RAGtime-PIANO: Efficient Secure Remote RAG

Antonia Januszewicz [1] **Jiachen Zhao** [1] **Meng Jiang** [1] **Taeho Jung** [1]

[1]

Abstract

Retrieval Augmented Generation (RAG) can enhance the performance of Large Language Models (LLMs) when used in conjunction with a comprehensive knowledge database. However, the space required to store the necessary information can be taxing when RAG is used locally. As such, the concept of RAG-as-a-Service (RaaS) has emerged, in which third-party servers can be used to process client queries via an external database. Unfortunately, using such a service would expose the client's query to a third party, making the product unsuitable for processing sensitive queries. Our scheme ensures that throughout the entire RAG processing, neither the query, any distances, nor retrieval information is known to the database hosting server. Using a two-pronged approach, we employ Fully Homomorphic Encryption (FHE) and Private Information Retrieval (PIR) to ensure complete security during RAG processing. FHE is used to maintain privacy during initial query processing, during which the query embedding is encrypted and sent to the server for k-means centroid scoring to obtain a similarity ranking. Then, a series of PIR queries is used to privately retrieve the centroid-associated embeddings and the top-ranked documents. A first-of-its-kind, lightweight, fully secure RAG protocol, RAGtime-PIANO, enables efficient secure RAG.

1. Introduction

As the popularity of RAG usage increases as an efficient way to improve the accuracy of Large Language Models (LLMs), so do concerns about privacy when using RAG, especially when combined with third parties. Lately, RaaS (RAG-as-a-Service) (Geniusee, 2024; Nuclia, 2024) has emerged as an option for those who lack the resources to operate RAG locally. If a client hands over a database for processing by a third party, it is understood that the third party is aware of its contents. Third parties may also offer RAG for clients who want to use a large, publicly available database published, stored, and processed by a third party for ease of use. However, even if the information stored in a database does not contain Personally Identifiable Information (PII), client queries can contain PII, as they may be intended for personal use. Even beyond the query itself, information about an individual's query can be leaked through either the query embedding (Morris et al., 2023; Zhuang et al., 2024) or the documents retrieved from the database. Even if the database is encrypted, clients' access patterns can still reveal information, compromising the privacy of their queries (Zhu et al., 2024).

A naïve solution would be for the client to compute the closest documents to their query locally using a local database of document embeddings, and then request the documents from the server. However, this method is suboptimal on two levels. First, the cost of calculating distances and maintaining the embedding database would be non-trivial, which is contrary to the RaaS business model, forcing clients to incur additional storage and computational costs. Second, the privacy of queries cannot be maintained if the server knows which documents are related to the queries.

In the literature, state-of-the-art protocols for completing the RAG process securely can be split into three categories: database restructuring protocols, partially secure protocols, and fully secure FHE-based protocols. Partially secure protocols such as Tiptoe (Henzinger et al., 2023b) and RemoteRAG (Cheng et al., 2025) are not considered fully secure because they leak information to the server about which documents are retrieved by the client. Though RemoteRAG has a fully secure protocol, the authors admit that the overhead and communication costs are prohibitively high as a real solution. Implementations that require the database to be stored as a graph or Oblivious RAM form, such as Compass (Zhu et al., 2024), GraphPIR (Wang et al., 2025), and PACMANN (Zhou et al., 2025), require significant database modifications and higher communication costs to traverse the database. Finally, for directly comparable works such as PIR-RAG (Wang et al., 2025), the heavy reliance on FHE results in poor database scaling. In summary, state-of-the-art solutions all share the limitations of limited security, heavy database modification, and/or poor scalability (See Table 1;

[1]Department of Computer Science, University of Notre Dame, South Bend, United States of America

Table 1. Comparison with related works.

Protocol	Primitives	No Embedding DB Download	No DB Modification	Complete Security
Compass (Zhu et al., 2024)	ORAM⋆	×	×	✓
RemoteRAG (Cheng et al., 2025)	DP† + FHE (Paillier)	✓	✓	×
Tiptoe (Henzinger et al., 2023a)	FHE (Regev)	✓	✓	×
GraphPIR (Wang et al., 2025)	PRF (PIANO)	✓	×	✓
PACMANN (Zhou et al., 2025)	PRF (PIANO)	✓	×	✓
PIR-RAG (Wang et al., 2025)	FHE (Paillier)	×	✓	✓
This work	PRF (PIANO) + FHE (CKKS)	✓	✓	✓

⋆ Oblivious Random Access Memory † Differential Privacy

a complete related work comparison is in Appendix C).

We implement a lightweight, scalable, and fully secure RAG that requires no database modifications or client-side storage. We achieve this secure process by combining Fully Homomorphic Encryption (FHE) with Private Information Retrieval (PIR) to form our novel two-step RAG protocol, which uses an Inverted File Index-Flat (IVF-Flat) structure. In this way, the server is privy only to FHE and PIR ciphertexts, which, by their indistinguishability, maintain the client's query privacy. The scheme is lightweight because we separate the RAG into two stages, enabled by the IVF-Flat structure, allowing us to first perform small-scale homomorphic FHE computations and then use an extremely scalable PIR scheme. Our novel two-stage design is more advantageous than existing approaches that use either PIR or FHE across the entire RAG process, because we can apply different primitives to different stages for better efficiency.

Our main contributions are as follows:

- We propose a novel two-stage fully secure encrypted RAG protocol that combines FHE and PIR for lightweight, quantum-secure retrieval without requiring significant client storage. The two-stage design limits use of expensive FHE to significantly improve efficiency.
- We provide a rigorous security definition, proof, and analysis based on the security primitives underlying our protocol.
- We provide a working implementation of our protocol, written in Python, that utilizes the CKKS (Cheon et al., 2017) scheme for FHE computations and the state-of-the-art non-FHE-based PIR scheme PIANO (Zhou et al., 2023), enabling faster, fully secure RAG. The anonymized source codes are released for reproducibility (Anonymous, 2026).
- We compare RAGtime-PIANO to PIR-RAG and GraphRAG, the state-of-the-art secure RAG protocols, to demonstrate that ours is superior with up to 323× lower communication costs and 40× lower latency with better accuracy.
- We provide retrieval task evaluations of RAGtime-PIANO over real-world benchmarks FEVER(Thorne et al., 2018a) and Natural Questions (NQ)(Kwiatkowski et al., 2019b), demonstrating accuracy comparable to plain insecure RAG.

2. Preliminaries

2.1. Retrieval Augmented Generation

Retrieval Augmented Generation (RAG) (Chase, 2022; Liu, 2022; Van Veen et al., 2023; Ram et al., 2023; Shi et al., 2024) can be used to improve LLM performance in questions that require the retrieval of additional information from external sources. RAG works by supplementing a user query with related documents from an external database. In our work, we select relevant documents using embeddings of the queries and database entries, which are vector representations of these elements. The distance between the query and database entry embeddings is calculated, and the documents with the closest embeddings to the query are chosen. These documents are then appended to the user query and fed into the LLM for a response (Lewis et al., 2020). For RAG, we introduce a basic notation:

- **RAG.Embed(e,M)** $\rightarrow$ **e'**: Given a string e and the embedding model **M**, generate e's embedding **e'**
- **RAG.Query(q,{doc},LLM)** $\rightarrow$ **r**: Given a query **q** and a set of documents **{doc}** fed into an LLM **LLM**, generate a response **r**.

2.2. Inverted File Index-Flat

Inverted File Index-Flat (IVF-Flat)(Jegou et al., 2010) is an Approximate Nearest Neighbor (ANN)(Li et al., 2020; Yang et al., 2017; Kalantidis & Avrithis, 2014) approach that accelerates vector-similarity searches within large databases by partitioning the database into smaller clusters and only searching the most relevant clusters to the query. ANN is integral to the RAG process, as it can be used to identify the most similar documents to the query and append them to the query during generation. Here, flat refers to the index structure used to store the information on disk, while IVF refers to the approach used to perform the ANN (Emanuilov & Dimov, 2024). For more information on IVF-Flat see Section B.1. For IVF-Flat, we introduce a basic notation:

- **IVF-Flat.Setup(DB',c)** $\rightarrow$ **C**: Given an embedding

database **DB'** and target number of centroids **c**, produce the set **C** of embedding centroids and mappings to the database entries associated with each centroid.
- **IVF-Flat.ANN(S,$\mathbf{q}_e$,k)** $\rightarrow \{\mathbf{idx}_s\}_{s\in\mathbf{S'}}$: Given a query embedding $\mathbf{q}_e$, target number **k**, and set of embeddings **S** produces the top-k nearest neighbours $\{\mathbf{idx}_s\}_{s\in\mathbf{S'}\subset\mathbf{S}}$.

2.3. Fully-Homomorphic Encryption

Fully Homomorphic Encryption (FHE)(Brakerski, 2012; Brakerski et al., 2014; Fan & Vercauteren, 2012) is a type of cryptography that allows for computations to be performed on encrypted data without requiring decryption. In an FHE scenario, once numerical data is encrypted, ciphertexts can be added and multiplied by other ciphertexts or plaintext numbers. Once the resultant ciphertext is decrypted, the result of the computations is revealed. In other words, arithmetics performed directly on ciphertexts are applied to the numbers inside the ciphertexts.

We use the CKKS scheme (Cheon et al., 2017), which is an FHE scheme that allows for approximate arithmetic on floating-point numbers, unlike other existing schemes like BGV(Brakerski et al., 2014) or BFV(Fan & Vercauteren, 2012). For more information on CKKS, see Section B.2. We provide a list of CKKS algorithms that we use below:

- **CKKS.Setup(pp,**1^λ**)** $\rightarrow$ **(pk,sk)**: Given a set of public parameters **pp** and security parameter 1^λ, generates public and secret keys **(pk,sk)**
- **CKKS.Enc(pk,x)** $\rightarrow [\![\mathbf{x}]\!]$: Encrypts given plaintext **x** using public key **pk** to get ciphertext of **x**, $[\![\mathbf{x}]\!]$.
- **CKKS.Dec(sk,** $[\![\mathbf{x}]\!]$**)** $\rightarrow$ **x**: Decrypts given ciphertext $[\![\mathbf{x}]\!]$ using secret key **sk** to get plaintext **x**.
- $\mathbf{x} \oplus [\![\mathbf{y}]\!] \rightarrow [\![\mathbf{x}+\mathbf{y}]\!]$: Addition of plaintext **x** and ciphertext $[\![\mathbf{y}]\!]$ to get encrypted sum $[\![\mathbf{x}+\mathbf{y}]\!]$
- $\mathbf{x} \otimes [\![\mathbf{y}]\!] \rightarrow [\![\mathbf{x}\times\mathbf{y}]\!]$: Multiplication of plaintext **x** and ciphertext $[\![\mathbf{y}]\!]$ to get encrypted product $[\![\mathbf{x}\times\mathbf{y}]\!]$

Note that additions and multiplications can be computed between ciphertexts in FHE schemes, but plaintext-ciphertext operations (i.e., the ones we listed above) are significantly more efficient than ciphertext-ciphertext operations, and we design our secure RAG carefully to avoid the expensive ciphertext-ciphertext operations.

2.4. Private Information Retrieval

Private Information Retrieval (PIR) (Chor et al., 1998; Henzinger et al., 2023b; Menon & Wu, 2022; Li et al., 2024; Mughees et al., 2021) is a cryptographic technique that enables private retrieval of information from third-party databases. Simply, PIR enables a client to query a database and receive the corresponding result without the database server knowing either the query or the result (Li et al., 2024).

In particular, we utilize the PIANO PIR scheme (Zhou et al., 2023), an extremely lightweight sublinear PIR scheme that does not require accessing all database elements to process a query. For more information on PIANO, see Section B.3. We provide a brief introduction to the basic notations below:

- **PIANO.Setup**$(1^\lambda,$ **DB)** $\rightarrow$ **p**: Completed offline by the client. Given a database **DB** and security parameter 1^λ, output a set of client parameters **p** for client storage, including the primary table, backup table, and replacement entries.
- **PIANO.Query(p, idx)** $\rightarrow$ **q**: The client uses parameters **p** and database index **idx** to generate query **q**.
- **PIANO.Response(DB,q)** $\rightarrow$ **r**: Given the query set **q**, the database **DB** returns response bits **r**.
- **PIANO.RecoverAndRefresh(p,idx,r)** $\rightarrow$ **doc**: Given client parameters **p**, query index **idx**, and response **r**, decrypt and return document **doc** and refresh for next query.

3. RAGtime-PIANO Description

3.1. Security Definition

Taking inspiration from traditional PIR security definitions (Zhou et al., 2023), we define the protocol security as:

Definition 3.1. A RAG protocol is defined to be secure if there exists a probabilistic polynomial-time simulator $\mathrm{Sim}(1^\lambda, n)$, such that for any probabilistic polynomial-time adversary $\mathbb{A}$ acting as the database hosting server, polynomially bounded n, and **DB**, $\mathbb{A}$'s views in the following two experiments are computationally indistinguishable:

- Real: An honest client interacts with $\mathbb{A}(1^\lambda, n, \mathbf{DB})$, who acts as the server. For every run of the protocol, $\mathbb{A}$ may choose the next query q for the client.
- Ideal: The simulated client, $\mathrm{Sim}(1^\lambda, n)$, interacts with $\mathbb{A}(1^\lambda, n, \mathbf{DB})$ who acts as the server. $\mathbb{A}$ may adaptively choose the next query q, but Sim randomly generates input for each step in the protocol communicated to the server.

This means that, under our definition of security, even if the server can choose the query for the client to use as input to the protocol, it will not be able to distinguish between protocol executions with that query and a random query. Simply put, if a RAG protocol is secure according to Definition 3.1, from the server's perspective, any two queries are indistinguishable, even when one of them is known.

3.2. Threat & Adversary Models

Assume a database **DB** which contains documents at each index. Another database, **DB'**, contains the vector embeddings corresponding to the documents in **DB** at corresponding indices. Assume both databases are publicly available.

We wish to process a RAG query, that is, find documents with the shortest distance from the client query while maintaining the privacy at two connected points:

1. The privacy of the query (that is, no information is leaked about the query, including the similarity distance from the query to any element in the database)
2. The privacy of the retrieval (that is, no information is leaked about what document is being retrieved from the database)

To do this, we assume an honest-but-curious adversary who controls both databases. Under the honest-but-curious adversary model, an adversary follows the specified protocols honestly but attempts to infer secret information during execution. In RAGtime-PIANO, our adversary is assumed to use what they have access to during the protocol to learn information about the query, including by trying to determine the retrieved documents, but will not deviate from the protocol itself.

3.3. Protocol Description of RAGtime-PIANO

RAGtime-PIANO has a query-independent pre-processing stage (Section 3.3.1), a query-dependent cluster level search & retrieval stage (Section 3.3.2), and a query-dependent document level search & retrieval stage (Section 3.3.3). The full description of RAGtime-PIANO is shown in Figure 2, and a diagram of the two-stage protocol that is run for each RAG query is provided in Figure 1.

3.3.1. Query-independent Pre-Processing

The preprocessing stage consists of the client setup and the server setup functions, and it is independent of the RAG queries. As PIANO does not require per-client storage or preprocessing, the client performs the PIANO preprocessing locally. As such, the database server can run and accept requests from any number of clients without additional preprocessing or storage. During the protocol, the client only needs to form the PIANO query, encrypt it, and send it to the server to receive the response. The PIANO setup may need to be performed again, depending on the number of client queries and the database size. As such, client setup will need to be repeated for every $\tilde{O}(\sqrt{n}/c)$ queries.

Similarly, the server setup also does not require per-client storage. To optimize FHE distance computations, in addition to the IVF-Flat setup, the server also pre-computes and stores the squared L2 norms of all centroids:

$$\|c_i\|^2 = \sum_{j=1}^{d} c_{i,j}^2$$

Once the setup is complete, it does not need to be modified for different clients or multiple queries. However, there is a per-database storage requirement on the client side, as the client must download a map of the clusters and their indices. We assume the map is publicly available to the client and that the indices of the embeddings and their corresponding documents are identical. The server setup is a one-time process that does not need to be repeated after pre-processing is complete.

3.3.2. Cluster Level Top-N Search and Retrieval with FHE

The client encrypts the query embedding q_e and its magnitude $||q_e||^2$ using the CKKS scheme by performing **CKKS.Enc** and sends $[\![q_e]\!]$ and $[\![\|q_e\|^2]\!]$ to the server. The server then computes the Euclidean distance homomorphically:

$$[\![D(q_e, c)]\!] \approx [\![\|q_e\|^2]\!] \oplus \|c\|^2 \oplus \Big(-2{<}[\![q_e]\!], c{>}\Big)$$

The dot product $-2{<}[\![q_e]\!], c{>}$ is computed as:

$$-2 \otimes \left(\sum_j \Big([\![q_e]\!][j] \otimes c[j] \Big) \right)$$

where j is the index of the embedding vector, and the summation is a series of homomorphic additions:

$$([\![q_e]\!][0] \otimes c[0]) \oplus ([\![q_e]\!][1] \otimes c[1]) \oplus ([\![q_e]\!][2] \otimes c[2]) \oplus \cdots$$

The resulting outcome is the set of encrypted Euclidean distances $\{[\![D(q_e, c)]\!]\}_{c \in \mathbf{C}}$ which is sent back to the client. Then, the client decrypts the data using their private key and uses the given distances to complete the **IVF-Flat.ANN** search to find the top-n closest centroids to the query. As the distance calculations are performed by the server, the client does not have to download the cluster embedding database, only receiving the smaller set of centroid distances.

3.3.3. Document Level Top-K Search and Retrieval with PIR

The client issues a set of PIR queries to retrieve the embeddings associated with the top n centroids from the embedding database. Once the embeddings are returned to the client, the client creates a smaller index and performs a local **IVF-Flat.ANN** search to obtain the indices of the top-k documents. The client then develops and sends a second set of PIR queries to retrieve the documents themselves. These documents can then be used directly to formulate the prompt based on the client's query and sent to the LLM.

3.4. IVF-FLAT Considerations

To securely compute the most relevant documents to the query using vector similarity search, we employed the IVF-Flat approach. Currently, the naïve FHE-kNN computation

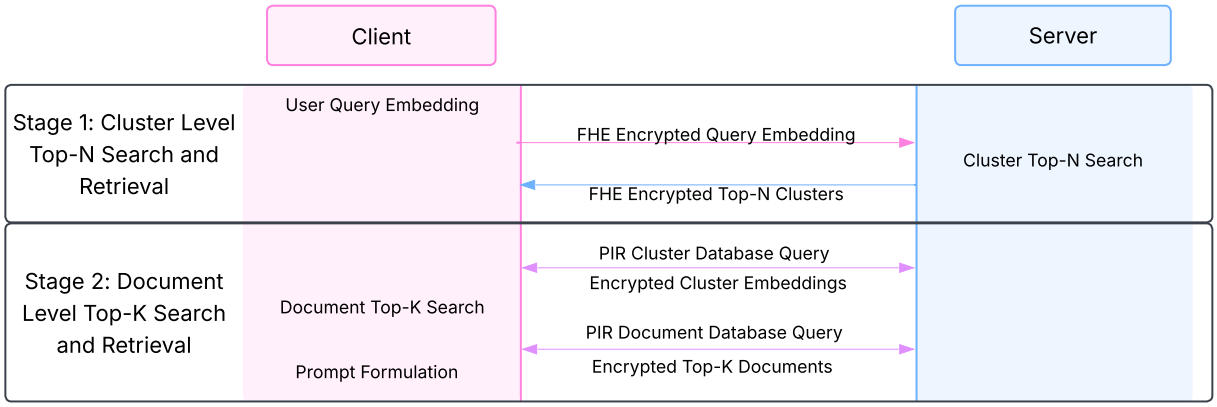


Figure 1. Overview of the per-query RAGtime-PIANO protocol. After a query-independent pre-processing stage, RAGtime-PIANO consists of two stages to process each query: 1) FHE-encrypted search for top-n cluster and encrypted database query for embeddings associated with the top-n clusters, and 2) local plaintext search for top-k and encrypted database query for top-k documents.

supports only $N < 15$ (Zama, 2023), and our database contains 100k entries; hence, we need to enable efficient FHE computation. When implementing IVF-Flat, we observed a significant effect of the number of clusters on computation time, as seen in Figure 4. As such, we limited clustering our 100k database into 4096 clusters.

The architecture of IVF-Flat makes our protocol FHE-friendly. When the client sends in the query, it sends the encrypted query embedding. The server computes vector similarities between the encrypted query and 4096 centroids, then returns the 4096 distances. The client then decrypts locally and selects the top-100 centroids. To balance the trade-off between speed and accuracy, as shown in Figure 3, we selected the top-100 centroids as the default option.

3.5. Security Analysis

Theorem 3.2. *RAGtime-PIANO satisfies security (i.e, Definition 3.1)*

Proof. We present a full proof in Appendix A and provide some high-level ideas here. As our protocol uses both PIR and FHE as black-box implementations, its security depends on the security of these primitives (as proven in (Cheon et al., 2017; Zhou et al., 2023)). In the first stage of the protocol, the query embedding is encrypted with FHE before the centroid distances are computed and returned to the client. Due to the indistinguishability of the underlying FHE scheme, neither the query nor the distance between the query and any given cluster is leaked to the server. That is, even if the server chooses a query, it will be unable to distinguish between an encryption of that chosen query embedding and an encryption of random data during the first stage of the protocol. Additionally, because any ciphertext is indistinguishable from any other, the distance results of the computations will not leak any information to the server.

In the second stage, as we can leverage the security guarantees of the underlying PIR, the use of PIR must be secure when retrieving documents. That is to say, the server cannot distinguish between a chosen and a random query of embeddings or documents, so security is maintained throughout the protocol. Even if the server chooses the query, it cannot determine whether it was used as input to the protocol. □

Table 2. Security Analysis showing RAGtime-PIANO in the highest level of security.

Method	No Security	Partial Security	Complete Security
Plaintext	✓		
RemoteRAG (Cheng et al., 2025)		✓	
Tiptoe (Henzinger et al., 2023a)		✓	
GraphPIR (Wang et al., 2025)			✓
PACMANN (Zhou et al., 2025)			✓
PIR-RAG (Wang et al., 2025)			✓
RAGtime-PIANO			✓

4. Evaluation

Full experiment setup can be found in Appendix Section E.

4.1. Security Guarantees Comparison

A formal comparison of the security of secure RAG protocols is difficult due to a lack of a uniform definition of security for secure RAG. To this end, we have separated comparable protocols into two categories, seen in Table 2: complete and partial security. Two papers, RemoteRAG and Tiptoe, define security more loosely, allowing the server to narrow down information about the retrieved database entries. Tiptoe does this by allowing the server to know which chunks of the database are retrieved after chunking, and RemoteRAG does this by providing the server a larger DP-generated list of similar documents.

Of the remaining protocols, none contains a formal security

Parameters: parties $\mathcal{S}$ and $\mathcal{C}$ corresponding to the server and the client, respectively. We denote the query held by the client as q and the database held by the server as **DB**.
Query-independent Pre-processing (run in parallel) (Section 3.3.1):

- Server Pre-Processing (every $\tilde{O}(c/\sqrt{n})$ queries)
  1. $\mathcal{S}$ generates the embedding database **DB'** by taking chosen embedding model M and running **RAG.Embed** for each element from the database $e \in$ **DB** producing $e' \in$ **DB'**.
  2. Then, to generate the cluster database $\mathbf{DB}_c$ the server runs **IVF-Flat.Setup** with **DB'** and target cluster number c.
  3. The final step of the server pre-processing is to store the square of the magnitude of the cluster embeddings and enter those into $\mathbf{DB}_c$.The mapping of the clusters to the associated indices in the document database is sent to $\mathcal{C}$.
- Client Pre-Processing (once per database)
  1. $\mathcal{C}$ runs the **CKKS.Setup** using the chosen parameters pp to get the public key pk and secret key sk
  2. Then, $\mathcal{C}$ also runs the **PIANO.Setup** using a security parameter λ_{PIANO} and a temporarily stored copy of the database **DB** to get a stored set of stored parameters $param_{PIANO}$.

Stage One: Cluster Level Top-N Search and Retrieval with FHE (Section 3.3.2):

1. $\mathcal{C}$ processes q by taking embedding model M and running **RAG.Embed** to produce the query embedding q_e. $\mathcal{C}$ also calculates $||q_e^2||$ and encrypts both values using the public key pk and function **CKKS.Enc** to produce the tuple $(\llbracket q_e \rrbracket, \llbracket ||q_e^2|| \rrbracket)$ of encrypted q_e and encrypted $||q_e^2||$ sent to the server $\mathcal{S}$.
2. $\mathcal{S}$ receives $(\llbracket q_e \rrbracket, \llbracket ||q_e^2|| \rrbracket)$, and for each of the centroid embeddings c calculates the encrypted distance $\llbracket d(q_e, c) \rrbracket = ||c^2|| + \llbracket ||q_e^2|| \rrbracket - 2 < c, \llbracket q_e \rrbracket >$ using **CKKS**, part of the **IVF-Flat.ANN** function. This set of encrypted distances $\{\llbracket d(q_e, c) \rrbracket\}_{c \in \mathbf{C}}$ is sent to the client $\mathcal{C}$.
3. $\mathcal{C}$ receives $\{\llbracket d(q_e, c) \rrbracket\}_{c \in \mathbf{C}}$ and uses sk to run **CKKS.Dec** and retrieve the plaintext distances $\{d(q_e, c)\}_{c \in \mathbf{C}}$. Once they are retrieved, $\mathcal{C}$ completes the **IVF-Flat.ANN** to find the top-n closest clusters $\mathbf{C}' \subset \mathbf{C}$ and, using the stored map of cluster to database indices, produces the set of indices $\{idx_c\}_{c \in \mathbf{C}'}$ belonging to the elements associated with those closest clusters.

Stage Two: Document Level Top-K Search & Retrieval with PIR (Section 3.3.3):

1. For each element of the set of indices $\{idx_c\}_{c \in \mathbf{C}'}$ and $params_{PIANO}$, $\mathcal{C}$ constructs q_{idx_c} = **PIANO.Query**$(idx_c, params_{PIANO})$ and sends it to $\mathcal{S}$. $\mathcal{S}$ sends back elements $r_e =$ **PIANO.Response(DB')**$, q_{idx_c})$. $\mathcal{C}$ then processes **PIANO.RecoverAndRefresh** for each r_e to get the embedding set E.
2. $\mathcal{C}$ runs **IVF-Flat.ANN** over the set of retrieved embeddings E and q' to get the top-k closest embeddings from that set and constructs the set $\{idx_e\}_{e \in E'}$ where $E' \subset E$.
3. For each element of the set of indices $\{idx_e\}_{e \in E'}$ and $params_{PIANO}$, $\mathcal{C}$ constructs a query for each element q_{idx_e} = **PIANO.Query**$(idx_e, params_{PIANO})$ and sends it to $\mathcal{S}$. $\mathcal{S}$ sends back r_{doc_e} = **PIANO.Response(DB')**$, q_{idx_e})$. $\mathcal{C}$ then processes **PIANO.RecoverAndRefresh** for that r_{doc_e} to get the set of documents $\{doc_e\}_{e \in E'}$.
4. $\mathcal{C}$ takes the set of document $\{doc_e\}_{e \in E'}$ and query q and runs **RAG.Query** using a chosen LLM model.

Figure 2. Full RAGtime-PIANO protocol

definition, but we have determined that, given their use of underlying primitives and their assumptions, these protocols would fit under our definition. In this case, GraphPIR, a PACMANN-inspired Python RAG implementation, and PACMANN itself both rely on PIANO and, therefore, OWF for their security. An assumption can also be made that PACMANN and GraphPIR share the same threat model, though the PACMANN paper does not present one. GraphPIR and PIR-RAG share the same threat model, similar to the one we defined. PIR-RAG has both a Paillier implementation and one using an LWE Linearly FHE scheme. All three comparable protocols fit under the same threat model and may fit under the same security definition, but a formal proof would be required for a formal comparison.

4.2. Accuracy Evaluation

To evaluate our accuracy, we first randomly sampled 100 queries from the FEVER and NQ datasets (Thorne et al., 2018b; Kwiatkowski et al., 2019a) and computed the ground-truth indices of the top-10 documents retrieved from the database. Then, we computed the unprotected, plaintext version of IVF-Flat retrieval and its accuracy against ground truth on the original 100k database. The results (Table 3) show that the secure protocol has a slight drop in accuracy compared to plaintext, while maintaining sufficient utility.

Additionally, we compared our protocols with those in related work (Wang et al., 2025) in Table 4. Due to the high latency of these comparative protocols and their hard-coded limitations, we constructed a smaller database containing

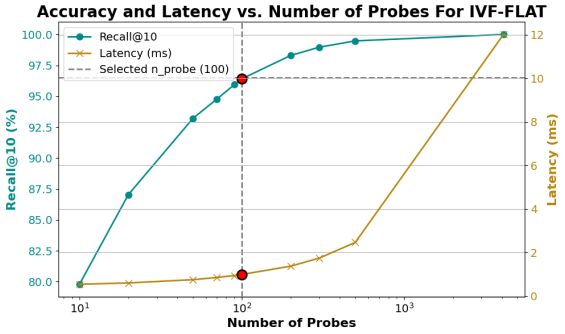


Figure 3. Accuracy and Latency vs. Number of Probes for IVF-FLAT speed-up: the accuracy–efficiency trade-off in our IVF-Flat protocol showing the reasoning behind our choice of probe represented by the dotted line.

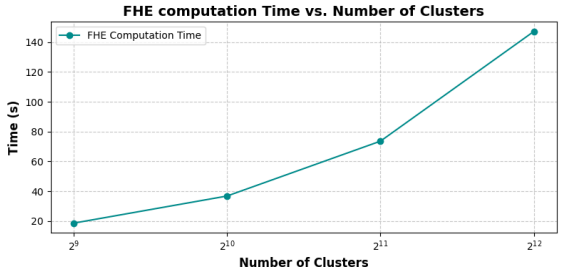


Figure 4. Time-Scaling of FHE computation with number of clusters, showing our reasoning behind cluster choices.

1000 entries sampled from the original 100k database, with the ground-truth most-related document included for each query. We used the three comparative protocols described in their paper and compared our performance against theirs. The ground truth for the tasks is established by retrieving the same queries in plaintext on the 1k database. We report both the hit ate at top-10, which measures whether at least one relevant document is retrieved, and the Accuracy for top-10, which measures agreement with the plaintext ground-truth top-10 ranking. While hit rate reflects retrieval success, reduced Top-10 Accuracy directly indicates greater deviation from plaintext retrieval behavior. A protocol may therefore exhibit a high hit rate but lower Top-10 Accuracy when it reliably identifies the most relevant document but is less precise in distinguishing among lower-relevance candidates within the top-10 ranking.

Note that our performance in Table 4 differs from that in Table 3 because we used a carefully constructed much smaller database for Table 4. This was inevitable because the current state-of-the-art secure RAG does not support large databases. We adjusted our protocol to reflect the database size drop, and therefore, the accuracy results between the two tables are not directly comparable. Additionally, we note that we only report retrieval accuracy rather than end-to-end LLM performance because our cryptographic contribution affects only the retrieval protocol. Metrics at the retrieval level provide a direct comparison of our performance with state-of-the-art cryptographic retrieval protocols, agnostic of choices on the reader model, prompting strategy, and/or generation process. Overall, the accuracy evaluation in Table 4 shows that our protocol provides greater utility than other existing PIR-based retrieval protocols.

Table 3. Comparison of plaintext IVF-Flat retrieval and our secure FHE+PIR+IVF-Flat system on the 100k database showing how additional security affects the accuracy of retrieval.

	Centroid Overlap (%)	**NQ Top-10 (%)**	**FEVER Top-10 (%)**
IVF-Flat (Insecure)	N/A	93	84.8
FHE+PIR+IVF-Flat (Secure)	75.6	88	80

Table 4. Top-10 retrieval performance on NQ and FEVER. We report Hit Rate (Hit) and Accuracy (Acc.) in % and show RAGtime-PIANO has both the highest hit rate and accuracy for both columns.

	NQ Top-10		**FEVER Top-10**	
Method	**Hit**	**Acc.**	**Hit**	**Acc.**
RAGtime-PIANO (Ours)	**100**	**94.6**	**100**	**95.8**
PIR-RAG (Wang et al., 2025)	96	36.2	97	35.6
Tiptoe (Henzinger et al., 2023a)	87	54.4	82	57.1
GraphPIR (Wang et al., 2025)	54	9.8	44	8.4

4.3. Comparison and Scaling Analysis

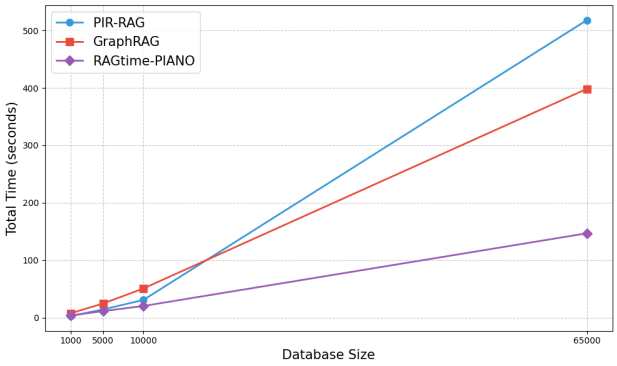


Figure 5. Total Per-Query Time vs Database Size. Here, RAGtime-PIANO shows significantly improved scaling due to the IVF-FLAT clustering and the excellent scaling of PIANO.

In Figure 5, we compare the timing of the per-query RAG processing with the database scale. Although our protocol does not perform as well on smaller databases, the reduction in retrieved entries, thanks to FHE, combined with the scaling of PIANO, is particularly evident for databases with more than 10,000 entries. It should be noted that the numbers collected for PIR-RAG use a default clustering size. When using 4096 clusters for the 65,000-entry database, their total RAG processing time is approximately 1.5 hours.

Another benefit of PIANO is that it does not involve ex-

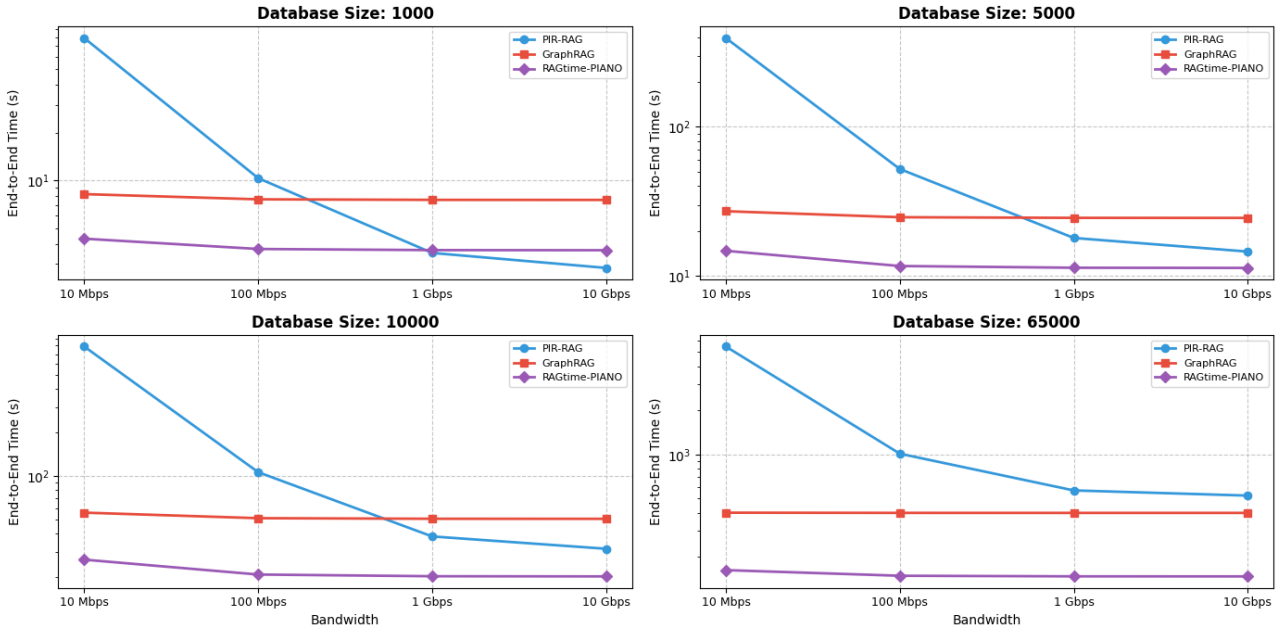


Figure 6. End-to-end latency on various network bandwidths and database sizes. RAGtime-PIANO shows better latency than PIR-RAG and GraphRAG on larger databases.

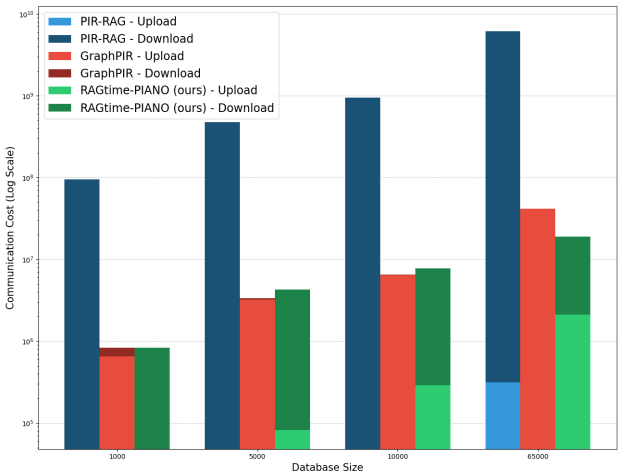


Figure 7. Per-Query Upload and Download Bytes vs Database Size. RAGtime-PIANO scales well for upload and download as the database size increases, showing communication costs similar to those of GraphPIR for smaller databases.

pensive FHE ciphertexts, thereby reducing communication costs, as shown in Figure 7. Our protocol also does not require retrieving more than a limited number of embeddings or the k-documents themselves. To better illustrate the differences among the protocols, we calculated the logs of both upload and download costs. During the upload process, RAGtime-PIANO outperforms GraphPIR because it requires less input for the multi-step traversal of the database graph. During the download process, RAGtime-PIANO outperforms PIR-RAG because PIR-RAG downloads all the documents associated with the top centroid cluster. Although our protocol incurs a higher download cost than GraphPIR and a higher upload cost than PIR-RAG, it achieves a significantly lower total cost than both GraphPIR and PIR-RAG. As such, for the 65,000-entry database, our protocol requires a total communication cost 323× lower than PIR-RAG and almost half of GraphPIR. In Figure 6, we show the end-to-end latency of RAGtime-PIANO across different bandwidths and database sizes. Here, RAGtime-PIANO shows superior end-to-end performance at higher bandwidths on larger databases.

5. Conclusion

In conclusion, our Privacy-Preserving RAG presents a novel, fully secure, privacy-preserving protocol that is both efficient and accurate compared to current state-of-the-art approaches, achieving speeds up to 40× faster, 323× lower communication costs, and significantly improved accuracy. To achieve this, we introduced IVF-Flat clustering to enable efficient FHE computation while avoiding the heavy storage required on the client side. Additionally, we used PIANO PIR, a sublinear PIR scheme made to scale for larger databases. We provide a threat model and a security definition, and demonstrate that, thanks to the secure underlying primitives, our protocol is secure under that definition. Due to the lack of per-client server storage and precomputation, our use case is ideal for large companies that wish to offer a secure RAG service to mobile devices without seeing user data.

Impact Statement

RAGtime-PIANO provides significant improvements in secure RAG efficiency and accuracy. By enabling improved scaling, we present a RAG option that is secure and viable for real-world RaaS scenarios. Other works in the same field either lack full privacy or require significant additional storage or overhead. Using a two-stage clustering and PIR approach, RAGtime-PIANO ensures that at no point during RAG processing is the client's information leaked to the server, either directly through information in the query or indirectly through the embedding or retrieved documents. Additionally, RAGtime-PIANO manages security while requiring minimal storage and eliminating the need for extensive server-side database preprocessing. Because all of the FHE processing can be reduced to plaintext-ciphertext operations, RAGtime-PIANO avoids the significant overhead of bootstrapping and ciphertext-ciphertext multiplication incurred by similar protocols. As a result, our work enhances the privacy and efficiency of a fully secure RAG, contributing to the broader adoption of security for RaaS scenarios.

References

Anonymous. Anonymous Source Codes of RAGtime-PIANO, January 2026. URL https://github.com/jennazhao7/PIRRAG.

Benaissa, A., Retiat, B., Cebere, B., and Belfedhal, A. E. Tenseal: A library for encrypted tensor operations using homomorphic encryption, 2021.

Boemer, F., Lao, Y., Cammarota, R., and Wierzynski, C. ngraph-he: a graph compiler for deep learning on homomorphically encrypted data. In *Proceedings of the 16th ACM International Conference on Computing Frontiers*, CF '19, pp. 3–13, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450366854. doi: 10.1145/3310273.3323047. URL https://doi.org/10.1145/3310273.3323047.

Brakerski, Z. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Annual cryptology conference*, pp. 868–886. Springer, 2012.

Brakerski, Z., Gentry, C., and Vaikuntanathan, V. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.

Byun, J., Park, S., Choi, Y., and Lee, J. Efficient homomorphic encryption framework for privacy-preserving regression. *Applied Intelligence*, 53(9):10114–10129, 2023.

Chase, H. Langchain. https://github.com/langchain-ai/langchain, October 2022.

Cheng, Y., Zhang, L., Wang, J., Yuan, M., and Yao, Y. RemoteRAG: A privacy-preserving LLM cloud RAG service. In Che, W., Nabende, J., Shutova, E., and Pilehvar, M. T. (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. URL https://aclanthology.org/2025.findings-acl.197/.

Cheon, J. H., Kim, A., Kim, M., and Song, Y. Homomorphic encryption for arithmetic of approximate numbers. In *International conference on the theory and application of cryptology and information security*, pp. 409–437. Springer, 2017.

Chor, B., Kushilevitz, E., Goldreich, O., and Sudan, M. Private information retrieval. *Journal of the ACM (JACM)*, 45(6):965–981, 1998.

Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P.-E., Lomeli, M., Hosseini, L., and Jégou, H. The faiss library. 2024.

Emanuilov, S. and Dimov, A. Billion-scale similarity search using a hybrid indexing approach with advanced filtering. *Cybernetics and Information Technologies*, 24(4):45–58, 2024.

Fan, J. and Vercauteren, F. Somewhat practical fully homomorphic encryption. *Cryptology ePrint Archive*, 2012.

Geniusee. Rag as a service. https://geniusee.com/retrieval-augmented-generation, 2024.

Henzinger, A., Dauterman, E., Corrigan-Gibbs, H., and Zeldovich, N. Private web search with tiptoe. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, pp. 396–416, New York, NY, USA, 2023a. Association for Computing Machinery. ISBN 9798400702297. doi: 10.1145/3600006.3613134. URL https://doi.org/10.1145/3600006.3613134.

Henzinger, A., Hong, M. M., Corrigan-Gibbs, H., Meiklejohn, S., and Vaikuntanathan, V. 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.

Jegou, H., Douze, M., and Schmid, C. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010.

Kalantidis, Y. and Avrithis, Y. Locally optimized product quantization for approximate nearest neighbor search. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2321–2328, 2014.

Kim, A., Song, Y., Kim, M., Lee, K., and Cheon, J. H. Logistic regression model training based on the approximate homomorphic encryption. *BMC medical genomics*, 11 (Suppl 4):83, 2018.

Kim, M., Song, Y., Li, B., and Micciancio, D. Semi-parallel logistic regression for gwas on encrypted data. *BMC Medical Genomics*, 13(Suppl 7):99, 2020.

Koirala, N., Paik, S., Martin, S., Berens, H., Januszewicz, T., Takeshita, J., Seo, J. H., and Jung, T. Select-then-compute: Encrypted label selection and analytics over distributed datasets using fhe. *NDSS*, 2025.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Kelcey, J., Devlin, J., and Lee, K. Natural questions: a benchmark for question answering research. In *Transactions of the Association for Computational Linguistics*, volume 7, pp. 453–466, 2019a.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Kelcey, M., Devlin, J., Lee, K., Toutanova, K. N., Jones, L., Chang, M.-W., Dai, A., Uszkoreit, J., Le, Q., and Petrov, S. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*, 2019b.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

Li, B., Micciancio, D., Raykova, M., and Schultz-Wu, M. Hintless single-server private information retrieval. In *Annual International Cryptology Conference*, pp. 183–217. Springer, 2024.

Li, W., Zhang, Y., Sun, Y., Wang, W., Li, M., Zhang, W., and Lin, X. Approximate nearest neighbor search on high dimensional data — experiments, analyses, and improvement. *IEEE Transactions on Knowledge and Data Engineering*, 32(8):1475–1488, 2020. doi: 10.1109/TKDE.2019.2909204.

Liu, J. LlamaIndex, November 2022. URL https://github.com/jerryjliu/llama_index.

Lyubashevsky, V., Peikert, C., and Regev, O. On ideal lattices and learning with errors over rings. In *Annual international conference on the theory and applications of cryptographic techniques*, pp. 1–23. Springer, 2010.

Martin, S., Koirala, N., Berens, H., Rozgonyi, T., Brody, M., and Jung, T. Hydia: Fhe-based facial matching with hybrid approximations and diagonalization. *Proceedings on Privacy Enhancing Technologies*, 2025.

Menon, S. J. and Wu, D. J. Spiral: Fast, high-rate single-server pir via fhe composition. In *2022 IEEE symposium on security and privacy (SP)*, pp. 930–947. IEEE, 2022.

Morris, J., Kuleshov, V., Shmatikov, V., and Rush, A. M. Text embeddings reveal (almost) as much as text. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12448–12460, 2023.

Mughees, M. H., Chen, H., and Ren, L. Onionpir: Response efficient single-server pir. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, pp. 2292–2306, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384544. doi: 10.1145/3460120.3485381. URL https://doi.org/10.1145/3460120.3485381.

Nuclia. Nuclia, the all-in-one rag as a service platform. https://nuclia.com/rag-as-a-service, 2024.

Park, S., Lee, J., Cheon, J. H., Lee, J., Kim, J., and Byun, J. Security-preserving support vector machine with fully homomorphic encryption. *SafeAI@ AAAI*, 2301, 2019.

Park, S., Byun, J., Lee, J., Cheon, J. H., and Lee, J. He-friendly algorithm for privacy-preserving svm training. *IEEE Access*, 8:57414–57425, 2020.

Park, S., Byun, J., and Lee, J. Privacy-preserving fair learning of support vector machine with homomorphic encryption. In *Proceedings of the ACM Web Conference 2022*, pp. 3572–3583, 2022.

Rajput, S., Mehta, N., Singh, A., Hulikal Keshavan, R., Vu, T., Heldt, L., Hong, L., Tay, Y., Tran, V., Samost, J., et al. Recommender systems with generative retrieval. *Advances in Neural Information Processing Systems*, 36: 10299–10315, 2023.

Ram, O., Levine, Y., Dalmedigos, I., Muhlgay, D., Shashua, A., Leyton-Brown, K., and Shoham, Y. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331, 2023.

Shi, E., Aqeel, W., Chandrasekaran, B., and Maggs, B. Puncturable pseudorandom sets and private information retrieval with near-optimal online bandwidth and time. In *Advances in Cryptology – CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part*

IV, pp. 641–669, Berlin, Heidelberg, 2021. Springer-Verlag. ISBN 978-3-030-84258-1. doi: 10.1007/978-3-030-84259-8_22. URL https://doi.org/10.1007/978-3-030-84259-8_22.

Shi, W., Min, S., Yasunaga, M., Seo, M., James, R., Lewis, M., Zettlemoyer, L., and Yih, W.-t. Replug: Retrieval-augmented black-box language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8371–8384, 2024.

Thorne, J., Vlachos, A., Christodoulopoulos, C., and Mittal, A. FEVER: a large-scale dataset for fact extraction and VERification. In *NAACL-HLT*, 2018a.

Thorne, J., Vlachos, A., Christodoulopoulos, C., and Mittal, A. 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 (NAACL-HLT)*, pp. 809–819, 2018b.

Van Veen, D., Van Uden, C., Blankemeier, L., Delbrouck, J.-B., Aali, A., Bluethgen, C., Pareek, A., Polacin, M., Reis, E. P., Seehofnerova, A., et al. Clinical text summarization: adapting large language models can outperform human experts. *Research Square*, pp. rs–3, 2023.

Wang, B., Lou, Q., Zheng, M., and Zhao, D. Pir-rag: A system for private information retrieval in retrieval-augmented generation, 2025. URL https://arxiv.org/abs/2509.21325.

Wang, Y., Ma, H., and Wang, D. Z. Lider: An efficient high-dimensional learned index for large-scale dense passage retrieval. *arXiv preprint arXiv:2205.00970*, 2022.

Xiao, S., Liu, Z., Zhang, P., and Muennighoff, N. C-pack: Packaged resources to advance general chinese embedding, 2023.

Yang, J., Zhao, W.-L., Deng, C.-H., Wang, H., and Moon, S. Fast nearest neighbor search based on approximate k-nn graph. In *International Conference on Internet Multimedia Computing and Service*, pp. 327–338. Springer, 2017.

Zama. Concrete-ml documentation. https://docs.zama.ai/concrete-ml, 2023. Accessed: 2025-xx-xx.

Zhou, M., Park, A., Shi, E., and Zheng, W. Piano: Extremely simple, single-server PIR with sublinear server computation. Cryptology ePrint Archive, Paper 2023/452, 2023. URL https://eprint.iacr.org/2023/452.

Zhou, M., Shi, E., and Fanti, G. Pacmann: Efficient private approximate nearest neighbor search. In Yue, Y., Garg, A., Peng, N., Sha, F., and Yu, R. (eds.), *International Conference on Representation Learning*, volume 2025, pp. 22791–22811, 2025.

Zhu, J., Patel, L., Zaharia, M., and Popa, R. A. Compass: Encrypted semantic search with high accuracy. Cryptology ePrint Archive, Paper 2024/1255, 2024. URL https://eprint.iacr.org/2024/1255.

Zhuang, S., Koopman, B., Chu, X., and Zuccon, G. Understanding and mitigating the threat of vec2text to dense retrieval systems. In *Proceedings of the 2024 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, pp. 259–268, 2024.

A. Theoretical Proofs

Theorem A.1. *Our secure RAG protocol satisfies security (i.e, Definition 3.1)*

Proof. We aim to show that the view of the adversary $\mathcal{A}$ who has chosen a query $q_{\mathcal{A}}$ as input for the protocol is indistinguishable from random input. Denote the distribution D_{FHE} as sampling a random set of real numbers forming a vector matching the dimension and scope set by the embedding model. Denote the distribution D_n as sampling a random set that draws distinct elements from $\{0, ..., n\}$ where n is the size of the database. The Ideal experiment is as follows:

- During the Cluster Level Top-N computation step, $\mathcal{A}$ chooses the query $q_{\mathcal{A}}$. The client processes the embedding q_A and sends the server a tuple of (a, b) randomly sampled from $\mathcal{D}_n$.
- During the Document Level Top-K computation step, $\mathcal{A}$ chooses sets $\{idx\}_{\mathcal{A}}$ and $\{idx'\}_{\mathcal{A}}$. For each element in the chosen sets, the client sends the **PIANO** queries associated with randomly selected $\{idx\}, \{idx'\} \leftarrow \mathcal{D}_n$.

We define hybrid experiment Hyb_1 as following:

- During the Cluster Level Top-N computation step, $\mathcal{A}$ chooses the query $q_{\mathcal{A}}$. The client processes the embedding q_A and sends the server a tuple of (a, b) randomly sampled from $\mathcal{D}_{FHE}$.
- During the Document Level Top-K computation step, $\mathcal{A}$ chooses sets $\{idx\}_{\mathcal{A}}$ and $\{idx'\}_{\mathcal{A}}$. For each element in the chosen sets, the client sends the **PIANO** queries associated with them to the server.

It is straightforward to prove that $\mathcal{A}$'s views in the Ideal and Hyb_1 are indistinguishable due to the Indistinguishability under Chosen-Plaintext Attack (IND-CPA) quality of CKKS. As in this case, in both Hyb_1 and Ideal, all that $\mathcal{A}$ is able to view are the CKKS ciphertexts; the views must be indistinguishable.

Without loss of generality, we define hybrid experiment Hyb_2 as following:

- During the Cluster Level Top-N computation step, $\mathcal{A}$ chooses the query $q_{\mathcal{A}}$. The client processes the embedding q_A and sends the server a tuple of (a, b) randomly sampled from $\mathcal{D}_{FHE}$.
- During the Document Level Top-K computation step, $\mathcal{A}$ chooses sets $\{idx\}_{\mathcal{A}}$ and $\{idx'\}_{\mathcal{A}}$. For each element in the chosen sets, the client sends the **PIANO** queries associated with randomly selected $\{idx\} \leftarrow \mathcal{D}_n$ and $\{idx'\}_{\mathcal{A}}$.

Here, the distribution of $\mathcal{A}$'s views in the Hyb_1 and Hyb_2 must be identical due to the strict considerations of the PIANO definition of security. Each successive round of PIANO for that specific instance of the scheme must follow this security definition.

Finally, we define hybrid experiment Real as following:

- During the Cluster Level Top-N computation step, $\mathcal{A}$ chooses the query $q_{\mathcal{A}}$. The client processes the embedding q_A and sends the server the tuple $([\![q_{\mathcal{A}}]\!], [\![||q_{\mathcal{A}}^2||]\!])$.
- During the Document Level Top-K computation step, $\mathcal{A}$ chooses sets $\{idx\}_{\mathcal{A}}$ and $\{idx'\}_{\mathcal{A}}$. For each element in the chosen sets, the client sends the **PIANO** queries associated with them to the server.

For the same reason as the previous step, here the views of $\mathcal{A}$ in Hyb_2 and Real are identically distributed. We have seen in this proof that Ideal $\approx \mathsf{Hyb}_1$, $\mathsf{Hyb}_1 \approx \mathsf{Hyb}_2$, and $\mathsf{Hyb}_2 \approx$ Real. Thus, by the transitive property, it must be true that Ideal and Real are computationally indistinguishable. □

B. Additional Background

B.1. IVF-Flat

The way IVF-Flat works is that once the documents in the database are processed into vector embeddings using a trained model, the vectors are then partitioned into Voronoi cells associated with a full-precision centroid vector. Thus, during an ANN search, the entire database need not be processed. Instead, each search selects the top few centroids based on similarity between the query and the centroids, then selects the top-k (or entire lists) associated with the centroids. After obtaining a large list of potential entries, another top-k selection is performed to identify the most likely candidates on the document vector embedding level (Rajput et al., 2023; Wang et al., 2022).

B.2. CKKS

The way that CKKS works is that it uses a Complex Canonical Embedding (CCE) along with a client's secret key in order to convert a plaintext float input into a polynomial. Since the polynomial is homomorphic to the plaintext space, computations on the ciphertext are reflected in the decrypted result. CKKS is based on the quantum-secure Ring Learning with Errors (RLWE) problem (Lyubashevsky et al., 2010) and has been used for other Privacy-Preserving Machine Learning (PPML) use cases (Kim et al., 2018; 2020; Boemer et al., 2019; Park et al., 2019; 2020; 2022; Byun et al., 2023). Unfortunately, like many FHE schemes, CKKS has difficulty with performing multiplication, which can be costly and may necessitate a costly resetting process called bootstrapping. As such, we seek to minimize the number of FHE multiplications in our protocol.

B.3. PIANO

The way PIANO works is that it stores a version of the database as a cluster of elements that have been bit XORed together. During the query, the client requests the XOR of all elements in the cluster, excluding the requested element. That way, during the recovery process, the client can XOR the cluster with the query result to get the element itself. PIANO relies on the One-Way Function (OWF) assumption, which the client uses as a secret key to generate the random clusters, for its security (Shi et al., 2021). Due to this clustering process and the lack of bulky ciphertexts, PIANO has improved scalability and can handle larger databases than many other PIR schemes.

C. Related work

A full comparison to related work is found in Table 1.

A field of research that is developing alongside our research is private search algorithms. These algorithms seek to allow users to retrieve links that match a private search query. Unfortunately, of the available protocols, Tiptoe (Henzinger et al., 2023a) and PACMANN (Zhou et al., 2025) are not directly suitable for RAG processing due to limitations. Tiptoe does not ensure complete security for the search, using a database chunking procedure that allows the server to know which chunks are retrieved. PACMANN, which uses similarity graphs, requires constructing a graph from the embedding database and performing remote graph traversal via PIR, which is not optimized for RAG.

Of the remaining related work, Compass (Zhu et al., 2024)

is a solution similar to PACMANN, where the server must construct a graph of the embedding database and perform remote graph traversal, this time via ORAM. Similar to Tiptoe, RemoteRAG does not provide complete security because it uses a search method that allows the server to narrow down which documents are retrieved from the database.

The main FHE-based secure RAG protocol, which can be directly compared to ours, is PIR-RAG (Wang et al., 2025), which also uses PIR and FHE to perform secure RAG. However, there are key differences that allow our protocol to scale better with database size and to mitigate the client processing required. When the document embeddings are generated and clustered, PIR-RAG requires the client to download a copy of the cluster database and process the clusters themselves. Following that, the client then requests the documents associated with those clusters using an FHE-based PIR scheme and performs a local re-ranking. In opposition, our protocol does not require the client to store any version of the document or embedding database. All processing required to associate the query with the top clusters is also performed by the server. Finally, the client must process only the embeddings associated with the top clusters, which requires less memory and processing power, and retrieve the top-k documents from their initial query.

D. Discussion

D.1. Batching Optimization

When the client submits multiple queries at once, batching optimizations can improve the protocol's throughput. Unfortunately, due to limitations in the current CKKS Python and PIANO implementations, we were unable to include experiments demonstrating how batched processing would work (Benaissa et al., 2021). However, we will include the methods to achieve batching here. CKKS is a scheme that allows multiple plaintext messages to be batched and processed together (Cheon et al., 2017). As such, when the client sends queries, multiple queries can be batched into a single ciphertext. Additionally, there are multiple methods for batching matrix multiplication with CKKS and for processing final additions via hoisted rotations (Martin et al., 2025).

For PIANO, there are two main methods for batching. In the first place, the retrieved documents may be associated with multiple queries, especially if all the queries concern the same topic. In this case, the user can note the indices of these documents and perform a single retrieval. Additionally, the authors of PIANO, in another work, provided a method for batching PIANO queries to reduce communication overhead (Zhou et al., 2025).

D.2. Further FHE Optimization

The current focus of RAGtime-PIANO has been to minimize the number of PIR queries. However, several methods can be used to reduce the computational cost of CKKS processing. One method is to modify the default number of clusters per database, currently set using the information in Figure 4. During FHE processing, batching can be used within a single query, as described in the above section. When the query is sent in, the query embedding and its square magnitude can be batched together for improved communication. Then, when the results are returned, multiple results can be batched together. Currently, we use as much batching as possible for computing matrix multiplication; however, this is limited by the Python FHE library we used (Benaissa et al., 2021).

Though we do not directly address it in the main paper, it would be possible for the server to perform the complete **IVF-Flat.ANN** processing using a threshold method. However, the required computations for threshold computation with CKKS are complex and incur multiple ciphertext multiplications (Koirala et al., 2025), which can result in even more costly operations like bootstrapping. This would result in a much lower download cost for the client and slightly faster computations on the client side, at the expense of a slower overall RAG retrieval.

E. Full Experiment Setup

For our accuracy tests, we selected two benchmarking datasets: FEVER and Natural Questions (Thorne et al., 2018a; Kwiatkowski et al., 2019b). FEVER is a fact-verification dataset where each claim is supported or refuted by one or more specific Wikipedia sentences. Natural Question (NQ) consists of real Google queries, their corresponding Wikipedia pages, and short answers. In our setting, we ignore the classifier/LLM reader and evaluate only the retriever process.

We used the standard BGE embeddings with a dimension of 768 (Xiao et al., 2023), and the FHE computations were performed using the TenSEAL (Benaissa et al., 2021) library. We use the CKKS scheme with a polynomial modulus degree (ring dimension) of 8192 by default, and coefficient modulus chains of $[60, 40, 40, 60]$.All parameter settings provide at least 128-bit security under standard RLWE assumptions. For the IVF-Flat component, we used Meta's Faiss library (Douze et al., 2024). Due to spatial constraints, we currently return only the first 1024 characters of documents; however, modifications could be made to store larger documents as chunks and retrieve them for reconstitution. The Go code used for the PIANO implementation can be found at: `https://github.com/nd-dsp-lab/Piano-PIR-RAG-MODIFIED` and the

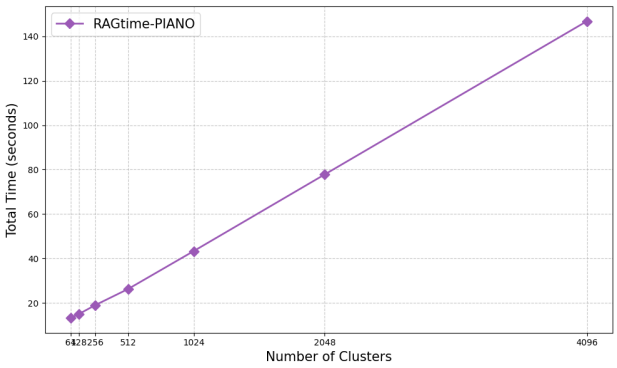


Figure 8. Total Per-Query RAG time vs Number of Clusters for Database of size 65,000. Increasing the number of clusters correlates linearly with the increased speed of the overall RAG.

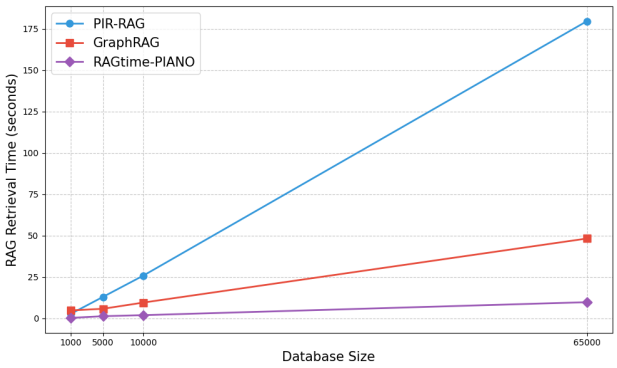


Figure 9. Per-Query RAG Retrieval vs Database Size. In this diagram, RAGtime-PIANO shows lower processing time due to the PIANO PIR queries, which are sublinear.

remaining Python source code can be found at: https://github.com/jennazhao7/PIRRAG.

Our comparison and scaling analysis were run using the source code provided by the PIR-RAG paper. To examine the timing and communication costs of our protocol, we compared it with PIR-RAG and the graph version of secure RAG, GraphPIR, a Python-based PACMANN RAG implementation (Wang et al., 2025). We used the source code provided by the authors of PIR-RAG to perform these experiments.

We conducted our timing experiments on an Intel Xeon W-1290P processor at 3.7 GHz with 20 cores, running in single-threaded mode. Both tests used a RAG database generated from Wikipedia entries from HuggingFace, available here: https://huggingface.co/royrin/wiki-rag.

F. Additional Experiments

In order to show how the number of clusters affects the total RAG time, we have included a separate experiment that shows the results for different clusterings for the 65k database. Combined with Figure 3, Figure 8 can be used to find the optimal number of clusters for a given required RAG speed. In Figure 8, the number of top-clusters that are retrieved from the database remains unchanged at 10. If more clusters were retrieved, accuracy could improve; however, that would further affect the number of PIR queries required.

To further demonstrate the effectiveness and scalability of PIR, we also compare the direct document retrieval performance of PIR-RAG, GraphPIR, and RAGtime-PIANO. Here, it is more clearly visible how directly PIANO is involved in maintaining the speed of the RAG for RAGtime-PIANO, and how retrieval-heavy FHE-based RAG schemes like PIR-RAG compare to other methods.