GATERAG: KEYWORD-GATED HYBRID RETRIEVAL FOR EFFICIENT PRIVATE RAG

Anonymous authors
Paper under double-blind review

ABSTRACT

While retrieval-augmented generation (RAG) significantly enhances the capabilities of large language models, protecting both sensitive user queries and server-owned corpora remains challenging. Existing private RAG systems either incur prohibitive cryptographic overheads for strict privacy, or sacrifice substantial confidentiality to achieve practical efficiency. In this paper, we present GATERAG, a private RAG retrieval framework that considerably reduces cryptographic cost without revealing query-dependent information to the server or server-owned plaintext documents to the client, while also improving retrieval quality. GATERAG achieves this through private keyword gating, which prunes the search space before expensive secure computation. The keyword matches obtained during gating are reused as lexical relevance signals and securely fused with semantic similarity over the reduced candidate set. This hybrid fusion exploits complementary semantic and lexical evidence to improve retrieval quality, while reusing the gating-derived lexical signals avoids a separate costly lexical computation. Experiments on three standard benchmarks (PubMedQA, HotpotQA, and FEVER) demonstrate that GATERAG substantially improves the retrieval quality–efficiency tradeoff over prior private RAG systems. A single parameter P controls this tradeoff, enabling different operating points along the quality–efficiency spectrum. At a quality-oriented operating point, GATERAG improves recall by 1.70–20.64 percentage points while reducing latency by 53.3–96.0% and communication by 89.5–98.6% compared with Pisces (Liang et al., 2026), the only prior system matching our privacy guarantees. At a more efficiency-oriented operating point, the latency and communication reductions reach 91.9–97.8% and 97.3–99.2%, respectively. GATERAG therefore provides a promising foundation for efficient private RAG.

1 INTRODUCTION

Retrieval-augmented generation (RAG) has become a fundamental paradigm for extending large language models (LLMs) beyond their parametric knowledge. By retrieving relevant documents from an external knowledge source, RAG can improve factuality and enable LLMs to leverage domain-specific or up-to-date information (Lewis et al., 2020; Gao et al., 2023). However, deploying RAG in real-world applications introduces substantial privacy concerns (Arzanipour et al., 2025; Bodea et al., 2026). In a conventional client-server architecture, the client must send sensitive queries to an untrusted service provider, risking the exposure of private information. Simultaneously, the server-side knowledge base often contains proprietary or confidential data that must remain shielded from the client. Addressing this bidirectional privacy challenge is critical for enabling RAG systems in privacy-sensitive domains such as healthcare (Zhao et al., 2025), finance (Choi et al., 2025), and legal services (Pipitone & Alami, 2024).

Consequently, the development of end-to-end private RAG systems has become an active area of research. While a substantial body of work on private LLMs (Gupta et al., 2024; Zheng et al., 2024; Xu et al., 2025; Hou et al., 2026; Moon et al., 2025) can be leveraged to secure the generation phase, the private retrieval phase tailored to RAG remains comparatively less mature (Cheng et al.,

Our code is available at `https://anonymous.4open.science/r/GateRAG_f/`.

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

Table 1: Functionality and privacy comparison with existing private RAG systems (●: Full support, ◑: Partial support, ○: No support). Pisces supports candidate reduction only in its semantic path, while query-content privacy in RemoteRAG is differential-privacy-bounded.

Method	Functionality		Privacy		
	Hybrid retrieval	Candidate reduction	Query content	Access pattern	Plaintext document
RemoteRAG	○	●	◑	○	○
ShieldRAG	○	●	●	○	○
PIR-RAG	○	●	●	●	○
Pisces	●	◑	●	○	●
GATERAG	●	●	●	●	●

2025; Chang et al., 2026b; Liang et al., 2026; Wang et al., 2025). Despite recent advances, existing retrieval approaches face two key challenges that impede their deployment in real-world scenarios.

The first challenge is reducing the cost of private retrieval without sacrificing privacy. Cryptographic retrieval is computationally intensive, especially when executed over large-scale corpora. Existing private RAG systems therefore attempt to reduce the search space before expensive secure computation. To construct a smaller query-relevant candidate set, RemoteRAG (Cheng et al., 2025) identifies documents near a differentially perturbed query embedding, ShieldRAG (Chang et al., 2026b) selects query-relevant locality-sensitive hashing buckets, and Pisces (Liang et al., 2026) employs an oblivious semantic filter operating over Hamming space. These mechanisms reduce computation, but expose query-dependent information that allows the server to infer which subsets of the database are semantically relevant to the client's query.

PIR-RAG (Wang et al., 2025), in contrast, performs private clustering-based candidate reduction that hides query-dependent information from the server. However, it severely compromises server-side confidentiality by revealing the entire set of unencrypted candidate documents to the client. RemoteRAG and ShieldRAG likewise expose server-owned documents in plaintext during retrieval. An important gap therefore remains in achieving efficient search-space reduction while revealing neither query-dependent information to the server nor plaintext documents to the client.

The second challenge is improving retrieval quality in private RAG by combining semantic and lexical signals without incurring prohibitive overhead. These two signals offer complementary notions of relevance, as semantic representations capture paraphrases and conceptual similarity, whereas lexical matching is particularly effective for rare entities, numbers, acronyms, and exact terminology (Bruch et al., 2023). This complementarity has led modern RAG systems to increasingly adopt hybrid semantic–lexical retrieval (Mandikal & Mooney, 2024; Muthusami & Saritha, 2026; Wang et al., 2026). Contrarily, most private RAG systems still rely exclusively on semantic similarity, leaving the benefits of lexical evidence largely unexplored (Cheng et al., 2025; Chang et al., 2026b; Wang et al., 2025).

Pisces (Liang et al., 2026) is a notable recent exception, introducing private hybrid retrieval with separate semantic and lexical paths. While its semantic path can reduce computation at the cost of query-dependent leakage, as discussed above, the lexical path provides no such option and always performs exhaustive full-corpus evaluation. Consequently, regardless of whether semantic search-space reduction is employed, the overall hybrid retrieval cost remains dominated by the full-corpus lexical path. This bottleneck motivates the need for a novel private retrieval design that captures the accuracy benefits of lexical signals without incurring substantial overhead.

In this work, we present GATERAG, a private RAG retrieval framework that addresses both challenges (Table 1). Our core mechanism—private keyword gating—securely isolates a small set of lexically matching candidates without revealing query-dependent access patterns to the server. While the keyword gating itself intrinsically filters for lexical similarity, GATERAG further refines the resulting candidates through a secure hybrid ranking phase. Semantic relevance scores are computed only over this reduced set, while the keyword matches obtained during gating are efficiently reused as the lexical relevance signal, thereby circumventing the need for a separate private lexical computation. The semantic and lexical scores are then securely fused to retrieve the documents with the highest hybrid relevance.

GATERAG securely constructs a reduced candidate set using oblivious pseudorandom function (OPRF)-protected keyword lookup and private information retrieval (PIR), without exposing query-dependent information to the server. Subsequent retrieval and ranking use PIR-to-share and secure multi-party computation (MPC), keeping server-owned plaintext documents and embeddings hidden from the client.

Our contributions are summarized as follows:

- To the best of our knowledge, GATERAG is the first private hybrid retrieval framework for RAG that avoids expensive full-corpus search without revealing query-dependent information to the server or server-owned plaintext data to the client.
- We introduce private keyword gating, which jointly reduces the search space and derives a lightweight lexical relevance signal from the same keyword matches. Reusing this signal for hybrid retrieval eliminates a separate full-corpus lexical scoring path.
- GATERAG achieves strong empirical efficiency and retrieval quality across all evaluated datasets. Relative to Pisces under the same query-privacy guarantee, it improves recall by up to 20.64 percentage points while reducing latency by up to 96.0% and communication by up to 98.6%.

2 RELATED WORK

In this work, we focus on the single-server RAG framework over a server-owned corpus, explicitly avoiding reliance on trusted hardware, non-collusion assumptions, or client ownership of the corpus. Appendix A further discusses private RAG under alternative trust and ownership models and related private similarity search outside the RAG setting.

RemoteRAG (Cheng et al., 2025) sends a differentially private perturbation of the query embedding to the server and computes cosine similarities using partially homomorphic encryption (PHE) over the k' documents nearest to the perturbed query. The privacy budget determines both the information exposed by the query and the size of the subsequent cryptographic search. Its strongest query-privacy endpoint, $k' = N$ (the total corpus size), does not allow candidate reduction and requires full-corpus processing. Similarly, ShieldRAG (Chang et al., 2026b) restricts retrieval to query-selected locality-sensitive hashing buckets and protects similarity computation using inner-product functional encryption, where completely removing its bucket-identity leakage eliminates candidate reduction. Both systems ultimately expose retrieved document contents to the client in plaintext.

Pisces (Liang et al., 2026) implements its semantic path using secret-sharing-based MPC and PIR, while its lexical path employs multi-instance labeled private set intersection (PSI) to obtain the term frequencies required for BM25 scoring (Robertson & Zaragoza, 2009). It does not return plaintext documents to the client, but the lexical path reveals corpus-wide term-frequency values for query tokens. Its coarse-to-fine semantic mode additionally reveals the entire candidate set to the server, while the fine-only variant removes this leakage but requires full-corpus semantic scoring. PIR-RAG (Wang et al., 2025) partitions the database into semantic clusters and privately downloads a query-relevant cluster using lattice-based PIR. While this approach hides query-dependent access patterns from the server, it exposes the full plaintext content of the retrieved cluster to the client.

In summary, prior private RAG systems either leak query-dependent information for efficiency, demand prohibitive full-corpus computations for strict privacy, or disclose server documents to the client. GATERAG bridges this gap by achieving query-private candidate reduction while hiding plaintext documents.

3 PRELIMINARIES

In this section, we briefly introduce the cryptographic primitives required to understand our protocol. Notation is deferred to Appendix B.

Secure multi-party computation (MPC). MPC enables multiple parties to jointly evaluate functions over their private inputs without revealing them. Additive secret sharing and garbled circuits (GCs) are widely used building blocks for two-party computation (Shamir, 1979; Yao, 1986). Under

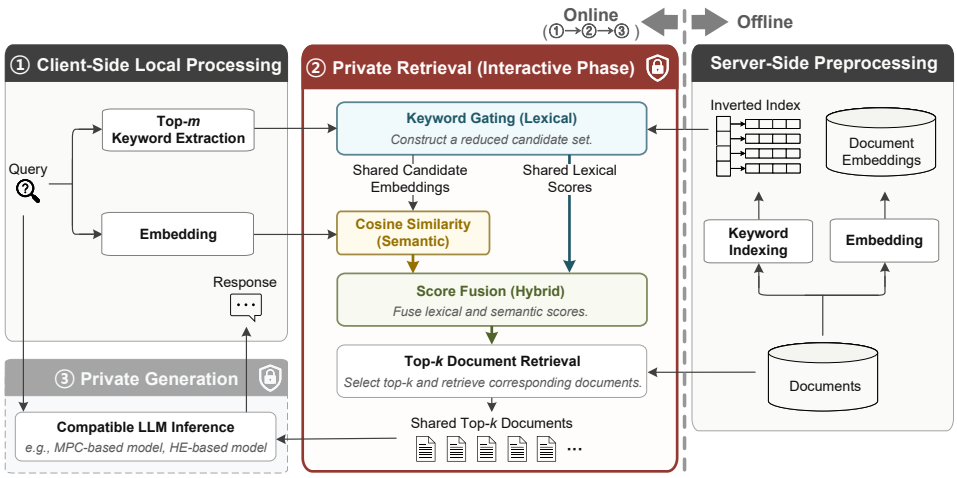


Figure 1: Overview of GATERAG.

additive secret sharing over a ring $\mathbb{Z}_Q$, a value x is split between a client $\mathcal{C}$ and a server $\mathcal{S}$ into secret shares $x_\mathcal{C}$ and $x_\mathcal{S}$ such that $x = x_\mathcal{C} + x_\mathcal{S} \pmod{Q}$. We denote this pair by $[\![x]\!]$, where neither party can independently reconstruct the underlying value x. Alternatively, GCs securely evaluate Boolean circuits and are suitable for comparison-based operations.

Oblivious pseudorandom function (OPRF). An OPRF allows a client holding x and a server holding a key K to jointly compute a pseudorandom function $F_K(x)$ (Freedman et al., 2005). The client obtains $F_K(x)$ without learning K, and the server learns nothing about x, enabling private lookups.

Private information retrieval (PIR). PIR allows a client to retrieve a record $D[i]$ from a server-held database D without revealing the index i to the server (Chor et al., 1998). While standard PIR returns plaintext records, a PIR-to-share protocol returns secret shares of the record, ensuring neither party learns the plaintext (Song et al., 2025). We employ tailored PIR protocols across our pipeline based on access patterns (Henzinger et al., 2023b; Liu et al., 2024; Song et al., 2025).

4 PRIVATE KEYWORD-GATED HYBRID RETRIEVAL

4.1 SYSTEM AND THREAT MODEL

We consider a RAG service where the server $\mathcal{S}$ owns a corpus of N documents, represented by a document database $\mathcal{D} = \{d_i\}_{i=1}^N$, an embedding database $\mathcal{E} = \{\mathbf{e}_i\}_{i=1}^N$, and an inverted index $(\mathcal{H}, \mathcal{B})$. The client $\mathcal{C}$ holds a private natural-language query q.

Figure 1 provides an end-to-end overview of GATERAG. The client locally extracts query keywords and embeds the query, while the server preprocesses its corpus into an inverted index and document embeddings. Private retrieval then proceeds through keyword gating, hybrid score computation, and top-k document retrieval. Keyword gating simultaneously restricts expensive secure computation to a reduced candidate set and provides the lexical signal reused in hybrid ranking. The resulting top-k relevant documents remain secret-shared and can be directly passed to a compatible private generation backend.

We assume a static, semi-honest two-party adversary model, where both $\mathcal{C}$ and $\mathcal{S}$ follow the prescribed protocol but may attempt to infer additional information from their respective views. Under this model, GATERAG provides asymmetric bidirectional privacy guarantees: the server's view is independent of the client's query, while server-owned data is protected under an explicitly characterized structural leakage profile.

Semi-Honest Server. GATERAG hides both the query contents and query-dependent access pattern from the server. The server therefore cannot infer which keywords were selected, which database records were accessed, or how many candidates matched.

Algorithm 1 Private Keyword Gating

Input: Client-selected keywords $\mathcal{K}_q$ with importance weights $\{w_t\}_{t\in\mathcal{K}_q}$; server inverted index $(\mathcal{H}, \mathcal{B})$, consisting of keyword header $\mathcal{H}$ and posting blocks $\mathcal{B}$, and embedding database $\mathcal{E}$
Output: Secret-shared candidate embeddings $\{[\![\mathbf{e}_i]\!]\}_{i\in\mathcal{C}_q}$; secret-shared lexical scores $\{[\![\ell_i]\!]\}_{i\in\mathcal{C}_q}$

for each $t \in \mathcal{K}_q$ **do**
    Privately obtain the posting block descriptor from $\mathcal{H}$ via OPRF-based header lookup
end for
Privately retrieve the corresponding posting blocks from $\mathcal{B}$ via PIR
Construct the candidate set $\mathcal{C}_q \leftarrow \bigcup_{t\in\mathcal{K}_q} \mathsf{Posting}(t)$ from the retrieved blocks
for each $i \in \mathcal{C}_q$ **do**
    $\ell_i \leftarrow \sum_{t\in\mathcal{K}_q} w_t\, \mathbb{I}[i \in \mathsf{Posting}(t)]$
end for
Additively secret-share $\{\ell_i\}_{i\in\mathcal{C}_q}$ between $\mathcal{C}$ and $\mathcal{S}$ to obtain $\{[\![\ell_i]\!]\}_{i\in\mathcal{C}_q}$
Retrieve the embeddings indexed by $\mathcal{C}_q$ from $\mathcal{E}$ via PIR-to-share, yielding $\{[\![\mathbf{e}_i]\!]\}_{i\in\mathcal{C}_q}$

Semi-Honest Client. GATERAG prevents the client from obtaining server-owned documents or document embeddings in plaintext. The client does, however, observe limited structural information through the postings associated with its selected keywords. We explicitly characterize this leakage and analyze the cost of exploiting it in Appendix F.

The remainder of this section describes the private retrieval protocol in detail. Section 4.2 presents private keyword gating, while Section 4.3 describes the subsequent secure hybrid scoring and top-k document retrieval over the resulting candidate set.

4.2 PRIVATE KEYWORD GATING

We propose private keyword gating, which uses a small set of query keywords selected by the client to identify a reduced set of lexically relevant documents. The server maintains an inverted index that maps each indexed term to the documents containing it. Conceptually, the client retrieves the postings lists associated with its selected keywords and forms the candidate set, while retaining the resulting keyword matches as lexical evidence for subsequent hybrid ranking. This allows the gating stage to serve both candidate reduction and lexical scoring.

The challenge is to perform this gating without revealing the selected keywords or query-dependent access patterns to the server. Postings lists are variable-sized, and both their locations and retrieval volumes depend on the private query. We address these issues using a blocked inverted-index representation with a fixed retrieval budget, instantiated through three consecutive private retrieval steps: an OPRF-based encrypted header lookup, PIR over posting blocks, and PIR-to-share over candidate embeddings. Algorithm 1 summarizes the resulting protocol.

Keyword selection. The client first assigns an importance weight w_t to each query term t using only the query and public word-frequency statistics, and selects the top-m terms as the query keywords $\mathcal{K}_q$. The keyword budget is capped at $m_{\max} = 5$, which bounds the rate at which the client can expose the structure of the inverted index. This operation is performed entirely locally and requires no interaction with the server. Appendix C details the client-local keyword weighting procedure. The same importance weights are later reused to quantify the lexical relevance of documents matching the selected keywords.

Private retrieval of variable-length postings. A key challenge is the highly nonuniform length of postings lists. Standard PIR operates over fixed-size records, so a naïve representation that stores each postings list as a single record would pad every list to the maximum length in the vocabulary. For long-tailed term frequencies, this makes the cost of retrieving even a short postings list depend on the globally longest one. Instead, we construct the inverted index by representing each postings list as a sequence of fixed-size blocks, following the general approach of block-based representations for variable-length private retrieval (Henry et al., 2013; Cash et al., 2014). Each block contains up to 256 document identifiers, and the length of each postings list is represented by the number of blocks that comprise it. This shifts padding from the entire postings list to the block level, avoiding

vocabulary-wide maximum padding. Appendix D quantifies the overhead of the naïve PIR and the resulting savings from the blocked representation.

Specifically, for each selected keyword $t \in \mathcal{K}_q$, an OPRF-based encrypted header lookup privately resolves t to a compact descriptor specifying the starting block and the number of associated blocks, without revealing t to the server (detailed in Appendix E). The client then uses this descriptor to retrieve the corresponding posting blocks through PIR (Henzinger et al., 2023b). While PIR conceals the identities of the accessed blocks, their number is still revealed to the server, exposing query-dependent volume information.

To eliminate this remaining channel, we fix the retrieval budget so that every execution issues a single PIR batch of exactly P posting blocks, irrespective of the query. Here, P is a public system parameter, and postings exceeding the budget are truncated in decreasing order of w_t, prioritizing the posting blocks associated with higher-weight keywords. When fewer than P blocks are required, the batch is padded with dummy requests. All subsequent candidate retrieval and secure scoring are likewise padded to the fixed capacity induced by P, rather than to the actual number of matched candidates. The server therefore observes an identical batch profile in every execution, and no stage of the protocol carries query-dependent volume information. Section 5.2 analyzes how P governs the resulting accuracy–efficiency tradeoff.

Candidate construction and lexical scoring. After recovering the postings lists for the selected keywords, the client constructs the deduplicated candidate set

$$\mathcal{C}_q = \bigcup_{t \in \mathcal{K}_q} \mathsf{Posting}(t). \tag{1}$$

The identifiers contained in the postings lists are not the original database indices. During preprocessing, the server applies a secret permutation to database identifiers, exposing only the resulting opaque identifiers throughout the retrieval protocol.

For each candidate $i \in \mathcal{C}_q$, the client computes its lexical score by accumulating the importance of the selected keywords that match the candidate:

$$\ell_i = \sum_{t \in \mathcal{K}_q} w_t \, \mathbb{I}[i \in \mathsf{Posting}(t)], \tag{2}$$

where $\mathbb{I}[\cdot]$ denotes the indicator function. Unlike conventional BM25-based lexical scoring (Robertson & Zaragoza, 2009), this score requires no per-document term-frequency retrieval or additional secure lexical computation, as it reuses the keyword matches already obtained during candidate reduction.

The resulting lexical scores are normalized to $[0, 1]$ and secret-shared between the client and server using additive secret sharing for subsequent hybrid score computation. The same keyword evidence used for gating therefore serves directly as the lexical component of the final ranking score, avoiding a separate full-corpus lexical retrieval path.

Candidate embedding retrieval. The client next uses the opaque identifiers in $\mathcal{C}_q$ as private indices into the server's embedding database $\mathcal{E}$. A multi-query PIR-to-share protocol based on Liu et al. (2024) retrieves the corresponding candidate embeddings directly in secret-shared form. The server therefore does not learn which candidate embeddings are accessed, while the client never receives the server-owned embeddings in plaintext.

4.3 Secure Hybrid Score Computation and Private Top-k Document Retrieval

Private keyword gating reduces the search space to the candidate set $\mathcal{C}_q$ and provides secret-shared candidate embeddings and lexical scores. The remaining retrieval operations compute a hybrid relevance score over these candidates and privately retrieve the top-k documents.

Hybrid score computation. Document embeddings are L_2-normalized during preprocessing, while the client locally normalizes the query embedding. Thus, cosine similarity can be evaluated as an inner product. For each candidate $i \in \mathcal{C}_q$, the parties securely compute the semantic score

$$s_i = \langle \mathbf{e}_q, \mathbf{e}_i \rangle = \cos(\mathbf{e}_q, \mathbf{e}_i), \tag{3}$$

using an MPC protocol over additive secret shares, obtaining $[\![s_i]\!]$. The query embedding $\mathbf{e}_q$ remains private to the client, while the candidate embedding is available only as shares $[\![\mathbf{e}_i]\!]$. Neither the candidate embedding nor the semantic score is reconstructed during the computation.

The semantic score is then normalized to $[0, 1]$ and combined with the lexical score obtained from private keyword gating:

$$[\![h_i]\!] = \alpha\,[\![s_i]\!] + (1 - \alpha)\,[\![\ell_i]\!], \tag{4}$$

where $\alpha \in [0, 1]$ is a public hyperparameter. By fusing these signals, the hybrid score inherently captures documents with high joint relevance, and can be computed directly over the secret shares using linear operations. Both semantic scoring and hybrid fusion are performed only over $\mathcal{C}_q$, rather than over the full document corpus.

Top-k document retrieval. Given the shared hybrid scores $[\![h_i]\!]_{i\in\mathcal{C}_q}$, the parties perform GC-based secure top-k selection (Li et al., 2025) to identify the highest-scoring candidates. The resulting opaque identifiers $\mathcal{R}_q^k$ are then used as private indices into the document database $\mathcal{D}$. A PIR-to-share protocol (Song et al., 2025) retrieves the corresponding documents directly as secret shares. Consequently, the server does not learn which documents are accessed, while the client does not reconstruct the retrieved documents in plaintext.

Interface to private generation. GATERAG outputs the top-k relevant documents as secret shares held between the client and server, providing a natural interface to privacy-preserving generation backends. These shares can be directly consumed by MPC-based private LLM inference protocols (Gupta et al., 2024; Zheng et al., 2024; Hou et al., 2026) or converted into homomorphically encrypted (HE) inputs for HE-based protocols using secure conversion techniques (Xu et al., 2025; Moon et al., 2025). Thus, the retrieved documents can be passed to downstream private generation without being reconstructed in plaintext. As our primary contribution lies in private retrieval, evaluation of the subsequent generation backend is beyond the scope of this work.

5 EVALUATION

5.1 EXPERIMENTAL SETUP

Datasets. We evaluate GATERAG on three retrieval benchmarks spanning diverse domains and corpus scales: PubMedQA (Jin et al., 2019) ($N = 0.274$M), HotpotQA (Yang et al., 2018) ($N = 1.795$M), and FEVER (Thorne et al., 2018) ($N = 5.396$M). For each dataset, we randomly sample 1,000 queries for evaluation and a disjoint set of 1,000 queries for validation. All hyperparameters are selected exclusively on the validation set. Dataset details are provided in Appendix G.

Models and metrics. For semantic representation, we use `all-MiniLM-L6-v2` from Sentence-Transformers (Reimers & Gurevych, 2019) to generate 384-dimensional document and query embeddings. For keyword gating, the client locally ranks query terms using our custom importance weights based on `wordfreq` (Speer, 2022) and lexical heuristics, as detailed in Appendix C. We evaluate retrieval accuracy using macro Recall@k with $k = 10$, averaging over all evaluation queries. We use a maximum keyword budget of $m = 5$ and select the hybrid weight α on the validation set to maximize recall, yielding $\alpha = 0.91$, 0.80, and 0.75 for PubMedQA, HotpotQA, and FEVER, respectively.

Baselines. We compare GATERAG against state-of-the-art private RAG systems: RemoteRAG (Cheng et al., 2025), ShieldRAG (Chang et al., 2026b), PIR-RAG (Wang et al., 2025), and Pisces (Liang et al., 2026). For a fair comparison, we evaluate all baselines under configurations that eliminate query-dependent leakage to the server. For the efficiency comparison, we further distinguish systems that expose plaintext document contents to the client (RemoteRAG, ShieldRAG, and PIR-RAG) from those that keep plaintext documents hidden (Pisces and GATERAG). Additional details on baseline implementation and evaluation are provided in Appendix H.

Execution environment. All latency experiments are executed on a single thread of an Intel Xeon Silver 4210R CPU. Reported latency and communication measure the execution time and total bytes exchanged between the client and server.

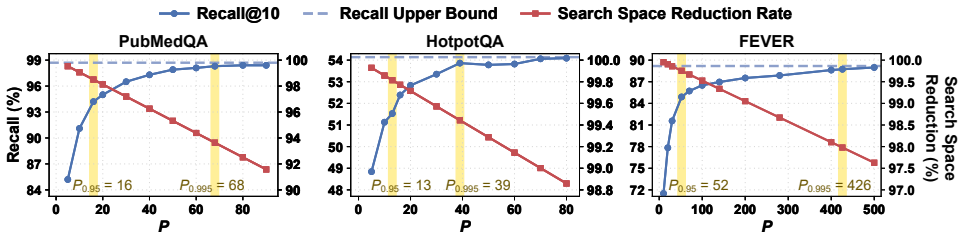


Figure 2: Tradeoff between retrieval accuracy and efficiency on the validation set. The yellow shaded regions mark the smallest P reaching 95% and 99.5% of the recall upper bound (dashed).

5.2 GateRAG Characterization and Design Analysis

We first characterize the accuracy–efficiency tradeoff of GateRAG and then quantify the end-to-end impact of its private keyword gating design.

Accuracy–efficiency tradeoff and operating-point selection. The fixed block PIR budget P directly controls the tradeoff between retrieval quality and efficiency. A larger P allows more posting blocks to contribute to the candidate set and therefore generally improves recall, while increasing the cost of keyword gating and the subsequent secure computation. Importantly, changing P does not affect the query-privacy guarantee.

Figure 2 characterizes this tradeoff on the held-out validation set. For a target quality-retention level τ, we define

$$P_\tau = \min\{P : R(P) \geq \tau R_\infty\}, \tag{5}$$

where $R(P)$ is the recall obtained with budget P, and R_∞ is the recall upper bound without truncation by the fixed block budget. We consider two practically relevant operating points. The balanced configuration uses $\tau = 0.95$, prioritizing candidate reduction while retaining most of the attainable retrieval quality, whereas the quality-oriented configuration uses $\tau = 0.995$ to retain near-maximum recall. The resulting $P_{0.95}/P_{0.995}$ values are $16/68$, $13/39$, and $52/426$ for PubMedQA, HotpotQA, and FEVER, respectively. Unless otherwise stated, subsequent analyses use the balanced configuration. Both operating regimes are reported in the cross-system comparison of Section 5.3.

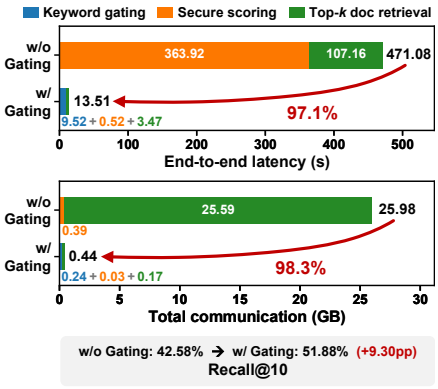


Figure 3: End-to-end impact of keyword gating on HotpotQA.

End-to-end impact of keyword gating. We examine how keyword gating changes the end-to-end cost structure of private retrieval. Figure 3 compares GateRAG against a semantic-only variant that removes keyword gating, which consequently eliminates both candidate reduction and the lexical signal used in hybrid ranking. We use HotpotQA as a representative benchmark and decompose both latency and communication into keyword gating, secure scoring, and top-k document retrieval.

By pruning the search space, our keyword gating drastically reduces the dominant costs of secure scoring and top-k document retrieval. The efficiency gains from this candidate reduction far outweigh the overhead introduced by keyword gating, resulting in a 97.1% reduction in end-to-end latency and a 98.3% reduction in communication. Keyword gating also provides the lexical signal for hybrid ranking, which increases recall from 42.58% to 51.88%. Hence, the substantial efficiency gains are achieved without sacrificing retrieval quality.

5.3 Comparison with Prior Private RAG Systems

Table 2 compares all systems under matched server-side query privacy. Under this privacy requirement, RemoteRAG, ShieldRAG, and Pisces fall back to full-corpus processing and therefore have a single applicable operating point. In contrast, PIR-RAG and GateRAG retain private candidate reduction, so we apply the common 95% and 99.5% quality-retention criteria described in Sec-

Table 2: End-to-end efficiency and recall comparison with prior private RAG systems under matched query privacy. *Comm.* represents the total communication, and best results are shown in **bold**.

	Method	PubMedQA (0.274M)			HotpotQA (1.795M)			FEVER (5.396M)		
		Latency	Comm.	Recall	Latency	Comm.	Recall	Latency	Comm.	Recall
Docs Exposed	RemoteRAG	31.42 h	0.76 GB	93.10 %	188.76 h	2.98 GB	42.58 %	590.84 h	10.87 GB	63.35 %
	ShieldRAG	25.87 h	41.60 GB	93.10 %	145.84 h	270.42 GB	42.58 %	534.32 h	815.03 GB	63.35 %
	PIR-RAG	97.46 s	3.62 GB	93.10 %	384.59 s	13.89 GB	42.58 %	0.39 h	52.02 GB	63.35 %
Docs Hidden	Pisces*	125.58 s	9.50 GB	96.30 %	628.14 s	57.82 GB	51.01 %	0.75 h	195.74 GB	69.03 %
	GateRAG ($P_{0.95}$)	**5.81 s**	**0.26 GB**	93.50 %	**13.51 s**	**0.44 GB**	51.88 %	**218.32 s**	**3.29 GB**	85.54 %
	GateRAG ($P_{0.995}$)	18.38 s	1.00 GB	**98.00 %**	25.35 s	0.81 GB	**53.45 %**	0.35 h	15.57 GB	**89.67 %**

*For Pisces, we assume perfect parallel execution of its semantic and lexical paths, reporting the maximum latency of the two. This assumption is favorable to Pisces.

tion 5.2. For PIR-RAG, both criteria select the same configuration, which is therefore reported as a single row. GATERAG yields distinct balanced and quality-oriented operating points, reported separately as $P_{0.95}$ and $P_{0.995}$.

Retrieval accuracy. Even at the more aggressively reduced $P_{0.95}$ setting, GATERAG achieves higher recall than all semantic-only baselines (RemoteRAG, ShieldRAG, and PIR-RAG) on all three datasets, by 0.40–22.19 percentage points. These results indicate that keyword gating preserves lexical evidence that semantic-only systems fail to capture. GATERAG also outperforms Pisces by 1.70–20.64 percentage points across all three datasets at $P_{0.995}$. At the more efficiency-oriented $P_{0.95}$ setting, it trails Pisces only on PubMedQA by 2.80 percentage points. Whereas Pisces selects documents based on the highest scores from either its semantic or lexical path, GATERAG securely fuses the two signals, allowing it to retrieve documents with high joint relevance that may not rank highly on either path alone. We further analyze this advantage in Appendix I.

Efficiency without plaintext document exposure. Among prior systems, Pisces is the only one that likewise hides plaintext documents from the client. Compared with Pisces, GATERAG reduces latency by 91.9–97.8% and communication by 97.3–99.2% at $P_{0.95}$. When retrieval quality is prioritized ($P_{0.995}$), GATERAG still remains 53.3–96.0% faster while reducing communication by 89.5–98.6%. This advantage stems from their different retrieval structures. Pisces performs computationally heavy semantic and lexical evaluation over the full corpus, whereas GATERAG privately prunes the search space and reuses the gating signal for lexical scoring.

Efficiency with plaintext document exposure. GATERAG also provides superior efficiency compared with systems that expose server-owned documents to the client. Against RemoteRAG and ShieldRAG, even the quality-oriented $P_{0.995}$ configuration is more than three orders of magnitude faster across all three datasets. GATERAG also reduces both latency and communication relative to PIR-RAG, which employs its own private candidate reduction, at both operating points.

Overall, GATERAG is the only evaluated system that combines query-private search-space reduction with protection against plaintext document disclosure. It further provides a tunable operating range from highly efficient retrieval to near-maximum retrieval quality. Additional operating point sweeps, including configurations of prior systems that relax query privacy, are provided in Appendix J.

6 CONCLUSION

In this work, we introduced GATERAG, a private RAG retrieval framework that uses keyword gating to reduce the cost of hybrid retrieval while preserving both query privacy and plaintext-document protection. Its gating stage jointly narrows the candidate set and supplies the lexical evidence used for hybrid scoring. This enables semantic and lexical relevance to be fused over the reduced candidate set without requiring a separate full-corpus lexical computation. Across three benchmarks, GATERAG substantially reduces retrieval costs relative to prior private RAG systems. Compared with the privacy-matched Pisces, the balanced setting achieves up to 97.8% lower latency and 99.2% lower communication. The quality-oriented setting prioritizes retrieval quality, improving recall by up to 20.64 percentage points while preserving substantial efficiency advantages. These results establish GATERAG as an effective framework for efficient and accurate private hybrid retrieval.

AI USE STATEMENT

We used generative AI tools as supporting aids for language translation, grammatical correction, and polishing the readability of author-written manuscript text. We also used them to assist in identifying potentially relevant prior work and in debugging software during implementation. All literature suggested by AI tools was independently verified against the original sources, and all AI-assisted debugging suggestions were reviewed and tested by the authors. Generative AI was not used to originate the core research idea or design the proposed methodology and experiments. We reviewed and take full responsibility for all content in this submission.

ETHICS STATEMENT

This work does not involve human-subject experiments, collection of personal data, or interaction with real users. All experiments use publicly released benchmark datasets and an openly available embedding model under their respective license terms. These resources are used solely for research evaluation, with the original sources appropriately cited. Our work aims to improve privacy-preserving retrieval and does not introduce or release new sensitive or personally identifiable data.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the proposed protocol and experimental methodology in the main paper, with additional implementation details and dataset preprocessing described in the appendices. All experiments use publicly available datasets and models. Our anonymized source code is available at `https://anonymous.4open.science/r/GateRAG_f/`. Hyperparameters and other configuration details required to reproduce the experiments are also provided in the released code.

REFERENCES

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

Atousa Arzanipour, Rouzbeh Behnia, Reza Ebrahimi, and Kaushik Dutta. RAG security and privacy: Formalizing the threat model and attack surface. In *2025 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 1387–1394, 2025.

Elaine Barker. Recommendation for key management: Part 1 – General. NIST Special Publication 800-57 Part 1 Rev. 5, National Institute of Standards and Technology, 2020.

Amina Bassit and Vishnu Boddeti. SecureRAG: End-to-end secure retrieval-augmented generation. In *The Second Workshop on GenAI for Health: Potential, Trust, and Policy Compliance (NeurIPS 2025 Workshop)*, 2025.

Andreea-Elena Bodea, Stephen Meisenbacher, Alexandra Klymenko, and Florian Matthes. SoK: Privacy risks and mitigations in retrieval-augmented generation systems. *arXiv preprint arXiv:2601.03979*, 2026.

Sebastian Bruch, Siyu Gai, and Amir Ingber. An analysis of fusion functions for hybrid retrieval. *ACM Transactions on Information Systems*, 42(1):1–35, 2023.

David Cash, Joseph Jaeger, Stanislaw Jarecki, Charanjit Jutla, Hugo Krawczyk, Marcel-Cătălin Roşu, and Michael Steiner. Dynamic searchable encryption in very-large databases: Data structures and implementation. In *Network and Distributed System Security Symposium (NDSS)*, 2014.

Yuan Chang, Tom H. Luan, Siran Wang, and Yuntao Wang. SafeRAG: Secure cloud-based retrieval-augmented generation for LLM-empowered voice assistants. *IEEE Transactions on Network Science and Engineering*, 13:6211–6224, 2026a.

Yuan Chang, Tom H. Luan, Yuntao Wang, Linkang Du, Chengyi Dong, and Zhou Su. An efficient privacy-preserving retrieval scheme for cloud-based retrieval-augmented generation. *IEEE Transactions on Cognitive Communications and Networking*, 12:9671–9686, 2026b.

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

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

Chanyeol Choi, Jihoon Kwon, Jaeseon Ha, Hojun Choi, Chaewoon Kim, Yongjae Lee, Jy-yong Sohn, and Alejandro Lopez-Lira. FinDER: Financial dataset for question answering and evaluating retrieval-augmented generation. In *Proceedings of the 6th ACM International Conference on AI in Finance*, pp. 638–646, 2025.

Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *Journal of the ACM*, 45(6):965–981, 1998.

Alex Davidson, Armando Faz-Hernandez, Nick Sullivan, and Christopher A. Wood. Oblivious pseudorandom functions (OPRFs) using prime-order groups. RFC 9497, 2023.

Michael J. Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In *Theory of Cryptography Conference (TCC)*, pp. 303–324, 2005.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

Kanav Gupta, Neha Jawalkar, Ananta Mukherjee, Nishanth Chandran, Divya Gupta, Ashish Panwar, and Rahul Sharma. SIGMA: Secure GPT inference with function secret sharing. *Proceedings on Privacy Enhancing Technologies*, 2024(4):61–79, 2024.

Ryan Henry, Yizhou Huang, and Ian Goldberg. One (block) size fits all: PIR and SPIR with variable-length records via multi-block queries. In *Network and Distributed System Security Symposium (NDSS)*, 2013.

Alexandra Henzinger, Emma Dauterman, Henry Corrigan-Gibbs, and Nickolai Zeldovich. Private web search with Tiptoe. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 396–416, 2023a.

Alexandra Henzinger, Matthew M. Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn, and Vinod Vaikuntanathan. One server for the price of two: Simple and fast single-server private information retrieval. In *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 3889–3905, 2023b.

Xiaoyang Hou, Jian Liu, Jingyu Li, Yuhan Li, Jiawen Zhang, Wen-jie Lu, Cheng Hong, and Kui Ren. CipherGPT: Secure two-party GPT inference. *IEEE Transactions on Dependable and Secure Computing*, 23(3):6748–6762, 2026.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. PubMedQA: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2567–2577, 2019.

Hugo Krawczyk and Pasi Eronen. HMAC-based extract-and-expand key derivation function (HKDF). RFC 5869, 2010.

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

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

Zhijun Li, Minghui Xu, Huayi Qi, Wenxuan Yu, Tingchuang Zhang, Qiao Zhang, GuangYong Shang, Zhen Ma, and Xiuzhen Cheng. PRAG: End-to-end privacy-preserving retrieval-augmented generation. *arXiv preprint arXiv:2604.26525*, 2026.

Xiaojian Liang, Lushan Song, Shishuai Du, Weicheng Zhu, Tan Faith, Jun Jie Sim, Haibing Jin, Zhenghao Wu, Yingting Liu, Xin Zhang, Jiangming Yang, and Pu Duan. Pisces: Cryptography-based private retrieval-augmented generation with dual-path retrieval. In *International Conference on Learning Representations*, pp. 61885–61904, 2026.

Jian Liu, Jingyu Li, Di Wu, and Kui Ren. PIRANA: Faster multi-query PIR via constant-weight codes. In *2024 IEEE Symposium on Security and Privacy (SP)*, pp. 4315–4330, 2024.

Priyanka Mandikal and Raymond Mooney. Sparse meets dense: A hybrid approach to enhance scientific document retrieval. *arXiv preprint arXiv:2401.04055*, 2024.

Yulong Ming, Mingyue Wang, Jijia Yang, Jie Xu, Zihan Wu, Cong Wang, and Xiaohua Jia. P^2RAG: Efficient privacy-preserving RAG service supporting arbitrary top-k retrieval. *arXiv preprint arXiv:2603.14778*, 2026.

Jungho Moon, Dongwoo Yoo, Xiaoqian Jiang, and Miran Kim. THOR: Secure transformer inference with homomorphic encryption. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pp. 3765–3779, 2025.

Rathinasamy Muthusami and Kandhasamy Saritha. Hybrid sparse-dense retrieval-augmented generation: An empirical analysis with SPLADE and DPR. *International Journal on Computational Engineering*, 3(1):10–19, 2026.

Nicholas Pipitone and Ghita Houir Alami. LegalBench-RAG: A benchmark for retrieval-augmented generation in the legal domain. *arXiv preprint arXiv:2408.10343*, 2024.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and trends in information retrieval*, 4(1-2):1–174, 2009.

Joseph Salowey, Abhijit Choudhury, and David McGrew. AES Galois counter mode (GCM) cipher suites for TLS. RFC 5288, 2008.

Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

Lushan Song, Qizhi Zhang, Yu Lin, Haoyu Niu, Daode Zhang, Zheng Qu, Weili Han, Jue Hong, Quanwei Cai, and Ye Wu. Suda: An efficient and secure unbalanced data alignment framework for vertical privacy-preserving machine learning. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 7663–7682, 2025.

Robyn Speer. rspeer/wordfreq: v3.0, 2022. URL `https://doi.org/10.5281/zenodo.7199437`.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: A large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 809–819, 2018.

Baiqiang Wang, Qian Lou, Mengxin Zheng, and Dongfang Zhao. PIR-RAG: A system for private information retrieval in retrieval-augmented generation. *arXiv preprint arXiv:2509.21325*, 2025.

Minghui Wang, Pengyuan Xiao, and Mingzhuo Yu. Sparse, dense, or hybrid? Comparing retrieval strategies for biomedical question answering with retrieval-augmented generation. *Journal of Science, Innovation & Social Impact*, 2(2):141–152, 2026.

Moyang Xie, Jiacheng Gao, Yuan Zhang, and Sheng Zhong. CoSecRAG: Efficient vector similarity search with database and query privacy for RAG. *Cryptology ePrint Archive, Paper 2026/1442*, 2026.

Tianshi Xu, Wen-jie Lu, Jiangrui Yu, Yi Chen, Chenqi Lin, Runsheng Wang, and Meng Li. Breaking the layer barrier: Remodeling private transformer inference with hybrid {CKKS} and {MPC}. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 2653–2672, 2025.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, 2018.

Andrew Chi-Chih Yao. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (SFCS 1986)*, pp. 162–167, 1986.

Huanyi Ye, Jiale Guo, Ziyao Liu, and Kwok-Yan Lam. Efficient privacy-preserving retrieval augmented generation with distance-preserving encryption. In *2025 3rd International Conference on Foundation and Large Language Models (FLLM)*, pp. 668–676, 2025.

Xuejiao Zhao, Siyan Liu, Su-Yin Yang, and Chunyan Miao. MedRAG: Enhancing retrieval-augmented generation with knowledge graph-elicited reasoning for healthcare copilot. In *Proceedings of the ACM on Web Conference 2025*, pp. 4442–4457, 2025.

Fei Zheng, Chaochao Chen, Zhongxuan Han, and Xiaolin Zheng. PermLLM: Private inference of large language models within 3 seconds under WAN. *arXiv preprint arXiv:2405.18744*, 2024.

Jinhao Zhou and Jun Wu. Efficient vector-multiplicative privacy-preserving retrieval-augmented generation for large language models. *IEEE Transactions on Dependable and Secure Computing*, 23(3):6947–6963, 2026.

Mingxun Zhou, Elaine Shi, and Giulia Fanti. Pacmann: Efficient private approximate nearest neighbor search. In *International Conference on Learning Representations*, volume 2025, pp. 22791–22811, 2025.

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

Guy Zyskind, Tobin South, and Alex Pentland. Don't forget private retrieval: Distributed private similarity search for large language models. In *Proceedings of the Fifth Workshop on Privacy in Natural Language Processing*, pp. 7–19, 2024.

APPENDIX

A EXTENDED RELATED WORK

This appendix discusses private RAG under alternative trust and ownership models and extends the scope of related work to private similarity search outside the RAG setting.

Non-colluding multi-server protocols. Several private RAG systems distribute the corpus or protocol state across multiple servers, operating under the assumption that the participating servers do not collude with one another (Zyskind et al., 2024; Xie et al., 2026; Ming et al., 2026). While this additional assumption significantly reduces the complexity of secure computation, the client's privacy strictly depends on this non-collusion premise. From a security perspective, systems relying on such strong trust assumptions are less trustworthy than those operating in a single-server setting.

Trusted authorities. Other systems reduce cryptographic overhead by relying on trusted hardware, key centers, or auxiliary authorities (Zhou & Wu, 2026; Chang et al., 2026a). While these approaches offer attractive performance, their security is fundamentally bound to the trusted component or party.

Table 3: Notation used in GATERAG.

Notation	Description
$\mathcal{C}, \mathcal{S}$	Client and server parties
q	Client query
$\mathbf{e}_q$	Embedding of query q
N	Number of documents in the server-held corpus
m	Number of query keywords selected for keyword gating
k	Number of documents returned by retrieval
P	Fixed per-query posting-block retrieval budget for block PIR
α	Weight for semantic–lexical score fusion
$\mathcal{K}_q$	Top-m keywords selected from query q
w_t	Importance weight of query term t
$(\mathcal{H}, \mathcal{B})$	Inverted index used for keyword gating
$\mathcal{H}$	Keyword header containing posting-block descriptors
$\mathcal{B}$	Fixed-size posting blocks of the inverted index
$\mathsf{Posting}(t)$	Postings list associated with term t
$\mathcal{E}$	Database of document embeddings
$\mathcal{D}$	Database of documents
$\mathcal{C}_q$	Candidate document set produced by keyword gating
d_i	i-th document
$\mathbf{e}_i$	Embedding of i-th document
ℓ_i	Lexical score of i-th document
s_i	Semantic score of i-th document
h_i	Hybrid score of i-th document
$\mathcal{R}_q^k$	Identifiers of the selected top-k documents
$\mathbb{I}[\cdot]$	Indicator function
$[\![x]\!]$	Additive secret shares of value x

Client-owned corpora. Another line of work assumes the client owns the corpus and outsources it in an encrypted format (Bassit & Boddeti, 2025; Li et al., 2026; Ye et al., 2025). Under this ownership model, documents can be returned to and decrypted by the client without violating corpus confidentiality, as the client already knows the underlying data. This relaxed confidentiality requirement significantly simplifies private retrieval by allowing unrestricted plaintext access to intermediate results.

Private similarity search outside RAG. Beyond private RAG systems, prior work has also explored private similarity search in more general retrieval settings (Li et al., 2025; Chen et al., 2020; Henzinger et al., 2023a; Zhou et al., 2025; Ahmad et al., 2021; Zhu et al., 2025). These works are technically related to our retrieval pipeline, but differ in the retrieval interface or data-ownership model required by our RAG setting. They generally terminate at retrieved scores or identifiers, or assume public corpora, and thus do not directly provide the retrieval interface required for private RAG.

B NOTATION TABLE

Table 3 lists the notation used throughout the GATERAG protocol.

C CLIENT-LOCAL KEYWORD WEIGHTING

Private keyword gating requires the client to rank query terms before accessing the server's inverted index. To avoid relying on corpus-dependent statistics, we compute all term-importance weights locally using only the query string and public word-frequency statistics. Thus, keyword selection requires no access to the server corpus and reveals no corpus statistics to the client.

Table 4: Overhead of a naïve PIR versus the blocked representation used in GATERAG for variable-length postings on HotpotQA. *Storage* indicates the size of the inverted index.

Metric	Naïve	Blocked
Latency (s)	274.10	0.55
Communication (MB)	29.16	1.31
Storage (GB)	510.62	0.57

For each query term t, we first compute a rarity score from its public English Zipf frequency provided by `wordfreq` (Speer, 2022):

$$S(t) = \max(0,\, 8 - \mathrm{ZipfFreq}(t))\,. \tag{6}$$

Consequently, uncommon terms receive larger scores than frequently used terms. $\mathrm{ZipfFreq}(t)$ is obtained from a public word-frequency resource and is independent of the server corpus.

We additionally assign small query-local bonuses to terms that are likely to carry discriminative information. Let $\mathbb{I}_{\mathrm{num}}(t)$ indicate that the term contains a number, $\mathbb{I}_{\mathrm{case}}(t)$ indicate that it is either an uppercase acronym or a capitalized word, and $\mathbb{I}_{\mathrm{long}}(t)$ indicate that its length is at least eight characters. The final importance weight is

$$w_t = S(t) + \beta_{\mathrm{num}}\,\mathbb{I}_{\mathrm{num}}(t) + \beta_{\mathrm{case}}\,\mathbb{I}_{\mathrm{case}}(t) + \beta_{\mathrm{long}}\,\mathbb{I}_{\mathrm{long}}(t), \tag{7}$$

where we use $\beta_{\mathrm{num}} = 0.5$, $\beta_{\mathrm{case}} = 0.5$, and $\beta_{\mathrm{long}} = 0.2$ throughout our experiments.

The client ranks the query terms in descending order of w_t and selects the top-m terms as query keywords. The entire importance computation itself is performed locally and requires no interaction with the server.

D Cost of Naïvely Padded PIR for Variable-Length Postings

Table 4 quantifies the overhead of a naïve PIR representation for variable-length postings. Such a construction pads every inverted-index row to the maximum postings-list length, resulting in substantial storage, latency, and communication overheads. In contrast, the blocked representation used in GATERAG stores postings as fixed-size blocks and retrieves the relevant blocks through an OPRF-based header lookup followed by block PIR. On HotpotQA, this representation reduces latency, communication, and storage by 99.8%, 95.5%, and 99.9%, respectively, compared with the naïvely padded representation.

E OPRF-Based Encrypted Header Lookup

This section details the OPRF-based encrypted header lookup used in private keyword gating.

E.1 OPRF-Based Keyword Lookup

An oblivious pseudorandom function (OPRF) allows a server holding a secret key K and a client holding an input t to evaluate

$$z_t = \mathrm{OPRF}_K(t) \tag{8}$$

such that the client obtains z_t without revealing t to the server (Freedman et al., 2005). For a fixed server key, the same input produces the same pseudorandom output.

Following the OPRF-based keyword-search approach of Freedman et al. (2005), we use z_t as the secret material for locating and decrypting the corresponding header entry. Specifically, a key-derivation function (KDF) expands z_t into multiple pseudorandom values with different purposes: a cache label used to locate the corresponding entry and cryptographic material used to encrypt and decrypt that entry. The header is protected using an authenticated-encryption scheme, which provides both confidentiality and integrity.

In our implementation, we instantiate these components with HKDF-SHA256 (Krawczyk & Eronen, 2010), AES-128-GCM (Salowey et al., 2008), and the OPRF using the `ristretto255-SHA512` ciphersuite (Davidson et al., 2023). Together, these primitives target 128-bit classical security.

Consequently, after completing the OPRF evaluation for a keyword, the client can independently derive the same cache label and decryption material, locate the corresponding encrypted entry, and decrypt it locally. Without the corresponding OPRF output, the remaining cache entries appear under unrelated pseudorandom labels and encryption keys.

E.2 Encrypted Header Cache

For each indexed keyword t, the server constructs a fixed-size header

$$H_t = (a_t, b_t), \tag{9}$$

where a_t is the starting row of its posting blocks and b_t is the number of corresponding blocks. Thus, the postings for t occupy rows $a_t, \ldots, a_t + b_t - 1$ in the posting-block storage.

During offline preprocessing, the server evaluates the OPRF for every indexed keyword to obtain z_t. It then applies HKDF-SHA256 to z_t to derive a cache label L_t and the keying material required by AES-128-GCM. The server encrypts H_t with AES-128-GCM and stores the resulting ciphertext under L_t. The complete encrypted cache, containing neither plaintext keywords nor plaintext header metadata, is distributed to the client before retrieval.

E.3 Header Lookup

For each selected query keyword t, the client and server execute the OPRF protocol, after which the client obtains z_t without revealing t to the server. The client derives the same cache label and encryption material from z_t, locally searches the encrypted cache for L_t, and decrypts the matching entry to recover (a_t, b_t). These values directly specify the posting blocks requested in the subsequent PIR stage.

Importantly, the server does not learn the queried keywords or the subsequently accessed cache entries. Conversely, the client can decrypt a header only after obtaining the corresponding OPRF output through the protocol, and therefore the encrypted cache does not expose the remaining headers in plaintext.

F Analysis of Server-Data Confidentiality

GateRAG keeps server-owned documents and embeddings cryptographically hidden throughout retrieval, while intentionally revealing limited structural information during private keyword gating. In this section, we explicitly characterize this leakage and analyze the extent to which it can be exploited to extract information about the server database.

Our formal threat model remains the semi-honest model described in Section 4.1. For this analysis, however, we consider a stronger adaptive client that deliberately chooses its queries across repeated executions to maximize the information learned about the server database. This provides a conservative analysis beyond the one-shot semi-honest setting.

Explicit client-side leakage. For analysis, we abstract the server's logical inverted index as a binary term–document incidence matrix. Let $\mathcal{V}$ denote the indexed vocabulary, with $V = |\mathcal{V}|$, and define

$$A \in \{0,1\}^{V \times N}, \qquad A_{t,i} = \mathbb{I}[t \in d_i]. \tag{10}$$

This matrix is only an analytical abstraction of the logical inverted index, as the actual GateRAG index is implemented using OPRF-protected headers and fixed-size posting blocks.

During preprocessing, the server applies a secret permutation π to the document identifiers. For each selected keyword $t \in \mathcal{K}_q$, the client obtains the corresponding header metadata H_t and retrieves the support of the corresponding row of A over permuted identifiers:

$$\mathrm{Posting}(t) = \{\pi(i) : A_{t,i} = 1\}. \tag{11}$$

Accordingly, the structural information intentionally revealed to the client in private keyword gating can be expressed as

$$\mathcal{L}_C(q) = \{(t, H_t, \mathrm{Posting}(t)) : t \in \mathcal{K}_q\}. \tag{12}$$

Table 5: Attacker-favorable lower bounds on the cost of exposing every vocabulary row of the server inverted index in GATERAG, at the capped keyword budget $m_{\max} = 5$.

Dataset	Vocabulary V	$Q_{\min}^{\text{GATERAG}}$	Computation	Communication
PubMedQA	256,624	51,325	82.8 h	13.3 TB
HotpotQA	508,999	101,800	382.0 h	44.8 TB
FEVER	2,850,781	570,157	34,577.0 h	1.88 PB

For a conservative upper-bound analysis, we grant the client the complete posting support of every selected keyword, even though the fixed retrieval budget may truncate the postings observed in an actual execution. Importantly, GATERAG does not reveal the plaintext contents or embeddings of the matching documents, their within-document term frequencies, or token positions.

$\mathrm{Posting}(t)$ can be completed into the corresponding binary row of A. Since each execution exposes at most m rows of A, the keyword budget directly governs the rate at which the index structure can be revealed. We fix $m_{\max} = 5$ as a server-enforced system parameter to bound the rate of structural leakage. Each execution comprises exactly m OPRF evaluations, and the server rejects requests exceeding this cap. All experiments in this work operate at this cap, with $m = 5$.

Resistance to adaptive inverted-index extraction. We first consider the strongest extraction strategy available through the prescribed interface. We grant the attacker the ability to force any desired indexed term into the selected keyword set, to select previously unobserved terms in every execution, and to maintain a fixed identifier permutation across all executions. These assumptions deliberately favor the attacker.

Proposition 1 (Query lower bound for complete index extraction). Suppose that each protocol execution reveals postings for at most m distinct indexed terms. Any client that recovers the complete binary incidence matrix A under a fixed document permutation requires at least

$$Q_{\min}^{\text{GATERAG}} = \left\lceil \frac{V}{m} \right\rceil \tag{13}$$

protocol executions.

Proof. Each execution exposes at most m rows of the V-row incidence matrix A. After Q executions, the attacker can therefore have observed at most mQ distinct rows. Complete recovery requires $mQ \geq V$, which directly yields Equation 13. As this is an attacker-favorable necessary condition, it should therefore be interpreted as a lower bound rather than an estimate of the number of executions required by a realistic attack.

Table 5 instantiates this bound at the capped budget $m = m_{\max} = 5$. Since the bound decreases monotonically in m, operating at the cap yields the most attacker-favorable instantiation the protocol admits. Even so, exhaustive extraction of the complete binary index requires more than 51K protocol executions for PubMedQA, 101K for HotpotQA, and 570K for FEVER. As every execution carries an identical fixed cost, these counts translate directly into the aggregate burden such an attack must impose, which the table reports as the computation it induces and the total communication it must transfer. Recovering the complete binary incidence matrix for FEVER, for instance, requires inducing roughly 34.6K hours of computation and transferring 1.88 PB. Costs of this magnitude severely bottleneck information extraction.

From index extraction to plaintext reconstruction. Reconstructing the complete binary inverted index is not equivalent to reconstructing the underlying plaintext document corpus. Even if an attacker obtained the entire incidence matrix A, it would learn only which indexed terms occur in each pseudonymous document.

GATERAG does not reveal the exact document length, within-document term multiplicities, token positions, sentence structure, punctuation, or terms removed during preprocessing. Consequently, many distinct plaintext documents induce exactly the same binary posting-membership representation. Therefore, even complete knowledge of A does not uniquely determine the server's plaintext document corpus. The mapping from plaintext documents to GATERAG's client-visible structural leakage is inherently many-to-one, establishing non-identifiability of exact plaintext reconstruction from the protocol leakage alone.

Table 6: Attacker-favorable comparison of the number of protocol executions required to expose lexical information for every indexed vocabulary term.

Dataset	$Q_{\min}^{\text{GATERAG}}$	$L_{\max}$	$Q_{\min}^{\text{Pisces}}$	Ratio
PubMedQA	51,325	135	1,901	27.0×
HotpotQA	101,800	141	3,610	28.2×
FEVER	570,157	117	24,366	23.4×

The non-identifiability result applies to exact plaintext reconstruction. With sufficiently strong external corpora or language priors, an attacker may still infer topics or plausible text fragments from accumulated keyword co-occurrences. Such auxiliary-information attacks are not ruled out by our leakage model.

Comparison with prior private RAG leakage. The structural leakage of GATERAG is substantially narrower than the server-data exposure of the prior private RAG systems considered in this work. RemoteRAG (Cheng et al., 2025), ShieldRAG (Chang et al., 2026b), and PIR-RAG (Wang et al., 2025) disclose server documents to the client in plaintext, whereas GATERAG keeps both documents and document embeddings hidden throughout retrieval.

Pisces (Liang et al., 2026) avoids plaintext-document disclosure but exposes substantially richer lexical information than ours. For each queried token, the client obtains per-document term-frequency values over the corpus. Such a term-frequency value strictly contains the corresponding binary membership information, since

$$A_{t,i} = \mathbb{I}\left[\mathrm{TF}(t, d_i) > 0\right]. \tag{14}$$

Thus, binary posting membership can be derived from term frequency, whereas term frequency cannot be reconstructed from binary membership. Moreover, Pisces exposes this information for all query tokens and over consistently identified documents, while GATERAG exposes only binary support for the selected m keywords over server-permuted document identifiers. Our leakage is therefore a strict subset of Pisces's.

To illustrate the difference in extraction rate, we use the maximum observed query lengths in our evaluation data as a practical per-query token budget for Pisces (135, 141, and 117 tokens for PubMedQA, HotpotQA, and FEVER, respectively). Under an attacker-favorable assumption that every query contains only previously unseen indexed terms, the number of executions required to expose lexical statistics for all V vocabulary terms is

$$Q_{\min}^{\text{Pisces}} = \left\lceil \frac{V}{L_{\max}} \right\rceil, \tag{15}$$

where $L_{\max}$ denotes the corresponding token budget.

Table 6 compares these attacker-favorable extraction rates. Under the stated query-length budget, Pisces requires approximately 23–28× fewer executions to cover the vocabulary, while each recovered vocabulary row contains exact per-document term frequencies rather than binary membership.

The two bounds measure different leakage objects. $Q_{\min}^{\text{GATERAG}}$ corresponds to extraction of the complete binary incidence matrix, whereas $Q_{\min}^{\text{Pisces}}$ corresponds to the richer term-frequency matrix. The comparison therefore reflects both a lower extraction rate and a coarser per-term leakage profile for GATERAG.

Practical defensibility. The preceding extraction analysis deliberately assumes that opaque document identifiers remain linkable across an unlimited number of executions and that the attacker already knows the complete indexed vocabulary and can successfully force arbitrary target terms into the selected keyword set. These assumptions are stronger than those required by our formal threat model and favor the attacker. Since the bound is derived under them, practical restriction on these capabilities for semi-honest clients drives the number of executions required for extraction far above the values in Table 5.

In deployment, cross-query accumulation can be further limited by periodically refreshing the server-secret permutation of document identifiers and rebuilding the corresponding private index. Such a refresh removes direct identifier continuity across epochs and forces the attacker to solve an

additional record-linkage problem before combining observations across epochs. Refreshing after every T executions therefore caps the coverage attainable within any single epoch at mT of the V vocabulary rows, irrespective of how much computation the attacker can afford. Authentication, per-client rate limiting, and abuse monitoring systems provide complementary restrictions against large-scale automated extraction.

Overall, GATERAG provides cryptographic confidentiality for plaintext documents and embeddings under the stated structural leakage profile. This leakage is narrower and slower to exploit than that of the prior private RAG systems, bounded by extraction costs that place large-scale corpus recovery beyond practical reach.

G DATASET DETAILS

To evaluate the retrieval accuracy and efficiency of GATERAG across diverse scales and domains, we utilize three standard knowledge-intensive NLP datasets: PubMedQA (Jin et al., 2019), HotpotQA (Yang et al., 2018), and FEVER (Thorne et al., 2018). These datasets represent a wide spectrum of corpus sizes and retrieval characteristics, providing a comprehensive benchmark for our private RAG pipeline.

PubMedQA is a domain-specific biomedical dataset focused on answering research questions using PubMed abstracts. The corpus contains 273,518 documents, and each query is associated with exactly one ground-truth relevant (gold) document.

HotpotQA requires multi-hop reasoning over Wikipedia articles. We used the training corpus, comprising 482,021 documents, and split them into individual sentences for retrieval, yielding 1,795,146 sentence-level chunks. This fine-grained chunking requires the retrieval system to identify precise supporting evidence rather than relying on broader document-level context.

FEVER is a large-scale dataset designed to verify claims against Wikipedia. It serves as our largest benchmark, featuring a massive corpus of 5,396,106 documents. This dataset reflects a typical large-scale open-domain retrieval environment.

H BASELINE IMPLEMENTATION AND EVALUATION

H.1 BASELINE IMPLEMENTATION AND PRIVACY CONFIGURATIONS

We implement each baseline following the protocol described in its original paper. All cryptographic components are instantiated to target 128-bit classical security, a standard modern security target consistent with NIST guidance (Barker, 2020). Below, we summarize the candidate-reduction parameter of each baseline and the configuration used to match the server-side query privacy of GATERAG.

RemoteRAG (Cheng et al., 2025). RemoteRAG controls candidate reduction through the perturbation distance r, which determines the search range k' around the perturbed query embedding. Increasing r provides stronger query concealment but expands the required search range. Its privacy-conscious endpoint corresponds to $r \to \infty$ (equivalently, $\epsilon \to 0$ and $k' = N$), where cryptographic retrieval is performed over the full corpus. We use this endpoint in the privacy-matched comparison.

ShieldRAG (Chang et al., 2026b). ShieldRAG reduces the search space by selecting query-relevant LSH buckets, with w controlling the bucket search range. Accessing only selected buckets reveals query-dependent bucket information to the server. We therefore use the full-bucket endpoint, $w \to \infty$, for the privacy-matched comparison, which removes this candidate reduction and its associated bucket-access leakage.

PIR-RAG (Wang et al., 2025). PIR-RAG partitions the corpus into n semantic clusters and privately retrieves the cluster selected by the client. Increasing n produces finer clusters and therefore reduces the number of documents retrieved per query. Since PIR hides the selected cluster from the server, this candidate reduction does not introduce query-dependent access leakage for any evaluated n. We therefore select n on the validation set using the same quality-retention criterion as GATERAG.

Pisces (Liang et al., 2026). Pisces uses a Hamming-distance threshold t in its coarse semantic filter, retaining documents whose $\mathcal{L}$-bit SimHash representations lie within distance t of the query. The resulting candidate set is available to the server, so candidate reduction through this filter introduces query-dependent leakage. For the privacy-matched comparison, we disable this reduction by setting $t = \mathcal{L} = 32$, corresponding to the fine-only semantic configuration and full-corpus secure scoring (we use $\mathcal{L} = 32$, following the original paper).

H.2 Evaluation Methodology

Matched evaluation conditions. All baselines are evaluated under the same experimental conditions as GateRAG, including the dataset splits, embedding model, retrieval depth, evaluation metric, and hardware environment specified in Section 5.1. The same evaluation queries are used for every method within each dataset to ensure a consistent cross-system comparison.

Evaluation of long-running baselines. The privacy-matched full-corpus configurations of RemoteRAG and ShieldRAG require tens to hundreds of hours per query, making execution over all 1,000 evaluation queries impractical. We therefore measure their latency and communication on three randomly sampled queries and report the average per-query cost. In these configurations, both methods process the entire corpus for every query, so their dominant cryptographic operation count and communication are query-independent for a fixed corpus.

Retrieval accuracy is evaluated over all 1,000 queries for every system. For RemoteRAG and ShieldRAG, we obtain retrieval results using functionally equivalent plaintext executions of their retrieval procedures, while their cryptographic implementations are used to measure latency and communication. This allows retrieval quality to be evaluated over the complete query set without requiring full cryptographic execution for every query.

I Analysis of Retrieval Accuracy

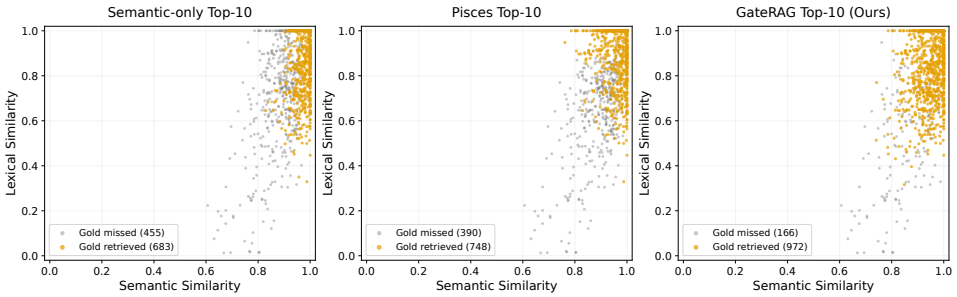
To better understand the source of the retrieval accuracy improvements achieved by GateRAG, we visualize the retrieval distribution of gold documents on the FEVER dataset. Figure 4 plots the distribution of missed and retrieved gold documents based on their normalized semantic similarity (x-axis) and lexical similarity (y-axis) across different retrieval strategies.

As shown in Figure 4(a), the semantic-only baseline (e.g., RemoteRAG, ShieldRAG, PIR-RAG) exclusively retrieves documents that exhibit high semantic similarity, failing to capture documents that rely heavily on exact keyword matches. Pisces mitigates this by maintaining a dual-path architecture (Liang et al., 2026). However, as it evaluates the semantic and lexical paths completely in isolation and simply takes the union of the top-scored documents from each, it primarily captures documents located at the extreme edges with either very high semantic or very high lexical similarity. Consequently, it misses a critical region of documents that possess moderate scores in both metrics. By contrast, GateRAG inherently supports hybrid score fusion securely. Since candidate documents are evaluated based on a unified score combining both lexical and semantic relevance, GateRAG successfully captures documents that neither path alone could identify.

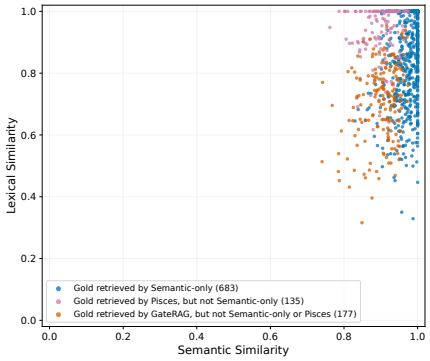
This advantage is most clearly illustrated in the combined scatter plot in Figure 4(b). The orange dots represent the 177 gold documents retrieved only by GateRAG, which were completely missed by both the semantic-only baseline and Pisces. These dots are distinctly clustered in the diagonal "middle" region, representing documents with sufficiently high joint relevance. These results show that our keyword gating and hybrid fusion mechanism is not merely an efficiency optimization, but a structurally superior approach for capturing complex retrieval signals.

J Prior Work Comparison Across Operating Points

Figure 5 extends the privacy-matched comparison in Section 5.3 to a broader set of operating points on FEVER, our largest benchmark. Using the candidate-reduction parameters defined in Appendix H.1, we report representative configurations spanning the practically relevant accuracy–efficiency range of each baseline: $r \in \{0, 0.1, 0.2, \infty\}$ for RemoteRAG, $w \in \{10, 20, 50, \infty\}$ for



(a) Retrieval distribution of individual methods



(b) Combined distribution

Figure 4: Scatter plots illustrating the distribution of gold documents in the FEVER dataset based on their semantic and lexical similarities. **(a)** The retrieved (yellow) and missed (grey) gold documents for the Semantic-only baseline, Pisces, and GATERAG ($P = P_{0.995} = 426$). **(b)** An overlay of these distributions, explicitly highlighting the documents retrieved exclusively by GATERAG (orange dots), which exhibit high joint relevance but lack extreme scores in any single metric.

ShieldRAG, $n \in \{1, 2, 3, 9\}$ for PIR-RAG, and $t \in \{8, 12, 16, 32\}$ for Pisces. We likewise show representative GATERAG budgets $P \in \{20, 52, 100, 426\}$. Marker shape indicates the type of query-dependent information exposed to the server, and dashed curves denote systems that reveal plaintext documents to the client.

The advantage of GATERAG persists even when prior systems are allowed to relax query privacy for efficiency. For Pisces, reducing t lowers recall from 69.03% to 34.37% while leaving latency essentially unchanged. Even its lowest-cost plotted configuration remains slower and more communication-intensive than every plotted GATERAG configuration. Similarly, increasingly aggressive candidate reduction in ShieldRAG substantially degrades recall while retaining very high computation and communication costs.

Among the plotted baseline configurations, RemoteRAG is the only system that undercuts GATERAG on either cost axis under relaxed query privacy. At $r = 0$, it requires only 19.55 s and 0.32 MB, but exposes the unperturbed raw query embedding to the server (Cheng et al., 2025). As r increases to provide stronger query concealment, its latency rises by over five orders of magnitude while recall remains fixed at 63.35%. Thus, the broader operating-point comparison confirms that GATERAG maintains a favorable accuracy–efficiency tradeoff across both privacy-matched and privacy-relaxed baseline settings.

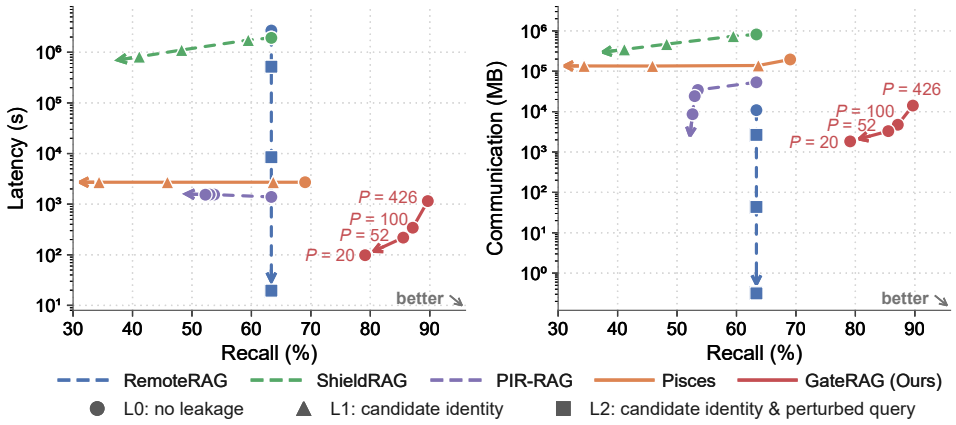


Figure 5: Comparison of representative operating points of prior private RAG systems on FEVER. Marker shape denotes the query leakage level of each configuration, while dashed curves indicate systems that reveal documents to the client in plaintext. Arrows indicate the direction of stronger candidate reduction along each system's parameter sweep. GATERAG is annotated with its block PIR budget P.