAN	1.08	80.73	0.97	82.78	2	138.07
			LAN	0.99	1.52	0.89	3.40		
	HE-Cluster	–	WAN	55621.65	24083.69	7716.07	87421.42	2	744096.35
			LAN	55980.91	4250.11	7708.48	67939.52		
	Compass Zhu et al. (2025)	$\geq$ 44.42	WAN	10.03	936.62	11.70	958.35	16.1048	11451.86
			LAN	13.74	36.85	14.43	65.02		
	Ours	512.74	WAN	1.45	323.32	1.34	**326.11**	2	788.82
			LAN	1.16	6.75	1.18	**9.09**		
LAION (100K, 512d)	HNSW Malkov & Yashunin (2018)	6.53	WAN	0.02	4.11	–	4.13	2	2.14
			LAN	0.02	0.07	–	0.09		
	No-Noise	25.64	WAN	1.20	163.11	1.19	165.50	2	391.82
			LAN	1.29	2.54	1.02	4.85		
	HE-Cluster	–	WAN	16408.31	8040.37	2217.09	26665.74	2	208941.03
			LAN	16408.31	1601.69	2228.37	20238.37		
	Compass Zhu et al. (2025)	$\geq$ 6.53	WAN	2.74	1068.67	11.18	1082.59	16.002	45989.36
			LAN	3.12	28.35	11.45	42.92		
	Ours	44.33	WAN	1.12	563.55	1.14	**565.81**	2	1303.57
			LAN	0.92	9.20	0.70	**10.82**		
TripClick (1.5M, 768d)	HNSW Malkov & Yashunin (2018)	1211.97	WAN	0.13	4.92	–	5.05	2	3.14
			LAN	0.07	0.12	–	0.19		
	No-Noise	1137.13	WAN	2.88	1206.90	1.32	1211.10	2	2914.82
			LAN	2.33	20.82	1.10	24.25		
	HE-Cluster	–	WAN	325800.00	29180.00	8620.00	363590.00	2	782504.63
			LAN	358757.20	5502.80	8897.02	373157.00		
	Compass Zhu et al. (2025)	$\geq$ 1211.97	WAN	134.82	4774.15	132.70	5041.67	18.0613	133042.91
			LAN	25.45	200.60	114.65	340.70		
	Ours	1586.19	WAN	6.46	3563.14	1.59	**3571.19**	2	8936.57
			LAN	2.97	18.01	1.62	**22.60**		
MS MARCO (8.8M, 768d)	HNSW Malkov & Yashunin (2018)	10728.84	WAN	0.22	5.05	–	5.27	2	3.15
			LAN	0.18	0.23	–	0.41		
	No-Noise	8676.12	WAN	4.33	1795.45	1.53	1801.31	2	4468.82
			LAN	3.03	32.33	1.52	36.88		
	HE-Cluster	–	WAN	–	–	–	–	–	–
			LAN	–	–	–	–		
	Compass Zhu et al. (2025)	$\geq$ 10728.84	WAN	122.32	7445.57	195.61	7763.50	18.0209	226657.48
			LAN	64.01	361.85	181.90	607.76		
	Ours	11741.53	WAN	4.20	5997.02	1.45	**6002.67**	2	18066.32
			LAN	6.98	120.47	2.84	**130.29**		

B.1 Experimental Parameters

The baseline comparison uses SIFT (1M, $128d$), LAION (100K, $512d$), TripClick (1.5M, $768d$), and MS MARCO (8.8M, $768d$), as summarized in Table 2. For a fair ablation comparison, No-Noise and MESS use the same multi-graph construction and HNSW parameters. Unless otherwise specified, MESS uses $M = 64$ servers, one graph shard per server, routes each database point to $t = 16$ selected servers, represents each database point on each selected shard with a $\kappa = 128$-bit code, and applies storage-side flip probability $p_D = 0.08$. The HNSW construction parameter is fixed to `efConstruction` $= 300$ across all four datasets. Compass is evaluated using the configuration reported in the original Compass paper.

For candidate generation, No-Noise uses smaller configured candidate-pool values because it searches over unperturbed binary embeddings. Its configured candidate-pool ranges are 250–400 on SIFT, 200–300 on LAION, 900–1050 on TripClick, and 1400–1600 on MS MARCO. In contrast, MESS uses larger configured values to compensate for query-side and storage-side perturbation: 1400–1600 on SIFT, 700–800 on LAION, 2900–3200 on TripClick, and 6000–6500 on MS MARCO.

C Candidate-Pool Size and Search Quality

Figure 5 studies how search quality changes with the candidate-pool setting. Here, B denotes the configured candidate-pool value recorded by the experiment; it is not the measured number of candidates returned before or after duplicate removal.

Candidate-recovery model. For query q, let $\mathcal{C}_q(B)$ be the aggregated and deduplicated candidate set returned under configured value B. For a database item j, introduce an effective recovery threshold T_{qj}, measured on the same configured-B scale. We use the monotone-threshold approximation

$$\mathbf{1}\{j \in \mathcal{C}_q(B)\} \approx \mathbf{1}\{T_{qj} \le B\}.$$

This relation is exact if the candidate sets are nested as B increases and otherwise models their overall monotone recovery trend. The latent threshold summarizes server-local HNSW traversal, candidate truncation, aggregation, and duplicate removal; it is not an observed absolute rank. We set $T_{qj} = \infty$ when j is not recoverable for the fixed dataset and perturbation configuration over the candidate-pool sweep.

To state pooled probabilities compactly, let Q be uniform over the evaluation queries and, for Recall, let J be uniform over the exact ground-truth Top-K set of Q. Probabilities also include any retrieval randomness; for a deterministic run, they reduce to empirical proportions over the evaluated query–item pairs. Define $T_{\mathrm{rec}} = T_{QJ}$ and $\gamma_{\mathrm{rec}} = \Pr[T_{\mathrm{rec}} < \infty]$. Within each panel, we use the empirical conditional-CDF approximation

$$\Pr[T_{\mathrm{rec}} \le B \mid T_{\mathrm{rec}} < \infty] \approx \Phi\left(\frac{B - \mu_{\mathrm{rec}}}{\sigma_{\mathrm{rec}}}\right).$$

It follows that the pooled candidate-recovery probability has the form

$$\Pr[J \in \mathcal{C}_Q(B)] \approx \gamma_{\mathrm{rec}} \Phi\left(\frac{B - \mu_{\mathrm{rec}}}{\sigma_{\mathrm{rec}}}\right),$$

where Φ is the standard normal CDF. This is an empirical approximation over the measured range of B, rather than a claim that an observed rank is exactly Gaussian.

Recall. Let $\mathcal{G}_q^K$ be the exact ground-truth Top-K set and let $\mathcal{O}_q^K(B)$ be the final Top-K output after exact client-side reranking. We use the same deterministic tie-breaking rule to construct both sets. Any recovered item in $\mathcal{G}_q^K$ must remain in the reranked Top-K, because fewer than K database items precede it in that common total order. Hence

$$\mathrm{Recall@K}_q(B) = \frac{|\mathcal{O}_q^K(B) \cap \mathcal{G}_q^K|}{K} = \frac{1}{K} \sum_{j \in \mathcal{G}_q^K} \mathbf{1}\{j \in \mathcal{C}_q(B)\}.$$

Averaging over the query set and taking expectations gives

$$\mathbb{E}[\mathrm{Recall@K}(B)] = \frac{1}{|\mathcal{Q}|K} \sum_{q \in \mathcal{Q}} \sum_{j \in \mathcal{G}_q^K} \Pr[j \in \mathcal{C}_q(B)] = \Pr[J \in \mathcal{C}_Q(B)] \approx \gamma_{\mathrm{rec}} \Phi\left(\frac{B - \mu_{\mathrm{rec}}}{\sigma_{\mathrm{rec}}}\right).$$

Thus Recall is exactly an average candidate-recovery probability, while the final CDF expression is the empirical threshold approximation.

MRR. For a query q, let $\mathcal{R}_q$ be its judged relevant set and define $A_q(B) = \{\mathcal{C}_q(B) \cap \mathcal{R}_q \neq \varnothing\}$. Let $\rho_q(B)$ be the rank of the first relevant document in the exactly reranked candidate list, with $\rho_q(B) = \infty$ when $A_q(B)$ does not occur. With the convention $1/\infty = 0$, its MRR@10 contribution is

$$\mathrm{RR@10}_q(B) = \frac{1}{\rho_q(B)} \mathbf{1}\{\rho_q(B) \leq 10\}.$$

Consequently, the query-averaged metric is exactly

$$\mathbb{E}[\mathrm{MRR@10}(B)] = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \sum_{r=1}^{10} \frac{1}{r} \Pr[\rho_q(B) = r].$$

Let $A(B) = A_Q(B)$. Since the reciprocal-rank contribution is zero outside $A(B)$,

$$\mathbb{E}[\mathrm{MRR@10}(B)] = \Pr[A(B)]\, \mathbb{E}[\mathrm{RR@10}_Q(B) \mid A(B)].$$

To connect the first factor to the candidate pool, define the query-level threshold

$$T_q^{\mathrm{rel}} = \min_{j \in \mathcal{R}_q} T_{qj}, \qquad A_q(B) \approx \{T_q^{\mathrm{rel}} \leq B\}.$$

Writing $\gamma_A = \Pr[T_Q^{\mathrm{rel}} < \infty]$, we model its aggregate recovery trend by

$$\Pr[A(B)] \approx \gamma_A \Phi\left(\frac{B - \mu_{\mathrm{mrr}}}{\sigma_{\mathrm{mrr}}}\right).$$

To construct the empirical fitting model, we approximate the conditional reciprocal-rank factor by a constant $\bar{c} \in [0, 1]$ over the measured range. Substitution into the preceding identity gives

$$\mathbb{E}[\mathrm{MRR@10}(B)] \approx \gamma_{\mathrm{mrr}} \Phi\left(\frac{B - \mu_{\mathrm{mrr}}}{\sigma_{\mathrm{mrr}}}\right), \qquad \gamma_{\mathrm{mrr}} = \bar{c}\, \gamma_A \in [0, 1].$$

Unlike the Recall identity above, this is an empirical approximation: newly recovered candidates can change the reranked position, so the CDF describes the aggregate growth and saturation trend rather than enforcing pointwise monotonicity of every measured MRR value.

Fitting. The two response models consequently use the common empirical form

$$f(B; \gamma, \mu, \sigma) = \gamma\, \Phi\left(\frac{B - \mu}{\sigma}\right),$$

where $f(B)$ is the fitted expected Recall or MRR@10 and $0 \leq \gamma \leq 1$, $\mu \in \mathbb{R}$, and $\sigma > 0$. For each dataset, perturbation setting, and evaluation metric, the three parameters are estimated independently from all measured points in that panel by nonlinear least squares; no parameters are shared across panels. The measured points in Figure 5 are sample estimates of these query-averaged responses.

The parameters μ and σ are the location and scale of the latent required-budget approximation, not the mean and standard deviation of an observed rank distribution. In particular, the unconstrained location parameter μ may be negative when that gives the best fit over the observed nonnegative range of B.

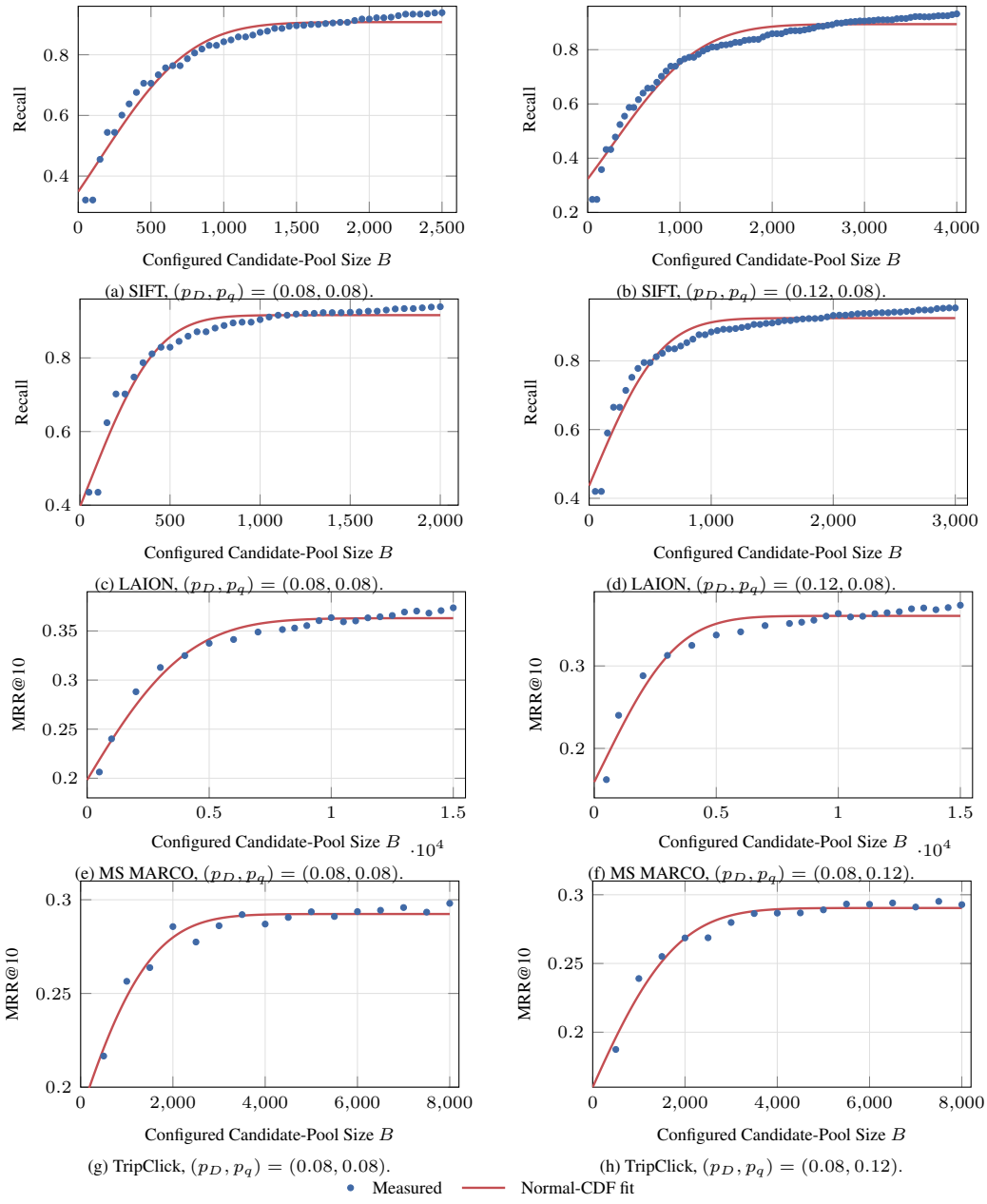


Figure 5: Measured search quality and empirical normal-CDF fits. Markers show measured Recall or MRR@10 at the recorded configured candidate-pool sizes B; lines show the corresponding non-linear least-squares fits.

D COMMUNICATION-EFFICIENT HOMOMORPHIC RERANKING

This appendix describes an optional future extension of MESS, rather than a component of the current implementation or evaluation. MESS currently searches perturbed binary codes, returns encrypted candidate payloads, and lets the client decrypt and rerank the corresponding vectors. Larger candidate pools can improve search quality, but transferring many high-dimensional vectors can become a communication bottleneck.

D.1 OPTIONAL HYBRID DESIGN

At setup, the data owner additionally encrypts each original vector and the fields needed for selection under a homomorphic-encryption (HE) public evaluation key in the same key domain used by authorized clients. The secret key is retained by the client (or another authorized decryptor), while servers receive only public evaluation material. An approximate-arithmetic scheme such as CKKS can support the vector operations required by this extension Cheon et al. (2017). These HE ciphertexts are separate from the randomized authenticated-encryption payloads used by the current MESS protocol.

At query time, the client sends the existing perturbed query reports and a fresh HE ciphertext $\mathsf{ct}_{\mathrm{HE}}(\boldsymbol{q})$. Server $\mathcal{S}_s$ first runs the unchanged MESS search and obtains its local candidate set $\mathcal{C}_s(q)$. It then evaluates a score only for those candidates:

$$\mathsf{ct}(z_{s,i}) \leftarrow \mathrm{Eval}_{\mathrm{HE}}(\mathrm{score},\, \mathsf{ct}_{\mathrm{HE}}(\boldsymbol{q}),\, \mathsf{ct}_{\mathrm{HE}}(\boldsymbol{x}_i))\,, \qquad i \in \mathcal{C}_s(q).$$

For cosine search, normalized vectors reduce the score to an inner product; for Euclidean search, the server can evaluate a squared distance and select the smallest values. Thus, HE computation is applied only after MESS has reduced the search space.

D.2 ENCRYPTED LOCAL SELECTION

After scoring, each server applies a data-independent encrypted Top-k selection network to the scores and the associated encrypted vectors, identifiers, and selection tags. Ranking and sorting constructions for CKKS Mazzone et al. (2025), as well as oblivious Top-k selection networks Cong et al. (2024), provide relevant building blocks. The server does not decrypt scores, comparison bits, identifiers, or selected vectors. In an implementation with variable-size candidate sets, the input should be padded to a fixed public length before selection so that the selection circuit itself does not reveal the local candidate count. Let $n_C = \sum_{s=1}^{M} |\mathcal{C}_s(q)|$ be the number of candidate copies before local selection, including cross-server duplicates. With exact homomorphic scores and a common deterministic tie-breaking rule, the global Top-k over the candidate union is contained in the union of the per-server Top-k sets (using all candidates when a local set has fewer than k items). Each server can therefore return at most k encrypted records, so the number of returned vector ciphertexts falls from n_C to at most Mk. The client decrypts these records, merges and deduplicates them, and performs the final exact ordering.

Approximate CKKS arithmetic can change the order of nearly tied scores; the resulting loss depends on scale, precision, and the comparison circuit. The bound above is therefore a correctness property of exact scores (or of an implementation whose numerical error is smaller than the relevant score gaps), not a new approximation guarantee.

D.3 COST AND SECURITY BOUNDARY

Reducing the number of returned records does not by itself guarantee a proportional reduction in bytes: HE ciphertexts can be substantially larger than the current encrypted payloads, and the query ciphertext adds communication overhead. Servers also incur HE storage, vector-arithmetic, and encrypted selection costs. The extension is consequently most attractive when client bandwidth is scarce and servers can absorb additional computation.

Each server still processes only its own MESS candidate set, so the existing XDP analysis continues to describe the perturbed-code candidate-generation view under the stated non-colluding model. Confidentiality of original vectors, the query, and intermediate scores relies on the computational

security of the chosen HE scheme. This optional design does not add a joint-view privacy claim, hide the MESS access pattern, or constitute a measured performance result; its practical benefit depends on the HE parameters and implementation.

E Supplementary Discussion of the Single-Server Setting

E.1 Deployment and Observation Scope

We consider a resource-constrained deployment in which one physical, semi-honest server stores all M logical graph shards. The logical construction is unchanged: each record independently samples a uniformly random route $\mathcal{R} \in \binom{[M]}{t}$ and is stored in exactly those t shards using shard-specific hashing, randomized response, labels, and graph construction. Unlike a fixed server in the main deployment, this physical server observes the union of all shard-local views and hence all t routed copies of a record. The non-colluding, per-server guarantees in Appendix A therefore do not apply directly to this combined view.

A complete execution could reveal copy correspondence through upload batches, insertion order, arrival times, or other transport metadata. To isolate residual information in the stored representation, the exploratory linkage analysis below suppresses these direct signals. It exposes only perturbed codes and graph structure, supplies one known target entry as an anchor, and allows a fixed set of non-target background correspondences as auxiliary seeds. The resulting observation is a metadata-suppressed code-and-graph view, not the complete physical-server transcript. Ciphertext contents remain covered separately by the randomized authenticated-encryption assumption.

E.2 Joint Randomized-Code View

For a challenge pair $q = (\boldsymbol{x}_0, \boldsymbol{x}_1)$, let $d_s(q) = d_H(H_s(\boldsymbol{x}_0), H_s(\boldsymbol{x}_1))$. If the observer knows the correspondence among all routed copies, then for a realized route $\mathcal{R}$,

$$u_{\mathcal{R}}(q) = \sum_{s \in \mathcal{R}} d_s(q), \qquad \delta^{\mathrm{full}}_{q|\mathcal{R}}(\varepsilon) = \delta^{\mathrm{RR}}_{u_{\mathcal{R}}(q)}(\varepsilon),$$

where δ^{RR}_u is the randomized-response profile defined in Appendix A. The equality follows because the shard copies use independent randomized-response coins and their differing coordinates are analyzed jointly. With random routing, the unconditional profile is the uniform mixture of these route-conditioned profiles over $\mathcal{R} \in \binom{[M]}{t}$.

There is no t/M amplification for this combined observation: the one physical server receives every routed copy, while routing only determines which t shard-specific distances contribute to $u_{\mathcal{R}}(q)$. This calculation is an explanatory randomized-code baseline under known correspondence, not an extension of the non-colluding theorem or a statement about plaintext payloads.

E.3 Exploratory Cross-Shard Linkage

E.3.1 Methods and Setting

The experiment asks whether residual geometry or graph structure can associate target copies when direct correspondence metadata is hidden. For each challenge pair, two neighboring worlds share the same background records and differ only in one target embedding. Routing, storage perturbation, graph construction, and linkage are rerun with fresh randomness. The procedures receive one target anchor and 20% of the non-target background correspondence pool as seeds, but receive neither the remaining target correspondences nor the world label.

Table 3: Cross-shard linkage methods in the metadata-suppressed view.

Method	Residual structure used for association
MDS	Aligns shard-local low-dimensional reconstructions obtained from Hamming distance geometry using the background seeds Torgerson (1952).
Topology	Compares local degree, neighbor-degree, and multi-hop graph-structure descriptors using seeded graph matching Narayanan & Shmatikov (2009); Fishkind et al. (2019); Kazemi et al. (2015).
GSGM	Propagates perturbed-code features over each graph and compares the resulting graph-filtered descriptors Liu et al. (2024).
Joint	Combines the three association scores and retains assignments that pass cross-shard cycle-consistency checks Swoboda et al. (2019).

All method settings are fixed before held-out evaluation. The main parameters match the system configuration, $M = 64$, $t = 16$, and $p_D = 0.08$. The evaluation uses index prefixes of 5k, 10k, and 20k records and challenge pairs with angular distance $d_\theta \le 0.05$.

E.3.2 Exploratory Diagnostic

The diagnostic records how much challenge-pair distance is represented by the correct target copies found by a method. For method $\mathcal{A}$, challenge pair q, neighboring world $b \in \{0, 1\}$, and $n_{q,b}$ trials, let $C^{(j,b)}_{\mathcal{A},q,s} = 1$ exactly when the inferred entry in shard s is the true target copy. Define

$$U^{(j,b)}_{\mathcal{A},q} = \sum_{s\in[M]} C^{(j,b)}_{\mathcal{A},q,s} d_s(q), \qquad \widehat{w}_{\mathcal{A},q,b}(u) = \frac{1}{n_{q,b}} \sum_{j=1}^{n_{q,b}} \mathbf{1}\Big\{U^{(j,b)}_{\mathcal{A},q} = u\Big\}.$$

At target δ_0, the RR-based diagnostic profile and score are

$$\Psi^{\mathrm{diag}}_{\mathcal{A},q,b}(\varepsilon) = \sum_u \widehat{w}_{\mathcal{A},q,b}(u)\delta^{\mathrm{RR}}_u(\varepsilon),$$
$$S^{\mathrm{diag}}_{\mathcal{A},q,b}(\delta_0) = \inf\Big\{\varepsilon \ge 0 : \Psi^{\mathrm{diag}}_{\mathcal{A},q,b}(\varepsilon) \le \delta_0\Big\}.$$

For each dataset–scale configuration, we report the maximum score over the four methods, eligible challenge pairs, and both worlds after computing each cell separately.

This score is only a distance-aware summary of successful associations on the RR-accountant scale. It is not an XDP parameter, an empirical privacy lower bound, or the privacy loss of the linkage procedure. In particular, association is selected using the perturbed codes and graph structure, so the distribution conditioned on a successful match need not be an unselected RR distribution; incorrect and missing matches also contribute zero to U.

E.3.3 Exploratory Observations

At $\delta_0 = 0.01$, the largest diagnostic score ranges from 31.586 to 78.110 across the valid configurations. For MS MARCO, the maxima are 71.812 at 5k using MDS, 60.444 at 10k using MDS, and 70.588 at 20k using GSGM. For SIFT, they are 60.921 at 5k using GSGM, 69.692 at 10k using GSGM, and 78.110 at 20k using Joint. TripClick yields 31.586 at 10k using Joint; this configuration contains only two eligible challenge pairs. No method is largest across all valid configurations. These limited observations indicate that the tested procedures can sometimes use residual code or graph structure to organize cross-shard evidence, but they do not quantify the privacy loss of the physical-server view.

LAION at 5k, 10k, and 20k and TripClick at 20k did not produce valid results because angular-distance validation failed; TripClick at 5k had no eligible challenge pair. We exclude these configurations and draw no linkage conclusion for them.