Pisces: Cryptography-based Private Retrieval-Augmented Generation with Dual-Path Retrieval

Anonymous authors
Paper under double-blind review

ABSTRACT

Retrieval-augmented generation (RAG) enhances the response quality of large language models (LLMs) when handling domain-specific tasks, yet raises significant privacy concerns. This is because both the user query and documents within the knowledge base often contain sensitive or confidential information. To address these concerns, we propose Pisces, the first practical cryptography-based RAG framework that supports dual-path retrieval, while protecting both the query and documents. Along the semantic retrieval path, we reduce computation and communication overhead by leveraging a coarse-to-fine strategy. Specifically, a novel oblivious filter is used to privately select a candidate set of documents to reduce the scale of subsequent cosine similarity computations. For the lexical retrieval path, to reduce the overhead of repeatedly invoking labeled PSI, we implement a multi-instance labeled PSI protocol to compute term frequencies for BM25 scoring in a single execution. Pisces can also be integrated with existing privacy-preserving LLM inference frameworks to achieve end-to-end privacy. Experiments demonstrate that Pisces achieves retrieval accuracy comparable to the plaintext baselines, within a 1.14% margin.

1 INTRODUCTION

Although large language models (LLMs) (Achiam et al., 2023; Liu et al., 2024) have achieved remarkable success in natural language processing tasks, they still exhibit significant limitations in domain-specific tasks (e.g., healthcare diagnostics). In particular, LLMs may produce hallucinations due to a lack of domain-specific knowledge (Huang et al., 2025; Li et al., 2024a). To mitigate these limitations, retrieval-augmented generation (RAG) (Lewis et al., 2020; Gao et al., 2023; Jiang et al., 2023) has emerged as a promising paradigm. It enhances the response quality of LLMs by retrieving relevant documents from external knowledge bases and integrating the query with these documents.

However, there are several significant privacy concerns (Huang et al., 2023; Zeng et al., 2024) during the RAG retrieval process. The query may contain sensitive and personal information, such as symptoms or genetic profiles, which the user wishes to be hidden from the knowledge base. The knowledge base, on the other hand, contains confidential and proprietary data, such as patient records, that should not be leaked in accordance with data privacy regulations like GDPR (Parliament & of the Council of the European Union), PIPL (Congress, a), and HIPAA (Congress, b).

Prior works (Grislain, 2025; He et al., 2025; Cheng et al., 2025; Yao & Li, 2025) primarily apply differential privacy (DP) to address privacy concerns about the RAG retrieval process, as listed in Table 1. Nevertheless, there are practical limitations to these works. Firstly, they consider only semantic retrieval, whereas a dual-path retrieval, such as a combination of semantic and lexical retrieval, is known to perform better (Kuzi et al., 2020). The noise introduced by DP disrupts exact term matching, making lexical retrieval difficult to support. Secondly, these works only ensure the privacy of either the query or the documents. As discussed above, it is essential that both the query and documents be protected.

In this paper, we ask: *How can we maintain retrieval performance while ensuring privacy for both the query and documents during retrieval?*

To this end, we propose Pisces, a cryptography-based private RAG framework that supports dual-path retrieval, while protecting the privacy of both the query and documents. Unlike approaches that merely combine existing components, Pisces introduces customized cryptographic protocols

Table 1: Comparison with prior works.

Framework	Retrieval Path		Privacy		Mechanism
	Semantic	Lexical	Query	Documents	
DP-RAG (Grislain, 2025)	✓	✗	✗	✓	DP
LPRAG (He et al., 2025)	✓	✗	✗	✓	DP
RemoteRAG (Cheng et al., 2024)	✓	✗	✓	✗	DP, Cryptography
(Yao & Li, 2025)	✓	✗	✓	✓	DP
`Pisces` (Ours)	✓	✓	✓	✓	Cryptography

tailored to the specific requirements of semantic and lexical retrieval paths, achieving significant improvements in both privacy and efficiency. Specifically, along the embedding-based semantic retrieval path, we propose a coarse-to-fine strategy to reduce computation and communication complexity when dealing with large-scale knowledge bases. Concretely, we first designed a novel oblivious filter over Hamming distance to privately select a candidate set of documents to significantly reduce the scale of potential matching documents. Then, the cosine similarities are computed between the query and the candidates with secure multi-party computation (MPC). For the lexical retrieval path, we design a multi-instance labeled PSI protocol that obtains all necessary term frequencies for best matching 25 (BM25) scoring in a single execution, avoiding the cost of repeated labeled PSI invocations. BM25 scoring is subsequently performed under MPC. Notably, `Pisces` can be seamlessly integrated with existing privacy-preserving LLM inference frameworks, enabling end-to-end private retrieval and generation. `Pisces` provides strong privacy guarantees for both the query and documents while maintaining high retrieval performance, offering a practical solution for privacy-sensitive RAG applications.

Our contributions are summarized as follows:

- We propose the first cryptography-based RAG retrieval framework with dual-path retrieval, while ensuring privacy for both the query and documents.
- We propose a coarse-to-fine strategy for the semantic retrieval path with an oblivious filter to reduce computation and communication complexity.
- We first leverage an efficient multi-instance labeled PSI protocol for the lexical retrieval path to reduce computation overhead.

We conducted comprehensive experiments to evaluate the performance of `Pisces`. For accuracy, the results show that `Pisces` achieves retrieval accuracy comparable to plaintext baselines over the ground-truth of the dataset, within a 1.14% margin. At the same time, we observe that combining semantic and lexical paths significantly improves retrieval accuracy. For efficiency, the experiments demonstrate that our coarse-to-fine strategy saves retrieval time by 41.21%, reduces upload and download overhead by 68.77% compared to the fine-only strategy on the large-scale dataset. Additionally, our proposed multi-instance labeled PSI outperforms state-of-the-art labeled PSI protocol (Yang et al., 2024), achieving 496.03× speedup in runtime, and reducing upload and download overhead by 70733× and 2.84×, respectively. Overall, `Pisces` is practical in both accuracy and efficiency.

2 PRELIMINARIES

In this work, we use a variety of cryptographic primitives to achieve a private RAG retrieval process. Below we briefly summarize each primitive, and further details can be found in Appendix B.1.

- **Secure Multi-Party Computation** (Ma et al., 2023). A cryptographic technology that enables multiple mutually distrustful parties to cooperatively compute a predefined function while keeping their data private.
- **Secret Sharing** (Keller, 2020). A critical primitive of MPC, that breaks a secret value into multiple shares held by different parties. The secret value can only be reconstructed when a sufficient number of shares are combined.

- **Labeled Private Set Intersection** (Chen et al., 2018). PSI (Jarecki & Liu, 2010) allows two parties to learn the intersection of their sets without revealing any information outside the intersection. Labeled PSI extends the traditional PSI by returning the label that is associated with each element in the intersection.
- **Oblivious Pseudorandom Function (OPRF)** (Naor et al., 1999). Enables two parties to jointly compute a pseudorandom function such that one party learns the output, while the other learns nothing about the input or output.
- **Oblivious Key-Value Store (OKVS)** (Garimella et al., 2021). A data structure that encodes a set of key-value pairs into a compact representation while preserving the privacy of both keys and values.
- **Batch PIR-to-Share** (Song et al., 2025). A cryptographic primitive that enables a client to privately retrieve the values corresponding to its queries from the server. After the execution, both parties obtain the secret shares of the retrieved values.

Additionally, we provide detailed descriptions of the semantic similarity and BM25 for lexical retrieval in Appendix B.2 and Appendix B.3, respectively.

3 Proposed Method

3.1 Overview

`Pisces` involves two parties: a server $\mathcal{S}$, who holds a sensitive knowledge base (a large corpus of textual documents D), and a user $\mathcal{C}$, who holds a private query Q. `Pisces` ensures that neither party learns the other's sensitive information during the RAG retrieval process in both retrieval paths (semantic and lexical).

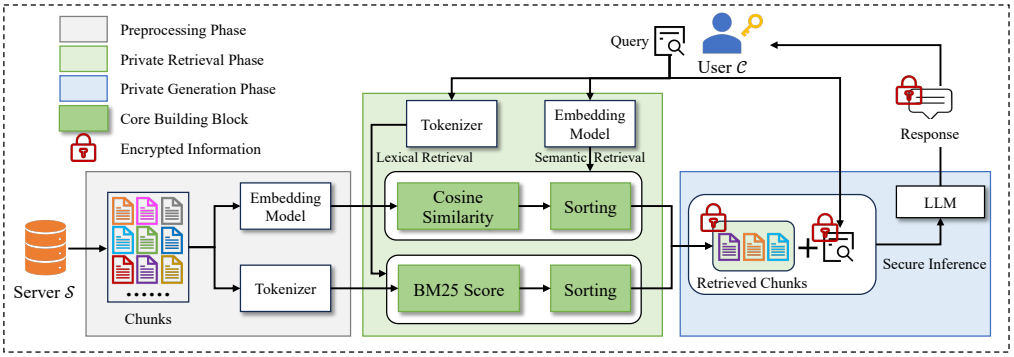

Figure 1: Overview of our proposed `Pisces`.

As shown in Figure 1, the whole process of `Pisces` consists of three phases:

Phase 1: Preprocessing Phase. In this phase, $\mathcal{S}$ preprocesses its private document corpus D for efficient retrieval.

- **Document Chunking.** $\mathcal{S}$ break down D into N smaller chunks of text, i.e. $D = \{c_1, c_2, \ldots, c_N\}$.
- **Vector Embedding.** $\mathcal{S}$ encodes each chunk c_i $(i \in [1,N])$ into vector representations using an embedding model, resulting in $D^v = \{I_i;\ \mathbf{v}_i;\ c_i\}_{i\in[1,N]}$, where I_i and $\mathbf{v}_i$ are the index and vector representation corresponding to the chunk c_i, respectively.
- **Tokenization & Term Frequencies.** $\mathcal{S}$ tokenizes each chunk c_i $(i \in [1,N])$ with a tokenizer and computes the term frequencies, resulting in $D^t = \left\{I_i; \{w_{i,l} : tf_{i,l}\}_{l\in[1,m_i]}; c_i\right\}_{i\in[1,N]}$, where m_i is the total number of unique tokens of c_i, $w_{i,l}$ is the l-th token in the chunk c_i and $tf_{i,l}$ is its term frequency.

Phase 2: Private Retrieval Phase. In this phase, $\mathcal{S}$ interacts with $\mathcal{C}$ to retrieve the relevant chunks for the query Q with privacy preservation.

- **Query's Processing.** $\mathcal{C}$ encodes its query Q into vector representations $\mathbf{q}$ and tokenizes Q to n tokens, i.e. $Q^t = \{q_1, q_2, \ldots, q_n\}$, utilizing the same embedding model and tokenizer applied during the preprocessing phase.
- **Private Semantic Similarity.** $\mathcal{S}$ and $\mathcal{C}$ invoke the private semantic similarity protocol $\prod_{\text{PrivateSS}}$ (Protocol 1), where $\mathcal{S}$ inputs D^v and $\mathcal{C}$ inputs $\mathbf{q}$. After execution, $\mathcal{S}$ obtains the encrypted top-K chunks with the highest similarity scores.
- **Private Lexical Matching.** $\mathcal{S}$ and $\mathcal{C}$ invoke the private BM25 protocol $\prod_{\text{PrivateBM25}}$ (Protocol 2), where $\mathcal{S}$ inputs D^t and $\mathcal{C}$ inputs Q^t. After execution, $\mathcal{S}$ obtains the encrypted top-K chunks with the highest BM25 scores.

Phase 3: Private Generation Phase. In this phase, $\mathcal{C}$ obtains the response to its query Q while preserving privacy.

- **Context Fusion.** Then $\mathcal{S}$ fuses the encrypted retrieved $2K$ chunks with the encrypted query.
- **Secure Inference.** $\mathcal{S}$ and $\mathcal{C}$ execute the secure LLM inference framework to generate an encrypted response to $\mathcal{C}$.

Notably, in this paper, we pay attention to the preprocessing phase and the private retrieval phase, where the private retrieval phase is our core contribution. Furthermore, `Pisces` can be integrated with the existing secure inference framework based on various technologies, such as HE Rovida & Leporati (2024); Moon et al. (2024), MPC (Xu et al., 2025; Lu et al., 2023; Pang et al., 2024), and DP (Koga et al., 2024), to achieve end-to-end privacy.

3.2 Private Semantic Similarity

Semantic retrieval aims to retrieve the top-K most semantically relevant chunks for a query issued by a user $\mathcal{C}$ from a set of chunks held by a server $\mathcal{S}$. We design an efficient and private semantic similarity protocol $\prod_{\text{PrivateSS}}$ (Protocol 1) that leverages a coarse-to-fine pipeline. Direct computation of the cosine similarity over the entire set of chunks with cryptographic protocols (e.g., MPC) is prohibitively expensive. To mitigate this, we first propose a novel oblivious filter (Protocol 3, described in Appendix B.4) that privately selects a subset of candidate chunks, significantly reducing the scale of subsequent cosine similarity computations.

Protocol 1: $\prod_{\text{PrivateSS}}$

Input: $\mathcal{S}$ inputs the embedded chunk set $D^v = \{I_i;\ \mathbf{v}_i;\ c_i\}_{i\in[1,N]}$, where I_i and $\mathbf{v}_i$ are the index and vector representation corresponding to the chunk c_i, respectively. $\mathcal{C}$ inputs embedded query $\mathbf{q}$.

Output: $\mathcal{S}$ learns the encrypted top-K chunks $\{Enc(c_{t1}), Enc(c_{t2}), \ldots, Enc(c_{tK})\}$ with the highest cosine similarities.

1: $\mathcal{S}$ computes $\mathbf{v}_i^b \leftarrow SimHash(\mathbf{v}_i) = \{0,1\}^{\mathcal{L}}$ for $i \in [1,N]$. $\mathcal{C}$ computes $\mathbf{q}^b \leftarrow SimHash(\mathbf{q}) = \{0,1\}^{\mathcal{L}}$.
2: $\mathcal{S}$ and $\mathcal{C}$ invoke the obvious filter $\prod_{\text{Oblivious\_Filter}}$ (Protocol 3) with $\{I_i;\ \mathbf{v}_i^b;\ c_i\}_{i\in[1,N]}$ and $\mathbf{q}^b$ as input, respectively. After execution, $\mathcal{S}$ learns the candidate chunk set D'
3: $\mathcal{S}$ and $\mathcal{C}$ securely compute the cosine similarity between each chunk in D' and the query using MPC protocols based on secret sharing (Ma et al., 2023), obtaining secret shares of the cosine similarities, respectively.
4: $\mathcal{S}$ and $\mathcal{C}$ invoke the secure sorting protocol (Li et al., 2024b) with the secret shares of cosine similarities as input. After execution, $\mathcal{C}$ learns the indices $I^K = \{I_{t1}, I_{t2}, \ldots, I_{tK}\}$ of top-K chunks with the highest cosine similarities.
5: $\mathcal{S}$ and $\mathcal{C}$ invoke the batch PIR-to-share protocol (Song et al., 2025) with D^v and I^K as input, respectively. After execution, $\mathcal{S}$ and $\mathcal{C}$ learn the secret shares $\langle D^K \rangle$ of top-K chunks corresponding to I^t, where $D^K = \{c_{t1}, c_{t2}, \ldots, c_{tK}\}$.
6: $\mathcal{C}$ encrypts $\langle D^K \rangle^C$ to $Enc\left(\langle D^K \rangle^C\right)$ using FHE and sends it to $\mathcal{S}$. $\mathcal{S}$ computes $Enc(D^K) \leftarrow Enc\left(\langle D^K \rangle^C\right) + \langle D^K \rangle^S$.

We describe the private semantic similarity protocol $\prod_{\text{PrivateSS}}$ (Protocol 1) as follows:

- **Step 1 (Lines 1-2) Private Coarse Matching.** To leverage the computational efficiency of Hamming distance in cryptographic protocols, particularly for large-scale knowledge bases, we first translate cosine similarity computations into Hamming distance. Concretely, $\mathcal{S}$ and $\mathcal{C}$ convert their vector embeddings $\mathbf{v}_i$ ($i \in [1, N]$) and $\mathbf{q}$ into $\mathcal{L}$-bit binary vectors $\mathbf{v}_i^b$ and $\mathbf{q}^b$, respectively, using SimHash (Charikar, 2002). They then invoke the obvious filter $\prod_{\mathsf{Oblivious\_Filter}}$ (Protocol 3) that operates over Hamming space to identify a candidate set of chunks, which is much smaller than the full chunk set, without revealing any sensitive information about the query or knowledge base.
- **Step 2 (Line 3) Private Cosine Similarity Computation.** After identifying the candidate set of chunks, $\mathcal{S}$ and $\mathcal{C}$ perform fine-grained matching by jointly computing the cosine similarity between each candidate chunk and the query utilizing MPC protocols (Ma et al., 2023) based on secret sharing.
- **Step 3 (Lines 4-6) Encrypted Top-K Chunk Retrieval.** Given the computed cosine similarities, $\mathcal{S}$ and $\mathcal{C}$ privately retrieve the corresponding top-K encrypted chunks. Concretely, $\mathcal{C}$ first obtains the indices of the top-K chunks with the highest cosine similarities utilizing a secure sorting protocol (Li et al., 2024b). $\mathcal{S}$ and $\mathcal{C}$ then retrieve these chunks in secret-shared form utilizing a batch PIR-to-share protocol (Song et al., 2025). Finally, they convert the secret shares of top-K chunks into homomorphic encryption ciphertexts. This conversion is optional and depends on the input type of the subsequent secure LLM inference framework.

3.3 Private Lexical Matching

Lexical matching adopted in this paper considers an alternative scoring metric as described in B.3 for the top-K chunks. To achieve lexical matching efficiently and privately, we design an efficient private BM25 protocol. We first explore labeled PSI to privately obtain term frequencies for BM25 scoring. Furthermore, to reduce the overhead of repeatedly invoking labeled PSI for each chunk, we introduce a multi-instance labeled PSI protocol $\prod_{\mathsf{MulLPSI}}$ (Protocol 4, and the details are shown in Appendix B.5) based on OPRF and OKVS, that computes all per-chunk query term frequencies in a single execution.

We describe the private BM25 protocol $\prod_{\text{PrivateBM25}}$ (Protocol 2) as follows:

- **Step 1 (Lines 1-4) Private BM25 Scores Computation.** Firstly, $\mathcal{C}$ privately obtains the term frequency of each query token in each chunk by invoking the multi-instance labeled PSI protocol (Protocol 4). From these term frequencies, $\mathcal{C}$ could compute the document frequency (i.e., the number of chunks in which q_j appears) for each query token q_j. Then $\mathcal{S}$ and $\mathcal{C}$ jointly compute the BM25 score for each chunk utilizing MPC protocols based on secret sharing (Ma et al., 2023).
- **Step 2 (Lines 5-7) Encrypted Top-K Chunk Retrieval.** Given the computed BM25 scores, $\mathcal{S}$ and $\mathcal{C}$ privately retrieve the corresponding top-K encrypted chunks. This step is similar to Step 3 in the private similarity matching protocol $\prod_{\text{PrivateSS}}$ (Protocol 1) and therefore we omit the details here.

3.4 Private Generation

`Pisces` can be integrated with various secure LLM inference frameworks.

Integrate with HE-based Secure Inference Frameworks. As discussed in Section 3.1, $\mathcal{S}$ receives the homomorphically encrypted retrieved chunks along with the encrypted query. It then executes the HE-based secure LLM inference framework Rovida & Leporati (2024); Moon et al. (2024) to compute an encrypted response, which is subsequently returned to $\mathcal{C}$.

Integrate with MPC-based Secure Inference Frameworks. $\mathcal{S}$ and $\mathcal{C}$ avoid converting the secret shares of the retrieved chunks into homomorphic ciphertexts, skipping Step 6 of the private semantic similarity protocol (Protocol 1) and Step 7 of the private BM25 protocol (Protocol 2). Instead, $\mathcal{C}$ secret shares its query with $\mathcal{S}$. They then use these shares directly to execute the MPC-based secure LLM inference framework (Xu et al., 2025; Lu et al., 2023; Pang et al., 2024), thereby jointly computing secret shares of the response.

Protocol 2: $\prod_{\text{PrivateBM25}}$

Input: $\mathcal{S}$ inputs the tokenized chunk set $D^t = \left\{I_i; \{w_{i,l} : tf_{i,l}\}_{l\in[1,m_i]}; c_i\right\}_{i\in[1,N]}$, where m_i is the total number of unique tokens of c_i, $w_{i,l}$ is the l-th token of chunk c_i and $tf^D_{i,l}$ is its term frequency. $\mathcal{C}$ inputs tokenized query $Q^t = \{q_1, q_2, \ldots, q_n\}$, where n is the number of tokens in Q.

Output: $\mathcal{S}$ learns the encrypted top-K chunks $\{Enc(c_{t1}), Enc(c_{t2}), \ldots, Enc(c_{tK})\}$ with the highest BM25 scores.

1: $\mathcal{S}$ and $\mathcal{C}$ invoke the multi-instance labeled PSI protocol $\prod_{\text{MulLPSI}}$ (Protocol 4) with $\{w_{i,l} : tf_{i,l}\}_{i\in[1,N],l\in[1,m_i]}$ and Q^t as input, respectively. After execution, $\mathcal{C}$ learns the term frequency $tf'_{i,j}$ of each token q_j ($j \in [1,n]$) in each chunk c_i ($i \in [1,N]$), where if $q_j = w_{i,l}$, $tf'_{i,j} \leftarrow tf_{i,l}$, and otherwise $tf'_{i,j} = 0$.

2: $\mathcal{C}$ computes the document frequency $df_j \leftarrow \sum_{i=1}^{N}(tf'_{i,j} > 0?1:0)$ for each token q_j ($j \in [1,n]$).

3: $\mathcal{C}$ and $\mathcal{S}$ locally computes $\log\left(1 + \frac{N-df_j+0.5}{df_j+0.5}\right) \cdot tf'_{i,j}$ and $k_1 \cdot \left(1 - b + b \cdot \frac{L_{c_i}}{L_{ave}}\right)$, respectively, for $i \in [1,N]$ and $j \in [1,n]$.

4: $\mathcal{S}$ and $\mathcal{C}$ secure computes the BM25 scores according Equation (1) utilizing MPC protocols based on secret sharing (Ma et al., 2023). Then $\mathcal{S}$ and $\mathcal{C}$ learns the secret shares of BM25 scores, respectively.

5: $\mathcal{S}$ and $\mathcal{C}$ invoke the secure sorting protocol (Li et al., 2024b) with the secret shares of BM25 scores as input. After execution, $\mathcal{C}$ learns the indices $I^K = \{I_{t1}, I_{t2}, \ldots, I_{tK}\}$ of tok-K chunks with the highest BM25 scores.

6: $\mathcal{S}$ and $\mathcal{C}$ invoke the batch PIR-to-share protocol (Song et al., 2025) with D^v and I^K as input, respectively. After execution, $\mathcal{S}$ and $\mathcal{C}$ learn the secret shares $\langle D^K \rangle$ of top-K chunks corresponding t I^t, where $D^K = \{c_{t1}, c_{t2}, \ldots, c_{tK}\}$.

7: $\mathcal{C}$ encrypts $\langle D^K \rangle^C$ to $Enc\left(\langle D^K \rangle^C\right)$ using FHE and sends it to $\mathcal{S}$. $\mathcal{S}$ computes $Enc(D^K) \leftarrow Enc\left(\langle D^K \rangle^C\right) + \langle D^K \rangle^S$.

Integrate with DP-based Secure Inference Frameworks. Upon receiving both the homomorphically encrypted retrieved chunks and the encrypted query, $\mathcal{S}$ injects differential privacy noise into the received encrypted result. This perturbed result is then sent to $\mathcal{C}$, who decrypts it and proceeds with the DP-based secure LLM inference framework (Koga et al., 2024) to produce the response.

4 Experiments

In this section, we first introduce the experimental settings. Then we evaluate the practicality of `Pisces` in two parts: (1) the accuracy of `Pisces` compared to the plaintext baseline, and (2) the efficiency of `Pisces` compared to state-of-the-art cryptographic techniques.

4.1 Experimental Settings

Embedding Model and Tokenizer. We employ an open-source embedding model, granite-embedding-small-english-r2[1] (Awasthy et al., 2025) to encode chunks and the query into 384-dimensional vector representations. Additionally, we utilize an open-source tokenizer BERT[2] (Devlin et al., 2019) for chunk and query tokenization.

Datasets. We use three datasets: ClapNQ, SQuAD, and HotpotQA as RAG datasets. The details of these datasets are shown in Table 7. For the Dev_answerable dataset (300 queries in total), we run 300 queries and take the average to obtain stable results, while for the other datasets, we run 1,000 queries.

[1] https://huggingface.co/ibm-granite/granite-embedding-small-english-r2

[2] https://github.com/google-research/bert

Baselines. To demonstrate the accuracy of Pisces, we compare Pisces against the plaintext baseline under the same RAG architecture. To demonstrate efficiency, we compare the semantic retrieval of Pisces against a semantic retrieval baseline without coarse matching and the lexical retrieval of Pisces against a lexical retrieval baseline with the labeled PSI protocol LSE (Yang et al., 2024).

Environment. All of our experiments are conducted on an Apple M4 Pro machine with 24 GB of RAM, running macOS 15.6.1 (24G90).

4.2 Accuracy Evaluation

We evaluate the accuracy of Pisces against the plaintext baseline through two complementary approaches.

First, for each of the two retrieval paths, we compare the chunks retrieved by Pisces with those by the corresponding plaintext retrieval paths. Tables 2 and 3 present semantic and lexical retrieval accuracy under the Top-5 and Top-10 settings, respectively, compared to the plaintext baseline. The results demonstrate that Pisces achieves semantic retrieval accuracy ranges from 75.23% to 87.47% for Top-5, and from 73.30% to 86.80% for Top-10. At the same time, lexical retrieval accuracy ranges from 90.06% to 98.22 % for Top-5 and from 89.48% to 98.02% for Top-10. The accuracy drop in the semantic retrieval path mainly stems from the information loss when approximating cosine similarity with Hamming distance via SimHash. The slight degradation in lexical retrieval accuracy is primarily due to precision loss during secure BM25 score computation.

Table 2: Semantic retrieval accuracy against the plaintext baseline.

Dataset		Top-5		Top-10	
		Accuracy	Time (s)	Accuracy	Time (s)
ClapNQ	Dev_answerable	87.47%	3.47	86.80%	3.56
	Train_answerable	77.90%	4.12	76.32%	4.17
	Train_single_answerable	84.90%	7.33	78.25%	7.88
SQuAD	Dev_v2.0	78.10%	3.37	75.94%	3.41
	Training_v2.0	75.23%	4.38	73.30%	4.46
HotpotQA	Dev_distractor	77.98%	18.91	77.76%	20.10
	Dev_fullwiki	77.84%	19.27	76.78%	20.76

Table 3: Lexical retrieval accuracy against the plaintext baseline.

Dataset		Top-5		Top-10	
		Accuracy	Time (s)	Accuracy	Time (s)
ClapNQ	Dev_answerable	97.47%	1.40	96.53%	1.44
	Train_answerable	95.62%	2.07	95.13%	2.24
	Train_single_answerable	95.64%	5.94	94.99%	6.86
SQuAD	Dev_v2.0	97.56%	1.39	97.32%	1.42
	Training_v2.0	98.22%	2.59	98.02%	2.82
HotpotQA	Dev_distractor	90.06%	21.46	89.48%	25.02
	Dev_fullwiki	90.58%	21.39	89.85%	25.61

Second, we evaluate the chunks retrieved by both Pisces and the plaintext baseline against the dataset ground-truth. Figure 2 compares the top-5 retrieval accuracy between Pisces and the plaintext baseline, indicating that (1) Pisces achieves retrieval accuracy comparable to the plaintext baseline, and (2) combining semantic and lexical paths improves overall retrieval performance. Additionally, the top-10 retrieval accuracy comparison is provided in Figure 5, with detailed accuracy values available in Table 8.

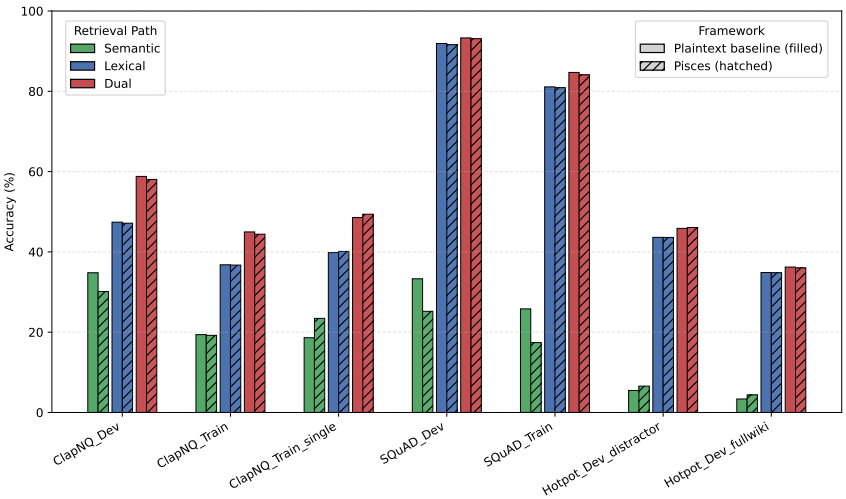

Figure 2: Top-5 Retrieval accuracy comparisons between `Pisces` and plaintext baseline over ground-truth.

4.3 Efficiency Evaluation

We evaluate the efficiency of the two retrieval paths of `Pisces`, respectively.

Table 4: Efficiency comparisons with Fine-only Strategy

Dataset		Fine-Only Strategy			
		Time (s)	**Upload (MB)**	**Download (MB)**	**Accuracy**
ClapNQ	Dev_answerable	1.855	26.64	36.41	99.5%
	Train_answerable	2.82	69.47	230.82	94.9%
	Train_single_answerable	10.29	278.77	1195.62	91.0%
SQuAD	Dev_v2.0	1.714	24.06	24.147	99.5%
	Train_v2.0	3.66	87.97	315.18	97.4%
HotpotQA	Dev_distractor	34.19	1008.39	4610.26	93.9%
	Dev_fullwiki	33.91	1031.45	4719.76	94.1%
Dataset		**Coarse-to-Fine Strategy**			
		Time (s)	**Upload (MB)**	**Download (MB)**	**Accuracy**
ClapNQ	Dev_answerable	3.56	23.95	23.32	86.80%
	Train_answerable	4.17	36.35	85.64	76.32%
	Train_single_answerable	7.88	113.07	442.97	78.25%
SQuAD	Dev_v2.0	3.41	21.96	13.93	75.94%
	Train_v2.0	4.46	43.51	118.29	73.30%
HotpotQA	Dev_distractor	20.10	324.90	1439.70	77.76%
	Dev_fullwiki	20.76	334.88	1487.09	78.02%

For the semantic retrieval path, we evaluate the efficiency of our proposed coarse-to-fine strategy with the fine-only strategy, i.e., without coarse matching. The results shown in Table 4 demonstrate that for a large-scale dataset, the coarse-to-fine strategy significantly saves retrieval time by 38.78% $\sim$ 41.21%, reduces the upload and download overhead by 67.53% $\sim$ 67.78% and 68.49% $\sim$ 68.77%, respectively. In contrast, when we deal with the small-scale dataset, directly computing the cosine similarities over the full chunk set outperforms the coarse-to-fine strategy. This is because in such scenarios, the coarse matching step, rather than the cosine similarity computation, becomes the bottleneck.

Table 5: Efficiency comparisons with labeled PSI

Dataset		Labeled PSI		
		Time (s)	Upload (MB)	Download (MB)
ClapNQ	Dev_answerable	3.89	1.62	0.60
	Train_answerable	27.57	4.21	11.35
	Train_single_answerable	138.89	21.22	57.77
SQuAD	Dev_v2.0	3.15	0.48	2.03
	Train_v2.0	45.89	6.99	30.05
HotpotQA	Dev_distractor	1051.98	161.61	382.26
	Dev_fullwiki	1179.58	176.89	414.16
Dataset		**Multi-instance Labeled PSI**		
		Time (s)	**Upload (MB)**	**Download (MB)**
ClapNQ	Dev_answerable	0.009	0.0003	0.58
	Train_answerable	0.056	0.0003	4.00
	Train_single_answerable	0.28	0.0003	21.04
SQuAD	Dev_v2.0	0.008	0.0004	1.49
	Train_v2.0	0.099	0.0004	22.64
HotpotQA	Dev_distractor	2.35	0.0006	79.82
	Dev_fullwiki	2.59	0.0006	81.44

For the lexical retrieval path, we evaluate the efficiency of our proposed multi-instance labeled PSI protocol $\prod_{\mathsf{MulLPSI}}$ (Protocol 4) with the state-of-the-art labeled PSI protocol LSE (Yang et al., 2024). The results shown in Table 4 demonstrate that our proposed multi-instance labeled PSI outperforms LSE by up to 496.03×, 70733×, and 2.84× in running time, upload overhead, and download overhead, respectively.

5 Related work

RAG with Dual-Path Retrieval. Multiple works (Kuzi et al., 2020; Gao et al., 2021; Li et al., 2022) demonstrate that leveraging semantic and lexical retrieval together significantly improves retrieval performance. Inspired by this, we aim to design `Pisces` that privately supports dual-path retrieval to guarantee the retrieval accuracy.

RAG with Retrieval Process Protection. Recent work applies differential privacy by injecting noise into embeddings to protect privacy during the retrieval process. Several works (Grislain, 2025; He et al., 2025) focus on protecting documents during the semantic retrieval, while Cheng et al. (Cheng et al., 2025) propose RemoteRAG to protect the query. Yao and Li (Yao & Li, 2025) further attempt to protect both the query and documents. However, all of these works only consider a single retrieval path, i.e., semantic retrieval. In contrast, `Pisces` supports dual-path retrieval, semantic and lexical, while protecting both the query and documents.

6 Conclusion

In this paper, we propose `Pisces`, the first practical cryptography-based RAG framework that supports dual-path retrieval while protecting both the query and documents. We design novel cryptographic protocols tailored for efficient semantic and lexical retrieval: a coarse-to-fine semantic strategy that employs a novel oblivious filter over Hamming distance, and an efficient multi-instance labeled PSI protocol that obtains BM25 term frequencies in a single execution. We comprehensively evaluate `Pisces` and find only a 1.14% deviation in retrieval accuracy relative to plaintext baselines. On large-scale datasets, our coarse-to-fine strategy reduces runtime by 41.21% and upload/download overhead by 68.77% compared to a fine-only strategy. Our proposed multi-instance labeled PSI further outperforms LSE by up to 496.03× in runtime, 70733× in upload overhead. These results demonstrate that `Pisces` is both accurate and efficient.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. arXiv preprint arXiv:2303.08774, 2023.

Parul Awasthy, Aashka Trivedi, Yulong Li, Meet Doshi, Riyaz Bhat, Vishwajeet Kumar, Yushu Yang, Bhavani Iyer, Abraham Daniels, Rudra Murthy, et al. Granite embedding r2 models. arXiv preprint arXiv:2508.21085, 2025.

Alexander Bienstock, Sarvar Patel, Joon Young Seo, and Kevin Yeo. Batch {PIR} and labeled {PSI} with oblivious ciphertext compression. In 33rd USENIX Security Symposium (USENIX Security 24), pp. 5949–5966, 2024.

Moses S Charikar. Similarity estimation techniques from rounding algorithms. In Proceedings of the thiry-fourth annual ACM symposium on Theory of computing, pp. 380–388, 2002.

Hao Chen, Zhicong Huang, Kim Laine, and Peter Rindal. Labeled psi from fully homomorphic encryption with malicious security. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 1223–1237, 2018.

Yihang Cheng, Lan Zhang, Junyang Wang, Mu Yuan, and Yunhao Yao. Remoterag: A privacy-preserving llm cloud rag service. arXiv preprint arXiv:2412.12775, 2024.

Yihang Cheng, Lan Zhang, Junyang Wang, Mu Yuan, and Yunhao Yao. RemoteRAG: A privacy-preserving LLM cloud RAG service. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), Findings of the Association for Computational Linguistics: ACL 2025, pp. 3820–3837, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.197.

Kelong Cong, Radames Cruz Moreno, Mariana Botelho da Gama, Wei Dai, Ilia Iliashenko, Kim Laine, and Michael Rosenberg. Labeled psi from homomorphic encryption with reduced computation and communication. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, pp. 1135–1150, 2021.

National People's Congress. Personal information protection law of the people's republic of china, a. URL http://en.npc.gov.cn.cdurl.cn/2021-12/29/c_694559.htm.

United States Congress. Health insurance portability and accountability act, b. URL http://www.cms.hhs.gov/hipaa/.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers), pp. 4171–4186, 2019.

Michael J Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In Theory of Cryptography: Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005. Proceedings 2, pp. 303–324. Springer, 2005.

Luyu Gao, Zhuyun Dai, Tongfei Chen, Zhen Fan, Benjamin Van Durme, and Jamie Callan. Complement lexical retrieval model with semantic residual embeddings. In European Conference on Information Retrieval, pp. 146–160. Springer, 2021.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. arXiv preprint arXiv:2312.10997, 2(1), 2023.

Gayathri Garimella, Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. Oblivious key-value stores and amplification for private set intersection. In Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part II 41, pp. 395–425. Springer, 2021.

Nicolas Grislain. Rag with differential privacy. In 2025 IEEE Conference on Artificial Intelligence (CAI), pp. 847–852. IEEE, 2025.

Longzhu He, Peng Tang, Yuanhe Zhang, Pengpeng Zhou, and Sen Su. Mitigating privacy risks in retrieval-augmented generation via locally private entity perturbation. Information Processing & Management, 62(4):104150, 2025.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. ACM Transactions on Information Systems, 43(2):1–55, 2025.

Yangsibo Huang, Samyak Gupta, Zexuan Zhong, Kai Li, and Danqi Chen. Privacy implications of retrieval-based language models. arXiv preprint arXiv:2305.14888, 2023.

Stanisław Jarecki and Xiaomin Liu. Fast secure computation of set intersection. In International Conference on Security and Cryptography for Networks, pp. 418–435. Springer, 2010.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pp. 7969–7992, 2023.

Marcel Keller. Mp-spdz: A versatile framework for multi-party computation. In Proceedings of the 2020 ACM SIGSAC conference on computer and communications security, pp. 1575–1590, 2020.

Tatsuki Koga, Ruihan Wu, and Kamalika Chaudhuri. Privacy-preserving retrieval-augmented generation with differential privacy. arXiv preprint arXiv:2412.04697, 2024.

Saar Kuzi, Mingyang Zhang, Cheng Li, Michael Bendersky, and Marc Najork. Leveraging semantic and lexical matching to improve the recall of document retrieval systems: A hybrid approach. arXiv preprint arXiv:2010.01195, 2020.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in neural information processing systems, 33: 9459–9474, 2020.

Hang Li, Shuai Wang, Shengyao Zhuang, Ahmed Mourad, Xueguang Ma, Jimmy Lin, and Guido Zuccon. To interpolate or not to interpolate: Prf, dense and sparse retrievers. In Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval, pp. 2495–2500, 2022.

Jiarui Li, Ye Yuan, and Zehua Zhang. Enhancing llm factual accuracy with rag to counter hallucinations: A case study on domain-specific queries in private knowledge-bases. arXiv preprint arXiv:2403.10446, 2024a.

Jingyu Li, Zhicong Huang, Min Zhang, Jian Liu, Cheng Hong, Tao Wei, and Wenguang Chen. Panther: Private approximate nearest neighbor search in the single server setting. Cryptology ePrint Archive, 2024b.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. arXiv preprint arXiv:2412.19437, 2024.

Wen-jie Lu, Zhicong Huang, Zhen Gu, Jingyu Li, Jian Liu, Cheng Hong, Kui Ren, Tao Wei, and WenGuang Chen. Bumblebee: Secure two-party inference framework for large transformers. Cryptology ePrint Archive, 2023.

Xing Han Lù. Bm25s: Orders of magnitude faster lexical search via eager sparse scoring. arXiv preprint arXiv:2407.03618, 2024.

Junming Ma, Yancheng Zheng, Jun Feng, Derun Zhao, Haoqi Wu, Wenjing Fang, Jin Tan, Chaofan Yu, Benyu Zhang, and Lei Wang. {SecretFlow-SPU}: A performant and {User-Friendly} framework for {Privacy-Preserving} machine learning. In 2023 USeNIX annual technical conference (USeNIX ATC 23), pp. 17–33, 2023.

Jungho Moon, Dongwoo Yoo, Xiaoqian Jiang, and Miran Kim. Thor: Secure transformer inference with homomorphic encryption. Cryptology ePrint Archive, 2024.

Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and kdcs. In International conference on the theory and applications of cryptographic techniques, pp. 327–346. Springer, 1999.

Qi Pang, Jinhao Zhu, Helen Möllering, Wenting Zheng, and Thomas Schneider. Bolt: Privacy-preserving, accurate and efficient inference for transformers. In 2024 IEEE Symposium on Security and Privacy (SP), pp. 4753–4771. IEEE, 2024.

European Parliament and of the Council of the European Union. Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation). URL `https://www.gdpr-info.eu`.

Stephen Robertson, Hugo Zaragoza, et al. The probabilistic relevance framework: Bm25 and beyond. Foundations and Trends® in Information Retrieval, 3(4):333–389, 2009.

Lorenzo Rovida and Alberto Leporati. Transformer-based language models and homomorphic encryption: An intersection with bert-tiny. In Proceedings of the 10th ACM International Workshop on Security and Privacy Analytics, pp. 3–13, 2024.

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.

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.

Yunbo Yang, Yiwei Hu, Ruofan Li, Xiaolei Dong, Zhenfu Cao, Jiachen Shen, and Shangmin Dou. Lse: Efficient symmetric searchable encryption based on labeled psi. IEEE Transactions on Services Computing, 17(2):563–574, 2024.

Dixi Yao and Tian Li. Private retrieval augmented generation with random projection. In ICLR 2025 Workshop on Building Trust in Language Models and Applications, 2025.

Shenglai Zeng, Jiankun Zhang, Pengfei He, Yue Xing, Yiding Liu, Han Xu, Jie Ren, Shuaiqiang Wang, Dawei Yin, Yi Chang, et al. The good and the bad: Exploring privacy issues in retrieval-augmented generation (rag). arXiv preprint arXiv:2402.16893, 2024.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. Qwen3 embedding: Advancing text embedding and reranking through foundation models. arXiv preprint arXiv:2506.05176, 2025.

A Appendix

A.1 Notation

We summarize the frequently used notation in Table 6.

Table 6: Notation Table

Symbol	Description
$\mathcal{S}$	The server, who holds a sensitive knowledge base.
$\mathcal{C}$	The user, who holds a private query.
D	Document set in the knowledge base.
Q	Query.
N	Chunk number of D.
n	Unique token number of Q.
c_i	i-th chunk.
I_i	Index of chunk c_i.
$\mathbf{v}_i$	Vector representation of chunk c_i.
m	Unique token number of chunk c_i.
$w_{i,l}$	l-th token of chunk c_i.
$tf_{i,l}$	Term frequency of $w_{i,l}$ in chunk c_i.
$D^v = \{I_i;\ \mathbf{v}_i;\ c_i\}_{i\in[1,N]}$	Embedded chunk set.
$D^t = \left\{I_i; \{w_{i,l} : tf_{i,l}\}_{l\in[1,m_i]}; c_i\right\}_{i\in[1,N]}$	Tokenized chunk set.
q_j	j-th token of query Q.
$\mathbf{q}$	Vector representation of query Q.
$Q^t = \{q_1, q_2, \ldots, q_n\}$	Tokenized query.

B Preliminaries

B.1 Cryptographic Primitives

B.1.1 Secert Sharing

Secret sharing (Shamir, 1979; Keller, 2020)is one of the critical primitives of MPC. In this paper, we adopt 2-out-of-2 arithmetic secret sharing technology. The main idea of it is to break a secret value into 2 shares, each of which is held by a party. For example, $\mathcal{S}$, who holds the secret value $x \in \mathbb{F}_p$, wants to secret share this secret value with another party $\mathcal{C}$. To do this, P_S first generates a random value $r \in \mathbb{F}_p$ as its share $\langle x\rangle^S = r$, and then sends $\langle x\rangle^C = x - r \bmod \mathbb{F}_p$ to another party $\mathcal{C}$. Therefore $x = \langle x\rangle^S + \langle x\rangle^C \bmod \mathbb{F}_p$, which, for simplicity, we denote as $x = \langle x\rangle^S + \langle x\rangle^C$.

B.1.2 Labeled Private Set Intersection

The PSI (Jarecki & Liu, 2010) allows two parties, a server $\mathcal{S}$ and a client $\mathcal{C}$, to learn the intersection of their respective element sets without revealing any additional information outside the intersection. Labeled PSI (Chen et al., 2018; Bienstock et al., 2024; Cong et al., 2021) extends the traditional PSI by allowing the server $\mathcal{S}$ to associate a label with each element, and the client $\mathcal{C}$ learns the labels for elements in the intersection. Formally, $\mathcal{S}$ inputs a set of key-value pairs $\{(x_i, l(x_i))\}$, where x_i is an element and $l(x_i)$ is its corresponding label, while $\mathcal{C}$ inputs a set of key Y. After execution, $\mathcal{C}$ learns a set of pairs $\{(y, l(y))\}$ for $y \in X \cap Y$.

B.1.3 Oblivious Pseudorandom Function

The oblivious pseudorandom function (OPRF) (Freedman et al., 2005) is a cryptographic primitive that enables two parties, a server $\mathcal{S}$ and a client $\mathcal{C}$, to jointly compute a pseudorandom function (PRF) $F_\cdot(\cdot)$. As shown in figure 3, $\mathcal{S}$ takes a PRF key k as input and learns nothing, while $\mathcal{C}$ takes x as input and learns the PRF value $F_k(x)$. Moreover, $\mathcal{C}$ learns nothing about the PRF key k and $\mathcal{S}$ learns nothing about the input or the output of $\mathcal{C}$.

Functionality $\mathcal{F}_{\text{OPRF}}$

Parameters: Two parties $\mathcal{S}$ and $\mathcal{C}$. A PRF $F_.(\cdot)$.

Functionality:

- Wait for input k from $\mathcal{S}$, where k is a PRF key.
- Wait for input x from $\mathcal{C}$.
- Output $F_k(x)$ to $\mathcal{C}$.

Figure 3: Ideal functionality of OPRF

B.1.4 Obviously Key-Value Store

The oblivious key-value store (OKVS) (Garimella et al., 2021) is a data structure that encodes a set of key-value pairs into a compact representation while preserving the privacy of both keys and values. The definition is as follows:

Definition 1 (Oblivious Key-Value Store)**.** *An OKVS parameterized by a key space $\mathcal{K}$ and a value $\mathcal{V}$ space, and consists of two algorithms:*

- Γ *or* $\perp \leftarrow \text{Encode}((k_1,v_1),(k_2,v_2),\ldots,(k_n,v_n))$*: The encode algorithm takes n key–value pairs* $\{(k_1,v_1),(k_2,v_2),\ldots,(k_n,v_n)\} \subset \{\mathcal{K}\times\mathcal{V}\}^n$ *as input, and outputs a structure* Γ *(or an error terminator* $\perp$ *with negligible probability).*
- $v \leftarrow \text{Decode}(\Gamma,k)$*: The decode algorithm takes an OKVS structure* Γ *and a key* $k \in \mathcal{K}$ *as input, and outputs the corresponding value* $v \in \mathcal{V}$*.*

Correctness: An OKVS is correct if, for all $X \subset \mathcal{K}\times\mathcal{V}$ with distinct keys such that $\text{Encode}(X) = \Gamma \neq \perp$ and $(k,v) \in X$, it holds that $\text{Decode}(\Gamma,k) = v$;

Computationally Obliviousness: An OKVS is computationally oblivious if, for any two key sets with n distinct keys $K = \{k_1,k_2,\ldots,k_n\} \subset \mathcal{K}$ and $K' = \{k'_1,k'_2,\ldots,k'_n\} \subset \mathcal{K}$ and a uniformly random value set $V = \{v_1,v_2,\ldots,v_n\} \subset \mathcal{V}$, a probabilistic polynomial-time adversary is not able to distinguish between $\text{Encode}((k_1,v_1),(k_2,v_2),\ldots,(k_n,v_n)) = \Gamma \neq \perp$ and $\text{Encode}((k'_1,v_1),(k'_2,v_2),\ldots,(k'_n,v_n)) = \Gamma' \neq \perp$.

This computationally obliviousness property ensures that the OKVS reveals no information about the encoded keys or values beyond the decoded results for given keys.

B.1.5 Batch Private Information Retrieval-to-Share

The batch private information retrieval-to-share (PIR-to-share) (Song et al., 2025) is a cryptographic primitive that enables a client $\mathcal{C}$ to privately retrieve the values corresponding to its queries from the server $\mathcal{S}$. After that, $\mathcal{S}$ and $\mathcal{C}$ obtain the secret shares of queried values, respectively. As shown in figure 4, $\mathcal{S}$ takes its data D of size N as input, while $\mathcal{C}$ takes its queries $I = \{I_1,I_2,\ldots,I_b\}$ (index set) as input. $\mathcal{S}$ learns data shares $\langle D[I_1]\rangle^S, \langle D[I_2]\rangle^S, \ldots, \langle D[I_b]\rangle^S$ corresponding to $\mathcal{C}$'s queries and $\mathcal{C}$ learns data shares $\langle D[I_1]\rangle^C, \langle D[I_2]\rangle^C, \ldots, \langle D[I_b]\rangle^C$. During this process, $\mathcal{S}$ learns nothing about $\mathcal{C}$'s queries, and $\mathcal{C}$ only learns the secret shares of the retrieved values rather than the raw data of $\mathcal{S}$.

B.2 Semantic Similarity

Semantic similarity (Awasthy et al., 2025; Zhang et al., 2025) is a measure of the degree to which the meanings of two linguistic units, such as words, phrases, sentences, or documents, are alike, based on their semantic content rather than lexical matching. It plays a fundamental role in many natural language processing tasks, including information retrieval and text summarization. Contemporary methods operationalize meaning via vector representations. Similarity is then measured with distance functions in embedding space, such as cosine similarity, Hamming distance, and Euclidean distance. In this paper, we choose cosine similarity as our similarity metric.

Functionality $\mathcal{F}_{\text{PIR2Share}}$

Parameters: Two parties $\mathcal{S}$ and $\mathcal{C}$.

Functionality:

- Wait for input D from $\mathcal{S}$.
- Wait for input $I = \{I_1, I_2, \ldots, I_b\}$ from $\mathcal{C}$.
- Sample $\langle D[I_1]\rangle^S, \langle D[I_2]\rangle^S, \ldots, \langle D[I_b]\rangle^S$ and $\langle D[I_1]\rangle^C, \langle D[I_2]\rangle^C, \ldots, \langle D[I_b]\rangle^C$ uniformly, such that $\langle D[I_1]\rangle^S + \langle D[I_1]\rangle^C = \langle D[I_1]\rangle, \ldots, \langle D[I_b]\rangle^S + \langle D[I_b]\rangle^C = \langle D[I_b]\rangle$.
- Output the shares $\langle D[I_1]\rangle^S, \langle D[I_2]\rangle^S, \ldots, \langle D[I_b]\rangle^S$ to P_S and $\langle D[I_1]\rangle^C, \langle D[I_2]\rangle^C, \ldots, \langle D[I_b]\rangle^C$ to P_C.

Figure 4: Ideal functionality of $\mathcal{F}_{\text{PIR2Share}}$

B.3 Best Matching 25

A popular algorithm to achieve lexical retrieval is BM25 (Robertson et al., 2009; Lù, 2024), which is a probabilistic information retrieval algorithm widely used to rank documents according to their relevance to a given query. It is an enhancement to the traditional term frequency-inverse document frequency (TF-IDF) algorithm, which measures the importance of a term within a set of documents. BM25 takes document length into account and introduces a saturation function to term frequencies, which helps prevent common terms from dominating the results to improve the ranking accuracy.

Given a document set $D = \{d_1, d_2, \ldots, d_N\}$ and a query $Q = \{q_1, q_2, \ldots, q_n\}$, where d_i denotes the i-th document in D, N is the total number of documents in D, q_j is the j-th term in Q, n is the total number of terms in Q, the BM25 relevance score for document d_i relative to this query is defined as:

$$\begin{aligned} Score(Q, d_i) &= \sum_{j=1}^{n} \text{IDF}(q_j) \cdot R(q_j, d_i) \\ &= \sum_{j=1}^{n} \log\left(1 + \frac{N - df_j + 0.5}{df_j + 0.5}\right) \cdot \frac{tf_{i,j}}{tf_{i,j} + k_1 \cdot \left(1 - b + b \cdot \frac{L_{d_i}}{L_{ave}}\right)} \end{aligned} \tag{1}$$

where $\text{IDF}(q_j)$ is the inverse document frequency of q_j and $R(q_j, d)$ is the relevance score for the document d_i relative to the term q_j. Besides, df_j is the document frequency for term q_j, i.e. the number of documents in the document set D in which q_j appears, $tf_{i,j}$ is the term frequency of q_j in the document d_i, L_{d_i} is the length of the document d_i, L_{ave} is the average length of the document set D, $k_1 > 0$ and $0 < b < 1$ are constant values, k_1 controls the saturation of the term frequency and b adjusts the impact of normalization of document length.

B.4 Oblivious filter

The core idea of the oblivious filter is to convert an approximate (fuzzy) matching problem into an exact matching task. In our protocol, both the knowledge base and the user should select the same projections to mask their binary vector(s). A chunk is considered a candidate match if its projected binary vectors match the query's projected binary vectors on at least two projections. This approach allows the knowledge base to identify a candidate set of chunks that are likely to match the query.

To achieve the threshold matching requirement cryptographically, we employ a 2-out-of-T Shamir secret sharing scheme. The client can only reconstruct a secret value if it obtains at least two shares for a chunk. Furthermore, the prevent the client to learn which specific chunks were matched, the knowledge base encrypts all shares with additive homomorphic encryption. As a result, the client would reconstruct the secret over the cipher space, which ensures the client could not learn any information throughout the oblivious filter.

Protocol 3: $\prod_{\text{Oblivious\_Filter}}$

Input: $\mathcal{S}$ inputs the set $D^v = \left\{(I_i, \mathbf{v}_i^{\mathbf{b}}, c_i)\right\}_{i\in[1,N]}$, where for each $i \in [1,N]$: I_i is an index, $\mathbf{v}_i^{\mathbf{b}} \in \{0,1\}^{\mathcal{L}}$ is a binary vector, and c_i is a chunk. $\mathcal{C}$ inputs binary vector $\mathbf{q}^{\mathbf{b}} \in \{0,1\}^{\mathcal{L}}$.

Output: $\mathcal{S}$ learns a set $D' = \left\{(I_i', \mathbf{v}_i^{\mathbf{b}\prime}, c_i')\right\}_{i\in[1,N]}$, where for all $i \in [1,N']$: $\mathsf{HD}(\mathbf{v}_i^{\mathbf{b}\prime}, \mathbf{q}^{\mathbf{b}}) \le t$ (i.e., Hamming distance at most t).

Setup Phase:

1: $\mathcal{S}$ generates a random keypair (pk, sk) for an additive homomorphic encryption scheme.
2: $\mathcal{S}$ sets $\ell \leftarrow \lceil\sqrt{t \cdot \mathcal{L}}\rceil$ (projection weight) and $T \leftarrow 160$ (number of projections). $\mathcal{S}$ randomly selects T projection masks $\{\mathbf{m}_i \in \{0,1\}^{\mathcal{L}}\}_{i\in[1,T]}$ such that $\|\mathbf{m}_i\| = \ell$ for all $i \in [1,T]$.
3: $\mathcal{S}$ selects $2N$ random numbers: $\{x_i\}_{i\in[1,N]}$ and $\{s_i\}_{i\in[1,N]}$, and initializes an empty collection $\mathbb{C}$.
4: $\mathcal{S}$ selects a random linear polynomial $P_i(x) = ax + s_i$ (with random coefficient a) for $i \in [1,N]$.
5: $\mathcal{S}$ computes ciphertext $v_{i,j} \leftarrow \mathsf{Enc}(pk, P_i(x_j))$ and key $k_{i,j} \leftarrow \mathsf{Hash}(\mathbf{v}_i^{\mathbf{b}} \wedge \mathbf{m}_j)$ for $i \in [1,N], j \in [1,T]$.
6: $\mathcal{S}$ inserts the pair $(k_{i,j}, v_{i,j})$ into $\mathbb{C}$.
7: $\mathcal{S}$ invokes $\mathsf{OKVS.Encode}(\mathbb{C})$ to obtain the OKVS structure Γ.

Interactive Phase:

1: $\mathcal{C}$ requests and receives from $\mathcal{S}$: the public key pk, projection masks $\{\mathbf{m}_i\}_{i\in[1,T]}$, OKVS structure Γ, and random numbers $\{x_i\}_{i\in[1,T]}$.
2: $\mathcal{C}$ computes for each $j = 1$ to T: $t_j \leftarrow \mathsf{Hash}(\mathbf{q}^{\mathbf{b}} \wedge \mathbf{m}_j)$
3: $\mathcal{C}$ invokes $\mathsf{OKVS.Decode}(\Gamma, \{t_j\}_{i\in[1,T]})$ to obtain values $\{d_i\}_{i\in[1,T]}$
4: $\mathcal{C}$ computes a candidate secret ciphertext: $s_{i,j} \leftarrow d_j - x_j \cdot \frac{d_i - d_j}{x_i - x_j}$ for each combination (i,j) from the $\binom{T}{2}$ possible pairs of indices from $[1,T]$.
5: $\mathcal{C}$ shuffles all computed ciphertexts $\{s_{i,j}\}$ to form the set $\mathbb{S}$ and sends $\mathbb{S}$ to $\mathcal{S}$.
6: $\mathcal{S}$ receives $\mathbb{S}$, decrypts each element: $\mathbb{P} \leftarrow \{\mathsf{Dec}(sk, s) \mid s \in \mathbb{S}\}$.
7: For each s_i (from the original setup) that appears in $\mathbb{P}$, $\mathcal{S}$ adds the corresponding item $(I_i, \mathbf{v}_i^{\mathbf{b}}, c_i)$ to the result set D'.
8: $\mathcal{S}$ returns D' as the final result.

B.5 Multi-Instance Labeled Private Set Intersection

We design a customized Multi-Instance Labeled PSI protocol to support repeated invocations of labeled PSI with the same small client query set. Our protocol features two key innovations. First, the setup phase only involves the knowledge base and produces a reusable OKVS structure Γ. It can be efficiently reused across multiple queries without recomputation. Second, the interactive phase minimizes computational overhead. It only requires a single, small-scale OPRF execution per query, independent of the server's data size. These optimizations significantly reduce both communication and computation costs compared to conventional labeled PSI protocols.

B.6 Detailed Dataset

B.7 Supplementary Accuracy Experimental Results

The top-10 retrieval accuracy comparison is shown in Figure 5, and the detailed accuracy values are available in Table 8.

Table 7: Details of datasets we evaluated in this paper. "Documents" denotes the number of documents in the dataset, and "Chunks" denotes the number of chunks generated from breaking down all the documents in the dataset.

Dataset		Documents	Chunks
ClapNQ	Dev_answerable	290	1990
	Train_answerable	1751	14010
	Train_single_answerable	8996	71363
SQuAD	Dev_v2.0	35	1204
	Training_v2.0	442	19029
HotpotQA	Dev_distractor	66581	269602
	Dev_fullwiki	66573	276013

Table 8: Retrieval accuracy comparisons between `Pisces` and plaintext baseline over ground-truth.

Dataset		Framework	Top-5		
			Semantic	Lexical	Dual-Path
ClapNQ	Dev_answerable	Plaintext	34.82%	47.42%	58.82%
		`Pisces`	30.14%	47.15%	58.05%
	Train_answerable	Plaintext	19.39%	36.78%	44.97%
		`Pisces`	19.22%	36.72%	44.41%
	Train_single_answerable	Plaintext	18.62%	39.82%	48.55%
		`Pisces`	23.44%	40.12%	49.39%
SQuAD	Dev_v2.0	Plaintext	33.30%	91.90%	93.30%
		`Pisces`	25.20%	91.60%	93.30%
	Training_v2.0	Plaintext	25.80%	81.10%	84.70%
		`Pisces`	17.40%	80.90%	84.10%
HotpotQA	Dev_distractor	Plaintext	5.48%	43.62%	45.86%
		`Pisces`	6.57%	43.60%	46.08%
	Dev_fullwiki	Plaintext	3.36%	34.86%	36.22%
		`Pisces`	4.36%	34.80%	36.00%
Dataset		**Framework**	**Top-10**		
			Semantic	**Lexical**	**Dual-Path**
ClapNQ	Dev_answerable	Plaintext	40.63%	54.84%	67.13%
		`Pisces`	36.56%	54.35%	65.43%
	Train_answerable	Plaintext	22.89%	42.95%	51.22%
		`Pisces`	21.95%	43.16%	50.75%
	Train_single_answerable	Plaintext	21.24%	46.83%	54.78%
		`Pisces`	27.63%	46.92%	54.78%
SQuAD	Dev_v2.0	Plaintext	40.10%	94.60%	95.70%
		`Pisces`	31.80%	94.40%	95.50%
	Training_v2.0	Plaintext	34.00%	85.10%	89.00%
		`Pisces`	22.10%	85.00%	88.40%
HotpotQA	Dev_distractor	Plaintext	6.23%	51.91%	53.96%
		`Pisces`	7.77%	52.48%	54.69%
	Dev_fullwiki	Plaintext	3.92%	40.18%	41.35%
		`Pisces`	5.16%	40.17%	41.40%

Protocol 4: Π_{MulLPSI}

Input: $\mathcal{S}$ inputs set $D^t = \{w_{i,l} : tf_{i,l}\}_{i\in[1,N],l\in[1,m_i]}$. $\mathcal{C}$ inputs set $Q^t = \{q_1, q_2, \ldots, q_n\}$.

Output: $\mathcal{C}$ learns $\{tf'_{i,j}\}_{i\in[1,N]],j\in[1,n]}$, where if $q_j = w_{i,l}$ then $tf'_{i,j} = tf_{i,l}$, and otherwise $tf'_{i,j} = 0$.

Setup Phase:

1: $\mathcal{S}$ selects a random PRF key k and two key derivation functions KDF_0 and KDF_1.
2: $\mathcal{S}$ initializes an empty set $\mathbb{S}$.
3: $\mathcal{S}$ computes $r_{i,l} \leftarrow \mathsf{PRF}(k, w_{i,l})$, $k_{i,l} \leftarrow \mathsf{KDF}_0(i, r_{i,l})$, $m_{i,l} \leftarrow \mathsf{KDF}_1(i, r_{i,l})$ and $c_{i,l} \leftarrow \mathsf{AES.Enc}(m_{i,l}, 0^\ell \parallel tf_{i,l})$ for $i \in [1,N], l \in [1,m_i]$.
4: $\mathcal{S}$ inserts the key-value pair $(k_{i,l}, c_{i,l})$ into $\mathbb{S}$ for $i \in [1,N], l \in [1,m_i]$.
5: $\mathcal{S}$ invokes $\mathsf{OKVS.Encode}(\mathbb{S})$ to obtain the OKVS structure Γ.

Interactive Phase:

1: $\mathcal{C}$ requests and receives from $\mathcal{S}$: the OKVS structure Γ and key derivation functions KDF_0, KDF_1.
2: $\mathcal{C}$ and $\mathcal{S}$ invoke an OPRF protocol with $Q^t = \{q_1, q_2, \ldots, q_n\}$ and PRF key k as inputs, respectively. After execution, $\mathcal{C}$ obtains the PRF results $\mathbb{D} = \{d_1, d_2, \ldots, d_n\}$.
3: $\mathcal{C}$ initializes $\mathbb{K}_i = \emptyset$ and $\mathbb{M}_i = \emptyset$ for $i \in [1,N]$.
4: $\mathcal{C}$ computes $k_{i,j} \leftarrow \mathsf{KDF}_0(i, d_j)$ and $m_{i,j} \leftarrow \mathsf{KDF}_1(i, d_j)$ for $i \in [1,N], j \in [1,n]$.
5: $\mathcal{C}$ adds $k_{i,j}$ to $\mathbb{K}_i$ and $m_{i,j}$ to $\mathbb{M}_i$ for $i \in [1,N], j \in [1,n]$.
6: $\mathcal{C}$ invokes $\mathsf{OKVS.Decode}(\Gamma, \mathbb{K}_i)$ to obtain ciphers $\{c_{i,j}\}_{i\in[1,N],j\in[1,n]}$.
7: $\mathcal{C}$ computes $p_{i,j} \leftarrow \mathsf{AES.Dec}(m_{i,j}, c_{i,j})$ for $i \in [1,N], j \in [1,n]$.
8: If $p_{i,j}$ starts with 0^ℓ (where ℓ is a security parameter), $\mathcal{C}$ parses $p_{i,j}$ as $0^\ell \parallel v_{i,j}$ and set $tf'_{i,j} \leftarrow v_{i,j}$. Otherwise, set $tf'_{i,j} \leftarrow 0$.
9: $\mathcal{C}$ returns $\{tf'_{i,j}\}_{i\in[1,N],j\in[1,n]}$ as the result.

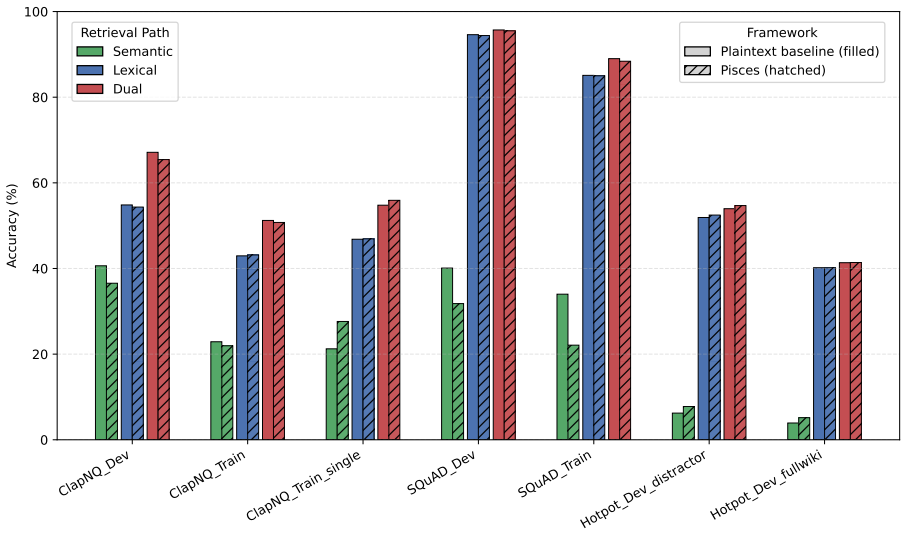

Figure 5: Top-10 Retrieval accuracy comparisons between `Pisces` and plaintext baseline over ground-truth.