Single Round Private Top-K Semantic Searchable Encryption

***Abstract*—In this paper**

***Index Terms*—Searchable Symmetric Encryption, Xi**

I. Introduction

WITH the rapid advancement of Large Language Models (LLMs), local deployment has emerged as a prevalent paradigm to safeguard data privacy. However, challenges such as limited knowledge coverage, hallucination, biases, and discriminatory outputs persist in LLMs. To address these issues, Retrieval-Augmented Generation (RAG) has been introduced. This technique enhances model performance by retrieving contextually relevant information from external knowledge bases for specific queries and integrating it into the LLM's input, thereby significantly improving response accuracy and reliability.

However, given the substantial storage demands, most organizations opt to store their data in the cloud. Given these privacy concerns, encryption becomes a mandatory safeguard for cloud-hosted data. Consequently, a critical challenge arises: enabling secure Retrieval-Augmented Generation (RAG) searches over encrypted cloud-based data. Addressing this challenge is essential for seamlessly integrating retrieved results into large language model (LLM) prompts while maintaining confidentiality and utility.

As a fundamental approach for querying encrypted cloud data, searchable encryption enables efficient ciphertext query and has emerged as a dominant paradigm in cloud storage systems. Modern implementations based on keyword indexing support sophisticated semantic search functionalities, including wildcard queries [1], multi-keyword search [2], conjunctive/disjunctive searches [3] and range query [4]. While these schemes offer diverse and mature solutions, it's important to highlight that they typically return only document identifiers containing matching keywords rather than the actual file contents. Consequently, users must engage in an additional interaction to retrieve the complete data.

Alternatively, Private Information Retrieval(PIR) techniques [5], [6] provide another mechanism for privacy-preserving cloud data queries. Compared to searchable encryption schemes, PIR emphasizes oblivious data access, enabling retrieval either by keywords or file locations while maintaining query privacy. However, their reliance on primitives like homomorphic encryption results in significantly higher communication costs compared to searchable encryption approaches.

While the integration of searchable encryption and private retrieval techniques can enable complex and secure semantic searches, this approach suffers from two critical limitations: (1) it necessitates two-round interactions, and (2) it fails to satisfy the stringent semantic accuracy requirements of modern Retrieval-Augmented Generation (RAG) systems. Traditional keyword-based retrieval—even with advanced constraints—cannot adequately address nuanced semantic disambiguation. For instance, the term "apple" may denote either the fruit or the technology company, a distinction that purely lexical analysis cannot reliably resolve. Such limitations inevitably introduce indexing inaccuracies and retrieval biases, compromising RAG performance. These challenges motivate our core research question: How can we design an efficient single-round semantic searchable encryption scheme specifically optimized for RAG applications?

We propose an affirmative solution to this challenge through a novel framework that combines vector embeddings with secure multi-party computation. Our approach consists of three key components: (1) representing data as high-dimensional vectors using neural embeddings, (2) developing an extended two-party comparison protocol with enhanced security guarantees, and (3) designing an efficient inner product computation algorithm for accurate similarity measurement. These innovations collectively enable our searchable encryption framework to support secure, single-round retrieval while maintaining semantic precision for RAG applications. Our principal contributions include:

- We present a augmented extended two-party comparison protocol for max specifically among many secretly shared data. The protocol can efficiently determine the maximal value and return the complete data entry corresponding to the identified maximum value while preserving data confidentiality. It allows substitution of the underlying

This work is supported by the National Natural Science Foundation of China (No.62072276, No.62302280), the Project Supported by Science Foundation of Hebei Normal University No.L2023B41, the S&T Program of Hebei No.22567606H, and the Natural Science Foundation of Shandong Province (No. ZR2023QF133).

Xi Zhang is with College of Computer and Cyber Security, Hebei Normal University, Shijiazhuang, 050024, China; Hebei Provincial Key Laboratory of Network and Information Security, Hebei Normal University, Shijiazhuang, 050024, China; School of Mathematics, Shandong University, Jinan, 250100, China.(e-mail: mathxz@163.com)

Cheng Huang is with School of Computer Science, Fudan University, 2005 Songhu Road, Yangpu District, Shanghai, 200438, China. (e-mail: chuang@fudan.edu.cn)

Ye Su is with School of Information Science and Engineering, Shandong Normal University, Jinan, 250358, China. She is also with School of Mathematics, Shandong University, Jinan, 250100, China.(e-mail: suye@sdnu.edu.cn)

Jing Qin is with School of Mathematics, Shandong University, Jinan, 250100, China.(e-mail: qinjing@sdu.edu.cn)

Corresponding author: Jing Qin.

comparison primitive, enabling optimization of online computation time.

- Building upon the extended two-party comparison protocol, we propose a single-round semantic searchable encryption scheme. In this scheme, vectors are clustered into multiple clusters, and a half-loop index structure based on secret sharing is introduced to support efficient search and matching. The scheme enables the retrieval of the top-K most similar results and is proven to be T-privacy.
- We conduct extensive simulation experiments and compare our solution with existing efficient schemes across different datasets in terms of time and accuracy. In the comparative experiments, we employ different two-party comparison protocols as foundational modules and perform multi-dimensional comparisons, demonstrating the efficiency and flexible of our scheme.

The remainder of the paper is organized as follows. Section II reviews related work. Section III formalizes the system model, adversarial model, security model, and design goals. Then, the preliminaries and the present augmented comparison protocol for max are proposed in Sections IV and V, followed by the construction and security proof in Sections VI and VII. The performance evaluation and comparison are shown in Section VIII. Finally, we draw the conclusion in Section **??**.

II. Related Work

Since Song et al. [7] first introduced the concept of searchable encryption (SE), the field has undergone significant advancements. Traditional SE schemes construct keyword-based indexes to enable search over encrypted data. To support more advanced semantic search capabilities, researchers have investigated various ways to represent and relate keywords. For instance, single-character (”_”) and multi-character (”%”) wildcards can be used to replace uncertain parts of keywords, enabling wildcard searchable encryption [1], [8]. When target documents contain multiple keywords, multi-keyword search techniques become essential. These approaches are often categorized based on the relationships between keywords. Boolean searchable encryption [2] supports queries with logical operators such as AND, OR, and NOT, while conjunctive and disjunctive searchable encryption [3] focus on set intersection and union, respectively. For dynamic scenarios requiring frequent index updates and complex semantic queries, mergeable searchable encryption [9] offers an effective solution.

However, although these keyword-based approaches ensure that the retrieved documents contain the specified keywords, they suffer from two fundamental limitations. First, they fail to resolve keyword ambiguity—where a single term may have multiple meanings depending on context (e.g., ”apple” referring to either the fruit or the technology company). Second, they rely on literal matching and lack the ability to comprehend the user's actual search intent. In essence, current methods retrieve documents based on the presence of exact keyword forms, rather than understanding their semantic meaning within context.

To overcome these limitations, semantic-aware encrypted [10] search has emerged as a promising direction. This approach leverages semantic models to enable conceptual-level similarity matching between queries and documents, moving beyond surface-level keyword matching. By incorporating techniques such as word embeddings and contextual language models, these systems can more accurately capture user intent and deliver more relevant search results—all while preserving the security of encrypted data.

In the field of semantic-aware encrypted search, recent research has leveraged several key technologies, including Word2Vec/Doc2Vec models [11], LDA models [12], [13], BERT models [10], [14], and knowledge graphs [15]. While traditional methods such as synonym expansion and WordNet-based extensions remain fundamentally keyword-driven and offer only limited improvements in retrieval accuracy, more advanced techniques have demonstrated significantly enhanced performance. For example, the conceptual graph approach improves semantic understanding by constructing vector spaces to represent documents and generating corresponding search vectors. Modern solutions based on Word2Vec/Doc2Vec, LDA, BERT, and knowledge graphs each offer distinct advantages in computational efficiency, topic modeling, deep semantic representation, and reasoning capabilities, respectively. At the same time, many of these semantic-aware approaches share similar cryptographic foundations and index structures with traditional searchable encryption (SE) schemes. For instance, Chen et al. [11] proposed a dual-layer index based on balanced binary trees, utilizing Word2Vec to generate distributed word vectors for semantic encrypted search. Similarly, Hu et al. [14] implemented privacy-preserving semantic search by encrypting document-topic and keyword-topic associations, which were derived from an enhanced BERT model incorporating TF-ITF weighting.

Despite significant advancements in semantic-aware encrypted search techniques through the integration of various deep learning models, these approaches still fundamentally rely on underlying keyword matching mechanisms. As a result, their retrieval accuracy remains limited and often falls short of meeting the high-precision semantic search requirements in retrieval-augmented generation (RAG) scenarios. Furthermore, both traditional searchable encryption and semantic retrieval schemes typically return file identifiers as search results. Even when the retrieved content is returned in plaintext, the exposure of file identifiers to cloud servers undermines the goal of preserving true data privacy.

The concept of Private Information Retrieval (PIR) was initially introduced by Benny et al. [16] to address the need for protecting user query privacy on public databases. Based on query strategies, PIR can be classified into two categories: index-based PIR [17] and keyword-based PIR [18], which are theoretically interconvertible. However, regardless of the classification, enhancing efficiency remains a central research challenge in the PIR domain. A widely adopted method to improve performance is Batch PIR [19], which allows users to retrieve multiple records within a single query, thereby reducing both communication rounds and computational overhead. Within the realm of single-server PIR, Simple PIR [5]

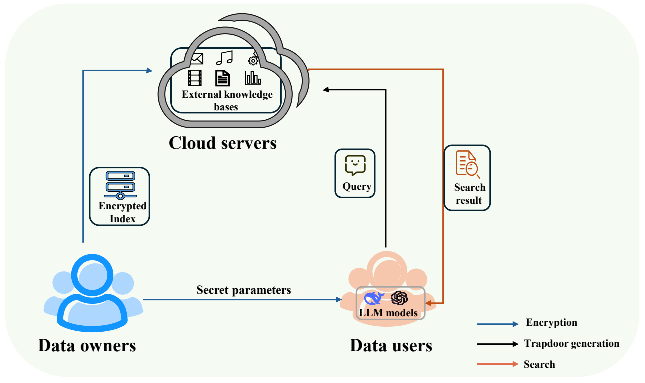

Fig. 1. The system model

has garnered significant attention. Constructed based on the Learning With Errors (LWE) assumption, Simple PIR acts as a foundational primitive for various PIR constructions. Another line of research emphasizes efficiency through controlled server-side data probing, which introduces acceptable privacy trade-offs to gain performance. Notable examples include Differentially Private PIR [20] and Distributional PIR [6]. The latter incorporates dataset popularity distributions to enhance the retrieval success rate for in-distribution queries while bounding privacy leakage.

It is important to highlight that traditional PIR schemes struggle to support semantic search capabilities. This limitation stems from their underlying design, which remains optimized for exact-match retrieval rather than for capturing and processing semantic relationships.

III. Models and Design Goals

A. System Model

The proposed private Top-K single round semantic search system is shown in Fig. This system is made up of by data owners(DOs), data users(DUs) and two cloud servers CS_1 and CS_2. Data owners upload their privacy data M in the way of secret sharing to cloud servers CS_1 and CS_2 and generate an index for data users to semantic search. Data users receives auxiliary information sent by the DO to help with semantic search. When a query happens, cloud servers received trapdoor from data users run a search protocol to find the top K chunks with the most similar semantics to the query. It is noted that both the chunk and index are stored in the secret sharing way, so data users upload trapdoor to two servers and receive chunk shares from them.

B. Adversarial Model & Security Model

1) Adversarial Model: In this system, data owners and data users are trusted. Cloud servers are honest but curious. It is assumed that either two cloud servers, or any cloud server and data user are not collude with each other. This assumption is based on the fact that data owners can choose different service providers and only grant the semantic search functionality to honest data users. Thus, both cloud servers and data users will merely follow the designed algorithms and protocol in our system but cloud servers meanwhile try their best to collect more private information. Specifically, cloud servers are curious about whether any two query is same and plain texts of chunks in encrypted index or search results for queries;

2) Security Model: To protect the privacy of our PTS system, we follow the framework and security model of an index-based searchable encryption scheme which contains four algorithms and a protocol.

- **Gen**$(\lambda) \to K$: A probabilistic key generation algorithm run by data owners on inputting a security parameter λ, outputs a secret key set K.
- **Enc**$(K, P, M) \to \mathbb{L}$: A probabilistic key generation algorithm run by data owners on inputting a secret key set K, a database M, a parameter set $P = \{\mathsf{K}, \mathsf{T}\}$ where K is the number of chunks in search result and T reflects the level of system security, outputs an encrypted index $\mathbb{L}$;
- **Trgen**$(q, K) \to \mathsf{t}$: A deterministic algorithm run by data users on inputting a query q and the key set K, outputs a trapdoor t of q.
- **Search**$(\mathsf{t}, K, \mathbb{L}) \to \mathbb{L}(q)$: A protocol run by cloud servers with inputting a trapdoor t, the key set K, and the encrypted index $\mathbb{L}$, outputs the search result $\mathbb{L}(q)$.
- **Dec**$(\mathbb{L}(q), K) \to \mathsf{m}$: A deterministic algorithm run by data users with inputting the search result $\mathbb{L}(q)$ and the key set K, outputs the responding data m.

We now present the definition of simulation-based security model, which is formalized using two games $\mathsf{Real}_{\mathcal{A}}(\lambda)$ and $\mathsf{Ideal}_{\mathcal{A},\mathcal{S}}(\lambda)$ [21]. Specifically, $\mathsf{Real}_{\mathcal{A}}(\lambda)$ is executed following the PTS defined above, while $\mathsf{Ideal}_{\mathcal{A},\mathcal{S}}(\lambda)$ is simulated using leakages in PTS. The leakages are defined with leakage functions $\mathcal{L} = (\mathcal{L}^{Enc}, \mathcal{L}^{Trgen}, \mathcal{L}^{Search})$, which captures information that could be collected and deduced by a probabilistic polynomial-time (PPT) adversary $\mathcal{A}$. If $\mathcal{A}$ cannot distinguish these two games, we can say that nothing except the defined leakages is leaked:

- $\mathsf{Real}_{\mathcal{A}}(\lambda)$: On inputting a database M chosen by $\mathcal{A}$, it outputs $\mathbb{L}$ by running **Gen**(λ) and **Enc**(K, P, M). Then, $\mathcal{A}$ can initiate queries and get the outputs of **Search**$(\mathsf{t}, K, \mathbb{L})$ as query results after invoking **Trgen**(q, K). At last, $\mathcal{A}$ outputs a bit.
- $\mathsf{Ideal}_{\mathcal{A},\mathcal{S}}(\lambda)$: On inputting a database M chosen by $\mathcal{A}$, a PPT simulator $\mathcal{S}$ outputs $\mathbb{L}$ with $\mathcal{L}^{Gen}, \mathcal{L}^{Enc}$. Then, it simulates the results of $\mathcal{A}$'s queries using $\mathcal{L}^{Trgen}, \mathcal{L}^{Search}$. At last, $\mathcal{A}$ outputs a bit.

Definition 1. *PTS is $\mathcal{L} - adaptively - secure$ if for every PPT adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}$ such that*

$$|Pr[\mathsf{Real}_{\mathcal{A}}(\lambda) = 1] - Pr[\mathsf{Ideal}_{\mathcal{A},\mathcal{S}}(\lambda) = 1]| \le negl(\lambda)$$

where $negl(\lambda)$ is a negligible function related to λ.

Definition 2. *(T-privacy) A $\mathcal{L} - adaptively - secure$ PTS achieves T-privacy if the leakage function $\mathcal{L}^{Gen}$, $\mathcal{L}^{Enc}$, $\mathcal{L}^{Trgen}$, $\mathcal{L}^{Search}$ can be written as*

$$\mathcal{L}^{Enc}(M) = \mathcal{L}'(\mathsf{N}, \mathsf{T}, \mathsf{P})$$
$$\mathcal{L}^{Trgen} = \mathcal{L}''(|\mathsf{B}|, |H|)$$
$$\mathcal{L}^{Search} = \mathcal{L}'''(\mathsf{T})$$

TABLE I
THE NOTATIONS

Notations	Descriptions
λ	the security parameters
N	the number of loops
K	the number of search results
T	the number of clusters accessed by cloud server each query
$\mathbb{L}$	the loop index
m_{ij}	the j-th chunK in cluster i
$[\cdot]$	the secret shares of a number or a vector
$[\cdot]_i$	the shares of cloud server CS_i
c_i	the center vector of a cluster
c	the center vector of clusters
P_1, P_2	the parties in protocols
$\mathsf{P.Enc}$	the encryption algorithm of Pailliar encryption
$\mathsf{S.Enc}, \mathsf{S.Dec}$	the encryption/decryption algorithm of Symmetric encryption

where $|\mathsf{B}|$ *is the length of the bloom filter and* $|\mathsf{H}|$ *is the number of the hash functions used in the trapdoor generation, then* T *represents the number of centroids accessed by cloud servers.*

C. Design Goals

The PTS system is designed with three objectives: functionality, privacy and efficiency.

- Functionality. The goal is to realize the semantic search that returns K search results, which is suitable for scenarios such as RAG, where the K results returned should be K chunks with the highest correlation with the query
- Privacy. The goal is to ensure achieve (L,T)- Adaptive semantic security for that there are two adversaries: cloud server and data users.
- Efficiency. The goal is to achieve single round communication. That is, it is not necessary for data users to interact another round with cloud servers to get the chunk data.

IV. PRELIMINARIES

A. Notations

Unless otherwise specified, we use lowercase letters to represent vectors v or numbers v, uppercase letters to represent sets D or some parameters that can be adjusted D. Specific symbols and descriptions to be explained are shown in the Table I.

B. Beaver triples and its generation

In secure multi-party computation, privately computing the product of two secret values is a fundamental task. To reduce communication and computational overhead, we adopt a Beaver triple generation scheme based on Paillier homomorphic encryption and integrate it with an arithmetic secret sharing-based multiplication protocol.

1) Beaver Triple Generation with Paillier Encryption: A Beaver triple is a tuple of random values $(\mathsf{a}, \mathsf{b}, \mathsf{c})$ such that $\mathsf{c} = \mathsf{a} \cdot \mathsf{b}$, where all three values are secret-shared across two parties P_1 and P_2. The challenge lies in securely computing the cross terms $[\mathsf{a}]_1[\mathsf{b}]_2$ and $[\mathsf{a}]_2[\mathsf{b}]_1$ without revealing individual shares.

To achieve this, we leverage the additive homomorphic property of the Paillier cryptosystem. Party P_1 encrypts its shares $[\mathsf{a}]_1$, $[\mathsf{b}]_1$ and sends them to P_2. Party P_2 then computes the encrypted cross terms and adds a random mask r to hide the result. The masked encrypted result is returned to P_1, who decrypts it to obtain the sum of the cross terms plus r. Both parties then adjust their local values accordingly to complete the sharing of c. This method is shown in Protocol1, and it ensures correctness without leaking any intermediate information.

Protocol 1 Beaver triple generation

```
1: $P_1$ sends $\mathsf{P.Enc}([\mathsf{a}]_1)$ and $\mathsf{P.Enc}([\mathsf{b}]_1)$ to $P_2$.
2: $P_2$ computes $\mathsf{d} = \mathsf{P.Enc}([\mathsf{a}]_1)^{[\mathsf{b}]_2} \cdot \mathsf{P.Enc}([\mathsf{b}]_1)^{[\mathsf{a}]_2} \cdot \mathsf{P.Enc}(\mathsf{r})$,
   and sends $\mathsf{d}$ back to $P_1$.
3: $P_1$ decrypts $\mathsf{d}$ to get $[\mathsf{a}]_1[\mathsf{b}]_2 + [\mathsf{b}]_1[\mathsf{a}]_2 + \mathsf{r}$.
4: $P_2$ computes $[\mathsf{c}]_2 = [\mathsf{a}]_2[\mathsf{b}]_2 - \mathsf{r}$.
5: $P_1$ sets $[\mathsf{c}]_1 = [\mathsf{a}]_1[\mathsf{b}]_1 + \mathsf{d}$.
```

2) Multiplication Protocol with Arithmetic Secret shares: Once the Beaver triple is established, secure multiplication of two secret shares values x and y can proceed efficiently. Given $(\mathsf{a}, \mathsf{b}, \mathsf{c})$ as the Beaver triple, each party computes:

$$[\mathsf{e}] = [\mathsf{x}] - [\mathsf{a}], \quad [\mathsf{f}] = [\mathsf{y}] - [\mathsf{b}]$$

They then exchange their shares and get e and f. Using the following formula:

$$\mathsf{z} = \mathsf{c} + \mathsf{e} \cdot \mathsf{b} + \mathsf{f} \cdot \mathsf{a} + \mathsf{e} \cdot \mathsf{f}$$

Each party computes its local share of the product $\mathsf{z} = \mathsf{x} \cdot \mathsf{y} = [\mathsf{z}]_1 + [\mathsf{z}]_2$. This approach requires only one round of communication and minimal local computation.

Protocol 2 Arithmetic Secret shares Multiplication Using Beaver Triples

```
1: Both compute $[\mathsf{e}] = [\mathsf{x}] - [\mathsf{a}]$, $[\mathsf{f}] = [\mathsf{y}] - [\mathsf{b}]$, and exchange their
   shares.
2: Both reconstruct complete values of $\mathsf{e}$ and $\mathsf{f}$.
3: $P_1$ compute $[\mathsf{z}]_1 = [\mathsf{c}]_1 + \mathsf{e} \cdot [\mathsf{b}]_1 + \mathsf{f} \cdot [\mathsf{a}]_1 + \mathsf{e} \cdot \mathsf{f}$.
4: $P_2$ compute $[\mathsf{z}]_2 = [\mathsf{c}]_2 + \mathsf{e} \cdot [\mathsf{b}]_2 + \mathsf{f} \cdot [\mathsf{a}]_2$.
```

This protocol strikes a balance between security and efficiency and can be applied in privacy-preserving tasks such as federated learning and secure similarity computation.

C. Bloom Filter

Bloom filter is a popular space-saving tool, to achieve an approximate set membership test. Generally, it consists of three polynomial-time algorithms.

- $\mathsf{BF.Gen}(\mathsf{p}, 4 \cdot \mathsf{N}) \rightarrow (H = \{h_i\}_{i=1}^{\mathsf{n}}, \mathsf{B})$: A probabilistic run by data users on inputting the false positive rate p and the number of centroids $4 \cdot \mathsf{N}$ in $\mathbb{L}$, outputs a hash functions set $H = \{h_1, \ldots, h_\mathsf{n}\}$, and the initialized bloom filter B as a $|\mathsf{B}| = \frac{4 \cdot \mathsf{N} \cdot \mathsf{n}}{\ln 2}$ bits zero string, where the number of hash functions $\mathsf{n} = -\log_2^{\mathsf{p}}$.
- $\mathsf{BF.Insert}(L'_\mathsf{l}, H, \mathsf{B}) \rightarrow \mathsf{B}'$: A deterministic algorithm run by the data users with inputting sets L'_l, H and B, outputs an updated bloom filter B' by setting $1 := \mathsf{B}[h_i(\mathsf{e})]$ where element $\mathsf{e} \in L'_\mathsf{l}$ and i changing from 1 to n.
- $\mathsf{BF.Check}(H, B', \mathsf{e}) \rightarrow b$: A deterministic algorithm with false positive rate p by inputting the hash function set H,

bloom filter B' and an element e, outputs e only when $b = \bigwedge B'[h_i(\mathsf{I})]_{i=1}^{\mathsf{n}} = 1$.

D. Comparison Protocol for Max

Rabbit [22] proposes a secure two-party comparison protocol Π_{LTS} to compare two secret values x and y under semi-honest security model. For simplified, the comparison protocol is notated as $\Pi_{\mathsf{LTS}}([\mathsf{x}],[\mathsf{y}]) \to [\mathsf{b}]$ or $\Pi_{\mathsf{LTS}}([\mathsf{x}],[\mathsf{y}]) \to \mathsf{b}$ where the input of Π_{LTS} is referred here as Arithmetic values x and y secret shared, such that party P_i hold $[\mathsf{x}]_i$ and $[\mathsf{y}]_i$; the output is referred here as either the Boolean value $[\mathsf{b}]$ or a bit b where $1(0)$ indicates that $\mathsf{y}(\mathsf{x})$ is the larger one by disclosing the Boolean value $[\mathsf{b}]$.

Protocol 3 Π_{exLTS}–The expanded Π_{LTS} Protocol

```
Input: Arithmetic values $x, y$ secret shared, such that party $P_i$ hold $[x]_i, [y]_i$.
Output: Arithmetic secret share $[Max]$.
1: Parties run $\Pi_{LTS}([x],[y]) \to [b]$.
2: Parties transform the Boolean secret share $[b]$ into Arithmetic secret share $[b^A]$ by invoking ABY protocol.
3: Parties run the Protocol1 to generate shares of $c$ where $a_1 = [b^A]_1$, $a_2 = [b^A]_2$, $b_1 = [y]_1 - [x]_1$, $b_2 = [y]_2 - [x]_2$
4: Parties $P_i$ output $[x]_i + [c]_i$.
```

If one wants to compare T secret values under Arithmetic secret share, the trivial way is to use Π_{LTS} compute the lager one of the first two number, then use it to compare with the third one and go it recursively. However, it needs to reveal maximums in each comparison for comparing the next two values. Note that, these leakage may promise some privacy, thus we expand the Π_{LTS} protocol by outputting a new secret shares of the lager number. That is, $\Pi_{\mathsf{exLTS}}([\mathsf{x}],[\mathsf{y}]) \to [\mathsf{Max}]$ where $\mathsf{Max} = y(\mathsf{Max} = x)$ if bit $\mathsf{b} = 1(\mathsf{b} = 0)$ in $\Pi_{\mathsf{LTS}}([\mathsf{x}],[\mathsf{y}]) \to \mathsf{b}$. The correctness analysis of Protocol 3 is shown below.

The correctness analysis of Protocol 3
Obviously, there is

$$\begin{aligned}\mathsf{Max} &= \mathsf{by} + (1-\mathsf{b})\mathsf{x}\\ &= ([\mathsf{b}^A]_1 + [\mathsf{b}^A]_2)([\mathsf{y}]_1 + [\mathsf{y}]_2)\\ &+ (1 - ([\mathsf{b}^A]_1 + [\mathsf{b}^A]_2))([\mathsf{x}]_1 + [\mathsf{x}]_2)\\ &= [\mathsf{b}^A]_1[\mathsf{y}]_1 + [\mathsf{x}]_1 - [\mathsf{b}^A]_1[\mathsf{x}]_1 (for\ P_1)\\ &+ [\mathsf{b}^A]_2[\mathsf{y}]_2 + [\mathsf{x}]_2 - [\mathsf{b}^A]_2[\mathsf{x}]_2 (for\ P_2)\\ &+ [\mathsf{b}^A]_1([\mathsf{y}]_2 - [\mathsf{x}]_2) + [\mathsf{b}^A]_2([\mathsf{y}]_1 - [\mathsf{x}]_1).\end{aligned}$$

Note that

$$\begin{aligned}\mathsf{c} = [\mathsf{c}]_1 + [\mathsf{c}]_2 &= ([\mathsf{b}^A]_1 + [\mathsf{b}^A]_2)(\underline{[\mathsf{y}]_1 - [\mathsf{x}]_1} + \underline{[\mathsf{y}]_2 - [\mathsf{x}]_2})\\ &= [\mathsf{b}^A]_1[\mathsf{y}]_1 - [\mathsf{b}^A]_1[\mathsf{x}]_1 + [\mathsf{b}^A]_2[\mathsf{y}]_2 - [\mathsf{b}^A]_2[\mathsf{x}]_2\\ &+ [\mathsf{b}^A]_1([\mathsf{y}]_2 - [\mathsf{x}]_b) + [\mathsf{b}^A]_2([\mathsf{y}]_1 - [\mathsf{x}]_1).\end{aligned}$$

Next,

$$\begin{aligned}&[\mathsf{b}^A]_1([\mathsf{y}]_2 - [\mathsf{x}]_2) + [\mathsf{b}^A]_2([\mathsf{y}]_1 - [\mathsf{x}]_1)\\ &= [\mathsf{c}]_1 - [\mathsf{b}^A]_1[\mathsf{y}]_1 + [\mathsf{b}^A]_1[\mathsf{x}]_1 (for\ P_1)\\ &+ [\mathsf{c}]_2 - [\mathsf{b}^A]_2[\mathsf{y}]_2 + [\mathsf{b}^A]_2[\mathsf{x}]_2 (for\ P_2).\end{aligned}$$

Thus,

$$\begin{aligned}\mathsf{Max} &= \mathsf{by} + (1-\mathsf{b})\mathsf{x}\\ &= ([\mathsf{b}^A]_1 + [\mathsf{b}^A]_2)([\mathsf{y}]_1 + [\mathsf{y}]_2)\\ &+ (1 - ([\mathsf{b}^A]_1 + [\mathsf{b}^A]_2))([\mathsf{x}]_1 + [\mathsf{x}]_2)\\ &= [\mathsf{b}^A]_1[\mathsf{y}]_1 + [\mathsf{x}]_1 - [\mathsf{b}^A]_1[\mathsf{x}]_1 (for\ P_1)\\ &+ [\mathsf{b}^A]_2[\mathsf{y}]_2 + [\mathsf{x}]_2 - [\mathsf{b}^A]_2[\mathsf{x}]_2 (for\ P_2)\\ &+ [\mathsf{c}]_1 - [\mathsf{b}^A]_1[\mathsf{y}]_1 + [\mathsf{b}^A]_1[\mathsf{x}]_1 (for\ P_1)\\ &+ [\mathsf{c}]_2 - [\mathsf{b}^A]_2[\mathsf{y}]_2 + [\mathsf{b}^A]_2[\mathsf{x}]_2 (for\ P_2)\\ &= [\mathsf{x}]_1 + [\mathsf{c}]_1 (for\ P_1) + [\mathsf{x}]_2 + [\mathsf{c}]_2 (for\ P_2)\end{aligned}$$

Based on protocol Π_{exLTS}, we propose an augmented protocol $\Pi_{\mathsf{aCMax}}(([\mathsf{x}_1],[\mathsf{D}_1]),([\mathsf{x}_2],[\mathsf{D}_2]),\ldots,([\mathsf{x}_\mathsf{T}],[\mathsf{D}_\mathsf{T}])) \to \mathsf{D}_\mathsf{j}$ shown in Protocol 4. D_j can be useful value which depends on specific application scenario.

Protocol 4 Π_{aCMax}–The augmented Comparison Protocol for Max

```
Input: Arithmetic value pairs $(x_1, D_1), (x_2, D_2), \ldots, (x_T, D_T)$ secret shared, such that party $P_i$ hold $([x_1]_i, [D_1]_i), ([x_2]_i, [D_2]_i), \ldots, ([x_T]_i, [D_T]_i)$.
Output: $[D_j]$ such that $x_j$ is the largest.
1: $j = 1$.
2: $[x_m]_i \leftarrow [x_1]_i$. $[D_m]_i \leftarrow [D_1]_i$.
3: for $j \le T - 1$ do
4:     Invoke $\Pi_{exLTS}([x_m],[x_{j+1}]) \to [x_m]$ and compute the new $[D_m]_i = [D_m]_i + [c]_i$ where $[c]_i$ is generated using Protocol1 for $a_1 = [b^A]_1$, $a_2 = [b^A]_2$, $b_1 = [D_{j+1}]_1 - [D_j]_1$, $b_2 = [D_{j+1}]_2 - [D_j]_2$.
5:     $j{+}{+}$.
6: end for
```

V. The Proposed Scheme

In this section, we show details of each algorithms and protocol to achieve single round private Top-K semantic search. To ensure the privacy, we invoke the Protocol4 as a basic module. In short, we use BERT to transfer chunks or queries into vectors and search for the first k vectors with the highest cosine similarity to the query vector and return the corresponding chunks ciphertext, so that users can directly decrypt them locally to obtain the plaintext.

It should be noted that the returned data may not be a simple ciphertext using symmetrically encrypt scheme. The encryption scheme can be carefully designed based on the next step of processing the data. For example, one can chooses homomorphic encryption in **Enc** to support next privacy calculation. To highlight our protocol design, we simply use symmetric encryption to process the data.

In **Gen** algorithm, the PTS system is initialized. Data owners generate a key set K based on the security parameter λ. The key set K contains a big prime P to secret share securely, a symmetric key k for encryption and hash function h.

Then data owners encrypt the database M and generate an encrypted index as shown in Algorithm 2. After transforming the database into chunks and responding vectors, data owners invoke a fast adaptive k-means algorithm to generate clusters. Finally we can set each cluster has K points, thus the $\mathsf{k}= \mathsf{M}/\mathsf{K}$ and the nearest cluster of query q is the search result. Based

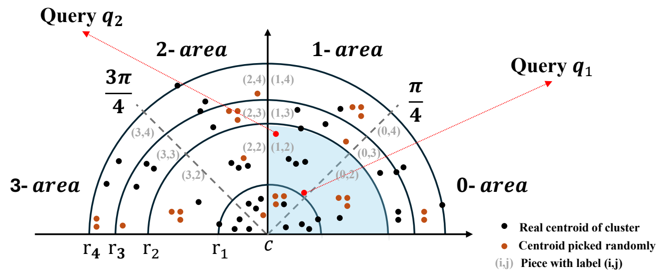

Fig. 2. The process of creating pieces of a loop index

on this assumption, we can pay more attention on the design of other algorithms and protocol.

To achieve the T-privacy, the search algorithm limits cloud servers comparing at less T centroid. To confuse the index cloud servers access for each query, we design a half-loop index based on two parameters: Euclidean distance and Cosine similarity. This index will not be affected by the dimensions of vectors. It is noted that it can be achieved using other index such as K-D tree, but the half-loop index can reflect our idea more clear and is easy to understand. Specifically, firstly, compute the center c of all centroids c_{i}. Second, divide all clusters into four part based on the cosine similarity of c and each centroid c_{i} and label them as f-area where $\mathsf{f} \in \{0, 1, 2, 3\}$ to indicate the angle θ between c and c_{i} is in the range $0 < \theta \leq \pi/4, \pi/4 < \theta \leq \pi/2, \pi/2 < \theta \leq 3\pi/4, 3\pi/4 < \theta \leq \pi$ respectively. Based on the partition, compute the Euclidean distance r between c and c_{i} and range them with ascending order. The minimum of r of top $\mathsf{T}/8$ c_{i} in each f-area is defined as r_1, therefore there are first four pieces of the loop index $\mathbb{L}$ with label $(1, 1), (2, 1), (3, 1), (4, 1)$. Thirdly, determine other r_j based on the same rule from the distance r_{j-1}. For pieces with label $\mathsf{l} = (\mathsf{f}, j)$ that less than $\mathsf{T}/8$ c_{i}, random choose centroids from pieces to $\mathsf{T}/8$ c_{i} whose label $\mathsf{l}' = (\mathsf{f}', j')$ meets $j' > j + 2$, $\mathsf{f}' = \mathsf{f} + 2 \mod 4$ or $j' < j + 2$ and $\mathsf{f}' = \mathsf{f} + 2 \mod 4$. The pseudocode algorithm is shown in Algorithm1. We use Fig.2 to illustrate the process. Set $\mathsf{T}/8 = 4$, all real centroids(centroids of clusters) are used solid black dots. Then we first divide all points into four areas based on the angle between c and r_{i}. Then, start from the center c, count 4 points and defined r_1. Do it until r_4. Because some pieces don't have four points, we choose randomly from other pieces and we use solid orange dots to indicate them. For example, for the piece with label $(1, 1)$, users can choose real points from other pieces in 1-area to make sure that the cosine similarity is positive number and similar with other points.

The index generation algorithm is shown in line 3 to 15 of Algorithm2. All centroids c_{i} are first unitized, then scaled into the Filed P by the same multiple, and then secretly shared, splitting them into two parts, one for each in two indexes $\mathbb{L}_1, \mathbb{L}_2$. For these centroids c_{i} that have two or more labels, they use different shares for pieces in $\mathbb{L}_1, \mathbb{L}_2$. The ciphertext of symmetric encryption is elements in the Filed P and are secretly shared too. Finally, use hash function to map each label as the identifier of each piece as a multiple map with the form of $[T_{h(\mathsf{k},\mathsf{l})}]_i = (h(k, \mathsf{r}_j\text{-}\mathsf{f}), \{([c_{\mathsf{i}}]_i, [\mathsf{D}_{\mathsf{i}}]_i)\})$.

Algorithm 1 Pieces–Determine the pieces

```
Input: All centroids $c_i$
Output: Label set $L_i$ for each centroid $c_i$
 1: Compute the center $c$ of all centroids $c_i$.
 2: Initiate four empty set f-area.
 3: for Each cluster i do
 4:     Compute the cosine similarity between $c_i$ and the center $c$.
 5:     Compute the Euclidean distance $r_i$ between $c_i$ and the center $c$.
 6:     Allocate the $(c_i, r_i)$ into f-area according to the angle responding to
        the cosine similarity.
 7: end for
 8: $j = 0$.
 9: A empty radius set $R$
10: while all f-areas are not empty do
11:     $j++$.
12:     for Each f-area do
13:         range all pairs $(c_i, r_i)$ with ascending order of $r_i$
14:         Set the $T/8$-th $r_i$ as r-f
15:     end for
16:     Select the minimum r-f as $r_j$.
17:     Add $(j, r_j)$ into $R$.
18:     Set labels $l = (f, j)$.
19:     for Each pair $(c_i, r_i)$ in the closure of ball with center $c$ and radius $r_j$
        from each f-area do
20:         Add the pair into the piece set $L_l = \{(c_i, r_i)\}$
21:         Kick out the pair from f-area.
22:     end for
23: end while
24: $N \leftarrow j$.
25: while $|L_l| = T/8$ where $l = (f, j)$ do
26:     random choose centroid $(c_i, r_i)$ from $L_{l'}$, $l' = (f, j')$.
27:     Add pair $(c_i, r_i)$ into $L_l$.
28: end while
```

Algorithm 2 Enc

```
Input: $K, P, M$
Output: $\mathbb{L} = (\mathbb{L}_1, \mathbb{L}_2)$
 1: Invoke BERT algorithm on datebase $M$ to transform it into M chunks,
    and get M vectors for each chunk.
 2: Invoke a ball $k$-means algorithm locally and get M/K clusters with a
    centroid $c_i$ and the responding chunk set $D_i$ for each cluster.
 3: Invoke Algorithm 1.
 4: Unify and scale each centroid $c_i$ into the Filed P by the same multiple.
 5: Update the new $c_i$ in piece sets and replace the $r_i$ by $D_i$ .
 6: for Each piece set $L_l = \{(c_i, D_i)\}$ do
 7:     Initiate empty multi map Tables $[T_{h(k,l)}]_i$ with identifier $h(k, l)$ where
        $i \in \{1, 2\}$.
 8:     for Each pair $(c_i, D_i)$ do
 9:         Encrypt chunk set $D_i$ under symmetric key k into $D_i$.
10:         Secret share each $D_i$ into $[D_i]_1$ and $[D_i]_2$.
11:         Secret share $c_i$ into $[c_i]_1$ and $[c_i]_2$.
12:         Add pair $([c_i]_i, [D_i]_)$ into $[T_{h(k,l)}]_i$
13:     end for
14:     $\mathbb{L}_i \leftarrow \mathbb{L}_i \cup [T_{h(k,l)}]_i$.
15: end for
```

To generate the trapdoor, data users compute the piece the query located and the $\mathsf{r}_{j'}$ and the angle line f' it nearest. Then, data users ascertain the search coverage composed by stable part and flexible one. The former consists of four fixed pieces with labels $(\mathsf{f}', j'), (\mathsf{f}', j' + 1), (\mathsf{f}' + 1, j'), (\mathsf{f}' + 1, j' + 1)$ and the latter is composed by four pieces chosen randomly from f-area and $\mathsf{f}' + 1$-area besides the stable coverage. Take queries q_1 and q_2 in Fig.2 as examples, the stable coverage of q_1 is four pieces with label $(0, 1)$, $(0, 2)$, $(1, 1)$, $(1, 2)$ (the light blue coverage) while that for q_2 is $(1, 2)$, $(1, 3)$, $(2, 2)$, $(2, 3)$. At last, choose the flexible coverage (four pieces) of pieces randomly besides of that in the stable coverage. Then, compute the bloom filter B containing labels of 8 pieces and the unitized secret sharing query vector $[q]_i$ as trapdoor.

Algorithm 3 Trgen

```
Input: $K, R, c, q$
Output: $\mathtt{t}$
1: Compute the piece the query $q$ located
2: Determine the $\mathsf{r}_{j'}$ and the angle line $\mathsf{f}'$ it nearest
3: Fix label $L_{\mathsf{l}}' = \{(\mathsf{f}', j'), (\mathsf{f}', j'+1), (\mathsf{f}'+1, j'), (\mathsf{f}'+1, j'+1)\}$ as stable coverage.
4: Choose four of the remaining pieces randomly as flexible coverage from $\mathsf{f}$-area and $\mathsf{f}'+1$-area besides the stable coverage and update $L_{\mathsf{l}}' = \{h(\mathsf{k}, \mathsf{l})\}$.
5: Invoke $(H = \{h_i\}_{i=1}^{\mathsf{n}}, \mathsf{B}) \leftarrow \mathsf{BF.Gen}(\mathsf{p}, 4 \cdot \mathsf{N})$
6: for Each $h(\mathsf{k}, \mathsf{l})$ do
7:     $\mathsf{B}' \leftarrow \mathsf{BF.Insert}(L_{\mathsf{l}}', H, \mathsf{B})$
8: end for
9: Unify the query $q$
10: Secret share $q$ into $[q]_1$ and $[q]_2$.
11: $\mathtt{t}_i = (\mathsf{B}', [q]_i)$
12: Send to $\mathtt{t}_i$ cloud server $CS_i$.
```

The search algorithm is shown in Algorithm4. Cloud servers use bloom filter to determine the pieces and use the unitized secret sharing query vector $[q]_i$ to compute the secret shares of Cosine similarity between q and c_{i}. To look for the nearest one, cloud servers invoke Π_{aCMax} protocol and output the secret shares of ciphertext as search result. Finally, data users decrypt and get the chunks.

Algorithm 4 Search

```
Input: $\mathtt{t}_i, \mathbb{L}_i$
Output: Updated $\mathsf{D_i}$.
1: A table set $[T_{h(\mathsf{k},\mathsf{l})}]_i \leftarrow \mathsf{BF.Check}(H, B', h(\mathsf{k}, \mathsf{l}))$ for all $h(\mathsf{k}, \mathsf{l})$.
2: for Each pair $([c_{\mathsf{i}}]_i, [\mathsf{D_i}]_i)$ in each table $[T_{h(\mathsf{k},\mathsf{l})}]_i$ do
3:     Invoke Protocol2 to compute secret shares of each dimension's multiplication of $c_{\mathsf{i}}$ and $q$.
4:     Compute the secret shares $\mathsf{x_i}$ of Cosine similarity of $c_{\mathsf{i}}$ and $q$ by summing the shared value of each dimension's multiplication.
5: end for
6: Get a set of $\mathsf{T}$ pairs $(\mathsf{x_i}, [\mathsf{D_i}])$.
7: Invoke the $\Pi_{\mathsf{aCMax}}$ protocol to get the shares Updated $\mathsf{D_i}$.
8: Send $[\mathsf{D_i}]$ to data users.
```

VI. Correctness and Security Analysis

A. Correctness Analysis

The correctness of the proposed scheme is mainly reflected in the search stage and the decryption stage. In **Search**, cloud servers receive a bloom filter B' to identified the pieces to be search and a secret share of the query vector $[q]_i$. Because both the cloud servers and users share the same hash set $H = \{h_i\}_{i=1}^{\mathsf{n}}$ and the same items $h(\mathsf{k}, \mathsf{l})$, both cloud servers could get the same pieces set L_{l}'' satisfying $L_{\mathsf{l}}' \subseteq L_{\mathsf{l}}''$. The cosine similarity between the query vector $q = (\dots, q_j, \dots)$ and a centroid $c_{\mathsf{i}} = (\dots, c_{\mathsf{i}_j}, \dots)$ is $q \cdot c_{\mathsf{i}}$ due to unitization. Then the correctness of $q_j \cdot c_{\mathsf{i}_j}$ under secret shares is ensured by the Protocol 2. Therefore, cloud servers can compute products of $q_j \cdot c_{\mathsf{i}_j}$ correctly and get secret shares of $q \cdot c_{\mathsf{i}}$. Then according to the Protocol 4, they can find the maximum and final get a new share of responding $\mathsf{D_i}$.

When users received shares $[\mathsf{D_i}]$, they first recover the ciphertext $\mathsf{D_i}$. Since users have the encrypted key used in **Enc**, they can decrypt $\mathsf{D_i}$ correctly and get the Top-K result.

B. Security Analysis

Theorem 1. *The proposed scheme can achieve* T*-privacy if pseudorandom functions is secure and the symmetric encryption is IND-CPA secure.*

Proof. In the proof, λ is the global security parameters, so all the advantages are under the same security parameters. Then, a serises of games are defined to provr the security of the proposed scheme.

Game G_0: We define G_0 is the same as $\mathsf{Real}_{\mathcal{A}}(\lambda)$ and so $Pr[\mathsf{Real}_{\mathcal{A}}(\lambda) = 1] = Pr[G_0 = 1]$.

Game G_1: In this game, we show how to use the leakage function $\mathcal{L}^{Enc}(M) = \mathcal{L}'(\mathsf{N}, \mathsf{T}, \mathsf{P})$ to simulate the output of **Enc** without changing other parts of the proposed scheme. For the dataset M, simulator $\mathcal{S}$ first revoke the ball k-means algorithm and the Algorithm 1 to generate all pieces. Then, unify and update each piece set $L_{\mathsf{l}} = \{(c_{\mathsf{i}}, D_{\mathsf{i}})\}$. To generate the identifiers $h(\mathsf{k}, \mathsf{l})$, $\mathcal{S}$ chooses random numbers and record them in a table for hash function T_h. To encrypt each chunk set D_{i}, $\mathcal{S}$ choose numbers randomly as the ciphertexts. Then, $\mathcal{S}$ runs the rest part as **Enc** to generate the encrypted index. For adversary, all identifiers $h(\mathsf{k}, \mathsf{l})$ and ciphertexts $\mathsf{D_i}$ are indistinguishable as shown that generated in Game G_0. Otherwise, it contradicts the security of hash functions h and the symmetric encryption scheme used in **Enc**. If $\mathtt{Adv}_h(\lambda)$ and $\mathtt{Adv}_{SE}(\lambda)$ are used to present the advantage of breaking the security of hash function and symmetric encryption, there is

$$|Pr[G_0 = 1] - Pr[G_1 = 1]| \leq \mathtt{Adv}_h(\lambda) + \mathtt{Adv}_{SE}(\lambda).$$

Game G_2: It is designed to be the same as G_1 except simulating the **Trgen**. For each query, simulator runs the line 1-4 by replacing the $h(\mathsf{k}, \mathsf{l})$ with numbers in table T_h. Then, $\mathcal{S}$ builds n tables T_{h_i} for each hash functions used in bloom filter and uses them to generate the bloom filter B'. Then, $\mathcal{S}$ runs the rest part as **Trgen** to generate the trapdoor. For adversary, all outputs of h_i are indistinguishable as that generated in Game G_1. Otherwise, it contradicts the security of hash functions h_i used in **Trgen**. If $\mathtt{Adv}_h(\lambda)$ is used to present the advantage of breaking the security of hash function, there is

$$|Pr[G_1 = 1] - Pr[G_2 = 1]| \leq \mathtt{nAdv}_h(\lambda).$$

Game G_3: To further simulate **Search** based on the Game G_2, $\mathcal{S}$ first invokes $\mathsf{BF.Check}$ to get all pieces of a query where hash functions are simulated by tables T_{h_i}. Then, $\mathcal{S}$ would access T centroids. At last, simulator invokes lines 2 to 8 and get the final search result. During the **Search**, adversary doesn't know the exactly centroid and its $\mathsf{D_i}$ due to the correctness of Protocol 4.

T is different for queries because of false positive rate and adversary cannot make sure whether two query is same, therefore the search pattern is hidden. For access pattern, adversary only knows T centroids but not the exact one, so the adversary guess it correctly with the probability $1/\mathsf{T}$ called T-privacy. Same as the Game 2, there are hash functions in bloom filter simulated by tables T_{h_i} merely and therefore

$$|Pr[G_2 = 1] - Pr[G_3 = 1]| \leq \mathtt{nAdv}_h(\lambda).$$

Game G_4: The only difference between G_3 and G_4 is **Dec**. Due to the IND-CPA secure of symmetric encryption scheme, the adversary gets nothing. That is,

$$|Pr[G_3 = 1] - Pr[G_4 = 1]| \leq \texttt{Adv}_{SE}(\lambda).$$

As shown in Game 4, Game G_4 is the ideal experiment of the proposed scheme simulated by $\mathcal{S}$ uses the leakage function merely. That is, $Pr[\mathsf{Ideal}_{\mathcal{A},\mathcal{S}}(\lambda) = 1] = Pr[G_4 = 1]$.

Thus, there is

$$|Pr[\mathsf{Real}_{\mathcal{A}}(\lambda) = 1] - Pr[\mathsf{Ideal}_{\mathcal{A},\mathcal{S}}(\lambda) = 1]| \leq 2\texttt{nAdv}_h(\lambda) + 2\texttt{Adv}_{SE}(\lambda)$$

Finally, for every adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}$ such that the leakage function is the same as Definition 1. Thus, the proposed scheme achieves T-privacy. □

VII. Performance Evaluation

To comprehensively evaluate the performance of our proposed scheme, we systematically compare it with three representative baseline approaches, with a focus on core metrics including computational efficiency, communication overhead, and privacy protection capabilities.

- Plaintext-RAG: The first baseline is a plaintext Retrieval-Augmented Generation (RAG) system without any privacy protections. Since it operates on unencrypted data and does not secure the computation process, it serves as an efficiency upper-bound reference. This baseline directly returns search result data.
- Boolean+simplePIR: The second baseline combines an efficient Boolean searchable encryption scheme with a Private Information Retrieval (PIR) protocol to privately fetch the search result data. This approach relies on keyword-based Boolean search and lacks semantic understanding. In our experiments, we decompose each query into multiple keywords with Boolean operators.
- Semantic+simplePIR: The third baseline integrates a semantic-aware encrypted search scheme with PIR. Through this comparison, we highlight the advantages of our proposed solution in security improvements—the baseline's KNN-based approach suffers from known vulnerabilities, whereas our scheme eliminates such risks.

In this section, we present a comprehensive theoretical analysis and empirical evaluation comparing our proposed scheme with baseline approaches, systematically examining the inherent trade-offs among computational efficiency, retrieval accuracy, and security guarantees.

A. Theoretical Comparison

Our theoretical analysis examines two key dimensions: scheme comparison and communication overhead. As shown in Table 2, the functional comparison focuses on four critical metrics: query formulation, encryption mechanism, similarity computation methodology, and returned results. The analysis reveals that while our scheme demonstrates functional similarities with both Semantic+simplePIR and Plaintext-RAG approaches, it fundamentally differs in its core computation mechanism. Specifically, whereas the baseline methods rely on topic/keyword scoring, our solution innovatively implements cosine similarity computation between semantic vectors, thereby achieving functional characteristics that more closely approximate the ideal Plaintext-RAG performance.

From a security perspective, we conduct a comprehensive evaluation focusing on three core attributes: component replaceability, foundational security assumptions, and privacy protection types. It should be particularly noted that the KNN algorithm employed in Semantic+simplePIR suffers from well-documented security vulnerabilities. In contrast, our proposed scheme is built upon more robust security assumptions while maintaining full functional integrity, thereby delivering enhanced security guarantees without compromising performance.

B. Performance Comparison

To demonstrate extensibility, we conduct modular tests by integrating **Long-term Storage (LTS)** and **Comparison Protocol (CMP)** as foundational components of our **Augmented Comparison Protocol for Max**. This showcases the architectural flexibility of our design.

VIII. Conclusion

Xi In this paper, we propose a.... **Correctness Analysis**:

References

[1] Q. Wang, D. Hu, M. Li, and G. Yang, "Secure and flexible wildcard queries," *IEEE Transactions on Information Forensics and Security*, 2024.

[2] K. Zhang, X. Wang, J. Ning, and X. Huang, "Dual-server boolean data retrieval for highly-scalable secure file sharing services," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 449–462, 2023.

[3] Z. Lin, H. Li, X. Chen, M. Xiao, and Q. Huang, "Identity-based encryption with disjunctive, conjunctive and range keyword search from lattices," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 8644–8657, 2024.

[4] H. Bao, L. Xing, H. Wu, M. Guan, N. Ruan, C. Huang, and H.-N. Dai, "Mkac: Efficient and privacy-preserving multi-keyword ranked query with ciphertext access control in cloud environments," *IEEE Transactions on Cloud Computing*, pp. 1–14, 2025.

[5] A. Henzinger, M. M. Hong, H. Corrigan-Gibbs, S. Meiklejohn, and V. Vaikuntanathan, "One server for the price of two: Simple and fast Single-Server private information retrieval," in *32nd USENIX Security Symposium (USENIX Security 23)*. Anaheim, CA: USENIX Association, Aug. 2023, pp. 3889–3905. [Online]. Available: https://www.usenix.org/conference/usenixsecurity23/presentation/henzinger

[6] R. Lehmkuhl, A. Henzinger, and H. Corrigan-Gibbs, "Distributional private information retrieval," Cryptology ePrint Archive, Paper 2025/132, 2025. [Online]. Available: https://eprint.iacr.org/2025/132

[7] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000*, 2000, pp. 44–55.

[8] X. Zhang, B. Zhao, J. Qin, W. Hou, Y. Su, and H. Yang, "Practical wildcard searchable encryption with tree-based index," *Int. J. Intell. Syst.*, vol. 36, no. 12, pp. 7475–7499, 2021. [Online]. Available: https://doi.org/10.1002/int.22595

[9] X. Zhang, C. Huang, Y. Su, and J. Qin, "Secure, dynamic, and efficient keyword search with flexible merging for cloud storage," *IEEE Transactions on Services Computing*, vol. 17, no. 5, pp. 2822–2835, 2024.

[10] Z. Fu, Y. Wang, X. Sun, and X. Zhang, "Semantic and secure search over encrypted outsourcing cloud based on bert," *Frontiers of Computer Science*, vol. 16, no. 2, pp. 1–8, 2022.

[11] L. Chen, Y. Xue, Y. Mu, L. Zeng, F. Rezaeibagha, and R. H. Deng, "Case-sse: Context-aware semantically extensible searchable symmetric encryption for encrypted cloud data," *IEEE Transactions on Services Computing*, vol. 16, no. 2, pp. 1011–1022, 2023.

TABLE II
THE COMPARISON OF SCHEMES

Scheme	Framework	Server Requeried	Security Model	Functionalities				Security		
				Query	Encryption	Similarity	Result	Flexible†	Assumption	Privacy
Plaintext-RAG	Centralized	1	Honest but curious	Semantic	-	Vector	Top-k	-	-	-
Boolean+simplePIR	Decentralized trust model	2		Keyword	Public	Keyword	All	-	PRF,DL,DDH	KGA-security
Semantic+simplePIR	Centralized	1		Semantic	Symmetric	Vector*	Top-k	-	PRF,secure KNN	Trapdoor Security
Our	Decentralized trust model	2		Semantic	Symmetric	Vector	Top-k	✓	PRF	T-privacy

$*$: keyword and .
†: keyword and .

TABLE III
THE COMPUTATION AND COMMUNICATION COMPLEXITY COMPARISON

Scheme	Merge Operation			Merging Consistency			Client Storage	Index Storage
	Communication	Computation	RT	Communication	Computation	RT		
BasicDPF	$O(t+1)$	$O(t+1)$	2	$O(u_f+1)$	$O(u_f+1)$	2	$O(\|\mathbb{W}\| \cdot \log(\|\mathbb{W}\|)) + O(t)$	$O(\|\mathbb{F}\| \cdot \|\mathbb{W}\|)$
Mitra	$O(t+1)$	$O(t+1)$	2	$O(t+2)$	$O(t+2)$	2	$O(\|\mathbb{W}\| \cdot \log(\|\mathbb{D}(w)\|)) + O(t)$	$O(\Sigma_{w \in \mathbb{W}} \|\mathbb{D}(w)\|)$
MSSE	$O(1)$	$O(1)$	1	$O(1)$	$O(1)$	1	$O(\|\mathbb{W}\| \cdot \log(\|\mathbb{W}\|))$	$O(\|\mathbb{W}\| \cdot \|\mathbb{F}\|/\beta)$

RT is the number of round trips.
$t = |W'|$ is the number of keywords in merge operation.
$\log$ is used to compute the length in binary bit string.
u_f is the number of updated files.

[12] X. Dai, H. Dai, C. Rong, G. Yang, F. Xiao, and B. Xiao, "Enhanced semantic-aware multi-keyword ranked search scheme over encrypted cloud data," *IEEE Transactions on Cloud Computing*, vol. 10, no. 4, pp. 2595–2612, 2020.

[13] Y. Hou, W. Yao, X. Li, Y. Xia, and M. Wang, "Lattice-based semantic-aware searchable encryption for internet of things," *IEEE Internet Things J.*, vol. 11, no. 17, pp. 28 370–28 384, 2024. [Online]. Available: https://doi.org/10.1109/JIOT.2024.3400816

[14] Z. Hu, H. Dai, Y. Liu, G. Yang, Q. Zhou, and Y. Chen, "Csmrs: An efficient and effective semantic-aware ranked search scheme over encrypted cloud data," in *2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, 2022, pp. 699–704.

[15] J. Li, H. Peng, and L. Li, "Sublinear smart semantic search based on knowledge graph over encrypted database," *Computers and Security*, vol. 151, p. 104319, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404825000082

[16] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in *Proceedings of IEEE 36th Annual Foundations of Computer Science*, 1995, pp. 41–50.

[17] B. Li, D. Micciancio, M. Raykova, and M. Schultz-Wu, "Hintless single-server private information retrieval," Cryptology ePrint Archive, Paper 2023/1733, 2023. [Online]. Available: https://eprint.iacr.org/2023/1733

[18] M. Hao, W. Liu, L. Peng, C. Zhang, P. Wu, L. Zhang, H. Li, and R. H. Deng, "Practical keyword private information retrieval from key-to-index mappings," Cryptology ePrint Archive, Paper 2025/210, 2025. [Online]. Available: https://eprint.iacr.org/2025/210

[19] M. H. Mughees and L. Ren, "Vectorized batch private information retrieval," in *2023 IEEE Symposium on Security and Privacy (SP)*, 2023, pp. 437–452.

[20] K. D. Albab, R. Issa, M. Varia, and K. Graffi, "Batched differentially private information retrieval," in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 3327–3344. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/albab

[21] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," *J. Comput. Secur.*, vol. 19, no. 5, pp. 895–934, 2011.

[22] E. Makri, D. Rotaru, F. Vercauteren, and S. Wagh, "Rabbit: Efficient comparison for secure multi-party computation," in *International conference on financial cryptography and data security*. Springer, 2021, pp. 249–270.

[23] S.-y. Xia, D. Peng, D. Meng, C. Zhang, G. Wang, E. Giem, and W. Wei, "A fast adaptive k-means with no bounds," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, pp. 1–1, 07 2020.

[24] Y. Ding, Y. Zhao, X. Shen, M. Musuvathi, and T. Mytkowicz, "Yinyang k-means: A drop-in replacement of the classic k-means with consistent speedup," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, ser. JMLR Workshop and Conference Proceedings, F. R. Bach and D. M. Blei, Eds., vol. 37. JMLR.org, 2015, pp. 579–587.

[25] J. Newling and F. Fleuret, "Fast k-means with accurate bounds," in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, ser. JMLR Workshop and Conference Proceedings, M. Balcan and K. Q. Weinberger, Eds., vol. 48. JMLR.org, 2016, pp. 936–944.