RAGs to Riches: Dual-Blind Retrieval for Distributed RAG

Anonymous ACL submission

Abstract

Retrieval-augmented generation (RAG) grounds large language models (LLMs) in external, updatable evidence at inference time, enabling factual and domain-specific responses without retraining. In a third-party RAG marketplace, however, this benefit is hard to realize: an LLM owner may need to retrieve from multiple proprietary providers, while providers must protect confidential corpora and users require strong query confidentiality. We introduce DUAL-BLIND RAG, a *bounded-leakage* decentralized retrieval pipeline for multi-provider RAG under a semi-honest threat model. The pipeline combines (i) query-side obfuscated SimHash screening, (ii) distributed top-candidate and block pruning, and (iii) CKKS-based encrypted reranking on a small final pool. By aggressively pruning before encrypted scoring, the system keeps the expensive stage bounded and concentrates most computation on pruned candidate buffers rather than full corpora. DUAL-BLIND RAG avoids raw embedding exchange at every stage and confines all per-query leakage to an explicit, bounded coordination transcript whose contents we characterize stage-by-stage. On MS MARCO, DUAL-BLIND RAG achieves an MRR@10 within 0.03 of a plaintext dense retriever and an nDCG@10 within 8%.

1 Introduction

Modern machine-learning progress depends not only on algorithmic advances, but also on access to *large*, *fresh*, and *domain-specific* data. Scaling results show that more data improves generalization (Sun et al., 2017; Hoffmann et al., 2022), while data acquisition and curation have become a major production cost in deployed ML systems (Mukherjee and Tong, 2024; Appen Limited and Poll, 2022). This has motivated data marketplaces for trading *raw* datasets (Dawex, 2025; Snowflake Inc., 2025a; Amazon Web Services, 2025). Yet raw-data exchange remains difficult to monetize: buyers must often pay before they can verify utility, while providers cannot easily price which parts of a corpus create downstream value. The result is a classic lemons-market problem that raises upfront costs and depresses trust (Azcoitia and Laoutaris, 2022; Snowflake Inc., 2025b; Akerlof, 1978).

Large language models (LLMs) make this problem even more acute. Their usefulness increasingly depends on access to fresh and specialized knowledge at inference time, and retrieval-augmented generation (RAG) supplies that knowledge on demand. This suggests a different market design: instead of transferring raw corpora, independent providers expose retrieval APIs and an LLM owner purchases only the documents actually used.[1] We call this setting **Third-Party RAG**. It improves freshness and long-tail coverage while keeping proprietary corpora in place, but it introduces two privacy constraints: *(i)* providers should not expose their corpora before purchase, and *(ii)* the LLM owner should not reveal its queries. Naive embedding exchange does not satisfy these constraints, since inversion attacks such as VEC2TEXT and ALGEN can recover text from embeddings even in low-data or black-box settings (Morris et al., 2023; Chen et al., 2025) (§2.1).

We present DUAL-BLIND RAG, a decentralized retrieval pipeline for Third-Party RAG that limits practically exploitable leakage without requiring providers to share raw embeddings or corpora. The design combines *privacy-aware approximate screening*, *distributed cascade pruning*, and *encrypted final-stage reranking*: providers interact first through query-obfuscated similarity surrogates, the system progressively prunes providers,

[1] Early commercial instances include AWS Bedrock's BYOI "third-party-RAG" knowledge bases (Amazon Web Services (AWS), 2025), LlamaCloud's managed retrieval service (LlamaIndex Team, 2024), and Voyage AI's retrieval API (Voyage AI, 2025).

blocks, and documents through decentralized top-k coordination, and only a small surviving candidate pool is passed to homomorphic encrypted scoring. We explicitly characterize per-party leakage and target a bounded-leakage envelope under a semi-honest, non-colluding threat model. Empirically, on MS MARCO (Nguyen et al., 2016), DUAL-BLIND RAG achieves 0.304 MRR@10 (within 9% of dense cosine) and remains within 10% on five out-of-domain BEIR benchmarks (Thakur et al., 2021).

2 Background

RAG couples a *retriever* that surfaces external evidence–typically passages of text, but increasingly images, tables, code, and other modalities–with a *generator* (usually an LLM) that conditions on that evidence to produce the final answer (Lewis et al., 2020; Izacard and Grave, 2021; Zhao et al., 2023). Existing deployments typically use dense dual-encoders for recall (Karpukhin et al., 2020), late-interaction models for robustness, and lightweight rerankers (e.g. ColBERT-v2) (Santhanam et al., 2022) for precision.

2.1 Challenges

Although RAG decouples model and data, it introduces a key hurdle: **privacy of the retrieval channel**. Naive dense indexes can leak private user queries and proprietary corpus contents.

In a third-party RAG marketplace, both the consumer's query and the provider's documents are proprietary; nevertheless, publishing an index built from *raw* embeddings exposes three well-studied attack surfaces. (1) **Membership-inference** asks whether a given string was part of the corpus; shadow-training attacks achieve $> 90\%$ precision on popular sentence encoders (Carlini et al., 2021). (2) **Attribute-inference** estimates sensitive labels—e.g. political stance or authorship—from the embedding alone (Song and Raghunathan, 2020). (3) A far more serious threat is **embedding inversion**: reconstructing the *plain text* (or a paraphrase) that produced a vector. The seminal work "*Text Embeddings Reveal (Almost) as Much as Text*" (Morris et al., 2023) trains a conditional language model that, given only a sentence embedding, recovers on average 92% of the original content words for MiniLM and 88% for SBERT across six domains. Crucially, the attack performance scales linearly with corpus size—making it perfectly suited to batch–extract an entire data provider's documents once the index is public. This work has inspired further embedding inversion attacks, including multilingual, transferable, and few-shot cross-model variants (Huang et al., 2024; Chen et al., 2024; Tragoudaras et al., 2025; Chen et al., 2025). These results render raw vector publication fundamentally incompatible with confidential third-party RAG, motivating a private retrieval scheme.

2.2 Related Work

Several cryptographic ANN systems for private semantic search may initially appear suitable for closing these leaks, but they face important limitations in the Third-Party RAG setting. *Compass* traverses HNSW inside Path-ORAM, requiring the client to decrypt full ORAM paths at each hop, increasing extra reads beyond the final top-k and client egress (Zhu et al., 2024). *Pacmann* preserves query privacy via PIR, but retrieving local HNSW subgraphs at each hop still exposes neighborhood structure beyond the purchased documents (Zhou et al., 2025a). *Wally* adds Laplace noise for differential privacy, yet under realistic budgets ($\varepsilon \approx 8$ in legal/medical settings) document membership remains inferable (Asi et al., 2024).

Two systems, *SANNS* and *Tiptoe*, encrypt both query and corpus but do not fit our third-party RAG setting. SANNS combines FHE with garbled circuits, but its precomputed tables scale only to $\leq 10^7$ vectors and would require parallel execution across providers plus a secure cross-provider top-k merge (Chen et al., 2020). Tiptoe combines linearly homomorphic encryption with cluster pruning and scales to 360M web pages on a *single* provider-controlled cluster, but its cost grows with the number of clusters and it requires a global clustering step that would force providers to share raw embeddings (Henzinger et al., 2023). Beyond ANN protocols, several privacy-oriented RAG systems are orthogonal to our setting: they either assume a centralized trusted store (Privacy-Aware RAG) (Zhou et al., 2025b), operate at training time (FedRAG) (Fajardo et al., 2025), rely on TEEs (C-FedRAG) (Addison et al., 2024), or defend only query privacy without protecting provider-side confidentiality (RemoteRAG) (Cheng et al., 2025).

None of these methods satisfies three core requirements of Third-Party RAG: (i) no raw query/embedding sharing, (ii) disclosure only for the retrieved top-k documents, (iii) composability across many mutually distrusting providers with-

Table 1: Can any private ANN act as a drop-in third-party RAG retriever? ✓= satisfies the criterion; ✗= fails; Δ = partially (leaks limited side-info).

Scheme (year)	Query private	Docs private	No extra vectors leaked	Global top-k across M prov?	Main technique
Compass '24 (Zhu et al., 2024)	✓	✗	Δ (10–100)	✗	ORAM-shielded HNSW
Pacmann '25 (Zhou et al., 2025a)	✓	✓	Δ (sub-graph)	✗	Piano-PIR over HNSW
Tiptoe '23 (Henzinger et al., 2023)	✓	✓	✓	✗*	LHE + cluster pruning
SANNS '20 (Chen et al., 2020)	✓	✓	✓	✗†	FHE + garbled circuits
Wally '24 (Asi et al., 2024)	✓	✗	✓	✗	DP + partially-HE
Dual-Blind RAG (ours)	✓	✓	✓	✓	LSH + CKKS FHE

* Tiptoe needs a *global* clustering step; providers would have to share raw embeddings, violating document privacy. † SANNS' garbled-circuit LUTs saturate memory beyond $\sim 10^7$ vectors, so per-provider shards cannot be merged securely.

out a shared global graph; Table 1 summarizes the gap. The mismatch is structural, a naive adaptation would necessarily modify the baselines in ways that defeat their own privacy guarantee. Our design satisfies the three requirements above natively.

2.3 Problem Statement

We consider a *decentralized* marketplace with one **LLM owner** $\mathcal{O}$ and a set of $M > 1$ **data-providers** $\{\mathcal{P}_1, \ldots, \mathcal{P}_M\}$. Provider $\mathcal{P}_i$ holds a private document corpus $P_i = \{d_{i,1}, \ldots, d_{i,n_i}\}$ and a vector index $E_i = \{\mathbf{e}_{i,1}, \ldots, \mathbf{e}_{i,n_i}\}$, where $\mathbf{e}_{i,j} = f(d_{i,j}) \in \mathbb{R}^d$ is the embedding of document $d_{i,j}$ under a shared embedding model $f(\cdot)$. The LLM owner stores a black-box generator $\mathcal{M}$ that produces an answer a for the query $\mathbf{q}$ conditioned on a (small) set of retrieved documents $R \subseteq \bigcup_i P_i$.
Privacy constraints: We capture this practical goal in two privacy constraints:

P1 Query privacy (coordination transcript). No provider should learn the query embedding (or any equivalent function of it) beyond (i) the per-query coordination transcript required by the protocol: the (noised) projection signatures it receives, its own pass/fail outcomes, and its *final encrypted reranking request workload* (i.e., how many/which local candidate identifiers it is asked to score), and (ii) which of its own documents are ultimately purchased and returned in the final top-k.

P2 Document privacy for non-returned content. The LLM owner should learn no provider embeddings beyond the purchased top-k results R. For candidates that are merely *scored* (e.g., in the overshoot set top-L) but not purchased, the owner learns only opaque identifiers and similarity scores, not document content or embeddings.

Threat model and explicit leakage. We assume a *semi-honest* (honest-but-curious) setting: the LLM owner and each provider follow the protocol but may try to infer extra information from the transcript. We assume secure channels and no collusion among providers; the non-collusion assumption is examined in §4.3 and Appendix F. Throughout, we treat document identifiers as *opaque pseudonyms* that are not linkable to document content.

Definition (dual-blind retrieval for non-returned content). We use the term *dual-blind* in an operational sense: (i) the owner learns no plaintext documents or document embeddings beyond the purchased top-k, and (ii) each provider learns no plaintext query or raw query embedding, but only the explicit coordination transcript required by the protocol. This notion is evaluated under the semi-honest, non-colluding threat model above.

Objective: Design a retrieval protocol under the stated constraints while minimizing: *(i)* additional leakages beyond **P1 & P2**, *(ii)* computational overhead on each provider, and *(iii)* communication cost per query. §3 details our construction.

3 Dual-Blind Decentralized Retrieval

Our goal is to retrieve the global top-k documents for a query embedding $\mathbf{q} \in \mathbb{R}^d$ held by the LLM owner while (i) avoiding exchange of raw document embeddings across providers and (ii) avoiding disclosure of the raw query embedding to providers. Rather than fully hiding the retrieval transcript, our design exposes only noised signatures, pruning outcomes, and bounded reranking requests. The four stages progressively narrow the search space: Stage 1 prunes irrelevant providers, Stage 2 prunes irrelevant blocks within them, Stage 3 computes a distributed surrogate top-L over the remaining hot blocks, and Stage 4 applies CKKS only to that bounded pool. All query/document embeddings are d-dimensional and ℓ_2-normalized; we use $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ to denote either a query embedding $\mathbf{q}$ or a document embedding $\mathbf{e}$ depending on context.

3.1 Approximating Cosine Similarity with Random Projection Signatures

Dense retrieval ranks documents by cosine similarity between a query embedding and document embeddings. However, exchanging raw embeddings (even without plaintext) is unsafe: embedding inversion attacks such as VEC2TEXT and ALGEN can reconstruct the underlying text from vectors (Morris et al., 2023; Chen et al., 2025). To enable decentralized screening without revealing raw embeddings, we approximate cosine similarity using locality-sensitive hashing (LSH) signatures. We use LSH (Indyk and Motwani, 1998) (SimHash (Charikar, 2002)) as an embedding-obscuring surrogate for cosine similarity between high-dimensional normalized embeddings via binary random-projection signatures. Given a normalized vector $\mathbf{x} \in \mathbb{R}^d$, we map it to a K-bit (or signed) signature by projecting onto K random hyperplanes and recording the sign of each projection. Concretely, we generate a random projection matrix $W \in \mathbb{R}^{K \times d}$ (Gaussian rows, or orthogonalized when specified in the pipeline) and define the *signed* signature $s(\mathbf{x}) = \mathrm{sign}(W\mathbf{x}) \in \{-1, +1\}^K$, where $s_i(\mathbf{x}) = +1$ iff $\langle \mathbf{x}, \mathbf{r}_i \rangle \geq 0$ and $s_i(\mathbf{x}) = -1$ otherwise, where $\mathbf{r}_i$ is the ith row of W.

The intuition is that two vectors $\mathbf{x}$ and $\mathbf{y}$ which are closely aligned (small angle θ) produce similar sign patterns, while those separated by a large angle disagree on more bits. In particular, for random hyperplanes, the collision probability is $1 - \theta/\pi$. If the Hamming distance between bit-signatures is d_H, then $\mathbb{E}[d_H/K] = \theta/\pi$. A convenient monotone similarity proxy is $\widetilde{\cos}(\mathbf{x}, \mathbf{y}) = 1 - \frac{2\, d_H}{K}$, which equals the normalized signed correlation $\frac{1}{K}\sum_{i=1}^{K} s_i(\mathbf{x})\, s_i(\mathbf{y})$ when using $s(\cdot) \in \{-1, +1\}^K$. Because retrieval only requires ranking, it suffices that $\widetilde{\cos}$ is monotone in the true cosine similarity in expectation, tightening as K increases. Importantly, the pipeline exchanges only signatures—not $\mathbf{x}$ or $\mathbf{y}$—which are many-to-one embedding surrogates. They avoid raw embedding exchange but are not secure encodings, since they still reveal a limited coordination signal and may permit coarse directional inference.

3.2 Private Semantic Similarity Reranking

To compute precise cosine similarities while preserving privacy, we use the CKKS FHE scheme (Cheon et al., 2017); with standard CKKS parameters, decrypted cosine scores match plaintext up to the small approximation error of rescaling. The LLM owner generates a key pair $(\mathsf{pk}, \mathsf{sk})$ and encrypts the query embedding $\mathbf{q}$ as $\mathsf{ct}_q = \mathsf{Enc}_{\mathsf{pk}}(\mathbf{q}) \in \mathsf{CKKS}(\Delta)$. We analyze what providers do and do not learn from observing ct_q in Stage 4, including under approximate-decryption attacks in §4.1, with details in Appendix B.

Packed (batched) encrypted dot products. A naive approach would return one CKKS ciphertext per candidate score, which makes bandwidth and ciphertext handling scale linearly with the number of reranked candidates. Rather than returning one ciphertext per candidate, providers use CKKS packing (SIMD) to amortize computation and bandwidth across multiple candidates at once. Given plaintext document embeddings $\{\mathbf{d}_i\}_{i=1}^{j}$, a provider groups them into packs of size p (up to the available CKKS slots) and homomorphically evaluates a *packed* dot-product operation that returns a ciphertext vector of p scores:

$$\begin{aligned}\mathsf{ct}^{(\ell)} &= \mathsf{EvalPackDot}(\mathsf{ct}_q, \mathbf{D}^{(\ell)}) \\ &\approx \mathsf{Enc}_{\mathsf{pk}}\big([\langle \mathbf{q}, \mathbf{d}_{(\ell,1)} \rangle, \ldots, \langle \mathbf{q}, \mathbf{d}_{(\ell,p)} \rangle]\big).\end{aligned}$$

where $\mathbf{D}^{(\ell)} \in \mathbb{R}^{d \times p}$ is the pack matrix whose columns are the p plaintext document embeddings in pack ℓ. The provider returns $\{\mathsf{ct}^{(\ell)}\}_{\ell=1}^{\lceil j/p \rceil}$ to the LLM owner, who decrypts and extracts the packed scores. Since all embeddings are pre-normalized ($\|\mathbf{q}\| = \|\mathbf{d}_i\| = 1$), the decrypted inner products are cosine similarities (up to CKKS numerical precision). Packing reduces the number of returned ciphertexts from j to $\lceil j/p \rceil$ for that provider. In Stage 4, if provider i is asked to score L_i of the global top-L candidates, the total number of returned ciphertexts across all providers is $\sum_i \lceil L_i/p \rceil$ (rather than L); this is smallest when the top-L is concentrated on a few providers and grows with the number of distinct contributing providers.

3.3 Decentralized Private Retrieval Pipeline

We combine the projection-based cosine approximation of §3.1 with the CKKS-based private cosine of §3.2. The four stages progressively narrow the search space: Stage 1 prunes irrelevant providers, Stage 2 prunes irrelevant blocks within them, Stage 3 computes a distributed surrogate top-L over the remaining hot blocks using a mid-length signature with lazy refinement for heap candidates, and Stage 4 applies CKKS only to that bounded pool. See Appendix A for bounds, derivations, and protocol details.

There are M providers $\mathcal{P}_1, \ldots, \mathcal{P}_M$, each holding a private corpus of document embeddings, and an owner holding a query vector $\mathbf{q} \in \mathbb{R}^d$. Since FHE cosine is too costly at full scale, we select an overshoot set of size $L > k$ using cheap surrogates, reranking only those L candidates by CKKS.

Envelope upper bound: Let $W \in \mathbb{R}^{K \times d}$ be an orthogonal projection and write the query signature as the clean signed vector $s_q = \mathrm{sign}(Wq) \in \{-1, +1\}^K$ (computed as in §3.1); in the actual protocol, Stages 1–3 transmit only its noised version, introduced later in §4.2. Each provider precomputes, for every row w_i of W, the extremes over its local set P: $m_i^{\max} = \max_{d \in P} \langle d, w_i \rangle$ and $m_i^{\min} = \min_{d \in P} \langle d, w_i \rangle$. Define:

$$u_i = \begin{cases} m_i^{\max}, & s_q[i] = +1, \\ -m_i^{\min}, & s_q[i] = -1, \end{cases} \quad U(W, P) = \tfrac{1}{K} \textstyle\sum_{i=1}^{K} u_i.$$

Then for any document $d \in P$, the surrogate $s_W(q, d) = \frac{1}{K} \sum_{i=1}^{K} s_q[i] \cdot \langle d, w_i \rangle$ satisfies $s_W(q, d) \leq U(W, P)$; thus $U(W, P)$ is an *envelope upper bound* valid for all $d \in P$. In Stage 1 we use $W = W_c$ with P the provider's corpus; in Stage 2, $W = W_b$ we get surrogate scores per block in P. Moreover, for isotropic random rows w_i, $\mathbb{E}[\, s_q[i] \langle d, w_i \rangle \,] = c \, \langle q, d \rangle$ for some constant $c > 0$ depending only on the row distribution, so $\mathbb{E}[s_W(q, d)]$ is monotone in the true cosine.

Stage 1: Provider envelope (coarse). Using a shared orthogonal projection W_c, the LLM owner sends a K*-bit signed signature* of $\mathbf{q}$ (not $\mathbf{q}$ itself). Each provider returns a single coarse upper bound over its entire corpus. Providers with $U(W_c, P_i) \leq \tau_{\mathcal{P}}$ are discarded.

Stage 2: Block envelope (refined granularity). Surviving providers partition their corpora into blocks, repeating the envelope test with a larger projection W_b: pruning blocks with $U(W_b, b) \leq \tau_B$. The remaining "hot" providers/blocks are exactly known.

Stage 3: Streaming merge via Threshold Algorithm (TA). The LLM owner and surviving providers use two Stage 3 projections: a mid-length projection W_m to cheaply score all documents in the hot blocks, and a longer refined projection W_r used lazily during TA for documents that enter the running heap (and for provider-side next-item bounds), with $K_r > K_m > K_b > K_c$. Providers expose only (i) a streaming PEEK of their next best candidates from hot blocks and (ii) an UB call reporting the best unseen score under the current refined surrogate. The LLM owner maintains a size-L heap and runs a distributed TA loop, freezing any provider whose best unseen score cannot enter the current top-L. When all providers are frozen, the heap contains the global top-L under the refined surrogate.

Stage 4: Exact top-k via packed encrypted reranking. Given the L candidates, the LLM owner encrypts the normalized query $\mathbf{q}$ once as ct_q and sends it to the involved providers, along with the identifiers of the candidates to score. Each provider $\mathcal{P}$ computes encrypted dot products *in packs* of size p using CKKS packing (§3.2), returning only $\lceil L_{\mathcal{P}}/p \rceil$ ciphertexts for its local share of the L candidates (where $L_{\mathcal{P}}$ is the number of requested candidates held by provider $\mathcal{P}$). The LLM owner decrypts the packed scores and selects the exact top-k. Since Stage 4 computes exact cosines only on the L surviving candidates, it recovers the true top-k only if those candidates are preserved by Stages 1–3. Thus, the final exactness guarantee is conditional on the surrogate cascade, whose earlier stages are designed for high recall rather than exact preservation of the global plaintext cosine ranking.

Privacy and safety: As per the leakage envelope in §4.1, Stages 1–3 exchange only signatures/surrogates and Stage 4 uses CKKS; pruning relies on surrogate scores and envelope bounds that are sound with respect to the transmitted surrogate representation. This ensures that no candidate that could enter the final surrogate top-L is removed before Stage 4, although true cosine top-k preservation remains empirical rather than formally guaranteed; Appendix D.2 measures the resulting ranking distortion stage-by-stage and confirms that CKKS itself contributes no measurable loss.

4 Security and Leakage Analysis

Our privacy claim is bounded leakage with explicit transcript accounting, not full end-to-end cryptographic privacy. It has three components, treated separately: *no raw embedding exchange* (Stages 1–3 send noised signatures, Stage 4 an encrypted ciphertext), *bounded transcript leakage under semi-honest assumptions* (per-stage in §4.1), and *empirical inversion resistance* of the released signatures (§5.3, Appendix B). Leakage thus consists of the per-query coordination transcript visible to each party and the inferences it enables, such as partial recovery of query direction from signatures (§4.2).

4.1 Explicit per-query leakage envelope

We analyze leakage under the semi-honest model in §2.3. Since the protocol is staged (§3.3), each stage reveals a distinct coordination transcript.

Explicit leakage to the owner (per query).

- Stage 1–2 synopsis values: per-provider envelope bounds $B_{\mathcal{P}_i}$ and per-block bounds $B_{\mathcal{P}_i,b}$ for providers/blocks it chooses to query.
- Stage 3 TA stream: identifiers of the documents surfaced by PEEK calls from hot blocks, together with their refined surrogate scores.
- Stage 4 reranking: exact cosine scores for the requested candidate identifiers (after decrypting packed CKKS outputs); no plaintext or embeddings unless a result is purchased/returned.
- Owner-side access pattern: which providers/blocks are contacted and which candidate identifiers are requested for Stage 4 scoring (visible to the owner by construction).

Explicit leakage to a provider $\mathcal{P}_i$ (per query).

- The obfuscated query signatures (randomized bit flips on the query side; see §4.2) used in Stages 1–3 and public protocol parameters (projection dimensions, L, H, CKKS public parameters, ciphertext sizes).
- Its own protocol outcomes and workload: whether it survives Stage 1, which of its blocks survive Stage 2, and the number/identifiers of its local candidates requested for Stage 4 scoring.
- Which of its own documents are ultimately purchased and returned in the final top-k (an unavoidable consequence of serving results).

Non-goal: hiding global access patterns. The owner sees which providers and blocks it contacts by construction; each provider sees only its own requests. We also do not hide provider sizes, timing, or distributional leakage from repeated queries.

Stage 4: what CKKS does and does not protect. Stage 4 is the only stage whose query-side confidentiality reduces to a standard cryptographic assumption. Under CKKS IND-CPA security, ct_q is computationally indistinguishable from an encryption of any other vector, so the provider learns nothing about $\mathbf{q}$ beyond the Stage 1–3 transcript. Active (malicious) provider behaviour and known CKKS approximate-decryption attacks (Li and Micciancio, 2021) are discussed in Appendix B.

4.2 Reconstruction Risk and Obfuscation

In Stages 1–3 the owner never transmits the query embedding $\mathbf{q} \in \mathbb{R}^d$ in the clear; instead it sends only *signed random-projection signatures* $\sigma_q^{(\ell)} = \mathrm{sign}(W_\ell \mathbf{q}) \in \{-1,+1\}^{K_\ell}$ (for stage-dependent projections W_ℓ). This is a strong practical improvement over sharing $\mathbf{q}$, but it is *not* one-way. Because it is a *many*-to-one mapping, a provider knowing the public projection(s) W_ℓ can treat $\sigma_q^{(\ell)}$ as 1-bit measurements and attempt partial recovery of the *direction* of $\mathbf{q}$ (e.g., coarse topic). In our threat model, the provider is the primary adversary: it observes query signatures, while the owner never sees document signatures or embeddings before purchase. To reduce this leakage without sacrificing Hamming/correlation ranking, we obfuscate each query signature with LSH∘RR ("LSHRR") (Fernandes et al., 2022). Before sending a signature $\sigma_q \in \{-1,+1\}^K$, the owner applies independent randomized response to each coordinate:

$$\tilde{\sigma}_{q,i} = \begin{cases} \sigma_{q,i}, & \text{w.p } \frac{e^\varepsilon}{e^\varepsilon+1}, \\ -\sigma_{q,i}, & \text{w.p } p_{\text{flip}} = \frac{1}{e^\varepsilon+1}, \end{cases} \quad i = 1,\dots,K$$

Each coordinate is an ε-randomized-response channel (i.e., ε-DP for one bit), and the resulting bitwise-RR construction corresponds to LSHRR (Fernandes et al., 2022). Importantly, *only the query-side signatures* are noised: providers neither perturb their document representations nor reveal document signatures or embeddings to the owner. This channel attenuates directional information by $\alpha = \tanh(\varepsilon/2) = \frac{e^\varepsilon-1}{e^\varepsilon+1}$, with $\mathbb{E}[\tilde{\sigma}_{q,i} \mid \sigma_{q,i}] = \alpha\,\sigma_{q,i}$. Thus, $\frac{1}{K}\sum_i \tilde{\sigma}_{q,i}\sigma_{d,i}$ stays monotone in expectation but grows noisier as ε decreases, capturing the privacy–utility trade-off. We study inversion risk in Appendix B (with the empirical study in § 5.3) and the privacy trade-off in Appendix D.6.

4.3 Collusion is outside the threat model

Our threat model (§2.3) assumes providers do not collude, and we briefly justify why this assumption is consistent with the setting rather than a defensive choice. The non-collusion assumption matches the parties' commercial incentives: each provider's private corpus is the asset it monetizes through the marketplace, so sharing document embeddings or signatures with another provider is equivalent to giving a competitor the ability to serve the same queries unpaid. Within this assumption, two of the Stage 1–4 collusion surfaces are structurally void — Stages 1–3 broadcast the *same* noised signature to every provider (forced by Stage 3's cross-provider score comparability), and Stage 4 reuses one CKKS ciphertext under IND-CPA security regardless of

coalition size — and the remaining access-pattern channel reduces, per provider, to bounds we characterize in Appendix B and F.

5 Experimental Evaluation

We first evaluate retrieval effectiveness against plaintext retrieval, then analyze privacy, computation, latency, and scalability. The ablation study is deferred to Appendix D, where we show the main quality–cost trade-offs and the mechanisms that keep computation and communication sublinear.

5.1 Experimental Setup

Datasets, embeddings, and baseline. We evaluate retrieval on MS MARCO-dev (6,980 queries) and five BEIR datasets (*nfcorpus*, *trec-covid*, *quora*, *arguana*, *scidocs*). All dense methods use `all-mpnet-base-v2` embeddings (768-d, L2-normalized for cosine). We compare against: (i) Dense-exact cosine similarity (brute-force), (ii) HNSW approximate dense search, and (iii) BM25 sparse retrieval. Implementation details, index parameters, and tuning ranges are in Appendix C.2. We report MRR@10, nDCG@10, Recall@100, and Recall@1 for: (i) Dual-Blind RAG light: Stages 1–3 (no encrypted reranking), and (ii) Dual-Blind RAG: Stages 1–4, our full pipeline. Approximation fidelity results that motivate signature lengths are in Appendix C.1.

5.2 Retrieval Effectiveness

Unless stated otherwise, we evaluate under a decentralized setup with $M = 10$ providers using a *non-IID* (size-skewed) corpus split (Dirichlet $\alpha = 1$). We use $B = 512$ blocks per provider in Stage 2, reranking depth $L = 50$, signature lengths $(K_c, K_b, K_m, K_r) = (124, 256, 512, 1024)$ for Stages 1–3, PRIVHASH/LSHRR noise $\varepsilon = 1.5$, Stage 1 provider threshold $\tau_P = 0.1$, and τ_B tuned to prune 80% of blocks (keep 20%). In Appendix D.1-D.6, we ablate each stage and knob (provider split, B, τ_P, τ_B and ε) to expose the accuracy–cost trade-offs.

MS MARCO (main). Table 2 reports end-to-end retrieval quality in the default decentralized setting. Despite dual privacy and decentralization, DUAL-BLIND RAG retains most of Dense-exact performance, reaching MRR@10= 0.304 and nDCG@10= 0.366 (91% and 93% of Dense-exact), while nearly matching Recall@100 (0.855 vs. 0.864). Recall@1 also matches HNSW (both 0.191) and is only 0.004 below Dense-exact, suggesting the cascade rarely filters out truly relevant items. Comparing DUAL-BLIND RAG LIGHT (Stages 1–3) with the full pipeline isolates the remaining gap. The light variant is close to BM25 on MS MARCO (MRR@10 0.245 vs. 0.241), indicating that Stage 3 mid-signature scoring is the main source of ranking distortion. Stage 4 reranking then substantially improves head quality while preserving near-dense recall, consistent with correcting the final order within a bounded candidate pool. Appendix D.2 decomposes the remaining 0.030 Dense-exact gap stage-by-stage: Stage 1–2 filtering accounts for ≈ 0.023, surrogate selection and top-L truncation a further ≈ 0.007, and CKKS reranking contributes *zero* measurable loss (8×10^{-8} score precision, zero rank changes). See Appendix D.1 for the on/off cost ablations.

Table 2: Retrieval effectiveness on MS MARCO. Best non-private baseline in **bold**, our results in gray.

Method	MRR@10	nDCG@10	Recall@100	Recall@1
Dense exact	**0.334**	**0.395**	**0.864**	**0.195**
BM25	0.241	0.277	0.655	0.103
HNSW	0.333	0.394	0.864	0.191
Dual-Blind RAG light	0.245	0.284	0.657	0.113
Dual-Blind RAG	0.304	0.366	0.855	0.191

BEIR (generalization). Figure 1 summarizes BEIR effectiveness across five diverse datasets. We first verified that our baseline runs reproduce standard BEIR reference numbers closely (typically within ≈ 0.01 absolute on key metrics), so differences are attributable to the privacy-preserving cascade rather than evaluation artifacts.

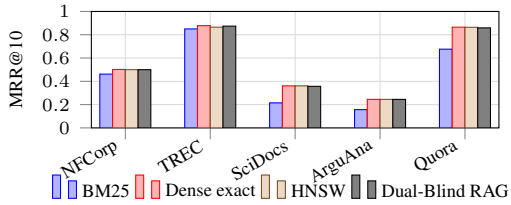

Figure 1: BEIR head-ranking effectiveness (MRR@10) across five tasks. Full breakdown in Appendix C.3.

Overall, DUAL-BLIND RAG preserves most of dense retrieval's advantage over BM25 under domain shift. Recall remains high (conservative Stage 1/2 screening), with the remaining Dense-exact gap concentrated in head ranking, which Stage 4 targets. See Appendix C.3 for the per-dataset breakdown, and Appendix E for a downstream-QA sanity check on 6,000 NQ-open questions confirming that the retrieval gap does not propagate to a measurable answer-quality gap.

5.3 Privacy-Preserving Analysis

Our protocol avoids raw embedding exchange, but the Stage 1–3 transcript still reveals query signatures and is thus not fully cryptographically private. To quantify inversion risk, we evaluate VEC2TEXT and ALGEN under our exact threat channel, using three attacker inputs: the raw query embedding, the clean Stage 3 refined signature, and the default noised signature (PRIVHASH/LSHRR, K_r=1024, ε=1.5). Figure 2 reports mean semantic cosine with 95% bootstrap confidence intervals on MS MARCO-dev. For both attacks, reconstruction drops monotonically from raw embeddings to clean signatures to noised signatures. Appendix B reports additional metrics (ROUGE-L, content-word F1, attack-success, retrieval-intent overlap) and a privacy-utility sweep along ε with a sharp retrieval-collapse transition at $\varepsilon < 1.0$ and a stable operating range ($1.0 \leq \varepsilon \leq 2.0$) containing our default. Overall, the Stage 1–3 transcript is substantially harder to invert than raw embeddings (0.21 vs 0.45 cosine similarity), though residual bounded leakage remains, consistent with §4.2.

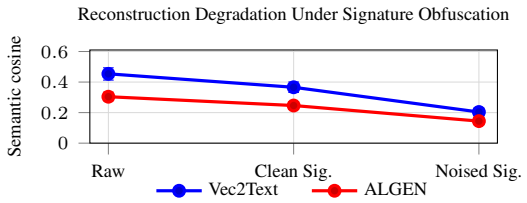

Figure 2: Query reconstruction quality (semantic cosine, 95% CI) under SOTA inversion attacks on `all-mpnet-base-v2`, across three attacker inputs: raw embedding, clean query signature, and noised signature.

5.4 Latency and Communication

We report costs under a *simulated distributed deployment*, separating one-time offline preprocessing from online per-query costs.

Offline one-time preprocessing. Offline preprocessing is amortized across queries; details are in Appendix C.4, and the dominant recurring costs are online (primarily Stages 3–4).

Online latency and bandwidth (per query). Figure 3 shows that our full pipeline takes 3.46 s per query online and exchanges 0.58 MB total (0.291 MB uplink + 0.291 MB downlink). Stage 3 dominates compute at 2.36 s (68.2%) and Stage 4 adds 1.08 s (31.3%) for CKKS reranking; Stages 0–2 together are negligible ($<$ 30 ms, $<$ 1%).

Stage 4 dominates communication and scales with the active top-L pool. Nearly all per-query traffic comes from Stage 4 ($\approx$ 0.58 MB); Stages 1–3 together require $<$ 1 KB. Figure 4 (left) shows that most head-quality gains are achieved by $L \approx 50$, so reranking is restricted to the candidate-owning providers in that small pool. Accordingly, Figure 4 (right) shows Stage 4 latency rising from 0.46 to 6.56 s/query and Stage 4 communication from 0.97 to 16.9 MB/query as M grows from 5 to 200. Stage 4 cost is governed by $\mathbb{E}[\#\text{S4 prov.}]$, the expected providers contributing to the top-L pool: this grows sublinearly to 26.5 at M=200, well below M, so total online latency grows only 2.7× for a 40× increase in M (3.13 → 8.49 s), versus the 125 s linear extrapolation. See Appendix D.7.

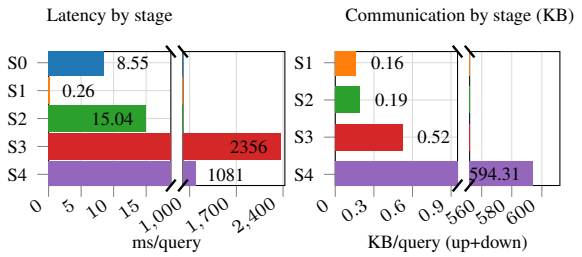

Figure 3: Stage-wise online (per-query) latency and communication under the main setting. S0 is query encoding.

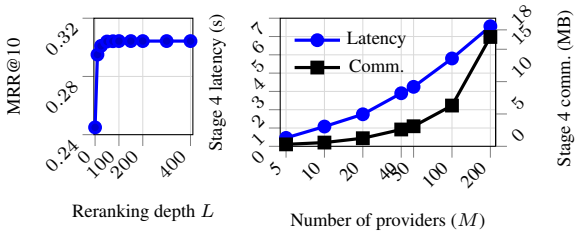

Figure 4: Sensitivity of the final reranking stage. Left: effect of reranking depth L on head quality (MRR@10). Right: provider scaling (M): Stage 4 latency (left axis) and Stage 4 communication (right axis) per query.

6 Conclusion and Future Work

Dual-Blind RAG enables decentralized, document-protecting retrieval with sublinear per-query cost, recovering over 90% of plaintext retrieval performance on MS MARCO and BEIR while reducing inversion attack-success rates by $\approx$ 6–12× relative to raw embedding exchange. A rent-only model–owners paying per grounded answer without acquiring the document–would complement the current buy-to-keep framework, while hybrid FHE/secure-enclave designs or faster homomorphic methods could further reduce reranking overhead. This supports practical third-party RAG without centralizing corpora or exposing raw embeddings.

Limitations

Our contribution is the *retrieval-protocol layer* of a Third-Party RAG marketplace: the mechanism by which an LLM owner obtains top-k results from multiple providers without exchanging raw embeddings. Marketplace-deployment concerns such as document-identifier choice, provider-identity routing, query-frequency patterns, and billing metadata are properties of the deployment substrate and are not addressed here; we use the marketplace setting as motivation, not as the contribution.

Our privacy claims are bounded-leakage and operational rather than full end-to-end cryptographic guarantees for the entire retrieval transcript. However, we explicitly characterize the coordination transcript exposed to each party and show empirically that strong privacy protection against raw embedding exchange can coexist with competitive retrieval quality. In Stages 1–3, providers observe noised query signatures, local pruning outcomes, and reranking workload, which may still support coarse semantic or directional inference, especially under repeated queries. We expect this leakage to overlap with the semantic inference a provider could already make from observing which of its own documents are highly scored or ultimately retrieved. Our results suggest that practical multi-provider RAG can be supported without centralizing corpora or exposing raw embeddings, while leaving stronger malicious-model and repeated-query protections to future work.

The parallel-latency estimates in Appendix D.7 are back-of-the-envelope (sequential latency divided by active S4 providers, ignoring RTT and stragglers); a measured deployment study with realistic networking is future work.

References

Parker Addison, Minh-Tuan H. Nguyen, Tomislav Medan, Jinali Shah, Mohammad T. Manzari, Brendan McElrone, Laksh Lalwani, Aboli More, Smita Sharma, Holger R. Roth, Isaac Yang, Chester Chen, Daguang Xu, Yan Cheng, Andrew Feng, and Ziyue Xu. 2024. C-FedRAG: A confidential federated retrieval-augmented generation system. *arXiv preprint arXiv:2412.13163*.

George A Akerlof. 1978. The market for "lemons": Quality uncertainty and the market mechanism. In *Uncertainty in economics*, pages 235–251. Elsevier.

Amazon Web Services. 2025. Data products—aws data exchange user guide. Accessed 6 May 2025.

Amazon Web Services (AWS). 2025. Evaluate models or rag systems using amazon bedrock evaluations. Demonstrates BYOI format with 'third-party-RAG' identifier.

Appen Limited and Harris Poll. 2022. State of AI and machine learning report 2022. Key finding quoted in Dataconomy, Aug. 18 2022.

Hilal Asi, Fabian Boemer, Nicholas Genise, Muhammad Haris Mughees, Tabitha Ogilvie, Rehan Rishi, Guy N. Rothblum, Kunal Talwar, Karl Tarbe, Ruiyu Zhu, and Marco Zuliani. 2024. Scalable private search with Wally. *arXiv preprint arXiv:2406.06761*.

Santiago Andrés Azcoitia and Nikolaos Laoutaris. 2022. A survey of data marketplaces and their business models. *ACM SIGMOD Record*, 51(3):18–29.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, and 1 others. 2021. Extracting training data from large language models. In *30th USENIX security symposium (USENIX Security 21)*, pages 2633–2650.

Moses S. Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing*, pages 380–388, Montreal, Quebec, Canada. ACM.

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

Yiyi Chen, Heather Lent, and Johannes Bjerva. 2024. Text embedding inversion security for multilingual language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7808–7827, Bangkok, Thailand. Association for Computational Linguistics.

Yiyi Chen, Qiongkai Xu, and Johannes Bjerva. 2025. ALGEN: Few-shot inversion attacks on textual embeddings via cross-model alignment and generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 24330–24348, Vienna, Austria. Association for Computational Linguistics.

Yihang Cheng, Lan Zhang, Junyang Wang, Mu Yuan, and Yunhao Yao. 2025. RemoteRAG: A privacy-preserving LLM cloud RAG service. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 3820–3837, Vienna, Austria. Association for Computational Linguistics.

Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 2017. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology—ASIACRYPT 2017*, volume 10624 of *Lecture*

Notes in Computer Science, pages 409–437, Hong Kong, China. Springer.

Dawex. 2025. Data marketplace. Accessed 6 May 2025.

Val Andrei Fajardo, David B. Emerson, Amandeep Singh, Marcelo Lotif, Veronica Chatrath, Izuki Matsuba, Chi Ho Cheung, and Ravi Theja Desetty. 2025. FedRAG: A framework for fine-tuning retrieval-augmented generation systems. In *ICML 2025 Workshop on Championing Open-source DEvelopment in Machine Learning (CODEML)*.

Jérôme Fernandes, Jérôme Pujol, Luca Oneto, and Fausto Sebé. 2022. PrivHash: Local differential privacy for approximate similarity search. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1234–1242. ACM.

Alexandra Henzinger, Emma Dauterman, Henry Corrigan-Gibbs, and Nickolai Zeldovich. 2023. Private web search with tiptoe. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, page 396–416, New York, NY, USA. Association for Computing Machinery.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, and 3 others. 2022. Training compute-optimal large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 30016–30030. Curran Associates, Inc.

Yu-Hsiang Huang, Yuche Tsai, Hsiang Hsiao, Hong-Yi Lin, and Shou-De Lin. 2024. Transferable embedding inversion attack: Uncovering privacy risks in text embeddings without model queries. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4193–4205, Bangkok, Thailand. Association for Computational Linguistics.

Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbor: Towards removing the curse of dimensionality. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 604–613, New York, NY, USA. ACM.

Gautier Izacard and Édouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, pages 874–880. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and Sebastian Riedel. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Baiyu Li and Daniele Micciancio. 2021. On the security of homomorphic encryption on approximate numbers. In *Advances in Cryptology – EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17–21, 2021, Proceedings, Part I*, page 648–677, Berlin, Heidelberg. Springer-Verlag.

LlamaIndex Team. 2024. Introducing llamacloud and llamaparse. Announces LlamaCloud's managed ingestion and retrieval APIs (private preview).

John Morris, Volodymyr Kuleshov, Vitaly Shmatikov, and Alexander Rush. 2023. Text embeddings reveal (almost) as much as text. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12448–12460, Singapore. Association for Computational Linguistics.

Supantha Mukherjee and Anna Tong. 2024. If your ai seems smarter, it's thanks to smarter human trainers. *Reuters*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated MAchine reading COmprehension dataset. In *Proceedings of the Cognitive Computation Workshop at NeurIPS 2016 (CoCo@NIPS)*.

Yaniv Plan and Roman Vershynin. 2013. One-bit compressed sensing by linear programming. *Communications in Information and Systems*, 12(1):51–72.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. Colbertv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3715–3734.

Snowflake Inc. 2025a. About snowflake marketplace. Accessed 6 May 2025.

Snowflake Inc. 2025b. Monitor listing use. Snowflake Documentation; accessed 6 May 2025.

Congzheng Song and Ananth Raghunathan. 2020. Information leakage in embedding models. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pages 377–390.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. 2017. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852.

Nandan Thakur, Nils Reimers, Andreas Rückle, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *NeurIPS 2021 Datasets and Benchmarks Track*.

Antonios Tragoudaras, Theofanis Aslanidis, Emmanouil Georgios Lionis, Marina Orozco González, and Panagiotis Eustratiadis. 2025. Information leakage of sentence embeddings via generative embedding inversion attacks. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '25, page 3234–3243, New York, NY, USA. Association for Computing Machinery.

Voyage AI. 2025. Introduction. Describes Voyage AI's standalone Retrieval API for embeddings and reranking.

Ruochen Zhao, Hailin Chen, Weishi Wang, Fangkai Jiao, Xuan Long Do, Chengwei Qin, Bosheng Ding, Xiaobao Guo, Minzhi Li, Xingxuan Li, and Shafiq Joty. 2023. Retrieving multimodal information for augmented generation: A survey. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 4736–4756, Singapore. Association for Computational Linguistics.

Mingxun Zhou, Elaine Shi, and Giulia Fanti. 2025a. Pacmann: Efficient private approximate nearest neighbor search. In *The Thirteenth International Conference on Learning Representations*. Poster.

Pengcheng Zhou, Yinglun Feng, and Zhongliang Yang. 2025b. Privacy-aware RAG: Secure and isolated knowledge retrieval. *arXiv preprint arXiv:2503.15548*.

Jinhao Zhu, Liana Patel, Matei Zaharia, and Raluca Ada Popa. 2024. Compass: Encrypted semantic search with high accuracy. Technical Report 2024/1255, IACR Cryptology ePrint Archive.

A Decentralized Private Retrieval Pipeline

Query-side PRIVHASH/LSHRR is applied in all signature stages. In our implementation, the owner never transmits a raw query signature in any stage. Whenever the protocol requires a signed query signature $\sigma_q = \mathrm{sign}(Wq) \in \{-1,+1\}^K$ for some stage-specific projection W, the owner first applies independent randomized response (PRIVHASH/LSHRR) to obtain a *noised* signature

$$\tilde{\sigma}_{q,i} = \begin{cases} \sigma_{q,i}, & \text{w.p. } \frac{e^{\varepsilon}}{e^{\varepsilon}+1}, \\ -\sigma_{q,i}, & \text{w.p. } p_{\text{flip}} = \frac{1}{e^{\varepsilon}+1}, \end{cases} \quad i = 1, \ldots, K$$

and *only* $\tilde{\sigma}_q$ is sent to providers (Stages 1–3). Document signatures are never noised and are never revealed to the owner.

Signed envelope pruning soundness (with noised query signature). Fix a projection matrix $W \in \mathbb{R}^{K\times d}$ with rows $w_1, \ldots, w_K$ and a set of documents P (either a provider corpus or a block), where all embeddings are ℓ_2-normalized. Let the owner compute the signed query signature $\sigma_q = \mathrm{sign}(Wq) \in \{-1,+1\}^K$ and transmit the *noised* signature

$$\tilde{\sigma}_q \in \{-1,+1\}^K$$

obtained by applying PRIVHASH/LSHRR as defined above. Each provider precomputes projection extremes on P:

$$m_i^{\max} = \max_{d\in P} \langle d, w_i \rangle, \qquad m_i^{\min} = \min_{d\in P} \langle d, w_i \rangle.$$

Define the per-row envelope contribution with respect to the received (noised) sign:

$$u_i = \begin{cases} m_i^{\max}, & \tilde{\sigma}_q[i] = +1, \\ -m_i^{\min}, & \tilde{\sigma}_q[i] = -1, \end{cases}$$

$$U(W, P; \tilde{\sigma}_q) = \frac{1}{K} \sum_{i=1}^{K} u_i.$$

Consider the signed surrogate score that is *actually evaluated against the received signature*:

$$\tilde{s}_W(q, d; \tilde{\sigma}_q) = \frac{1}{K} \sum_{i=1}^{K} \tilde{\sigma}_q[i] \cdot \langle d, w_i \rangle.$$

For each i and any $d \in P$: if $\tilde{\sigma}_q[i] = +1$, then $\tilde{\sigma}_q[i]\langle d, w_i \rangle = \langle d, w_i \rangle \le m_i^{\max} = u_i$; if

$\tilde{\sigma}_q[i] = -1$, then $\tilde{\sigma}_q[i]\langle d, w_i\rangle = -\langle d, w_i\rangle \leq -m_i^{\min} = u_i$. Summing over i and dividing by K yields $\tilde{s}_W(q, d; \tilde{\sigma}_q) \leq U(W, P; \tilde{\sigma}_q)$ for all $d \in P$. Therefore, pruning any set P whenever $U(W, P; \tilde{\sigma}_q) \leq \tau$ is *sound* with respect to the surrogate induced by the transmitted (noised) signature.

A.1 Stage 1: Provider Envelope Test

1) **Pre-computation (by each provider $\mathcal{P}$).** For the current provider $\mathcal{P}$, let P denote its local corpus. Before any queries arrive, each provider computes and stores $(m_i^{\min}, m_i^{\max})$ for every row w_i of $W_c \in \mathbb{R}^{K_{\text{coarse}} \times d}$:

$$m_i^{\max} = \max_{d \in P} \langle d, w_i \rangle,$$
$$m_i^{\min} = \min_{d \in P} \langle d, w_i \rangle, \qquad i = 1, \ldots, K_{\text{coarse}}.$$

This is $2K_{\text{coarse}}$ float32 values, i.e., $2K_{\text{coarse}} \cdot 4\text{B}$ (about 1 kB when K_{coarse}=124).

2) **Query arrives (owner → providers).** The owner computes the signed coarse signature $\sigma_q^c = \text{sign}(W_c q) \in \{-1, +1\}^{K_{\text{coarse}}}$ and transmits the *noised* version

$$\tilde{\sigma}_q^c \in \{-1, +1\}^{K_{\text{coarse}}},$$

obtained by applying PRIVHASH/LSHRR (defined at the start of this section). Only $\tilde{\sigma}_q^c$ is sent.

3) **Envelope computation (at provider $\mathcal{P}$).** Given $\tilde{\sigma}_q^c$, the provider forms

$$u_i = \begin{cases} m_i^{\max}, & \tilde{\sigma}_q^c[i] = +1, \\ -m_i^{\min}, & \tilde{\sigma}_q^c[i] = -1, \end{cases}$$

$$B_{\mathcal{P}} = U(W_c, P; \tilde{\sigma}_q^c) = \frac{1}{K_{\text{coarse}}} \sum_{i=1}^{K_{\text{coarse}}} u_i.$$

By the soundness argument above, $B_{\mathcal{P}}$ upper-bounds the induced surrogate $\tilde{s}_{W_c}(q, d; \tilde{\sigma}_q^c)$ for all $d \in P$.

4) **Filtering (owner side).** The provider returns the single float $B_{\mathcal{P}}$. The owner keeps provider $\mathcal{P}$ iff $B_{\mathcal{P}} > \tau_{\mathcal{P}}$ (where $\tau_{\mathcal{P}}$ is the Stage 1 threshold).

A.2 Stage 2: Block–Level Envelope Test

Each surviving provider partitions its corpus into B blocks. We re-apply the signed envelope test within each block using $W_b \in \mathbb{R}^{K_{\text{block}} \times d}$ with $K_{\text{block}} > K_{\text{coarse}}$.

1) **Block extremes (pre-computation).** For each block b and each row $w_{b,i}$ of W_b, store

$$m_{b,i}^{\max} = \max_{d \in b} \langle d, w_{b,i} \rangle,$$
$$m_{b,i}^{\min} = \min_{d \in b} \langle d, w_{b,i} \rangle, \qquad i = 1, \ldots, K_{\text{block}}.$$

Storage per provider is $2BK_{\text{block}}$ float32 values.

2) **Block signature (owner → provider).** Owner computes $\sigma_q^b = \text{sign}(W_b q) \in \{-1, +1\}^{K_{\text{block}}}$ and transmits the *noised* version

$$\tilde{\sigma}_q^b \in \{-1, +1\}^{K_{\text{block}}},$$

via PRIVHASH/LSHRR (defined at the start of this section).

3) **Envelope per block (at provider).** For each block b, define

$$u_{b,i} = \begin{cases} m_{b,i}^{\max}, & \tilde{\sigma}_q^b[i] = +1, \\ -m_{b,i}^{\min}, & \tilde{\sigma}_q^b[i] = -1, \end{cases}$$

$$B_{\mathcal{P},b} = \frac{1}{K_{\text{block}}} \sum_{i=1}^{K_{\text{block}}} u_{b,i}.$$

Then for all $d \in b$, $\tilde{s}_{W_b}(q, d; \tilde{\sigma}_q^b) \leq B_{\mathcal{P},b}$.

4) **Block filtering (owner side).** Each provider returns $\{B_{\mathcal{P},b}\}$ (one float per block). The owner discards any block b with $B_{\mathcal{P},b} \leq \tau_B$; the remaining "hot" blocks advance to Stage 3.

A.3 Stage 3: Distributed Streaming with K_m Ordering and K_r Heap Scoring

After Stage 2, the owner knows the surviving providers and their hot blocks. Stage 3 builds a global overshoot set of size L using a two-signature procedure: a shorter *mid* signature (K_m) to define provider-local stream order, and a refined signature (K_r) to score documents inserted into the global heap.

Two Stage-3 signature lengths. Let

$$W_m \in \mathbb{R}^{K_m \times d}, \qquad W_r \in \mathbb{R}^{K_r \times d},$$

with $K_r > K_m$ in our implementation. Each provider precomputes and stores both document signatures:

$$\sigma_m(d) = \text{sign}(W_m d) \in \{-1, +1\}^{K_m}, \quad (1)$$
$$\sigma_r(d) = \text{sign}(W_r d) \in \{-1, +1\}^{K_r}, \quad (2)$$

packed as bits. For each query, the owner computes $\sigma_m(q), \sigma_r(q)$ and transmits only their noised versions

$$\tilde{\sigma}_m(q), \tilde{\sigma}_r(q)$$

after applying PRIVHASH/LSHRR (as defined at the start of this section).

1) **Provider-local stream order (uses K_m).** For each surviving provider $\mathcal{P}$ and its hot documents $\mathcal{D}_{\mathcal{P}}^{\text{hot}}$, define

$$\tilde{s}_m(q, d) = 1 - \frac{2}{K_m} d_H\big(\tilde{\sigma}_m(q), \sigma_m(d)\big).$$

Provider $\mathcal{P}$ sorts $\mathcal{D}_{\mathcal{P}}^{\text{hot}}$ by $\tilde{s}_m$ (descending), yielding a stream

$$\pi_{\mathcal{P}}(1), \pi_{\mathcal{P}}(2), \ldots$$

(implemented as a merged ordering over hot blocks).

2) **PEEK/UB scores for heap updates (uses K_r).** At stream pointer $t_{\mathcal{P}}$, provider $\mathcal{P}$ returns the next H docs from the K_m-ordered stream and scores them with the refined signature:

$$\tilde{s}_r(q, d) = 1 - \frac{2}{K_r} d_H\big(\tilde{\sigma}_r(q), \sigma_r(d)\big).$$

Interface:

1. $\text{PEEK}_{\mathcal{P}}(H) \rightarrow \{(d, \tilde{s}_r(q, d))\}$ for $d \in \{\pi_{\mathcal{P}}(t_{\mathcal{P}}+1), \ldots, \pi_{\mathcal{P}}(t_{\mathcal{P}}+H)\}$.

2. $\text{UB}_{\mathcal{P}}() \rightarrow \tilde{s}_r\big(q, \pi_{\mathcal{P}}(t_{\mathcal{P}}+1)\big)$ (or $-\infty$ if exhausted).

So, documents pushed into the global heap are scored with K_r signatures (not K_m).

3) **Owner-side heap loop.** The owner maintains a size-L min-heap keyed by refined scores $\tilde{s}_r$ and threshold

$$\tau_L = \text{current } L\text{-th best refined score}$$

($-\infty$ until heap is full). Repeatedly: (i) call $\text{PEEK}_{\mathcal{P}}(H)$ on active providers and update heap; (ii) query $\text{UB}_{\mathcal{P}}()$ and freeze provider $\mathcal{P}$ when $\text{UB}_{\mathcal{P}}() \leq \tau_L$. When all providers are frozen, the heap contents form the Stage 3 overshoot candidate set, which is sent to Stage 4 for exact reranking.

A.4 Stage 4: Final Top-k reranking with Packed CKKS

By the end of Stage 3, the owner has L candidate document identifiers. Let provider $\mathcal{P}$ hold $L_{\mathcal{P}}$ of these candidates.

1. The owner encrypts the normalized query $\mathbf{q}$ once as $\mathsf{ct}_q = \mathsf{Enc}_{\mathsf{pk}}(\mathbf{q})$ and sends ct_q to each involved provider, along with the identifiers of the candidates to score.

2. Each provider groups its $L_{\mathcal{P}}$ candidates into packs of size p (up to CKKS slot capacity) and evaluates a packed dot-product:

$$\mathsf{ct}^{(\ell)} \approx \mathsf{Enc}_{\mathsf{pk}}\Big([\langle \mathbf{q}, \mathbf{d}_{(\ell,1)} \rangle, \ldots, \langle \mathbf{q}, \mathbf{d}_{(\ell,p)} \rangle]\Big),$$
$$\ell = 1, \ldots, \left\lceil \frac{L_{\mathcal{P}}}{p} \right\rceil.$$

   The provider returns only $\lceil L_{\mathcal{P}}/p \rceil$ ciphertexts instead of $L_{\mathcal{P}}$.

3. The owner decrypts the packed ciphertexts, extracts the (approximate) cosine scores, and selects the exact top-k.

Packing reduces Stage 4 bandwidth and ciphertext handling from $O(L_{\mathcal{P}})$ to $O(\lceil L_{\mathcal{P}}/p \rceil)$ ciphertexts per provider, while preserving the same final top-k outcome (up to CKKS numerical precision).

B Risk of Reconstruction and Obfuscation Measures Discussion

What an attacker sees. For each stage $\ell \in \{1, 2, 3\}$ the provider observes only a (potentially noised) query signature $\tilde{\sigma}_q^{(\ell)} \in \{-1, +1\}^{K_\ell}$ together with its *local* coordination transcript (e.g., whether it survives, and how many candidate ids it is asked to score). It does *not* observe the plaintext query embedding $\mathbf{q}$, any document embeddings from other providers, or any decrypted Stage 4 scores. The reconstruction risk we consider here is: given public W_ℓ and observed $\tilde{\sigma}_q^{(\ell)}$, how well can a provider estimate the *direction* of $\mathbf{q}$?

Baseline intuition: one-bit observations can recover coarse direction. The one-bit compressed-sensing analysis of Plan–Vershynin (Plan and Vershynin, 2013) implies that, for random hyperplanes and suitable estimators, directional recovery improves with the number of bits: the mean-square error scales as $\Theta(d/k)$ and thus the expected angular error decreases like $\sqrt{d/k}$. For $d \in \{768, 1024\}$

and $k \approx 1000$ this corresponds to a fairly wide but still semantically informative cone, often sufficient for coarse topical inference. This theoretical intuition motivates our use of query-side obfuscation, but by itself it does not tell us how vulnerable the actual Stage 1–3 transcript is to modern text reconstruction attacks.

Query-side obfuscation via bitwise randomized response (PRIVHASH/LSHRR). To mitigate inversion, we apply independent ε-randomized response (RR) to *query signatures only* before transmission:

$$\tilde{\sigma}_{q,i} = \begin{cases} \sigma_{q,i}, & \text{w.p. } \frac{e^{\varepsilon}}{e^{\varepsilon}+1}, \\ -\sigma_{q,i}, & \text{w.p. } p_{\text{flip}} = \frac{1}{e^{\varepsilon}+1}, \end{cases}$$

which is exactly Warner's RR channel per bit and is the "bitwise RR" component in LSHRR (Fernandes et al., 2022).[2] Define $\alpha = \tanh(\varepsilon/2) = \frac{e^{\varepsilon}-1}{e^{\varepsilon}+1}$ so that $\mathbb{E}[\tilde{\sigma}_{q,i} \mid \sigma_{q,i}] = \alpha\, \sigma_{q,i}$. For similarity estimation one can debias by dividing by α; however our pipeline only requires ranking, so we treat ε as a direct privacy–utility knob and evaluate both the induced inversion resistance (Figure 2) and the retrieval trade-off separately in Appendix D.6.

Empirical protocol with SOTA inversion attacks. The Stage 1–3 threat channel is not access to raw embeddings but access to observed query signatures. We therefore evaluate inversion under three attacker inputs: (i) the raw query embedding, which serves as an upper-bound baseline, (ii) the clean signature σ_q, and (iii) the noised signature $\tilde{\sigma}_q$ used in our default protocol (K=1024, ε=1.5). We use the MS MARCO-dev queries and `all-mpnet-base-v2`. As downstream textual inversion attacks we use VEC2TEXT (Morris et al., 2023) and ALGEN (Chen et al., 2025). Because both attacks natively operate on dense embeddings, for clean and noised signatures we first recover a dense proxy embedding via optimization-based one-bit recovery and then apply the same inversion pipeline. We score each reconstruction by re-encoding the generated text with the corresponding victim encoder and measuring cosine similarity to the original query embedding. Figure 2 reports means with 95% bootstrap confidence intervals.

[2]We report ε as the *per-bit* RR parameter. Under naive composition, releasing k independently randomized bits yields a worst-case $k\varepsilon$ bound in standard local DP; Fernandes *et al.* analyze the combined LSH∘RR construction under an extended-DP view tailored to angular similarity and show that this worst-case bound is pessimistic for the similarity task.

Observed degradation from signatures and RR noise. Across bothattacks we observe the same monotone ordering: raw embedding > clean signature > noised signature. For `all-mpnet-base-v2`, VEC2TEXT falls from 0.454 on raw embeddings to 0.366 on clean signatures and 0.204 on noised signatures, while ALGEN falls from 0.304 to 0.246 to 0.144. This corresponds to raw-to-noised reductions of approximately 55% and 53%, respectively.

Two attack pipelines: optimization-based and linear inversion. Recovering a dense proxy embedding from a sign signature admits two reasonable implementations. *Optimization-based (opt)* treats each bit as a half-space constraint and recovers the proxy by gradient-based one-bit recovery before feeding it to VEC2TEXT/ALGEN; this is the variant used to produce Figure 2. *Linear* treats the signature bits as ridge-regression features against a public set of (embedding, signature) calibration pairs and reads out the regressed embedding directly. We find these two variants are not uniformly ordered: the optimization-based attack works better on clean or near-clean signatures, while ridge regression works better as randomized-response noise increases. At our default ε=1.5, the linear attack produces *higher* semantic cosine (0.246) than the optimization-based attack (0.199). Below we report both variants and treat the maximum-per-ε as the conservative attacker; Figure 2 corresponds to the optimization-based variant, and so underreports the worst-case noised reconstruction by approximately 0.04 semantic cosine at the default ε.

Multi-metric reconstruction at the default operating point. Table 3 reports an expanded set of metrics on 500 MS MARCO-dev queries at ε=1.5, K_r=1024, using `all-mpnet-base-v2` as the victim encoder and VEC2TEXT as the inversion attack. *Sem. cosine*: cosine between the re-encoded reconstruction and the original query embedding (the metric in Figure 2). *ROUGE-L*: token-level overlap between the reconstructed text and the original query string. *CW-F1*: F1 over content words (stopwords removed). $\Pr[\text{Atk}\geq 0.5]$ and $\Pr[\text{Atk}\geq 0.7]$: fraction of queries whose reconstruction exceeds the indicated semantic-cosine threshold (the attack-success-rate framing used in some prior work). Across every metric, moving from raw embedding to clean signature to noised signature degrades the attack monotonically, with the noised variants reducing $\Pr[\text{Atk}\geq 0.5]$ from 37% (raw) to 6% (lin-

Table 3: Multi-metric reconstruction at ε=1.5 on 500 MS MARCO-dev queries (`all-mpnet-base-v2`, VEC2TEXT). Bracketed values are 95% bootstrap confidence intervals.

Attacker input	Sem. cosine [95% CI]	ROUGE-L	CW-F1	Pr[Atk≥0.5]	Pr[Atk≥0.7]
Raw embedding	0.449 [0.424, 0.475]	0.186	0.157	37%	10%
Clean signature (opt)	0.370 [0.345, 0.391]	0.162	0.117	25%	3%
Noised signature, linear (ε=1.5)	0.246 [0.225, 0.266]	0.107	0.063	6%	1%
Noised signature, opt (ε=1.5)	0.199 [0.180, 0.218]	0.090	0.043	3%	0%

Table 4: Retrieval intent overlap on 500 MS MARCO-dev queries (`all-mpnet-base-v2`). RECALL@100 is the fraction of the original query's top-100 documents recovered by retrieval over the reconstructed query; JACCARD@100 is the symmetric set overlap.

Attacker input	Recall@100 [95% CI]	Jaccard@100
Oracle (original query)	1.000	1.000
Clean signature (opt)	0.787 [0.774, 0.800]	0.658
Noised, linear (ε=1.5)	0.616 [0.596, 0.636]	0.461
Noised, opt (ε=1.5)	0.499 [0.476, 0.523]	0.350

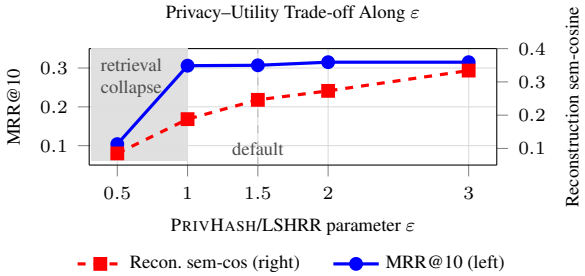

Figure 5: Privacy–utility trade-off along ε. Solid blue: retrieval MRR@10 of the full pipeline. Dashed red: conservative-max reconstruction semantic cosine (max over linear and opt attack variants, VEC2TEXT on `all-mpnet-base-v2`, 500 queries). Shaded grey: retrieval-collapse zone ($\varepsilon < 1.0$). Dashed vertical line: paper default ε=1.5.

ear) or 3% (opt) and Pr[Atk≥0.7] from 10% to at most 1%.

Operational metric: retrieval intent overlap. Semantic cosine measures *intrinsic* reconstruction quality. A more deployment-relevant question is whether an attacker who used a reconstructed query could *retrieve the same documents* as the original query — the operational notion of "learning the user's intent." We measure this directly: for each query we issue both the original and the reconstructed text to a plaintext dense retriever and compare the returned document sets (Table 4). Even on the clean signature, retrieval intent RECALL@100 falls to 0.787 (Jaccard 0.658). On the noised default (ε=1.5), the linear attack recovers 61.6% of the original retrieval set and the optimization-based attack recovers 49.9%. Under the conservative (linear) attacker, an adversary who inverts the noised signature and re-issues the reconstructed query recovers fewer than two-thirds of the documents the original query would have surfaced.

Privacy–utility along the ε axis. The default ε=1.5 is one point on a continuum. Figure 5 sweeps $\varepsilon \in \{0.5, 1.0, 1.5, 2.0, 3.0\}$ along a single axis, reporting on the left the retrieval MRR@10 of the full pipeline and on the right the conservative-max reconstruction semantic cosine (Vec2Text on `all-mpnet-base-v2`, maximum over the linear and opt variants at each ε). Three regions are visible. *(i) Retrieval collapse zone ($\varepsilon < 1.0$):* MRR@10 falls from 0.306 at ε=1.0 to 0.104 at ε=0.5, because at $p_{\text{flip}} \approx 0.38$ the query signatures become too noisy for the envelope upper bounds to distinguish providers and blocks; the cascade no longer functions. *(ii) Operating range ($1.0 \leq \varepsilon \leq 2.0$):* retrieval quality is essentially flat (0.306–0.315 MRR@10), while reconstruction sem-cosine rises from 0.188 to 0.273. The paper-default ε=1.5 sits in the middle of this range. *(iii) Privacy erosion zone ($\varepsilon > 2.0$):* retrieval is unchanged but reconstruction approaches clean-signature quality (0.334 at ε=3.0 vs. 0.370 on the clean signature). A practitioner who values reconstruction-resistance over the safety margin we chose may prefer ε=1.0, which sits at the floor of the operating range and gives sem-cosine 0.188 at no measurable retrieval cost; we use ε=1.5 as the default to keep a buffer above the phase transition.

Implication for the threat model. This experiment directly matches the provider-side inversion channel in our protocol: providers see signatures, not raw embeddings, and query-side RR noise further perturbs those signatures before transmission. The results therefore provide empirical support for the design claim that signature compression already reduces direct inversion risk, and PRIVHASH/LSHRR noise further suppresses

semantic recoverability while preserving enough ranking signal for coordination. At the same time, noised-signature recovery is *not* zero, which is consistent with our stated non-goal of eliminating all semantic leakage from the coordination transcript.

Natural semantic leakage remains (by design). Even without any explicit inversion attempt, a provider can sometimes infer coarse query intent from *which of its own documents* appear most similar under the (possibly noised) signature—e.g., if the provider's top candidates are medical abstracts, the query is likely medical. This is an instance of the *coordination transcript leakage* we declare in the main threat model: our goal is to prevent disclosure of plaintext embeddings or non-purchased document content, not to eliminate inevitable high-level semantic inference from the (few) candidates that become highly scored.

Stage 4 security argument and CKKS approximate-decryption attacks. Stage 4 is the only stage of the protocol whose query-side confidentiality reduces to a standard cryptographic assumption. The provider's per-query view in Stage 4 consists of: the ciphertext ct_q, the identifiers of its local candidates to be scored, the public encryption parameters, and the (encrypted) returned packed scores — none of which it decrypts. Under CKKS's IND-CPA security, which reduces to the Ring-LWE assumption (Cheon et al., 2017), ct_q is computationally indistinguishable from an encryption of any other vector of the same dimension. Informally, a simulator that does not know $\mathbf{q}$ can produce a view distributed identically to the provider's real view by encrypting an arbitrary message of the same shape, so the provider gains no information about $\mathbf{q}$ beyond what the Stage 1–3 transcript already reveals (§4.1).

Approximate-decryption attacks (Li and Micciancio, 2021). A natural objection to relying on CKKS for confidentiality is that CKKS is an *approximate* HE scheme: decryption returns $m + e$ for a small noise term e, and Li and Micciancio (Li and Micciancio, 2021) showed that an adversary with *decryption-oracle access* can use this noise to recover the secret key, distinguishing CKKS's standard IND-CPA security from the stronger IND-CPA^{D} notion. The attack model requires that the adversary be able to (i) submit ciphertexts of its choice and (ii) observe the corresponding decrypted plaintexts. In Dual-Blind RAG this model does not match the threat surface: the only party that ever decrypts is the owner, who holds the secret key, and the only party we treat as an adversary is the provider, who has neither secret-key access nor visibility into any decryption output. The attack therefore does not apply to the provider-as-adversary setting under our protocol; we would need to revisit the security argument only if a deployment exposed decryption outputs (or differences between repeated decryptions of the same ciphertext) to providers, which our protocol does not.

Packed ciphertexts and the linear-algebra reduction. Providers return *packed* ciphertexts (SIMD): instead of sending one ciphertext per candidate, a provider batches up to p candidates per ciphertext and returns only $\lceil j/p \rceil$ ciphertexts for its j requested candidates. Packing changes the ciphertext layout but not the IND-CPA security claim, since the encryption operation and the underlying RLWE problem are unchanged. As usual, if an adversary were to obtain sufficiently many *plaintext* inner products $\langle \mathbf{q}, \mathbf{y}_i \rangle$ against known linearly independent $\mathbf{y}_i$, then $\mathbf{q}$ could be recovered by linear algebra; in our protocol such plaintext values exist only after the owner decrypts, and are never visible to providers.

Out of scope: active (malicious) providers. The above argument assumes semi-honest providers (§2.3). A malicious provider could in principle return ciphertexts that are not honest packed dot products — for example, ciphertexts crafted to encode a different function of $\mathbf{q}$ that the owner will decrypt as a number. This would not violate CKKS security but would violate the protocol's correctness; defending against it would require verifiable computation (zk-SNARKs over CKKS circuits or a public-coin proof of correct evaluation), which we leave to future work.

C Evaluation Details

C.1 Approximation Fidelity Details

As shown in Fig. 6, global correlations increase steeply from $K = 128 \rightarrow 512$ and then plateau at $K = 1024$. The binned MSE concentrates in the high-similarity tail ($[0.6, 1.0]$), which motivates using short signatures for aggressive early pruning and a longer signature right before exact reranking. Errors are essentially flat below 0.3 cosine and rise

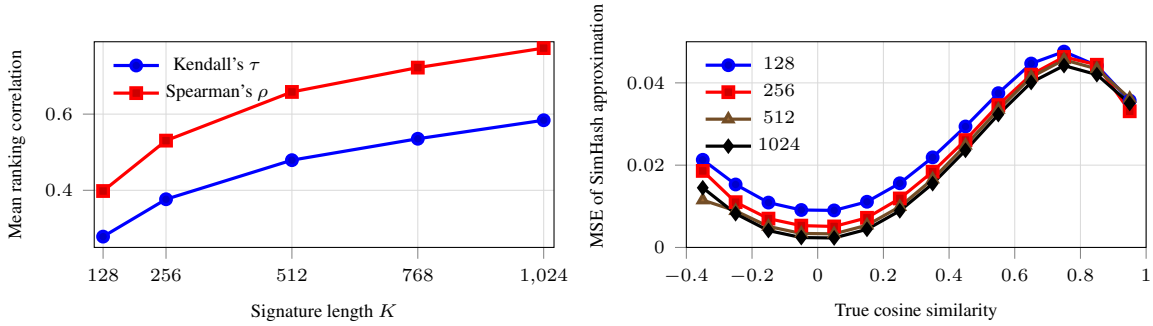

Figure 6: **Approximation fidelity of SimHash.** Left: global ranking fidelity (Kendall's τ and Spearman's ρ) improves with signature length and begins to plateau at larger K. Right: binned MSE of cosine approximation versus true cosine for $K \in \{128, 256, 512, 1024\}$, with largest errors in the high-similarity region.

again in the negative tail (approx. $[-0.4, -0.2]$), consistent with LSH behavior.

C.2 Baseline and Index Hyper-Parameters

Table 5 collects baseline/index settings and the default cascade configuration used throughout §5.2 unless explicitly swept. Dense exact uses FAISS in exhaustive mode; BM25 follows standard Anserini defaults with RM3; HNSW settings follow common practice and were lightly tuned on MS MARCO `train`.

Unless noted, hyperparameters are tuned once on MS MARCO-`train` and then frozen for MS MARCO-dev and BEIR.

C.3 BEIR Results

Across general-domain (NFCorpus) and biomedical (TREC-COVID) retrieval, dense embeddings substantially outperform BM25, and our private cascade preserves most of this gain. For instance, on NFCorpus, Recall@100 rises from 0.209 (BM25) to 0.333 (Dense-exact), while our pipeline incurs only a modest drop relative to dense retrieval (about 0.03 in Recall@100 and 0.005 in nDCG@10). Similarly on TREC-COVID, embeddings improve Recall@100 from 0.040 to 0.051, and the private cascade remains close, suggesting that successive provider- and block-level screening filters aggressively without systematically eliminating relevant passages even under domain shift. SciDocs is a challenging scientific-paper retrieval task where absolute scores are low for all methods (e.g., nDCG@10 $\approx$ 0.077 under dense baselines). Here, we observe that our pipeline is competitive with (and in our runs slightly exceeds) HNSW on Recall@100 (e.g., 0.089 vs. 0.081), indicating that the cascade can maintain broad candidate coverage in difficult regimes where approximate index traversal may miss relevant regions. ArguAna exhibits very high Recall@100 but near-zero Recall@1 across baselines, reflecting that many candidates are topically similar and the single truly relevant sentence is hard to place at rank 1. Our method increases Recall@100 (e.g., to 0.903) but Recall@1 remains low (e.g., 0.022), suggesting that this dataset's bottleneck is fine-grained answer selection rather than neighborhood search. Finally, on the large Quora corpus ($\approx$ 552K passages), embedding-based methods saturate at high recall, and our approach trails Dense-exact only slightly (e.g., $\approx$ 0.02 in Recall@100 and $\approx$ 0.05 in MRR@10), consistent with decentralization and privacy imposing only a modest penalty at scale.

C.4 Offline Preprocessing Details

Table 6 reports the one-time preprocessing cost of materializing stage-specific signatures and synopsis structures used by the cascade. As in the main paper, these costs are paid once and amortized across queries. Storage is dominated by signature materialization and grows roughly linearly with signature length K, while the Stage 1–2 synopsis structures remain comparatively compact.

Peak preprocessing memory is 28.1 GB RSS. Overall, this supports the claim in §5.4 that the main recurring overheads are online—especially Stages 3–4 of streaming and encrypted reranking—while the offline footprint is amortized.

D Additional Ablations and Sensitivity

We report sensitivity studies and detailed ablations under the same overall evaluation protocol. Our ablations show that (i) the final encrypted Stage 4 reranking provides the largest gain in head ranking quality but drives most bandwidth, (ii) Stage 2

Table 5: Baseline and index hyper-parameters (default unless otherwise stated).

Component	Setting
Dense exact (FAISS)	IVF-Flat with n_{list}=1 (exhaustive), inner-product
HNSW	M=32, `ef_construction`=256, `ef_search`=64
BM25 (Anserini)	k_1=0.9, b=0.4, RM3 enabled (default params)
Embeddings	`all-mpnet-base-v2`, d=768, L2-normed for cosine
Our cascade	$(K_c, K_b, K_m, K_r) = (124, 256, 512, 1024)$, B=512, $\tau_P = 0.1$, τ_B=80% of blocks pruned, L=50, M=10, ε=1.5
CKKS rerank	Depth 1, modulus 218-bit, scale 2^{40} (TenSEAL)

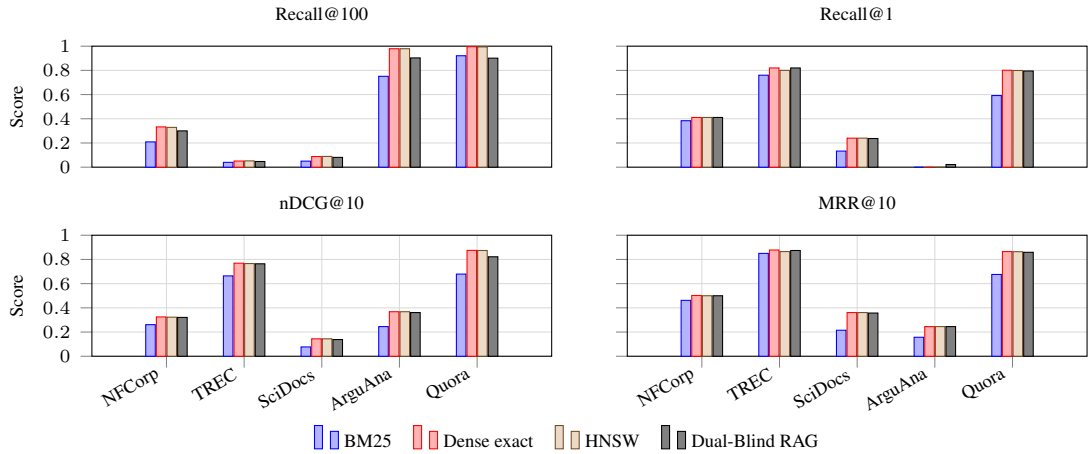

Figure 7: **BEIR effectiveness across five tasks.** Metrics shown: Recall@100, Recall@1, nDCG@10, and MRR@10.

Table 6: Offline preprocessing and storage footprint (one-time, amortized). Peak RSS: 28.1 GB.

Artifact	Variant	Build time	Size on disk
Signatures Stage 1	coarse (K_c=124)	4.22 s	135.2 MiB
Signatures Stage 2	block (K_b=256)	6.69 s	270.3 MiB
Signatures Stage 3	mid (K_m=512)	10.76 s	539.6 MiB
	refined (K_r=1024)	20.49 s	1079.3 MiB
Provider envelopes	S1 synopsis	18.86 s	0.06 MiB
Block synopses	S2 synopsis	27.50 s	44.0 MiB

block pruning and Stage 3 TA streaming are the key mechanisms that keep compute and communication sublinear, and (iii) the main quality–cost knobs are the block pruning aggressiveness and reranking depth L. We also find that retrieval quality is stable as the number of providers grows, but cryptographic cost increases when the top-L pool spans many providers; non-IID provider splits typically help by enabling earlier pruning. Two key messages we want readers to remember: (1) effectiveness close to dense/HNSW with dual privacy, and (2) a controllable accuracy–cost frontier driven mainly by Stage 2 pruning and Stage 4 depth L, with Stage 3 streaming dominating online compute. Unless stated otherwise, results use the default decentralized configuration: M=10 providers, a non-IID Dirichlet split with α=1.0, B=512 blocks/provider, Stage 1 $\tau_P = 0.1$, Stage 2 keep= 20% (prune 80%), L=50, $(K_c, K_b, K_m, K_r) = (124, 256, 512, 1024)$, and PrivHash/LSHRR noise ε=1.5. All latency/-communication numbers are measured on a fixed profiled query subset and are intended to support *relative* comparisons.

D.1 Do the stages matter? (Component Ablations)

Table 7 isolates the contribution of Stage 2 pruning, Stage 3 TA pruning, and Stage 4 reranking (FHE) under the default setting.

Stage 4 (FHE) improves head quality at a clear cost. Comparing *Light* to *Full*, Stage 4 increases MRR@10 from 0.245 to 0.304 and nDCG@10 from 0.284 to 0.366, indicating that exact reranking corrects hash-induced mis-orderings in the head. This gain comes with additional compute and bandwidth (2.38 s → 3.46 s and 0.001 MB → 0.58 MB), consistent with Stage 4 dominating per-query communication.

Stage 2 is primarily a cost reducer (and is conservative). Removing block pruning increases end-

Table 7: Component ablations on MS MARCO-dev.

Variant	MRR@10	nDCG@10	R@100	R@1	Total ms	Comm MB
Full (Stg. 1–4, FHE)	0.304	0.366	0.855	0.191	3460	0.58
Light (Stg. 1–3)	0.245	0.284	0.657	0.113	2380	0.001
w/o Stage 2 (no block prune)	0.305	0.367	0.856	0.191	4180	0.61
w/o TA pruning (scan-all hot blocks)	0.298	0.358	0.860	0.186	10450	6.97

to-end latency by $\sim$21% (3.46 s $\rightarrow$ 4.18 s) with essentially unchanged quality, showing that Stage 2 filters substantial downstream work while rarely over-pruning at the tuned keep rate.

TA pruning in Stage 3 is essential for scalability. Disabling TA termination causes a $\sim$3$\times$ latency blow-up (3.46 s $\rightarrow$ 10.45 s) and a large communication increase (0.58 MB $\rightarrow$ 6.97 MB), because scanning the full hot region expands both the amount of streaming coordination in Stage 3 and the number of candidate-owning providers that must participate in Stage 4. This confirms that TA pruning is not merely an optimization but a key mechanism keeping compute and coordination sublinear in the hot region size.

D.2 Where Does the Quality Gap Come From? (Retrieval Gap Decomposition)

Appendix D.1 isolates each stage's *cost* contribution by switching it on or off. This subsection complements that view by isolating each stage's *quality* contribution: we replace each cascade stage in turn with a plaintext oracle and measure the resulting MRR@10. The row-to-row differences then attribute the gap between Dense-exact and the full pipeline to specific stages.

Setup. We run all variants on 2,000 MS MARCO-dev queries using the paper's default cascade configuration (Dirichlet α=1.0, $\tau_{\mathcal{P}}$=0.10, Stage 2 keep= 20%, L=50, (K_c, K_b, K_m, K_r)=(124, 256, 512, 1024), ε=1.5). At this sample size, the absolute MRR@10 levels reproduce Table 2 to three decimal places (Dense-exact 0.334, full pipeline 0.304), so the decomposition runs on the same operating point as the main results.

The five variants. Each row replaces one cascade component with a plaintext oracle:

1. **Dense exact.** No filtering, no surrogate. Brute-force cosine over the entire corpus.
2. **Stage 1–2 + dense over hot blocks.** Stages 1–2 filter providers and blocks as in the real pipeline, but the surviving "hot" documents are ranked with *plaintext* cosine (no surrogate). This isolates the irrecoverable loss from filtering — documents that the cascade discards before any scoring stage sees them.
3. **Stage 1–3 pool + dense rerank.** Stage 3 selects the candidate pool via the surrogate as in the real pipeline, but the final ranking inside that pool uses plaintext cosine. Isolates the additional loss from the surrogate *selecting* the pool, on top of the filtering loss already counted in (2).
4. **Stage 1–3 top-L + plaintext rerank.** As (3), but truncates the pool to size L=50 before plaintext reranking. Isolates the cost of the top-L cutoff. This row is also our plaintext stand-in for the full Stage 1–4 pipeline; the CKKS diagnostic below confirms the equivalence to 8×10^{-8} score precision.
5. **Stage 1–3 surrogate only.** As (4), but ranking within the top-L uses the surrogate score (no plaintext rerank). The gap between (4) and (5) is the quality recovered by an exact final stage; this is exactly what Stage 4 provides, and matches the "Dual-Blind RAG light" vs. "Dual-Blind RAG" row in Table 2.

Where the gap actually comes from. Three findings stand out. *(i)* The Stage 1–2 block filter is gentle: it costs -0.023 absolute MRR@10 ($\approx$ 7% relative to Dense-exact), much smaller than one might expect given that the filter discards 80% of blocks per provider. *(ii)* The surrogate-based pool selection in Stage 3 contributes only -0.005, and the top-L=50 truncation contributes a further -0.002, consistent with the L-elbow in Figure 4 (left). *(iii)* The single largest component is the gap between surrogate ranking and exact ranking within the top-L pool: 0.245 $\rightarrow$ 0.304, i.e. $+0.059$ recovered by an exact final-stage rerank. This is the contribution of Stage 4. In short, the cascade's filtering and pool selection lose little; what Stage 4 buys back is head ranking inside an already well-chosen candidate pool.

Table 8: Retrieval gap decomposition on 2,000 MS MARCO-dev queries under the paper default (Dirichlet α=1.0, Stage 2 keep= 20%). Each row replaces one cascade component with a plaintext oracle; Δ MRR@10 is the loss vs. the row above. Row 4 corresponds to the full Stage 1–4 pipeline (CKKS-equivalent to plaintext rerank up to 8×10^{-8} score precision; Table 9); the full-pipeline MRR@10 reproduces Table 2 (0.304).

Variant	**MRR@10**	**Δ MRR@10**	**nDCG@10**	**R@100**	**R@1**
Dense exact	0.334	–	0.395	0.864	0.195
+ Stage 1–2 filter, dense over hot	0.311	−0.023	–	–	–
+ Stage 3 pool, dense rerank	0.306	−0.005	–	–	–
+ Stage 3 top-L, plaintext rerank	0.304	−0.002	0.366	0.855	0.191
+ Stage 3 surrogate (no rerank)	0.245	−0.059	0.284	0.657	0.113

What survives each stage. We additionally tracked the fraction of dense top-10 and judged-relevant documents that survive each cascade stage. Of the dense top-10, 83.9% survive Stage 1–2 hot-block selection and 79.6% remain in the Stage 3 top-L. Of judged-relevant documents, 85.7% survive Stage 1–2 and 71.6% remain in the top-L. The relevant-survival rate slightly exceeds the dense-top-10 survival rate through Stage 1–2 (85.7% vs. 83.9%), suggesting the envelope-based filter is at least as gentle on relevance as on raw cosine rank; the ordering reverses at Stage 3 because the surrogate-based pool selection favours surrogate-cosine alignment over judged relevance, which is the ranking-distortion symptom Stage 4 corrects.

CKKS contributes no measurable quality loss. To separate the Stage 4 *rerank* effect from the CKKS approximation itself, we re-ran the Stage 3 top-L pool through either plaintext rerank or CKKS rerank and compared the resulting rankings and scores (Table 9). MRR@10 is identical to four decimal places, and the per-query rank changes in the top-1 and top-10 positions are exactly zero. The mean absolute difference between plaintext and CKKS-decrypted scores is 8.15×10^{-8}, dominated by CKKS rescaling noise. The full 0.030 gap in Table 2 between Dual-Blind RAG and Dense-exact is therefore entirely attributable to the SimHash filtering and surrogate-ranking stages, with the homomorphic reranker contributing zero.

Table 9: CKKS reranking diagnostic on MS MARCO-dev queries. Δ top-1 / Δ top-10 count per-query rank changes between plaintext and CKKS-decrypted scores; *mean abs. score err.* is the average per-candidate $|s_{\text{plain}} - s_{\text{CKKS}}|$.

Rerank	**MRR@10**	**Δ top-1**	**Δ top-10**	**MAE**
Plaintext	0.304	–	–	–
CKKS	0.304	0.000	0.000	8.15×10^{-8}

Reducing the gap: which knob. The decomposition leaves a Dense-exact → Dual-Blind RAG gap of 0.030 absolute MRR@10, split roughly 0.023 Stage 1–2 filtering plus 0.007 surrogate selection and truncation. The Stage 4 rerank knob is already at its quality elbow (Figure 4 left), and the CKKS approximation contributes nothing to close. The only knob whose contribution is large enough to materially close the remaining gap is therefore the Stage 2 keep fraction (Appendix D.4), which trades Stage 1–2 filtering loss against Stage 3/4 work; the existing keep= 20% default sits near the knee of that trade-off curve.

D.3 Stage 1 Provider Threshold ($\tau_{\mathcal{P}}$) Diagnostic

The Stage 1 provider threshold $\tau_{\mathcal{P}}$ controls how aggressively the cascade discards entire providers based on the coarse envelope upper bound $B_{\mathcal{P}} = U(W_c, P_i; \tilde{\sigma}_q^c)$ (see §A.1). Because every document of a discarded provider is permanently lost from the candidate pool, this knob is asymmetric: a too-loose value costs only Stage 2/3 work, while a too-tight value can drop genuinely relevant documents before any later stage can recover them. Following the Stage 1 setup of §5.2, we therefore evaluate $\tau_{\mathcal{P}}$ as a *safety* parameter rather than a quality–cost knob: we measure, per split, the expected number of providers pruned per query and the conditional probability that a pruned provider held a judged-relevant document.

Setup. We sweep $\tau_{\mathcal{P}} \in \{0.00, 0.05, 0.10, 0.12, 0.15, 0.18\}$ over 500 judged MS MARCO-dev queries and the four provider splits used in Appendix D.5: IID (random-even), Dirichlet α=1.0 (the paper default), Dirichlet α=0.1 (heavily skewed), and topic-even. For each query we record (i) the number of providers pruned by the Stage 1 gate, and (ii) for each pruned provider, whether it

contained any document judged relevant for that query in the MS MARCO qrels. The conditional probability $\Pr[\text{rel} \mid \text{pruned}]$ is averaged across queries, restricted to queries for which at least one provider was pruned (entries are 0 if every provider survives across all queries). All other pipeline parameters match the main setting (M=10, ε=1.5, (K_c, K_b, K_m, K_r)=$(124, 256, 512, 1024)$, B=512).

Result 1: zero relevant-provider loss at the default. At the paper default $\tau_{\mathcal{P}}$=0.10, no judged-relevant provider is pruned across any of the four splits over all 500 queries ($\Pr[\text{rel} \mid \text{pruned}]$=0 on all splits).

Result 2: the gate fires meaningfully only under skew. The Dirichlet α=0.1, 1.0 columns are the diagnostic cases. Here the corpus mass is concentrated on a few providers and the envelope upper bound of the remaining providers falls below typical query similarities, so Stage 1 discards on average 5.70, 4.90 providers respectively per query at $\tau_{\mathcal{P}}$=0.10. None of these pruned providers held a relevant document, which empirically confirms the design intuition: Stage 1 removes providers that are unlikely to contribute any high-similarity candidate, and under the heavy-skew regime this redundancy is real rather than nominal.

Result 3: a wide operating margin. Moving from $\tau_{\mathcal{P}}$=0.10 to 0.12 continues to prune zero relevant providers under all splits while increasing the average prune rate on Dirichlet α=1.0 from 4.9 to 5.15. At $\tau_{\mathcal{P}}$=0.15, risk first becomes non-zero but remains small: $\Pr[\text{rel} \mid \text{pruned}]$=0.004 on Dirichlet α=1.0 and 0.0028 on Dirichlet α=0.1 (i.e., roughly 0.3% per query, averaged over queries that triggered a prune). The paper-default $\tau_{\mathcal{P}}$=0.10 is therefore safely *below* this first non-zero point.

Result 4: a sharp cliff at $\tau_{\mathcal{P}}$=0.18. At $\tau_{\mathcal{P}}$=0.18 the threshold exceeds the coarse envelope bound for nearly every query on the balanced splits (random-even, Dirichlet α=1.0, topic-even), pruning all 10 providers per query and disabling the cascade entirely. We mark these entries as degenerate rather than as data points: when every provider is pruned, no candidate enters Stage 2 and the conditional $\Pr[\text{rel} \mid \text{pruned}]$ is operationally vacuous (every relevant document is lost, but the system has also stopped serving any result). Only the skewed Dirichlet α=0.1 column avoids total collapse here (because some providers retain envelope bounds above 0.18), and even there the risk rises to 1.4% – an order of magnitude above $\tau_{\mathcal{P}}$=0.15.

How the default was chosen. $\tau_{\mathcal{P}}$=0.10 was selected once on MS MARCO-`train` using the same selection protocol as the other cascade hyperparameters (Appendix C.2): we chose the smallest value that produced non-trivial Stage 1 activity on the default split we evaluate (Dirichlet α=1.0) without inducing measurable relevant-document loss on `train`. The above test-set sweep was run only after the default was frozen and confirms the choice rather than tuning it. We leave the main quality–cost trade-off to the Stage 2 block threshold τ_B (Appendix D.4) and the reranking depth L (§5.4), neither of which interacts with Stage 1 at $\tau_{\mathcal{P}}$=0.10 on any split where Stage 1 is a no-op.

D.4 Stage 2 Block Pruning: Trade-off and Block Design

We ablate Stage 2 along two axes: (i) *pruning aggressiveness*, implemented as a target keep-fraction (keep $\in \{10\%, 20\%, 25\%, 50\%, 80\%\}$ at B=512), and (ii) *block construction*, varying the number of blocks per provider $B \in \{64, 256, 512, 1024\}$ and using either random or LSH-based partitions. Unless noted, we keep ε=1.5, L=50, M=10, and the default Dirichlet α=1.0 split.

Pruning aggressiveness yields a clean accuracy–latency knob. Figure 8(a) shows a Pareto frontier: moderate pruning (keep $\geq$ 25%) preserves head quality, while aggressive pruning (keep= 10%) further reduces latency but begins to degrade head ranking, reflecting over-pruning once the hot region becomes too small.

Block granularity and partitioning shape downstream Stage 3 work. Figure 8(b) shows Stage 3 latency at keep= 25%. Finer partitions (larger B) can reduce downstream exposure by allowing more precise block pruning. LSH partitions typically tighten the hot region relative to random at the same keep rate.

D.5 Robustness to Provider Splits

We evaluate robustness to how the corpus is partitioned across providers by sweeping: (i) IID (random-even), (ii) size-skewed Dirichlet splits with varying concentration α (smaller α = more skew), and (iii) topic-based splits with different topic resolutions ("bits"). We keep the total corpus fixed and use the same protocol settings (M=10, ε=1.5, L=50, B=512). Table 11 reports effective-

Table 10: Stage 1 provider threshold sweep on 500 judged MS MARCO-dev queries across four provider splits. **Pruned**: expected number of providers discarded by Stage 1 per query (out of M=10). Pr[**rel** | **pruned**]: conditional probability that a pruned provider held a judged-relevant document, averaged across queries where at least one provider was pruned (– when no provider was ever pruned, since the conditional is undefined). The shaded row is the paper default. Italicised entries at $\tau_{\mathcal{P}}$=0.18 are degenerate: the threshold exceeds the envelope upper bound for almost every query, pruning all (or nearly all) providers and disabling retrieval.

	$\tau_{\mathcal{P}}$=0.00		$\tau_{\mathcal{P}}$=0.05		$\tau_{\mathcal{P}}$=0.10 (default)		$\tau_{\mathcal{P}}$=0.12		$\tau_{\mathcal{P}}$=0.15		$\tau_{\mathcal{P}}$=0.18	
Split	Prun.	Pr[rel | ·]	Prun.	Pr[rel | ·]	Prun.	Pr[rel | ·]	Prun.	Pr[rel | ·]	Prun.	Pr[rel | ·]	Prun.	Pr[rel | ·]
Random-even	0.00	–	0.00	–	0.00	–	0.00	–	0.00	–	*10.00*	*n/a*
Dirichlet (α=1.0)	0.00	–	0.00	–	4.90	0.0000	5.15	0.0000	5.92	0.004	*10.00*	*n/a*
Dirichlet (α=0.1)	0.00	–	0.60	0.0000	5.70	0.0000	5.86	0.0000	6.06	0.0028	7.00	0.014
Topic-even	0.00	–	0.00	–	0.00	–	0.00	–	0.00	–	*10.00*	*n/a*

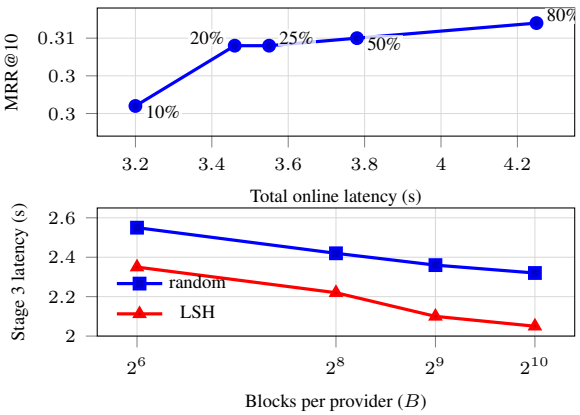

Figure 8: Stage 2 block pruning ablation. (a) Quality–latency trade-off induced by pruning aggressiveness at B=512 (labels denote keep%). (b) Effect of block granularity B on Stage 3 latency at keep= 25%, comparing random vs. LSH partitions.

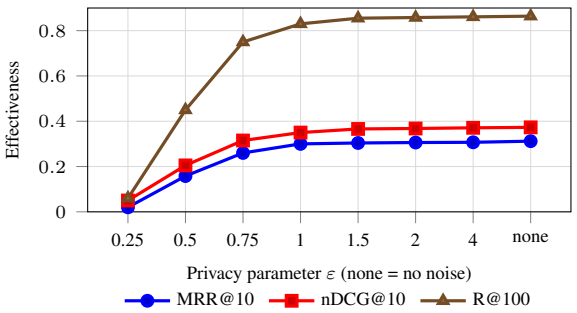

Figure 9: Effect of PrivHash signature noise on effectiveness. We sweep ε and place "none" (no noise) after the largest ε for reference.

ness, pruning statistics, and the resulting Stage 4 (FHE) cost.

Interpretation (high level). Near-IID splits are the hardest case for pruning (many providers survive Stage 1/2), so Stage 4 cost is highest. As non-IID structure increases, the cascade concentrates computation on fewer providers, reducing Stage 4 coordination while maintaining effectiveness.

D.6 Privacy/Noise Sensitivity (PRIVHASH/LSHRR)

We sweep the PrivHash privacy parameter ε (equivalently, the signature bit-flip probability p_{flip}) to quantify the privacy–utility trade-off, keeping the rest of the pipeline fixed (M=10, L=50, B=512, same split). Figure 9 shows a clear elbow around $\varepsilon \approx 1$: utility collapses at very strong noise, recovers rapidly between $\varepsilon \in [0.75, 1.5]$, and then saturates. We use ε=1.5 as the default point beyond the elbow.

The same ε axis, now paired with reconstruction semantic cosine measured by VEC2TEXT (Appendix B, Figure 5), reveals that the elbow at $\varepsilon \approx 1$ is a genuine phase transition rather than a gradual degradation: below ε=1.0 the cascade collapses because the envelope upper bounds can no longer rank providers reliably, while above ε=1.0 retrieval is essentially flat and the only quantity that changes is reconstruction-resistance. The practical operating range is therefore $\varepsilon \in [1.0, 2.0]$; we use ε=1.5 as a default with safety margin above the cliff, but ε=1.0 would yield stronger reconstruction-resistance at no measurable retrieval cost.

D.7 Scaling with the number of providers

Figure 4 (right) shows Stage 4 cost as the marketplace becomes more fragmented. Here we report the full sweep: $\mathbb{E}[\#\text{prov S2}]$ (Stage 2 survivors) and $\mathbb{E}[\#\text{S4 prov.}]$ (providers who actually contribute to the Stage 4 top-L pool). Stage 4 cost is governed by the latter: the active candidate-owning providers, not by M directly.

Setup. We sweep $M \in \{5, 10, 20, 40, 100, 200\}$ on MS MARCO with the default cascade ($\tau_{\mathcal{P}}$=0.10, Stage 2 keep= 20%, L=50, ε=1.5, Dirichlet α=1.0 split). Each row is averaged over

Table 11: Provider split robustness. The **Dirichlet** α=1.0 row corresponds to the default setting used in the main paper.

Split	MRR@10	nDCG@10	R@100	R@1	$\mathbb{E}[\#\text{prov}_{\text{S1}}]$	$\mathbb{E}[\#\text{prov}_{\text{S2}}]$	S4 ms	S4 MB
IID (random-even)	0.292	0.354	0.848	0.187	10.00	9.60	2190	1.74
Dirichlet (α=10)	0.296	0.358	0.850	0.188	10.00	9.10	2050	1.60
Dirichlet (α=1.0)	0.304	0.366	0.855	0.191	5.10	2.90	1080	0.58
Dirichlet (α=0.1)	0.306	0.369	0.858	0.192	4.30	2.30	950	0.52
Topic-even (bits=10)	0.289	0.349	0.846	0.184	10.00	10.00	1980	1.52
Topic-even (bits=14)	0.297	0.357	0.850	0.187	10.00	10.00	1880	1.45
Topic-even (bits=16)	0.302	0.363	0.852	0.189	10.00	10.00	1760	1.36
Topic+Size (single-bucket, bits=16)	0.305	0.367	0.856	0.191	8.60	3.60	1120	0.62

Table 12: Provider-count sweep on MS MARCO-dev. $\mathbb{E}[\#\text{S4 prov.}]$ is the expected number of providers contributing candidates to the Stage 4 top-L pool. *S4 parallel est.* is the sequential S4 latency divided by $\mathbb{E}[\#\text{S4 prov.}]$, a back-of-the-envelope estimate that ignores RTT, synchronisation, and stragglers. MRR@10 varies within experimental noise ($\approx \pm 0.005$ across M).

M	Total seq. lat.	$\mathbb{E}[\#\text{S4 prov.}]$	S4 seq. lat.	S4 parallel est.	S4 comm.	MRR@10
5	3.13 s	1.5	0.46 s	$\sim$305 ms	0.97 MB	0.308
10	3.07 s	2.0	0.59 s	$\sim$294 ms	1.29 MB	0.303
20	3.74 s	4.6	1.19 s	$\sim$259 ms	2.90 MB	0.309
40	5.04 s	10.2	2.59 s	$\sim$254 ms	6.49 MB	0.308
100	7.14 s	20.6	5.13 s	$\sim$249 ms	13.17 MB	0.301
200	8.49 s	26.5	6.56 s	$\sim$248 ms	16.93 MB	0.307

$6,890$ MS MARCO-dev queries. Total sequential latency includes all four stages run on a single machine. Stage 4 parallel-latency estimates are back-of-the-envelope: we divide the sequential Stage 4 latency by the average number of active S4 providers, which approximates wall-clock time under the optimistic assumption of independent parallel execution with no network RTT and no stragglers. We report these to indicate the order-of-magnitude effect of parallelism, not as a measured deployment.

Findings. End-to-end sequential latency grows much slower than linearly in M: a $40\times$ increase in M ($5 \rightarrow 200$) produces only a $2.7\times$ increase in total online latency ($3.13 \rightarrow 8.49$ s). Linear extrapolation from M=5 would predict ≈ 125 s at M=200; the actual measured cost is $14\times$ smaller. Two mechanisms drive this. First, $\mathbb{E}[\#\text{S4 prov.}]$ grows sublinearly in M: at M=200 the top-L=50 candidates are held by 26.5 providers on average, not by all 200. Second, the per-provider Stage 4 cost is roughly constant in the parallel estimate (≈ 250 ms), reflecting that each active provider's work depends on its share of the top-L pool, not on the total provider count.

Retrieval quality is essentially flat across M (MRR@10 0.301–0.309), confirming that fragmentation does not degrade candidate quality once the cascade is correctly configured.

Bandwidth is not flattened by parallelism. The Stage 4 communication column scales with the total candidate count delivered to the owner, which grows with $\mathbb{E}[\#\text{S4 prov.}]$ regardless of how the providers execute internally. At M=200 Stage 4 communication reaches 16.9 MB per query, which a parallel deployment would not reduce. This bandwidth growth is the dominant cost component for heavily fragmented or IID marketplaces.

Deployment implications. Three observations follow. (i) For semantically specialised provider corpora (the natural case in a Third-Party RAG marketplace where providers compete on domain coverage), the cascade concentrates the top-L on a small number of relevant providers and both latency and bandwidth remain modest. (ii) For heavily fragmented IID marketplaces (hundreds of providers each holding small slices of the same distribution), Stage 4 communication grows substantially and the design is less practical without further compression of the encrypted score return. (iii) Parallel execution helps Stage 4 latency but not Stage 4 bandwidth; the latter is the constraint most likely to bind in fragmented settings. We note that the parallel-latency estimates above are back-of-the-envelope; a measured deployment study with realistic RTT and straggler behaviour is left to future work.

E Downstream QA Evaluation

Goal. This appendix evaluates whether the 9% relative MRR@10 gap between DUAL-BLIND RAG and Dense-exact (§5.2, Table 2) translates into a measurable answer-quality gap on a downstream RAG task.

Cross-corpus caveat (lead). The evaluation pairs Natural Questions-open queries (Kwiatkowski et al., 2019) with an MS MARCO passage corpus. NQ answers are grounded in Wikipedia, while MS MARCO is built from anonymised web-search results. Absolute retrieval and answer metrics are therefore systematically lower than NQ on its native corpus: passage relevance is approximate, and many gold short answers do not appear verbatim in any MS MARCO passage. The deployment-relevant quantity is therefore the *relative* gap between two retrieval systems evaluated on the same corpus with the same generator, not the absolute score level.

Setup. We evaluate two retrieval configurations: (i) Dense-exact (brute-force plaintext cosine), and (ii) DUAL-BLIND RAG. The cascade runs with M=10 providers, Dirichlet α=1.0 split, $\tau_{\mathcal{P}}$=0.10, ε=1.5, L=50, and Stage 2 keep= 20%. For each retrieved set, the top-5 passages are fed to the generator `Qweb2.5-1.5B-Instruct` with the prompt template *"Answer the following question in a few words based on the passages below. Give only the answer, no explanation"*. We sample 2000 NQ-open development questions uniformly at random and use 3 seeds. All 6000 resulting examples use the same scoring script: `recall@k` is the fraction of questions whose gold short answer string appears in any of the top-k retrieved passages (case-insensitive substring match); `token F1` and `exact match` are standard SQuAD-style metrics on the generated answer string.

Interpretation. The retrieval gap is real and consistent across seeds: Dense retrieves roughly 2 pp more relevant passages at k=5 (0.482 vs. 0.461), matching the relative MRR gap reported in Table 2. The answer-level metrics, however, are statistically indistinguishable. Token F1 is identical to three decimal places (Δ=0.001, std $\approx$ 0.02), and exact-match is marginally higher for DUAL-BLIND RAG ($\Delta = -0.002$, std $\approx$ 0.015). Both answer-level gaps are smaller than a single standard deviation across seeds, so we treat them as a tie rather than as a reversal. Mechanistically, this is the expected behaviour when the generator reads multiple passages: the marginal passage at rank 4–5 that Dense retrieves but the cascade does not is rarely the only passage containing the gold span, and the generator extracts the same answer from the overlapping content of the top-3 passages that both systems reliably retrieve.

Limitations of this evaluation. Three caveats. *(i) Cross-corpus.* As noted above, NQ-on-MS MARCO under-counts true downstream quality because many gold answers are absent from any retrievable passage; the comparison is internally valid but the absolute numbers should not be read as NQ benchmarks. *(ii) Generator coupling.* Results are reported with one specific generator and prompt; a much weaker generator might be more sensitive to retrieval-quality differences, whereas a much stronger one would be less so. We report on a representative current open-weights model rather than claiming generator-independence.

F Collusion analysis

This appendix expands §4.3 for readers interested in what would change if the non-collusion assumption were relaxed. The treatment is structural rather than experimental: collusion is declared outside our threat model, and the discussion below is provided as orientation rather than as a defended-against attack.

F.1 Per-surface analysis

(i) Signature pooling: structurally void. Stages 1–3 broadcast the *same* noised signature $\tilde{\sigma}_q^{(\ell)}$ to every active provider. This is forced by Stage 3's TA-style global top-L merge: the owner maintains a size-L heap keyed by refined surrogate scores $\tilde{s}_r(\mathbf{q}, d)$, which are only comparable across providers if all providers score against the same query signature. A protocol variant that gave each provider an independently-noised $\tilde{\sigma}_q^{(\ell,i)}$ would render the heap incoherent. Consequently, K colluding providers observe K copies of a single noised signature, not K independent draws; the bit-flip noise added by PRIVHASH/LSHRR cannot be averaged out.

(ii) Stage 4 ciphertexts: cryptographically void. CKKS encrypts $\mathbf{q}$ once as ct_q and broadcasts it to all providers whose candidates appear in the top-L. Under CKKS IND-CPA security (Appendix B,

Table 13: Three-seed aggregated downstream QA results (6000 questions total). *Gap* is Dense minus DUAL-BLIND RAG; positive means Dense is higher.

Metric	Dense (mean±std)	DB-RAG (mean±std)	Gap
Recall@1	0.333 ± 0.009	0.312 ± 0.012	$+0.011$
Recall@3	0.437 ± 0.015	0.420 ± 0.025	$+0.017$
Recall@5	0.482 ± 0.016	0.461 ± 0.018	$+0.021$
Recall@10	0.525 ± 0.011	0.513 ± 0.021	$+0.012$
Token F1	0.235 ± 0.023	0.235 ± 0.014	$+0.001$
Exact match	0.135 ± 0.019	0.137 ± 0.011	-0.002

"Stage 4 security argument"), ct_q is computationally indistinguishable from an encryption of any other vector; K identical copies of the same ciphertext do not improve this distinguishability.

(iii) Access-pattern inference: per-provider, already bounded. The remaining collusion channel is each provider's per-query access-pattern view: which of its blocks survived Stage 2, which of its candidates were peeked, scored, and ultimately purchased. The view splits into two cases by provider role, and in both cases the information bound is one we have already characterized.

- *Providers with documents in the Stage 3 top-L.* Such a provider observes which of its own documents the cascade ranked highly — e.g., that the query returned several medical abstracts from its catalog. This is precisely the "natural semantic leakage from one's own retrieved documents" channel that Appendix B (paragraph "Natural semantic leakage remains") declares a non-goal of the protocol: serving a query inherently reveals something about its topic to a provider that contributes results. Pooling this information across K colluding providers gives the coalition a broader topical picture, but only over documents *some* member contributed — which are documents those members would learn about anyway by serving the query honestly.

- *Providers fully pruned by Stage 1 or Stage 2.* Such a provider receives the noised signature, returns its envelope bound, and observes that none of its candidates are requested. Its query-side view reduces to the noised signature alone. The inversion bound for this view is the noised-signature reconstruction in Table 3 (≈ 0.246 semantic cosine under the conservative-max attack at ε=1.5), and collusion adds no bits because all coalition members hold the same signature.

Where collusion *would* add bits, and why providers won't. The only way for a coalition to extract genuinely new information about the query would be for member providers to share their *document-side* information — their corpus embeddings, block signatures, or precomputed envelope extrema — with one another. This is structurally different from pooling query-side observations, but it is also directly incentive-incompatible: the private corpus is the asset a provider monetizes through the marketplace, and sharing its embeddings with a competitor is equivalent to giving the competitor the ability to serve the same queries without compensation.

F.2 Incentive alignment

Each party's incentive aligns with semi-honest behaviour. A provider's revenue depends on (a) contributing documents that the owner purchases, which requires participating honestly in Stages 1–4 (the only way to be paid is to be selected), and (b) keeping its corpus exclusive enough to remain worth subscribing to, which directly forbids sharing embeddings with competitors. The owner pays per result and benefits from correct top-k selection; deviating from the protocol saves nothing and is empirically detectable in any real marketplace (dispute mechanisms, repeated queries, public reference benchmarks). Honest participation is the cheapest way for either party to obtain its objective.

F.3 What full collusion-resistance would cost

For completeness, two natural points further along the collusion-resistance spectrum:

- *Oblivious access patterns.* Replacing each Stage 3 PEEK/UB call with an ORAM-style oblivious read would hide which candidates were peeked from each provider, at the cost of increasing communication from sublinear in the hot region to polylogarithmic in the entire corpus per call. This eliminates surface (iii) above but substantially changes the cost profile in §5.4.

- *Dishonest-majority MPC over the top-L merge.* The global top-L heap could be maintained as a secret-shared structure across providers under dishonest-majority MPC. This eliminates per-provider visibility into candidate identifiers but adds rounds proportional to the security parameter, currently far slower than CKKS for our score scales.

Our protocol sits at a different point on this spectrum: bounded-leakage retrieval under semi-honest, non-colluding providers, with structural and incentive-based arguments that two of the three collusion surfaces remain void and the third is bounded by already-acknowledged channels.