SetRkNN: Efficient and Privacy-Preserving Set Reverse kNN Query in Cloud

Yandong Zheng, *Member, IEEE*, Rongxing Lu, *Fellow, IEEE*, Hui Zhu, *Senior Member, IEEE*, Songnian Zhang, Yunguo Guan, Jun Shao, *Senior Member, IEEE*, Fengwei Wang, *Member, IEEE*, and Hui Li, *Member, IEEE*

***Abstract*— The advance of cloud computing has driven a new paradigm of outsourcing large-scale data and data-driven services to public clouds. Due to the increased awareness of privacy protection, many studies have focused on addressing security and privacy issues in outsourced query services. Although many privacy-preserving schemes have been proposed for various query types, the set reverse k nearest neighbors (RkNN) query is still an unexplored area. Even if some existing schemes can be adapted to achieve privacy-preserving set RkNN queries, they will suffer from linear search efficiency. As a steppingstone, in this paper, we propose an efficient and privacy-preserving set RkNN query scheme over encrypted data with sublinear query efficiency. Specifically, we first design an inverted prefix index to organize the set dataset and propose an algorithm to traverse the index with sublinear search efficiency. Then, we propose two oblivious data comparison protocols based on a symmetric homomorphic encryption (SHE) scheme and design the private filter/refinement protocols to preserve the privacy of index searching. After that, we propose an access pattern privacy-preserving set RkNN query scheme by using private filter/refinement protocols. Rigorous security analysis demonstrates that our scheme can protect data privacy and access pattern privacy. Experimental results indicate that our scheme is more efficient than the available naive solution in terms of computational costs and communication overheads.**

Manuscript received 19 May 2022; revised 25 September 2022 and 5 November 2022; accepted 11 December 2022. Date of publication 22 December 2022; date of current version 30 December 2022. This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFB3103400; in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grants (04009, RGPIN-2022-03244); in part by NSFC under Grant 61972304, Grant 61932015, and Grant U22B2030; in part by the China Postdoctoral Science Foundation under Grant 2022M722498; in part by the Zhejiang Natural Science Foundation (ZJNSF) under Grant LQ22F020022, in part by the OPPO Research Fund; and in part by the Henan Key Laboratory of Network Cryptography Technology under Grant LNCT2022-A18. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Zekeriya Erkin. *(Corresponding author: Hui Zhu.)*

Yandong Zheng is with the State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China, and also with the Henan Key Laboratory of Network Cryptography Technology, Zhengzhou 450000, China (e-mail: zhengyandong@xidian.edu.cn).

Rongxing Lu, Songnian Zhang, and Yunguo Guan are with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB E3B 5A3, Canada (e-mail: rlu1@unb.ca; szhang17@unb.ca; yguan4@unb.ca).

Hui Zhu, Fengwei Wang, and Hui Li are with the State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China (e-mail: zhuhui@xidian.edu.cn; wangfengwei@xidian.edu.cn; lihui@mail.xidian.edu.cn).

Jun Shao is with the School of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou 310018, China (e-mail: chn.junshao@gmail.com).

Digital Object Identifier 10.1109/TIFS.2022.3231785

***Index Terms*— Set RkNN, encrypted data, inverted prefix filter index, homomorphic encryption, access pattern privacy.**

I. Introduction

LOW maintenance costs and high computing power advantages of cloud computing have inspired organizations to outsource large-scale data and data-driven services to public clouds. However, cloud servers usually belong to third-party enterprises and are not fully trusted. Direct outsourcing plaintext data to clouds will unavoidably expose the sensitive information of the data and may even threaten relevant personnel's life and property safety [1], [2]. Consequently, many studies have been devoted to addressing security and privacy issues for outsourced data while guaranteeing data availability. An effective approach is to leverage encryption techniques to encrypt data and design trapdoors to support data availability.

Many studies have been proposed for various privacy-preserving query types, including k nearest neighbor (kNN) query [3], [4], [5], skyline query [6], aggregation query [7], and predication query [8], etc. However, the reverse kNN (RkNN) query only receives limited attention, yet it complements the kNN query and plays a vital role in taxi dispatch and the targeted push of media information. Essentially, the kNN query retrieves top-k records nearest to query records and can further mine a wealth of information from these nearest neighbors. While the RkNN query retrieves all records regarding query records as their k nearest neighbors. Meanwhile, RkNN queries have different categories based on their target data types, e.g., location based RkNN query [9], [10], [11], road network based RkNN query [12], multi-dimensional data based RkNN query [13], etc. In this work, we target set records and study set RkNN queries, where the similarity between two sets is measured by Jaccard similarity.

Currently, many studies have paid attention to privacy-aware location based RkNN queries. As the first initiative, Du et al. [9] proposed a data blurring method to hide dataset records and query records into rectangle regions and then designed a "Voronoi Cell for Regions" structure to index the data for improving RkNN query efficiency. However, it trades off the query accuracy for data privacy. Lin et al. [10] used a private information retrieval (PIR) technique to retrieve query results and enhanced the query efficiency by using RkNN-HG and RkNN-HRT indexes, respectively. Unfortunately, it does

not preserve the dataset's privacy. Li et al. [11] integrated Delaunay triangulation, structured encryption, and a reference-locked encryption scheme to design a secure RkNN query scheme, which can protect both data privacy and query privacy. Nevertheless, the location based RkNN query schemes in [9], [10], and [11] are not applicable to set RkNN queries. Similarly, the RkNN query schemes in [12] and [13] are designed for road networks and multi-dimensional data with specialized indexes. It is nontrivial to adapt them to handle set RkNN queries. In addition, many existing privacy-preserving Jaccard similarity computation and kNN query schemes [14], [15], [16], [17], [18], [19], [20], [21] (More details will be discussed in Section VIII) are available to check whether a data record satisfies an RkNN query request. However, using such schemes to achieve RkNN queries has to access all dataset records and will suffer from linear search efficiency. Therefore, attaining sub-linear search efficiency in set RkNN queries over encrypted data is still an unexplored area that needs to be studied urgently.

As a steppingstone, in this paper, we propose an efficient and privacy-preserving set RkNN query (SetRkNN) scheme over encrypted data with sublinear query efficiency. The main idea of our scheme is to build an inverted prefix based on the prefix and length filters of set similarity computation. Roughly, if two sets have high similarity, their prefix elements should have an intersection when all elements in them are sorted in a global order. Meanwhile, they should also be similar in size. The index reduces our search scope to a smaller collection of candidate sets and thus brings our scheme to sublinear query efficiency. After that, we focus on protecting the data privacy of inverted prefix index based set RkNN queries. We first design two oblivious data comparison protocols in the two-server model based on an efficient symmetric homomorphic encryption (SHE) and its public-key version [19], [22], [23]. Based upon these two protocols, we further design a private filter protocol and a private refinement protocol, which are respectively employed for searching an entry of the inverted index and verifying whether a candidate set satisfies the query request or not. Subsequently, we propose our efficient and privacy-preserving set RkNN query scheme over encrypted data based on the private filter/refinement protocols. Since the access pattern leakage may violate data privacy [24], [25], we also consider the access pattern privacy. Because protecting access pattern privacy of the whole dataset is unnecessary and outrageously expensive [26], our scheme introduces the security level of t-access pattern unlinkability, which guarantees that one record in the index and dataset is indistinguishably accessed with $(t-1)$ records. Specifically, our contributions are four folds as follows.

- First, we design an inverted prefix index to organize the set dataset by leveraging prefix and length filters of set similarity computation. Then, we elaborately propose an algorithm to efficiently perform set RkNN queries over the index, which reaches the sublinear search efficiency.
- Second, we propose two oblivious data comparison protocols, i.e., an oblivious less than comparison (OLTC) protocol and an oblivious greater than comparison (OGTC) protocol based on an SHE scheme and its public-key version. Then, we design the private filter/refinement protocols to protect the privacy of index searching and candidate records verification.
- Third, we propose our SetRkNN scheme based on private filter/refinement protocols. To efficiently protect access pattern privacy, we introduce the security level of t-access pattern unlinkability and carefully design our scheme to guarantee the t-access pattern unlinkability in every aspect of our scheme.
- Finally, we rigorously prove the security of our scheme in a simulation-based real/ideal worlds model. The results show that our scheme can protect data privacy and achieve t-access pattern unlinkability. In addition, we experimentally evaluate the performance of our scheme. The results show that our scheme is more efficient than the naive solution in computational costs and communication overheads.

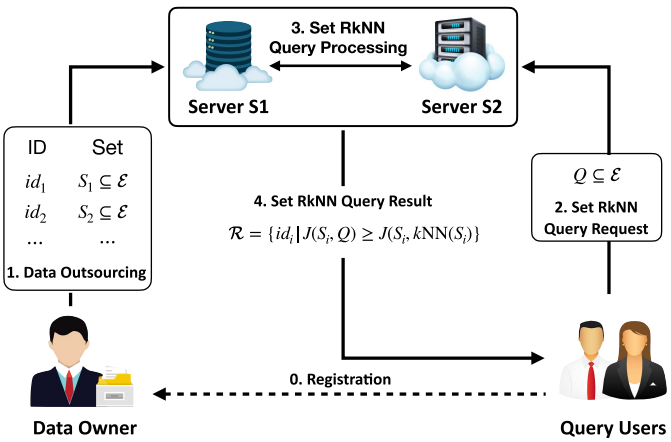

Fig. 1. Set RkNN query model in our system.

The remainder of this paper is organized as follows. In Section II, we introduce our system model and security model. Then, we describe some preliminaries in Section III and propose some building blocks in Section IV. In Section V, we present our scheme, followed by security analysis and performance evaluation in Section VI and Section VII, respectively. In Section VIII, we present some related work. Finally, we draw our conclusion in Section IX.

II. System Model and Security Model

In this section, we formalize the system model and security model considered in our work.

A. System Model

Our system focuses on a set RkNN query model in the outsourced scenario with three types of participants, including a data owner, two cloud servers, and many query users, as shown in Fig. 1.

1) Data Owner: The data owner has a set dataset $\mathcal{S} = \{(id_i, S_i)\}_{i=1}^{n}$ with n records, where id_i is the identity of S_i. Without loss of generality, we assume that all elements in each set $S_i \in \mathcal{S}$ are integers [27]. To fully exploit the benefit of the dataset, the data owner offers set RkNN query services over $\mathcal{S}$ to users in need. Restricted by computing and storage resources, the data owner outsources its dataset $\mathcal{S}$ to two cloud servers and hires the servers to offer set RkNN query services

to users on behalf of itself. The data owner will build an index for the dataset to improve the query efficiency. Then, it encrypts the index and the dataset before outsourcing them to the cloud to protect data privacy.

2) Two Cloud Servers: Two cloud severs, denoted as S1 and S2, have rich computing and storage resources. They collaboratively process set RkNN query requests received from query users. Let (Q, k) be a set RkNN query request, where Q is a query set. On receiving (Q, k), S1 and S2 search on the outsourced dataset for sets regarding Q as their kNN. Finally, the cloud servers return the result to the query user.

3) Query Users: The system offers set RkNN query services for multiple query users. The data owner will authorize users in their registration phase to ensure that only legitimate users can enjoy query services.

B. Security Model

Since the data owner is the creator of our entire system, we assume that it is trusted. For the query users, since the data owner has authorized them, we consider that they are *honest*, i.e., sincerely launch set RkNN query requests to cloud servers based on our scheme. For the two servers, since third parties run them, we assume that they are *honest-but-curious*. They faithfully offer query services to users but may be curious about some private information to them, including the plaintexts of sets in the outsourced dataset, in the query requests, and even in the query results; and the access pattern of the outsourced data. Meanwhile, we assume that there is no collusion between two servers. The assumption is reasonable and widely acknowledged in the security community [28], [29] because different cloud service providers are likely to have conflicts of interest. Since our work focuses on privacy preservation, other active attacks are beyond the scope of this work and will be discussed in future. Note that, as discussed in [30], many systems like ours can later be enhanced to resist malicious adversaries using verifiable secret sharing and zero knowledge proofs, if the steep increase in computational complexity is acceptable.

III. Preliminaries

In this section, we recall the Jaccard similarity, define set RkNN query, review a symmetric homomorphic encryption (SHE) scheme and its public-key version (PHE), and introduce the security level of t-access pattern unlinkability.

A. Jaccard Similarity and Set RkNN Query

This section reviews Jaccard similarity and two filter strategies widely used for Jaccard-based set similarity queries, and formalizes the definition of the set RkNN query.

Definition 1 (Jaccard Similarity): The Jaccard similarity between two sets S_i and S_j is defined as $J(S_i, S_j) = \frac{|S_i \cap S_j|}{|S_i \cup S_j|}$. Based on the definition, two filter strategies are widely used to improve the Jaccard based set similarity query, including a prefix filter and a length filter.

Theorem 1 (Prefix Filter [31]): Let S_i and S_j be two sets, and their elements are placed in a global order (e.g., frequency order). If $J(S_i, S_j) \geq \tau$, we have $\psi(S_i) \cap \psi(S_j) \neq \emptyset$, where $\psi(S_i)$ (resp. $\psi(S_j)$) denotes the first $\lfloor (1-\tau)|S_i| \rfloor + 1$ (resp. $\lfloor (1-\tau)|S_j| \rfloor + 1$) elements of S_i (resp. S_j).

Theorem 2 (Length Filter [31]): Given two sets S_i and S_j, if $J(S_i, S_j) \geq \tau$, we have $\tau|S_j| \leq |S_i| \leq \frac{|S_j|}{\tau}$.

Definition 2 (Set RkNN Query): Suppose that $\mathcal{S} = \{(id_i, S_i)\}_{i=1}^{n}$ is a set dataset, and $k\text{NN}(S_i)$ is the k-th nearest neighbor of S_i over $\mathcal{S}$. Let (Q, k) denote a set RkNN query request, where Q is a query set. The query result of (Q, k) will be $\mathcal{R} = \{id_i | J(S_i, Q) \geq J(S_i, k\text{NN}(S_i)); S_i \in \mathcal{S}\}$.

B. SHE Scheme and PHE Scheme

This section introduces the SHE scheme and its public-key version (PHE).

1) SHE Scheme: The SHE scheme is a homomorphic encryption scheme, which was proposed by Mahdikhani et al. [22]. It relies on a new assumption referred to the $(\mathtt{L}, \mathtt{p})$-based decision problem. In Section VI-A, we show how to instantiate it with a parameter set that withstands three known attacks. In this section, we focus on presenting the SHE scheme, including key generation, encryption, and decryption. Its security proof will be introduced in Section VI.

- $\mathsf{SHE.KeyGen}(\mathtt{k}_{\mathcal{M}}, \mathtt{k}_\mathtt{r}, \mathtt{k}_\mathtt{L}, \mathtt{k}_\mathtt{p}, \mathtt{k}_\mathtt{q})$: On input security parameters $\mathtt{k}_{\mathcal{M}}$, $\mathtt{k}_\mathtt{r}$, $\mathtt{k}_\mathtt{L}$, $\mathtt{k}_\mathtt{p}$, and $\mathtt{k}_\mathtt{q}$ with $\mathtt{k}_{\mathcal{M}} \ll \mathtt{k}_\mathtt{L}$, the key generation algorithm sets the message space to be $\mathcal{M} = \{\mathtt{m} | \mathtt{m} \in [-2^{\mathtt{k}_{\mathcal{M}}-1}, 2^{\mathtt{k}_{\mathcal{M}}-1})\}$. Then, it randomly chooses a $\mathtt{k}_\mathtt{L}$-bit number $\mathtt{L}$ and a $\mathtt{k}_\mathtt{p}$-bit prime number $\mathtt{p}$. After that, it randomly chooses a set of $\mathtt{k}_\mathtt{p}$-bit prime numbers $\{\mathtt{q}_i | 1 \leq i \leq \lceil \frac{\mathtt{k}_\mathtt{q}}{\mathtt{k}_\mathtt{p}} \rceil\}$, sets $\mathtt{q} = \prod_{i=1}^{\lceil \frac{\mathtt{k}_\mathtt{q}}{\mathtt{k}_\mathtt{p}} \rceil} \mathtt{q}_i$, and computes $\mathtt{N} = \mathtt{p} * \mathtt{q}$. Finally, the algorithm outputs the public parameter $\mathtt{pp} = \{\mathtt{k}_{\mathcal{M}}, \mathtt{k}_\mathtt{r}, \mathtt{k}_\mathtt{L}, \mathtt{k}_\mathtt{p}, \mathtt{k}_\mathtt{q}, \mathtt{N}\}$, the secret key $\mathtt{sk} = \{\mathtt{p}, \mathtt{L}\}$, and the message space $\mathcal{M}$.
- $\mathsf{SHE.Enc}(\mathtt{m}, \mathtt{sk})$: A message $\mathtt{m}$ is encrypted by the secret key $\mathtt{sk}$ as $[\![\mathtt{m}]\!] = (\mathtt{r} * \mathtt{L} + \mathtt{m})(1 + \mathtt{r}' * \mathtt{p}) \mod \mathtt{N}$, where $\mathtt{r} \in \{0,1\}^{\mathtt{k}_\mathtt{r}}$ and $\mathtt{r}' \in \{0,1\}^{\mathtt{k}_\mathtt{q}}$ are random numbers.
- $\mathsf{SHE.Dec}(\mathtt{sk}, [\![\mathtt{m}]\!])$: Given a ciphertext $[\![\mathtt{m}]\!]$, the plaintext $\mathtt{m}$ is recovered by i) calculating $\mathtt{m}' = ([\![\mathtt{m}]\!] \mod \mathtt{p}) \mod \mathtt{L}$; and ii) setting $\mathtt{m} = \mathtt{m}'$ if $\mathtt{m}' < \frac{\mathtt{L}}{2}$ and $\mathtt{m} = \mathtt{m}' - \mathtt{L}$ otherwise.

The SHE scheme allows homomorphic addition and multiplication over ciphertexts, including

- Add-I: $([\![\mathtt{m}_1]\!] + [\![\mathtt{m}_2]\!]) \mod \mathtt{N} \rightarrow [\![\mathtt{m}_1 + \mathtt{m}_2]\!]$;
- Add-II: $([\![\mathtt{m}_1]\!] + \mathtt{m}_2) \mod \mathtt{N} \rightarrow [\![\mathtt{m}_1 + \mathtt{m}_2]\!]$;
- Multiply-I: $[\![\mathtt{m}_1]\!] * [\![\mathtt{m}_2]\!] \mod \mathtt{N} \rightarrow [\![\mathtt{m}_1 * \mathtt{m}_2]\!]$;
- Multiply-II: $[\![\mathtt{m}_1]\!] * \mathtt{m}_2 \mod \mathtt{N} \rightarrow [\![\mathtt{m}_1 * \mathtt{m}_2]\!]$ $(\mathtt{m}_2 > 0)$.

The SHE scheme is a leveled homomorphic encryption scheme. Its multiplicative depth over ciphertexts is limited by the security parameters and is up to $\lfloor \frac{\mathtt{k}_\mathtt{p}}{\mathtt{k}_\mathtt{r} + \mathtt{k}_\mathtt{L}} \rfloor - 1$.

2) PHE Scheme: The SHE scheme is a symmetric encryption scheme. Based on its homomorphic properties, we can construct a public-key homomorphic encryption (PHE) scheme, which has different key generation and encryption algorithms from the SHE scheme, as described below.

- $\mathsf{PHE.KeyGen}(\mathtt{k}_{\mathcal{M}}, \mathtt{k}_\mathtt{r}, \mathtt{k}_\mathtt{L}, \mathtt{k}_\mathtt{p}, \mathtt{k}_\mathtt{q})$: On input security parameters $\mathtt{k}_{\mathcal{M}}$, $\mathtt{k}_\mathtt{r}$, $\mathtt{k}_\mathtt{L}$, $\mathtt{k}_\mathtt{p}$, and $\mathtt{k}_\mathtt{q}$, the key generation algorithm first generates the keys and message space of the SHE scheme as

$$\mathtt{pp}, \mathtt{sk}, \mathcal{M} \leftarrow \mathsf{SHE.KeyGen}(\mathtt{k}_{\mathcal{M}}, \mathtt{k}_\mathtt{r}, \mathtt{k}_\mathtt{L}, \mathtt{k}_\mathtt{p}, \mathtt{k}_\mathtt{q}). \quad (1)$$

Then, it generates two ciphertexts of zero as

$$[\![0]\!] \leftarrow \mathsf{SHE.Enc}(0, \mathtt{sk}) \text{ and } [\![0]\!]' \leftarrow \mathsf{SHE.Enc}(0, \mathtt{sk}). \quad (2)$$

Finally, it outputs the public key $\mathtt{pk} = \{\mathtt{pp}, [\![0]\!], [\![0]\!]'\}$, the secret key $\mathtt{sk}$, and the message space $\mathcal{M}$.

- $\mathsf{PHE.Enc}(\mathtt{m}, \mathtt{pk})$: A message $\mathtt{m}$ is encrypted by $\mathtt{pk}$ as

$$[\![\mathtt{m}]\!] \leftarrow (\mathtt{m} + \mathtt{r}_1 * [\![0]\!] + \mathtt{r}_2 * [\![0]\!]') \bmod \mathtt{N}, \quad (3)$$

where $\mathtt{r}_1, \mathtt{r}_2 \in \{0,1\}^{\mathtt{k_r}}$.

Note that the decryption algorithm of the PHE scheme is the same as that of the SHE scheme. Meanwhile, the PHE scheme is also IND-CPA secure, as proved in [23].

Remark. Besides the SHE scheme, we can also leverage other fully homomorphic encryption schemes, such as BGV [32] and CKKS [33], to protect the privacy of our scheme.

C. t-Access Pattern Unlinkability

In our scheme, to balance the query efficiency and data privacy, we focus on achieving t-access pattern unlinkability in set RkNN queries, as defined in Definition 3.

Definition 3 (t-Access Pattern Unlinkability): Our set RkNN query scheme achieves t-access pattern unlinkability *iff* one record in both the index and dataset is indistinguishably accessed with at least $(t-1)$ records.

IV. Building Blocks

We first propose an inverted prefix index and a set RkNN query algorithm. Then, we design two oblivious data comparison protocols, a private filter protocol, and a private refinement protocol to preserve the privacy of set RkNN queries.

A. Inverted Prefix Index

Based on the prefix and length filters, we introduce an element-set inverted prefix index for the dataset. Let $\mathcal{S} = \{(id_i, S_i)\}_{i=1}^n$ be a dataset. An inverted prefix index for it can be built by following steps below, also shown in Alg. 1.

Step 1. We first figure out the complete set of elements in all sets, i.e., $\mathcal{E} = \cup_{i=1}^n S_i$. Then, we sort these elements in ascending order of their frequency in $\mathcal{S}$. The order will be regarded as the global order of elements. Without loss of generality, suppose that the global order of elements is $\mathcal{E} = \{e_1, e_2, \ldots, e_d\}$, and the elements in each $S_i \in \mathcal{S}$ are sorted based on the global order.

Step 2. We retrieve nearest neighbors for sets in $\mathcal{S}$. Let $k_{\max}$ denote the maximum value of k that users can issue. For each $S_i \in \mathcal{S}$, we search on $\mathcal{S}$ for its k-th nearest neighbor $k\mathrm{NN}(S_i)$ and compute $\tau_{i,k} = J(S_i, k\mathrm{NN}(S_i))$ for $1 \le k \le k_{\max}$. Then, we construct a similarity vector for S_i as

$$\mathbf{s}_i = (\tau_{i,1}, \tau_{i,2}, \ldots, \tau_{i,k_{\max}}). \quad (4)$$

Step 3. We identify prefix sets for sets in $\mathcal{S}$. Specifically, for each $S_i \in \mathcal{S}$, we construct a set $\psi(S_i)$ using the first $(\lfloor(1-\tau_{i,k_{\max}}) * |S_i|\rfloor + 1)$ elements of S_i.

Step 4. We build an element-set inverted index according to the constructed prefix sets as follows.

Algorithm 1: Inverted Prefix Index Building

Input: $\mathcal{S} = \{(id_i, S_i)\}_{i=1}^n$: the dataset;
$k_{\max}$: the maximum number of k;
Output: Inverted index $\mathcal{I}$;

```
E = ∪_{i=1}^n S_i;
Obtain a global order of elements by sorting all of them in E based on their frequency, denoted by E = {e_1, e_2, ..., e_d};
Sort elements in each S_i based on the global order;
for each S_i ∈ S do
    Initialize a vector s_i = (τ_{i,1}, τ_{i,2}, ..., τ_{i,k_max});
    for k = 1, 2, ..., k_max do
        τ_{i,k} = J(S_i, kNN(S_i));
    Construct ψ(S_i) using the first (⌊(1 − τ_{i,k_max}) * |S_i|⌋ + 1) elements of S_i;
for each S_i ∈ S do
    for each e_j ∈ ψ(S_i) do
        Initialize a vector b_{j,i} = (b_{j,i,1}, b_{j,i,2}, ..., b_{j,i,k_max});
        for k = 1, 2, ..., k_max do
            if Loc(e_j|S_i) ≤ ⌊(1 − τ_{i,k}) * |S_i|⌋ + 1 then
                b_{j,i,k} = 1;
            else
                b_{j,i,k} = 0;
        w_{j,i} = (id_i, s_i, b_{j,i}, |S_i|);
Set I = ∅;
for e_j ∈ E do
    W_j = {w_{j,i} | e_j ∈ ψ(S_i)};
    I = I ∪ {(e_j, W_j)};
return I;
```

(1) For each $e_j \in \psi(S_i)$, we construct a $k_{\max}$-dimensional vector $\mathbf{b}_{j,i} = (b_{j,i,1}, b_{j,i,2}, \ldots, b_{j,i,k_{\max}})$ as

$$b_{j,i,k} = \begin{cases} 1 & \text{If } \mathrm{Loc}(e_j|S_i) \le \lfloor(1-\tau_{i,k}) * |S_i|\rfloor + 1 \\ 0 & \text{Otherwise,} \end{cases} \quad (5)$$

where $\mathrm{Loc}(e_j|S_i)$ denotes the relative location of e_j in S_i, e.g., the relative location of 4 in the set $\{1, 4, 20\}$ is 2.

(2) We construct a four-tuple $\mathbf{w}_{j,i} = (id_i, \mathbf{s}_i, \mathbf{b}_{j,i}, |S_i|)$ for each pair of e_j and $\psi(S_i)$.

(3) For each $e_j \in \mathcal{E}$, we construct a set $\mathcal{W}_j = \{\mathbf{w}_{j,i} | e_j \in \psi(S_i)\}$. Then, we build an element-set inverted prefix index as $\mathcal{I} = \{(e_j, \mathcal{W}_j) | e_j \in \mathcal{E}\}$, where e_j is the set element, and $\mathcal{W}_j$ is a set associated with all prefix sets containing e_j.

B. Inverted Prefix Index Based Set RkNN Query

Suppose that a dataset $\mathcal{S} = \{(id_i, S_i)\}_{i=1}^n$ has been represented to an inverted prefix index $\mathcal{I} = \{(e_j, \mathcal{W}_j) | e_j \in \mathcal{E}\}$. Let (Q, k) be a set RkNN query request, where Q is a query set and its elements have been sorted based on the global order. Then, we can perform the set RkNN query (Q, k) over the index $\mathcal{I}$. Before performing the query, we first generate query tokens for the query request (Q, k).

1) Token Generation: The query tokens include a filter token and a refinement token, which are respectively used for searching $\mathcal{I}$ for the candidate result and further refining the candidate result. The filter token is in the form of $F_Q = \{\mathbf{w}_{q,j} = (e_j, \texttt{Loc}(e_j|Q))|e_j \in Q\}$, where $\texttt{Loc}(e_j|Q)$ is the relative location of e_j in Q. The refinement token is Q itself. The query tokens is $\{F_Q, Q, k\}$.

2) Query Processing: As shown in Alg. 2, the query processing algorithm includes two stages, i.e., a filter stage and a refinement stage.

• *Filter Stage.* In the filter stage, we use the filter token F_Q to search on the index $\mathcal{I}$ for the candidate result as follows.

Step 1. For each $\mathbf{w}_{q,j} \in F_Q$, we search on $\mathcal{I}$ to find $\mathcal{W}_j$, where $\mathcal{W}_j = \{\mathbf{w}_{j,i}|e_j \in \psi(S_i)\}$.

Step 2. For each $\mathbf{w}_{j,i} \in \mathcal{W}_j$, we determine whether it satisfies the prefix filter and length filter. First, we have $\mathbf{w}_{j,i} = (id_i, \mathbf{s}_i, \mathbf{b}_{j,i}, |S_i|)$ and $\mathbf{w}_{q,j} = (e_j, \texttt{Loc}(e_j|Q))$. If S_i is the set RkNN of Q, we have $J(S_i, Q) \geq \tau_{i,k}$. Meanwhile, S_i and Q satisfy the prefix filter and length filter

$$\begin{cases} \psi(S_i) \cap \psi(Q) \neq \emptyset \\ |S_i| * \tau_{i,k} \leq |Q| \leq \frac{|S_i|}{\tau_{i,k}}. \end{cases} \tag{6}$$

For the prefix filter, we have

$$\psi(S_i) \cap \psi(Q) \neq \emptyset \Leftrightarrow \{e_j \in \psi(S_i) \wedge e_j \in \psi(Q)\}. \tag{7}$$

Meanwhile, we have

$$\begin{aligned} e_j \in \psi(S_i) &\Leftrightarrow \texttt{Loc}(e_j|S_i) \leq \lfloor (1-\tau_{i,k}) * |S_i| \rfloor + 1 \\ &\Leftrightarrow b_{j,i,k} = 1 \end{aligned} \tag{8}$$

and

$$e_j \in \psi(Q) \Leftrightarrow \texttt{Loc}(e_j|Q) \leq \lfloor (1-\tau_{i,k}) * |Q| \rfloor + 1. \tag{9}$$

In this case, the prefix filter can be transformed to

$$\begin{cases} b_{j,i,k} = 1 \\ \texttt{Loc}(e_j|Q) \leq \lfloor (1-\tau_{i,k}) * |Q| \rfloor + 1. \end{cases} \tag{10}$$

Finally, the filter conditions will be

$$\begin{cases} b_{j,i,k} = 1 \\ \texttt{Loc}(e_j|Q) \leq \lfloor (1-\tau_{i,k}) * |Q| \rfloor + 1 \\ |S_i| * \tau_{i,k} \leq |Q| \leq \frac{|S_i|}{\tau_{i,k}}. \end{cases} \tag{11}$$

Thus, in the filter stage, we will pick out the sets whose $\mathbf{w}_{j,i} \in \mathcal{W}_j$ and $\mathbf{w}_{q,j} = (e_j, \texttt{Loc}(e_j|Q))$ satisfy the filter conditions in Eq. (11) for $e_j \in Q$. The candidate result will be

$$\mathcal{C} = \{id_i|\mathbf{w}_{j,i} \text{ and } \mathbf{w}_{q,j} \text{ satisfy Eq. (11) for } e_j \in Q\}. \tag{12}$$

• *Refinement Stage.* In the refinement stage, we will refine the candidate result by verifying whether each candidate set S_i satisfying $J(S_i, Q) \geq \tau_{i,k}$. If yes, we will put it into the final query result, i.e., $\mathcal{R} = \mathcal{R} \cup \{id_i\}$.

Algorithm 2: Inverted Index Based Set RkNN Query

Input: Inverted index $\mathcal{I}$; Query tokens: (F_Q, Q, k);
Output: The query result $\mathcal{R}$;

▷*Filter stage* ◁
Candidate result $\mathcal{C} = \emptyset$;
for each $\mathbf{w}_{q,j} \in Q$ **do**
  Retrieve $\mathcal{W}_j = \{\mathbf{w}_{j,i}|e_j \in \psi(S_i)\}$;
  for each $\mathbf{w}_{j,i} = (id_i, \mathbf{s}_i, \mathbf{b}_{j,i}, |S_i|) \in \mathcal{W}_j$ **do**
    ▷*Prefix filter and length filter* ◁
    if $b_{j,i,k} = 1$ && $\texttt{Loc}(e_j|Q) \leq \lfloor (1-\tau_{i,k}) * |Q| \rfloor + 1$ && $|S_i| * \tau_{i,k} \leq |Q| \leq \frac{|S_i|}{\tau_{i,k}}$ **then**
      $\mathcal{C} = \mathcal{C} \cup \{id_i\}$;
Query result $\mathcal{R} = \emptyset$;
▷*Refinement stage* ◁
for *each* $id_i \in \mathcal{C}$ **do**
  if $J(S_i, Q) \geq \tau_{i,k}$ **then**
    $\mathcal{R} = \mathcal{R} \cup \{id_i\}$;
return $\mathcal{R}$;

C. Oblivious Data Comparison

This section presents two oblivious data comparison protocols, i.e., an oblivious less than comparison (OLTC) protocol and an oblivious greater than comparison (OGTC) protocol.

1) OLTC Protocol: The OLTC protocol is run by S1 and S2. S1 has a ciphertext $[\![m]\!]$, and S2 has the corresponding secret key sk that is used to encrypt $[\![m]\!]$. They can obliviously determine whether $m \leq 0$ or not. Finally, S1 can obtain an encrypted bit $[\![b]\!]$, where

$$b = \begin{cases} 1 & \text{If } m \leq 0 \\ 0 & \text{If } m > 0. \end{cases} \tag{13}$$

Detailed steps are shown as follows.

Step 1: S1 chooses two random numbers $\{r_1, r_2\} \in \mathcal{M}$ satisfying $r_1 > r_2 > 0$. Then, it chooses a number f, where f is set to be either $[\![1]\!]$ or $[\![-1]\!]$ with equal probability. After that, it computes

$$[\![m']\!] = f * (r_1 * [\![m]\!] - r_2) \mod \texttt{N}, \tag{14}$$

and sends $[\![m']\!]$ to S2.

Step 2: On receiving $[\![m']\!]$, S2 decrypts $[\![m']\!]$ and obtains m'. Then, S2 sends an encrypted ciphertext $[\![b']\!]$ to S1, where

$$b' = \begin{cases} 1 & \text{If } m' < 0 \\ 0 & \text{If } m' > 0. \end{cases} \tag{15}$$

Step 3: With $[\![b']\!]$, S1 computes a value $[\![b]\!]$ as

$$[\![b]\!] = \begin{cases} [\![b']\!] & \text{If } f = [\![1]\!] \\ 1 + [\![-1]\!] * [\![b']\!] & \text{If } f = [\![-1]\!]. \end{cases} \tag{16}$$

2) Correctness: We prove the correctness of the OLTC protocol by proving that Eq. (13) holds. 1) When $m \leq 0$ and $f = [\![1]\!]$, we have $m' = (r_1 * m - r_2)$. Since $r_1 > r_2 > 0$, we can infer that $m' < 0$ and $b' = 1$. Based on "$f = [\![1]\!]$", we have $[\![b]\!] = [\![b']\!] = [\![1]\!]$. 2) When $m > 0$ and $f = [\![1]\!]$, we have $m' > 0$ and $b' = 0$. Then, we have $[\![b]\!] = [\![b']\!] = [\![0]\!]$. 3) When $m \leq 0$ and $f = [\![-1]\!]$, we have $m' = -(r_1 * m - r_2)$. Since $r_1 > r_2 > 0$, we have $m' > 0$ and $b' = 0$. Based on "$f = [\![-1]\!]$", we have $[\![b]\!] = 1 + [\![-1]\!] * [\![b']\!] = [\![1]\!]$. 4) When

$m > 0$ and $f = [\![-1]\!]$, we have $m' < 0$ and $b' = 1$. Then, we have $[\![b]\!] = 1 + [\![-1]\!] * [\![b']\!] = [\![0]\!]$. Therefore, Eq. (13) holds, and the OLTC protocol is correct.

3) OGTC Protocol: The OGTC protocol is run by S1 and S2. S1 has a ciphertext $[\![m]\!]$, and S2 has the corresponding secret key sk. They can obliviously determine whether $m \geq 0$ or not. Finally, S1 can obtain an encrypted bit $[\![b]\!]$, where

$$b = \begin{cases} 1 & \text{If } m \geq 0 \\ 0 & \text{If } m < 0. \end{cases} \tag{17}$$

The OGTC protocol is almost the same as that of the OLTC protocol except that $[\![m']\!]$ is computed as

$$[\![m']\!] = f * (r_1 * [\![m]\!] + r_2) \mod \mathtt{N}. \tag{18}$$

The correctness of the OGTC protocol can be easily inferred from that of the OLTC protocol.

D. Private Filter Protocol

This section introduces a private filter protocol to privately determine whether $\mathbf{w}_{j,i}$ and $(\mathbf{w}_{q,j}, |Q|, k)$ satisfy the filter conditions in Eq. (11), where $\mathbf{w}_{j,i} = (id_i, \mathbf{s}_i, \mathbf{b}_{j,i}, |S_i|)$ and $\mathbf{w}_{q,j} = (e_j, \mathtt{Loc}(e_j|Q))$. The protocol is executed between two servers S1 and S2, and the privacy is protected by the SHE scheme. The server S1 has the ciphertexts of $\mathbf{w}_{j,i}$ and $(\mathbf{w}_{q,j}, |Q|, k)$. The server S2 has the corresponding secret key $\mathtt{sk_f}$ used for producing the ciphertexts in S1. They cooperate to determine whether three conditions in Eq. (11) hold or not. Since determining $b_{j,i,k} = 1$ is easy, we only focus on showing how to determine whether $\mathtt{Loc}(e_j|Q) \leq \lfloor(1 - \tau_{i,k}) * |Q|\rfloor + 1$ and $|S_i| * \tau_{i,k} \leq |Q| \leq \frac{|S_i|}{\tau_{i,k}}$.

(1) Determination of $\mathtt{Loc}(e_j|Q) \leq \lfloor(1 - \tau_{i,k}) * |Q|\rfloor + 1$. Since our SHE scheme can only encrypt integers, we first represent $\tau_{i,k}$ into the fraction form, i.e., $\tau_{i,k} = \frac{\alpha_{i,k}}{\beta_{i,k}}$, where both $\alpha_{i,k}$ and $\beta_{i,k}$ are positive integers. Then, the condition $\mathtt{Loc}(e_j|Q) \leq \lfloor(1 - \tau_{i,k}) * |Q|\rfloor + 1$ is equivalent to

$$\begin{aligned} &\mathtt{Loc}(e_j|Q) \leq \lfloor(1 - \frac{\alpha_{i,k}}{\beta_{i,k}}) * |Q|\rfloor + 1 \\ &\quad \Leftrightarrow \ \alpha_{i,k} * |Q| - \beta_{i,k} * (|Q| + 1 - \mathtt{Loc}(e_j|Q)) \leq 0. \end{aligned} \tag{19}$$

(2) Determination of $|S_i| * \tau_{i,k} \leq |Q| \leq \frac{|S_i|}{\tau_{i,k}}$. First, determining $|S_i| * \tau_{i,k} \leq |Q| \leq \frac{|S_i|}{\tau_{i,k}}$ is equivalent to determining

$$(|Q| - |S_i| * \tau_{i,k}) * (|Q| - \frac{|S_i|}{\tau_{i,k}}) \leq 0. \tag{20}$$

Meanwhile, based on $\tau_{i,k} = \frac{\alpha_{i,k}}{\beta_{i,k}}$, the determination can be further transformed to

$$(|S_i| * \alpha_{i,k} - \beta_{i,k} * |Q|) * (\alpha_{i,k} * |Q| - |S_i| * \beta_{i,k}) \geq 0. \tag{21}$$

Based on Eq. (19) and Eq. (21), "$\mathbf{w}_{j,i} = (id_i, \mathbf{s}_i, \mathbf{b}_{j,i}, |S_i|)$" in our scheme will be encrypted into ciphertexts

$$[\![\mathbf{w}_{j,i}]\!] = ([\![id_i]\!], [\![\boldsymbol{\alpha}_i]\!], [\![-\boldsymbol{\beta}_i]\!], [\![\mathbf{b}_{j,i}]\!], [\![|S_i|]\!]) \tag{22}$$

with

$$\begin{cases} [\![\boldsymbol{\alpha}_i]\!] = ([\![\alpha_{i,1}]\!], \ldots, [\![\alpha_{i,k_{\max}}]\!]) \\ [\![-\boldsymbol{\beta}_i]\!] = ([\![-\beta_{i,1}]\!], \ldots, [\![-\beta_{i,k_{\max}}]\!]) \\ [\![\mathbf{b}_{j,i}]\!] = ([\![b_{j,i,1}]\!], \ldots, [\![b_{j,i,k_{\max}}]\!]). \end{cases} \tag{23}$$

Meanwhile, $\mathbf{w}_{q,j} = (e_j, \mathtt{Loc}(e_j|Q))$ will be encrypted into ciphertexts as

$$[\![\mathbf{w}_{q,j}]\!] = ([\![e_j]\!], -\mathtt{Loc}(e_j|Q)). \tag{24}$$

Thus, the server S1 has the encrypted ciphertext $[\![\mathbf{w}_{j,i}]\!] = ([\![id_i]\!], [\![\boldsymbol{\alpha}_i]\!], [\![-\boldsymbol{\beta}_i]\!], [\![\mathbf{b}_{j,i}]\!], [\![|S_i|]\!])$ and $([\![\mathbf{w}_{q,j}]\!], |Q|, k)$. Moreover, we assume that S1 has a ciphertext $[\![-1]\!]$. Then, the server S1 can determine whether $\mathbf{w}_{j,i}$ and $(\mathbf{w}_{q,j}, |Q|, k)$ satisfy the filter conditions in Eq. (11) with the help of S2. The protocol is run as follows.

Step 1. S1 $\rightarrow$ S2: The server S1 first uses homomorphic properties to compute

$$[\![u]\!] \leftarrow [\![\alpha_{i,k}]\!] * |Q| + [\![-\beta_{i,k}]\!] * (|Q| + 1 - \mathtt{Loc}(e_j|Q)) \tag{25}$$

and

$$\begin{aligned} [\![v]\!] \leftarrow &([\![|S_i|]\!] * [\![\alpha_{i,k}]\!] + [\![-\beta_{i,k}]\!] * |Q|) \\ &* ([\![\alpha_{i,k}]\!] * |Q| + [\![-\beta_{i,k}]\!] * [\![|S_i|]\!]). \end{aligned} \tag{26}$$

Then, on inputs of $[\![u]\!]$ (resp. $[\![v]\!]$) and $\mathtt{sk_f}$, S1 and S2 runs the OLTC (resp. OGTC) protocol such that S1 obtains an encrypted $[\![b_u]\!]$ (resp. $[\![b_v]\!]$), where

$$b_u = \begin{cases} 1 & \text{If } u \leq 0 \\ 0 & \text{If } u > 0 \end{cases} \qquad b_v = \begin{cases} 1 & \text{If } u \geq 0 \\ 0 & \text{If } u < 0. \end{cases} \tag{27}$$

Step 3. S1: With $\{[\![b_u]\!], [\![b_v]\!]\}$, S1 computes

$$[\![z]\!] \leftarrow [\![b_{j,i,k}]\!] * [\![b_u]\!] * [\![b_v]\!]. \tag{28}$$

Note that if $z = 1$, we have $\mathbf{w}_{j,i}$ and $(\mathbf{w}_{q,j}, |Q|, k)$ satisfy the filter conditions in Eq. (11), and they do not satisfy otherwise.

Theorem 3: The private filter protocol is correct.

Proof: The protocol is correct *if*

$$z = 1 \Leftrightarrow \begin{cases} b_{j,i,k} = 1 \\ \mathtt{Loc}(e_j|Q) \leq \lfloor(1 - \tau_{i,k}) * |Q|\rfloor + 1 \\ |S_i| * \tau_{i,k} \leq |Q| \leq \frac{|S_i|}{\tau_{i,k}}. \end{cases} \tag{29}$$

Next, we prove its correctness. Based on Eq. (28), we have

$$\begin{aligned} z = 1 &\Leftrightarrow (b_{j,i,k} = 1) \text{ AND } (b_u = 1) \text{ AND } (b_v = 1) \\ &\Leftrightarrow (b_{j,i,k} = 1) \text{ AND } (u \leq 0) \text{ AND } (v \geq 0). \end{aligned} \tag{30}$$

Based on the OLTC and OGTC protocols, we have

$$u \leq 0 \Leftrightarrow \alpha_{i,k} * |Q| - \beta_{i,k} * (|Q| + 1 - \mathtt{Loc}(e_j|Q)) \leq 0$$

and

$$\begin{aligned} &v \geq 0 \\ \Leftrightarrow \ &(|S_i| * \alpha_{i,k} - \beta_{i,k} * |Q|)(\alpha_{i,k} * |Q| - |S_i| * \beta_{i,k}) \geq 0. \end{aligned}$$

In this case, we have

$$\begin{cases} \mathtt{Loc}(e_j|Q) \leq \lfloor(1 - \frac{\alpha_{i,k}}{\beta_{i,k}}) * |Q|\rfloor + 1 \\ (|Q| - |S_i| * \tau_{i,k}) * (|Q| - \frac{|S_i|}{\tau_{i,k}}) \leq 0. \end{cases} \tag{31}$$

By combining $b_{j,i,k} = 1$, we can infer that Eq. (29) is correct. Therefore, the private filter protocol is correct. □

E. Private Refinement Protocol

This section introduces a private refinement protocol to privately determine whether a candidate set S_i and the query request (Q, k) satisfy $J(S_i, Q) \geq \tau_{i,k}$ or not. The protocol is executed between two servers S1 and S2, and the privacy is protected by the SHE scheme with a different pair of public and secret keys from those of the private filter protocol. The server S2 has the ciphertexts of $(S_i, \mathbf{s}_i)$ and (Q, k), where $\mathbf{s}_i = (\tau_{i,1}, \tau_{i,2}, \cdots, \tau_{i,k_{\max}})$. The server S1 has the corresponding secret key $\mathtt{sk}_\mathtt{r}$ used for encrypting the ciphertexts in S2. The core idea of the protocol is to determine whether $J(S_i, Q) \geq \tau_{i,k}$ or not.

Based on the representation $\tau_{i,k} = \frac{\alpha_{i,k}}{\beta_{i,k}}$, determining $J(S_i, Q) \geq \tau_{i,k}$ is equivalent to determining

$$\begin{aligned}\frac{|S_i \cap Q|}{|S_i \cup Q|} \geq \frac{\alpha_{i,k}}{\beta_{i,k}} &\Leftrightarrow \frac{|S_i \cap Q|}{|S_i| + |Q| - |S_i \cap Q|} \geq \frac{\alpha_{i,k}}{\beta_{i,k}} \\ &\Leftrightarrow |S_i \cap Q| * (\alpha_{i,k} + \beta_{i,k}) - (|S_i| + |Q|) * \alpha_{i,k} \geq 0.\end{aligned} \tag{32}$$

In the protocol, $(S_i, \mathbf{s}_i)$ is encrypted into ciphertexts

$$[\![(S_i, \mathbf{s}_i)]\!] \leftarrow ([\![S_i]\!], [\![|S_i|]\!], [\![-\boldsymbol{\alpha}_i]\!], [\![\boldsymbol{\alpha}_i + \boldsymbol{\beta}_i]\!]) \tag{33}$$

with

$$\begin{cases}[\![S_i]\!] = \{([\![e_l^2]\!], [\![-2e_l]\!]) | e_l \in S_i\} \\ [\![-\boldsymbol{\alpha}_i]\!] = ([\![-\alpha_{i,1}]\!], \cdots, [\![-\alpha_{i,k_{\max}}]\!]) \\ [\![\boldsymbol{\alpha}_i + \boldsymbol{\beta}_i]\!] = ([\![\alpha_{i,1} + \beta_{i,1}]\!], \cdots, [\![\alpha_{i,k_{\max}} + \beta_{i,k_{\max}}]\!]).\end{cases} \tag{34}$$

Meanwhile, Q is encrypted as

$$[\![Q]\!] = \{([\![e_j]\!], [\![e_j^2]\!]) | e_j \in Q\}. \tag{35}$$

S2 holds these ciphertexts $[\![(S_i, \mathbf{s}_i)]\!]$ and $([\![Q]\!], |Q|, k)$. S1 has the corresponding secret key $\mathtt{sk}_\mathtt{r}$ and assists S2 to achieve the determination of Eq. (32) as follows.

Step 1. S2 computes $[\![|S_i \cap Q|]\!]$ as follows.

(1) S2 → S1: For each pair of $(e_l, e_j) \in (S_i, Q)$, S2 first computes

$$[\![x_{l,j}]\!] \leftarrow ([\![e_l^2]\!] + [\![-2e_l]\!] * [\![e_j]\!] + [\![e_j^2]\!]) \mod \mathtt{N}. \tag{36}$$

Then, on inputs of $[\![x_{l,j}]\!]$ and $\mathtt{sk}_\mathtt{r}$, S1 and S2 run the OLTC protocol such that S2 obtains an encrypted bit $[\![b_{l,j}]\!]$, where

$$b_{l,j} = \begin{cases}1 & \text{If } x_{l,j} \leq 0 \\ 0 & \text{If } x_{l,j} > 0.\end{cases} \tag{37}$$

Finally, S2 obtains a set $[\![\mathcal{B}_i]\!] = \{[\![b_{l,j}]\!] | e_l \in S_i, e_j \in Q\}$.

(2) S2: With $[\![\mathcal{B}_i]\!]$, S2 computes

$$[\![|S_i \cap Q|]\!] \leftarrow \Big(\sum_{e_l \in S_i, e_j \in Q} [\![b_{l,j}]\!]\Big) \mod \mathtt{N}. \tag{38}$$

The correctness of Eq. (38) will be proved in Theorem 4.

Step 2. S2 achieves the determination of Eq. (32). Specifically, S2 first computes

$$[\![z_i]\!] \leftarrow [\![|S_i \cap Q|]\!] * [\![\alpha_{i,k} + \beta_{i,k}]\!] + ([\![|S_i|]\!] + |Q|) * [\![-\alpha_{i,k}]\!].$$

Then, on inputs $[\![z_i]\!]$ and $\mathtt{sk}_\mathtt{r}$, S1 and S2 run the OGTC protocol such that S2 can obtain an encrypted bit $[\![b_i]\!]$, where

$$b_i = \begin{cases}1 & \text{If } z_i \geq 0 \\ 0 & \text{If } z_i < 0.\end{cases} \tag{39}$$

Meanwhile, "$b_i = 1$" denotes $J(S_i, Q) \geq \tau_{i,k}$ and "$b_i = 0$" denotes $J(S_i, Q) < \tau_{i,k}$.

Theorem 4: The private refinement protocol is correct.

Proof: The protocol is correct *if*

$$\begin{cases}[\![|S_i \cap Q|]\!] \leftarrow \sum_{e_l \in S_i, e_j \in Q} ([\![b_{l,j}]\!]) \mod \mathtt{N} \\ b_i = 1 \Leftrightarrow J(S_i, Q) \geq \tau_{i,k}.\end{cases} \tag{40}$$

(1) Correctness of $[\![|S_i \cap Q|]\!] \leftarrow \sum_{e_l \in S_i, e_j \in Q} ([\![b_{l,j}]\!])$ mod N. When $b_{l,j} = 1$, we have $x_{l,j} \leq 0$. Based on Eq. (36) and homomorphic properties, we have $x_{l,j} = (e_l - e_j)^2$. In this case, we have $x_{l,j} \leq 0 \Leftrightarrow e_l = e_j$ and $[\![|S_i \cap Q|]\!] \leftarrow \sum_{e_l \in S_i, e_j \in Q} ([\![b_{l,j}]\!]) \mod \mathtt{N}$.

(2) Correctness of $b_i = 1 \Leftrightarrow J(S_i, Q) \geq \tau_{i,k}$. When $b_i = 1$, we have $z_i \geq 0$ and can further deduce that $J(S_i, Q) \geq \tau_{i,k}$. Therefore, the private refinement protocol is correct. □

V. Our Proposed Scheme

Based on private filter/refinement protocols, we present our SetRkNN scheme. In this scheme, we elaborately design each step to guarantee the security of t-access pattern unlinkability. Specifically, our scheme has four phases, including System Initialization, Data Outsourcing, Token Generation, and Set RkNN Query Processing.

A. Phase I: System Initialization

The data owner initializes the system. It first chooses security parameters $\{\mathtt{k}_\mathcal{M}, \mathtt{k}_\mathtt{r}, \mathtt{k}_\mathtt{L}, \mathtt{k}_\mathtt{p}, \mathtt{k}_\mathtt{q}\}$ and generates the message space and two pairs of public and secret keys as

$$\begin{cases}\mathtt{pk}_\mathtt{f}, \mathtt{sk}_\mathtt{f}, \mathcal{M} \leftarrow \mathsf{PHE.KeyGen}(\mathtt{k}_\mathcal{M}, \mathtt{k}_\mathtt{r}, \mathtt{k}_\mathtt{L}, \mathtt{k}_\mathtt{p}, \mathtt{k}_\mathtt{q}) \\ \mathtt{pk}_\mathtt{r}, \mathtt{sk}_\mathtt{r} \leftarrow \mathsf{PHE.KeyGen}(\mathtt{k}_\mathcal{M}, \mathtt{k}_\mathtt{r}, \mathtt{k}_\mathtt{L}, \mathtt{k}_\mathtt{p}, \mathtt{k}_\mathtt{q}),\end{cases}$$

where $\{\mathtt{pk}_\mathtt{f}, \mathtt{sk}_\mathtt{f}\}$ and $\{\mathtt{pk}_\mathtt{r}, \mathtt{sk}_\mathtt{r}\}$ are respectively used for protecting the privacy of index search and candidate records refinement. Then, to assist two servers performing RkNN queries, the data owner generates two ciphertexts of -1, i.e.,

$$\begin{cases}[\![-1]\!]_\mathtt{f} \leftarrow \mathsf{SHE.Enc}(-1, \mathtt{sk}_\mathtt{f}); \\ [\![-1]\!]_\mathtt{r} \leftarrow \mathsf{SHE.Enc}(-1, \mathtt{sk}_\mathtt{r}).\end{cases} \tag{41}$$

Note that the data owner here calls the SHE scheme to encrypt -1 rather than the PHE scheme. This is because using the PHE scheme to encrypt the message requires one time homomorphic multiplication between $\{[\![0]\!], [\![0]\!]'\}$ and random numbers $\{\mathtt{r}_1, \mathtt{r}_2\}$ as shown in Eq. (3), which will decrease the multiplicative depth of the PHE scheme by one.

After that, the data owner publishes the public keys $\mathtt{pk}_\mathtt{f}$, $\mathtt{pk}_\mathtt{r}$, and the ciphertexts $[\![-1]\!]_\mathtt{f}$, $[\![-1]\!]_\mathtt{r}$, sends $\mathtt{sk}_\mathtt{f}$ to the server S2, and sends $\mathtt{sk}_\mathtt{r}$ to the server S1. When query

users register to the system, the data owner will authorize them by some common authorization methods such as account/password and send $\mathcal{E}$ to them.

B. Phase II: Data Outsourcing

The data owner outsources its dataset $\mathcal{S} = \{(id_i, S_i)\}_{i=1}^{n}$ to two servers as follows.

Step 1. The data owner builds an element-set inverted prefix index $\mathcal{I} = \{(e_j, \mathcal{W}_j)|e_j \in \mathcal{E}\}$ as Alg. 1.

Step 2. The data owner uses the secret key $\mathtt{sk_f}$ to encrypt the index $\mathcal{I}$ by encrypting each entry $(e_j, \mathcal{W}_j)$ into ciphertexts $(\llbracket e_j \rrbracket, \llbracket \mathcal{W}_j \rrbracket)$, where $\llbracket \mathcal{W}_j \rrbracket \leftarrow \{\llbracket \mathbf{w}_{j,i} \rrbracket | \mathbf{w}_{j,i} \in \mathcal{W}_j\}$ and $\mathbf{w}_{j,i}$ is encrypted according to Eq. (22).

Step 3. The data owner uses the secret key $\mathtt{sk_r}$ to encrypt the dataset $\mathcal{S}$ by encrypting each set $S_i \in \mathcal{S}$ and its similarity vector $\mathbf{s}_i$ into ciphertexts $\llbracket (S_i, \mathbf{s}_i) \rrbracket$ as Eq. (33).

Step 4. The data owner outsources the encrypted index, denoted by $\llbracket \mathcal{I} \rrbracket = \{(\llbracket e_j \rrbracket, \llbracket \mathcal{W}_j \rrbracket)|e_j \in \mathcal{E}\}$, to the server S1, and outsources the encrypted dataset, denoted by $\llbracket \mathcal{S} \rrbracket = \{(id_i, \llbracket (S_i, \mathbf{s}_i) \rrbracket)|S_i \in \mathcal{S}\}$, to the server S2.

C. Phase III: Token Generation

When a query user launches a set RkNN query request (Q, k) with $k \leq k_{\max}$, it generates the query token as follows.

Step 1. The query user generates the query tokens (F_Q, Q, k) as the token generation process in Section IV-B.

Step 2. The query user uses the public key $\mathtt{pk_f}$ to encrypt the filter token $F_Q = \{\mathbf{w}_{q,j}\}_{e_j \in Q}$ into ciphertexts as $\llbracket F_Q \rrbracket \leftarrow \{\llbracket \mathbf{w}_{q,j} \rrbracket | e_j \in Q\}$, where $\mathbf{w}_{q,j}$ is encrypted based on Eq. (24). Then, the query user uses the public key $\mathtt{pk_r}$ to encrypt the refinement token Q into ciphertexts $\llbracket Q \rrbracket$ as Eq. (35).

Step 3. The query user sends the encrypted filter token $(\llbracket F_Q \rrbracket, |Q|, k)$ to the server S1 and sends the encrypted refinement token $(\llbracket Q \rrbracket, |Q|, k)$ to the server S2.

D. Phase IV: Set RkNN Query Processing

In the query processing phase, two servers utilize the query tokens to search for the query result by a filter stage and a refinement stage.

• *Filter Stage.* In the filter stage, the server S1 uses the filter token $(\llbracket F_Q \rrbracket, |Q|, k)$ to search on the index $\llbracket \mathcal{I} \rrbracket$ for the candidate query result with the help of the server S2, where

$$\begin{cases} \llbracket F_Q \rrbracket \leftarrow \{\llbracket \mathbf{w}_{q,j} \rrbracket = (\llbracket e_j \rrbracket, \mathtt{Loc}(e_j|Q))|e_j \in Q\} \\ \llbracket \mathcal{I} \rrbracket = \{(\llbracket e_j \rrbracket, \llbracket \mathcal{W}_j \rrbracket)|e_j \in \mathcal{E}\}. \end{cases} \tag{42}$$

Step 1. Based on Alg. 2, for each $\llbracket \mathbf{w}_{q,j} \rrbracket \in \llbracket F_Q \rrbracket$, the server S1 first searches on $\llbracket \mathcal{I} \rrbracket$ to find $\llbracket \mathcal{W}_j \rrbracket$. Specifically, S1 will search for $\llbracket \mathcal{W}_j \rrbracket$ by comparing $\llbracket e_j \rrbracket$ with each $\llbracket e_l \rrbracket \in \llbracket \mathcal{I} \rrbracket$ as the following steps.

(1) S1 → S2: S1 computes

$$\llbracket g_{j,l} \rrbracket \leftarrow r_1 * (\llbracket e_j \rrbracket^2 + \llbracket -1 \rrbracket * 2 * \llbracket e_j \rrbracket * \llbracket e_l \rrbracket + \llbracket e_l \rrbracket^2) - r_2,$$

where $r_1, r_2 \in \mathcal{M}$ are random numbers satisfying $r_1 > r_2 > 0$. It is easy to observe that when $e_j = e_l$, $g_{j,l} < 0$. Otherwise, $g_{j,l} > 0$. Then, S1 constructs a set $\llbracket \mathcal{G}_j \rrbracket = \{(\mathtt{psu}_{j,l}, \llbracket g_{j,l} \rrbracket)|e_l \in \mathcal{I}\}$, where $\mathtt{psu}_{j,l}$ is a random identity chosen for $\llbracket g_{j,l} \rrbracket$ and is just for simplifying the description. Then, S1 permutates $\llbracket \mathcal{G}_j \rrbracket$ and sends the permutated $\llbracket \mathcal{G}_j \rrbracket$ to the server S2.

(2) S2 → S1: On receiving $\llbracket \mathcal{G}_j \rrbracket$, S2 uses the secret key $\mathtt{sk_f}$ to decrypt $\llbracket \mathcal{G}_j \rrbracket$ and obtains the set $\mathcal{G}_j = \{g_{j,l}|e_l \in \mathcal{I}\}$. Then, S2 identifies the element satisfying $g_{j,l} < 0$, which is associated with the element e_l with $e_l = e_j$. Then, to achieve t-access pattern unlinkability, S2 randomly chooses $t-1$ elements satisfying $g_{j,l} > 0$. After that, it uses the identifies of these elements to form a set $\mathcal{P}_j$ and returns it to S1.

(3) S1: Based on the received $\mathcal{P}_j$, S1 first does a reverse permutation and constructs the set $\llbracket \mathbf{W}_j \rrbracket = \{\llbracket \mathcal{W}_l \rrbracket | \mathtt{psu}_{j,l} \in \mathcal{P}_j\}$, where $\llbracket \mathcal{W}_l \rrbracket \leftarrow \{\llbracket \mathbf{w}_{l,i} \rrbracket | \mathbf{w}_{l,i} \in \mathcal{W}_l\}$. Obliviously, $\llbracket \mathbf{W}_j \rrbracket$ can be further represented to

$$\llbracket \mathbf{W}_j \rrbracket = \{\llbracket \mathbf{w}_{l,i} \rrbracket | \mathtt{psu}_{j,l} \in \mathcal{P}_j, \mathbf{w}_{l,i} \in \mathcal{W}_l\}. \tag{43}$$

Step 2. For each $\llbracket \mathbf{w}_{l,i} \rrbracket \in \llbracket \mathbf{W}_j \rrbracket$, S1 and S2 run the private filter protocol to determine whether it satisfies the filter condition. Meanwhile, S1 can get $\llbracket z_{l,i} \rrbracket$, where "$z_{l,i} = 1$" denotes that $\llbracket \mathbf{w}_{l,i} \rrbracket$ satisfies the filter condition, and "$z_{l,i} = 0$" denotes that $\llbracket \mathbf{w}_{l,i} \rrbracket$ does not satisfy the filter condition.

After that, S1 constructs a set

$$\llbracket \mathcal{B}_j \rrbracket \leftarrow \{(\llbracket id_i \rrbracket, \llbracket z_{l,i} \rrbracket) | \llbracket \mathbf{w}_{l,i} \rrbracket \in \llbracket \mathbf{W}_j \rrbracket\}. \tag{44}$$

Finally, S1 can obtain a set $\llbracket \mathcal{B}_j \rrbracket$ for each $\llbracket \mathbf{w}_{q,j} \rrbracket \in \llbracket F_Q \rrbracket$. Then, it can obtain a set $\llbracket \mathcal{B} \rrbracket = \cup_{\llbracket \mathbf{w}_{q,j} \rrbracket \in \llbracket F_Q \rrbracket} \llbracket \mathcal{B}_j \rrbracket$.

Step 3. S1 identifies the candidate result and sends it to S2. Specifically, it first computes the number of elements that have not been filtered. Based on this value, it adds a certain number of dummy identifies into $\llbracket \mathcal{B} \rrbracket$ and forms the candidate result. Detailed steps are shown as follows.

(1) S1 computes $\llbracket n_q \rrbracket = \sum_{\llbracket z_{l,i} \rrbracket \in \llbracket \mathcal{B} \rrbracket} \llbracket z_{l,i} \rrbracket$. Then, S1 chooses two random numbers $r_1, r_2 \in \mathcal{M}$ and computes $\llbracket n'_q \rrbracket = r_1 * \llbracket n_q \rrbracket + r_2$. After that, S1 sends $\llbracket n'_q \rrbracket$ to S2.

(2) After receiving $\llbracket n'_q \rrbracket$, S2 uses the secret key $\mathtt{sk_f}$ to recover n'_q by decrypting $\llbracket n'_q \rrbracket$. Then, it sends n'_q back to S1.

(3) With n'_q, r_1, and r_2, S1 computes $n_q = \frac{n'_q - r_2}{r_1}$. To achieve t-access pattern unlinkability, S1 randomly chooses $(t-1)*n_q$ dummy identifies, denoted by $\mathcal{D} = \{id_{q,i}\}_{i=1}^{(t-1)*n_q}$ and uses the public key $\mathtt{pk_f}$ to encrypt them into ciphertexts

$$\llbracket \mathcal{D} \rrbracket = \{\llbracket id_{q,i} \rrbracket | 1 \leq i \leq (t-1) * n_q\}. \tag{45}$$

(4) For each pair of $(\llbracket id_i \rrbracket, \llbracket z_{l,i} \rrbracket) \in \llbracket \mathcal{B} \rrbracket$, S1 computes $\llbracket id_i * z_{l,i} \rrbracket$ and organizes them into a set

$$\llbracket \mathcal{B}' \rrbracket = \{\llbracket id_i * z_{l,i} \rrbracket | \llbracket id_i \rrbracket \in \llbracket \mathcal{B} \rrbracket\}. \tag{46}$$

(5) S1 sends the candidate result $\llbracket \mathcal{C} \rrbracket = \llbracket \mathcal{D} \rrbracket \cup \llbracket \mathcal{B}' \rrbracket$ to S2.

• *Refinement Stage.* On receiving the candidate result $\llbracket \mathcal{C} \rrbracket$, S2 refines it as the following steps.

Step 1. S2 uses the secret key $\mathtt{sk_f}$ to decrypt each ciphertext in it and obtains a set of identities

$$\mathcal{C}' = \{id_{q,i}|1 \leq i \leq (t-1) * n_q\} \cup \{id_i | z_{i,j} = 1, id_i \in \mathcal{B}\}.$$

Since the number of identities in $\{id_i|z_{i,j} = 1, id_i \in \mathcal{B}\}$ is n_q, there are $t*n_q$ identities in $\mathcal{C}'$. For simplifying the description, we denote $\mathcal{C}'$ as $\mathcal{C}' = \{id_{q,i}|1 \leq i \leq t * n_q\}$.

Step 2. For each $id_{q,i} \in \mathcal{C}'$, S2 and S1 run the private refinement protocol to verify whether $(\llbracket S_{q,i} \rrbracket, \llbracket \mathbf{s}_{q,i} \rrbracket)$ in S2

satisfies the query request $(\llbracket Q \rrbracket, |Q|, k)$. Meanwhile, S2 can obtain a value $\llbracket b_{q,i} \rrbracket$, where "$b_{q,i} = 1$" denotes that $id_{q,i}$ satisfies the query request and "$b_{q,i} = 0$" denotes $id_{q,i}$ does not satisfy the query request. Then, for each $id_{q,i} \in \mathcal{C}'$, S2 computes $\llbracket h_{q,i} \rrbracket \leftarrow id_{q,i} * \llbracket b_{q,i} \rrbracket + r_{q,i}$, where $r_{q,i} \in \mathcal{M}$ is a random number. Finally, S2 sends the set of ciphertexts $\llbracket \mathcal{R}_1 \rrbracket = \{\llbracket h_{q,i} \rrbracket | id_{q,i} \in \mathcal{C}'\}$ to the server S1 and the set of random numbers $\mathcal{R}_2 = \{r_{q,i} | id_{q,i} \in \mathcal{C}'\}$ to the query user.

Step 3. On receiving $\llbracket \mathcal{R}_1 \rrbracket$, S1 recovers the set $\mathcal{R}_1$ by decrypting each $\llbracket h_{q,i} \rrbracket$. Then, it sends $\mathcal{R}_1 = \{h_{q,i} | id_{q,i} \in \mathcal{C}'\}$ to the query user.

Step 4. The query user uses $\mathcal{R}_1$ and $\mathcal{R}_2$ that are respectively from S1 and S2 to recover the final query result. Specifically, the query user can compute

$$h_{q,i} - r_{q,i} = \begin{cases} id_{q,i} & \text{If } b_{q,i} = 1 \\ 0 & \text{If } b_{q,i} = 0. \end{cases} \tag{47}$$

Then, the query result will be $\mathcal{R} = \{h_{q,i} - r_{q,i} | h_{q,i} - r_{q,i} \neq 0, h_{q,i} \in \mathcal{R}_1\}$. The query result is correct because all sets with $id_{q,i} \in \mathcal{C}'$ satisfy the query request and the corresponding identity $id_{q,i}$ will be the query result.

Remark. It is worth noting that we use homomorphic encryption to protect data privacy, because it has two advantages over the competent privacy-preserving techniques (e.g., secret sharing) in our scheme. On the one hand, our scheme involves many multiplication operations, which can be easily achieved using homomorphic properties of the homomorphic encryption and is communication free. On the other hand, using homomorphic encryption allows the data owner to separately store the index and set records in S1 and S2. Such storage approach contributes to the deign of our t-access pattern unlinkability strategy. That is, S2 can randomly add dummy entries to access the index in S1, and S1 can randomly add dummy entries to access set records in S2.

VI. Security Analysis

In this section, we first revise the security of the SHE scheme and then prove the security of our SetRkNN scheme.

A. Revised Security Proof for the SHE Scheme

Here, we first introduce the $(\mathtt{L}, \mathtt{p})$-based decision problem and revise its intractability and the way of the parameter setting for resisting the potential approximate great common divisor (AGCD) problem. Then, we prove the security of the SHE scheme.

Definition 4 ($(\mathtt{L}, \mathtt{p})$-based decision problem): Suppose that there are two sets $\mathbb{S}$ and $\overline{\mathbb{S}}$ with

$$\begin{cases} \mathbb{S} : \{x = (\alpha \mathtt{L} + \beta \mathtt{p}) \bmod \mathtt{N} | \alpha, \beta \in Z_{\mathtt{N}}, \alpha \mathtt{L} < \mathtt{p}\} \\ \overline{\mathbb{S}} = Z_{\mathtt{N}} \setminus \mathbb{S} : \{x = (\alpha \mathtt{L} + \beta \mathtt{p}) \bmod \mathtt{N} | \alpha, \beta \in Z_{\mathtt{N}}, \alpha \mathtt{L} \geq \mathtt{p}\}. \end{cases}$$

Given $(\mathtt{k_r}, \mathtt{k_L}, \mathtt{k_p}, \mathtt{k_q})$ and a set of instances in $\mathbb{S}$, i.e., $\{x_i = (\alpha_i \mathtt{L} + \beta_i \mathtt{p}) \bmod \mathtt{N} \in \mathbb{S} | i = 1, 2, \cdots, l\}$, the $(\mathtt{L}, \mathtt{p})$-based decision problem is to determine that a random $x \in Z_{\mathtt{N}}$ is in $\mathbb{S}$ or $\overline{\mathbb{S}}$.

• **Intractability of $(\mathtt{L}, \mathtt{p})$-based decision problem.** The $(\mathtt{L}, \mathtt{p})$-based decision problem is to determine whether a random number $x \in Z_{\mathtt{N}}$ is in $\mathbb{S}$ or $\overline{\mathbb{S}}$. To solve this decision problem, the distinguisher $\mathcal{B}$ can attempt to find the values of $(\mathtt{L}, \mathtt{p})$ in the following three ways.

$*$ First, since $\mathtt{N} = \mathtt{p} * \mathtt{q}$, $\mathcal{B}$ can try to factor $\mathtt{N}$ to obtain $\mathtt{p}$, where $\mathtt{q} = \prod_{i=1}^{\lceil \frac{\mathtt{k_q}}{\mathtt{k_p}} \rceil} \mathtt{q}_i$. With $\mathtt{p}$, $\mathcal{B}$ can recover $\mathtt{L}$ from $\{x_i = (\alpha_i \mathtt{L} + \beta_i \mathtt{p}) \bmod \mathtt{N} \in \mathbb{S} | i = 1, 2, \cdots, l\}$. Specifically, $\mathcal{B}$ can compute $x_i \bmod \mathtt{p}$ for $1 \leq i \leq l$ and obtains $\mathtt{L}$ by computing the great common divisor of all values in $\{\alpha_i \mathtt{L} | i = 1, 2, \cdots, l\}$. Once $\mathcal{B}$ knows $(\mathtt{L}, \mathtt{p})$, the $(\mathtt{L}, \mathtt{p})$-based decision problem can be solved. Therefore, we need to choose proper parameters $\{\mathtt{k_p}, \mathtt{k_q}\}$ to ensure the large integer factoring problem is hard.

$*$ Second, $\mathcal{B}$ can exhaust the values of $\{\alpha, \mathtt{L}\}$ and compute $\gcd(x - \alpha \mathtt{L}, \mathtt{N})$ to obtain $\mathtt{p}$. With $(\mathtt{L}, \mathtt{p})$, the $(\mathtt{L}, \mathtt{p})$-based decision problem can be solved. Thus, we need to choose proper $\mathtt{k_r}$ and $\mathtt{k_L}$ such that exhausting $\{\alpha, \mathtt{L}\}$ is hard.

$*$ Third, $\mathcal{B}$ can try to obtain $\mathtt{p}$ from $\{x_i = (\alpha_i \mathtt{L} + \beta_i \mathtt{p}) \bmod \mathtt{N} \in \mathbb{S} | i = 1, 2, \cdots, l\}$ by solving the AGCD problem. Specifically, since $x_i = (\alpha_i \mathtt{L} + \beta_i \mathtt{p}) \bmod \mathtt{N}$ and $\alpha_i \mathtt{L} < \mathtt{p}$, computing $\mathtt{p}$ based on $\{x_i = (\alpha_i \mathtt{L} + \beta_i \mathtt{p}) \bmod \mathtt{N} \in \mathbb{S} | i = 1, 2, \cdots, l\}$ can be reduced to the AGCD problem, as described in [34]. Meanwhile, it is well known that the orthogonal based approach in [35] is one of the most effective approaches to solve the AGCD problem, which reduces the AGCD problem to the shortest non-zero vector solving problem in the lattice with the rank of γ, and γ should satisfy that

$$\gamma \geq \frac{\mathtt{k_q} + \mathtt{k_p} - (\mathtt{d_m} + 1) * (\mathtt{k_r} + \mathtt{k_L})}{\mathtt{k_p} - (\mathtt{d_m} + 1) * (\mathtt{k_r} + \mathtt{k_L})} \tag{48}$$

where i) $\mathtt{k_q} + \mathtt{k_p}$ is the bit length of x_i; ii) $(\mathtt{d_m} + 1) * (\mathtt{k_r} + \mathtt{k_L})$ is the bit length of $\alpha_i * \mathtt{L}$, which is related to the multiplicative depth $\mathtt{d_m}$ of the SHE scheme; and iii) $\mathtt{k_p}$ is the bit length of $\mathtt{p}$. As a result, to guarantee the security of our scheme, we must carefully set $\{\mathtt{k_r}, \mathtt{k_L}, \mathtt{k_p}, \mathtt{k_q}\}$ such that γ must be large enough to guarantee that the shortest non-zero vector solving problem over the lattice with the rank of γ is hard.

Summarizing the above three ways, when the security parameters $\{\mathtt{k_r}, \mathtt{k_L}, \mathtt{k_p}, \mathtt{k_q}\}$ are properly chosen, the $(\mathtt{L}, \mathtt{p})$-based decision problem is intractable. An example parameter setting is that $\mathtt{k_r} = 80$, $\mathtt{k_L} = 80$, $\mathtt{k_p} = 1024$, $\mathtt{k_q} = 236896$, where the maximum multiplicative depth is $\mathtt{d_{max}} = \lfloor \frac{\mathtt{k_p}}{\mathtt{k_r} + \mathtt{k_L}} \rfloor - 1 = 5$. In this parameter setting, the $(\mathtt{L}, \mathtt{p})$-based decision problem is intractable, as shown below.

(1) "$\mathtt{k_r} = 80$" and "$\mathtt{k_L} = 80$" guarantee the hardness of exhausting $\{\alpha, \mathtt{L}\}$;
(2) "$\mathtt{k_p} = 1024$" and "$\mathtt{k_q} = 236896$" guarantee the hardness of factoring $\mathtt{N}$ $(\mathtt{N} = \mathtt{p} * \prod_{i=1}^{\lceil \frac{\mathtt{k_q}}{\mathtt{k_p}} \rceil} \mathtt{q}_i)$;
(3) When "$\mathtt{k_r} = 80$", "$\mathtt{k_L} = 80$", "$\mathtt{k_p} = 1024$", and "$\mathtt{k_q} = 236896$", we can deduce that

$$\frac{\mathtt{k_q} + \mathtt{k_p} - (\mathtt{d_m} + 1) * (\mathtt{k_r} + \mathtt{k_L})}{\mathtt{k_p} - (\mathtt{d_m} + 1) * (\mathtt{k_r} + \mathtt{k_L})} \in [275, 3702] \tag{49}$$

where $\mathtt{d_m} \in [0, \mathtt{d_{max}}] \triangleq \mathtt{d_m} \in [0, 5]$. It means that γ should be equal to or greater than 275 to solve the AGCD problem. When $\gamma \geq 275$, the shortest non-zero vector solving problem in the lattice is hard [36]. That is, the AGCD problem is also hard under this parameter setting, and the $(\mathtt{L}, \mathtt{p})$-based decision problem is intractable.

Based on the intractability of the $(\mathtt{L}, \mathtt{p})$-based decision problem, we can define the $(\mathtt{L}, \mathtt{p})$-based decision assumption, as shown in the following Definition 5.

Definition 5 ($(\mathtt{L}, \mathtt{p})$-Based Decision Assumption): $(\mathtt{L}, \mathtt{p})$-based decision problem satisfies $(\mathtt{L}, \mathtt{p})$-based decision assumption if for any polynomial time algorithm, its advantage in solving the $(\mathtt{L}, \mathtt{p})$-based decision problem is a negligible function in $\mathtt{k_r}$, $\mathtt{k_L}$, $\mathtt{k_p}$, and $\mathtt{k_q}$.

Theorem 5: The SHE scheme is semantically secure against chosen-plaintext attacks (CPA) under the $(\mathtt{L}, \mathtt{p})$-based decision assumption.

Proof: Suppose that there exists a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ that has a non-negligible advantage ε to break the semantic security of the SHE scheme. We can construct a distinguisher $\mathcal{B}$, which has access to $\mathcal{A}$ and can have a non-negligible advantage to break the $(\mathtt{L}, \mathtt{p})$-based decision problem. Let $z \in \{0, 1\}$ be a random bit, and an $(\mathtt{L}, \mathtt{p})$-based decision instance $(\mathtt{k_r}, \mathtt{k_L}, \mathtt{k_p}, \mathtt{k_q}, \mathtt{N}, x)$ is given to $\mathcal{B}$, where x is randomly chosen from $\mathbb{S}$ if $z = 0$, and x is randomly chosen from $\overline{\mathbb{S}}$ if $z = 1$. Then, the $(\mathtt{L}, \mathtt{p})$-based decision problem is to guess z.

With the $(\mathtt{L}, \mathtt{p})$-based decision instance $(\mathtt{k_r}, \mathtt{k_L}, \mathtt{k_p}, \mathtt{k_q}, \mathtt{N}, x)$, $\mathcal{B}$ first chooses $\mathtt{k}_{\mathcal{M}}$ such that $\mathtt{k}_{\mathcal{M}} \ll \mathtt{k_L}$ and sets $\mathcal{M} = \{\mathtt{m} | \mathtt{m} \in [-2^{\mathtt{k}_{\mathcal{M}}-1}, 2^{\mathtt{k}_{\mathcal{M}}-1})\}$ as the message space. Then, $\mathcal{B}$ sends $(\mathtt{k}_{\mathcal{M}}, \mathtt{k_r}, \mathtt{k_L}, \mathtt{k_p}, \mathtt{k_q}, \mathtt{N}, x)$ and $\mathcal{M}$ to $\mathcal{A}$.

On receiving $(\mathtt{k}_{\mathcal{M}}, \mathtt{k_r}, \mathtt{k_L}, \mathtt{k_p}, \mathtt{k_q}, \mathtt{N}, x)$ and $\mathcal{M}$, $\mathcal{A}$ selects two messages $\mathtt{m}_0, \mathtt{m}_1 \in \mathcal{M}$ and sends them to $\mathcal{B}$. Then, $\mathcal{B}$ flips a bit $b \in \{0, 1\}$, computes $\mathtt{c} = \mathtt{m}_b + x$, and returns $\mathtt{c}$ as a ciphertext to $\mathcal{A}$.

After receiving $\mathtt{c}$, $\mathcal{A}$ returns $\mathcal{B}$ a bit $b' \in \{0, 1\}$ as the guess of b. $\mathcal{B}$ then guesses $z = 0$ if $b' = b$. Obviously, when $z = 0$, i.e., $x \in \mathbb{S}$, we have $\mathtt{c} = (\mathtt{m}_b + x) \bmod \mathtt{N} = (\mathtt{m}_b + \alpha\mathtt{L} + \beta\mathtt{p}) \bmod \mathtt{N}$ is a valid ciphertext. In this case, $\mathcal{A}$ can exert his capability and will guess b correctly with the probability $\frac{1}{2} + \varepsilon$. Then, $\Pr[\mathcal{B}\ \text{success} | z = 0] = \frac{1}{2} + \varepsilon$. On the other hand, when $z = 1$, i.e., $x \in \overline{\mathbb{S}}$, $\mathtt{c} = (\mathtt{m}_b + x) \bmod \mathtt{N} = (\mathtt{m}_b + \alpha\mathtt{L} + \beta\mathtt{p}) \bmod \mathtt{N}$ is no longer a valid ciphertext. Then, the probability that $\mathcal{A}$ can guess b correctly is only $\frac{1}{2}$, i.e., $\Pr[\mathcal{B}\ \text{success} | z = 1] = \frac{1}{2}$. Summarizing the above two cases, we have $\Pr[\mathcal{B}\ \text{success}] = \frac{1}{2}(\frac{1}{2} + \varepsilon) + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2} + \frac{\varepsilon}{2}$. Since ε is non-negligible, the result contradicts with the $(\mathtt{L}, \mathtt{p})$-based decision assumption. Thus, the SHE scheme is semantically secure against CPA. □

B. Security Analysis of Our SetRkNN Scheme

In this section, we formally prove the security of our SetRkNN query scheme. Since our scheme is a searchable encryption scheme, we prove its security based on a simulation-based real/ideal worlds model. The real world model works as our SetRkNN scheme, and the ideal world is simulated based on the leakages of our scheme. If the views of adversaries (i.e., two cloud servers) in real and ideal worlds are indistinguishable, our scheme is secure under the leakages that are used for simulating the ideal world. Next, we first define the leakages of our scheme and use them to simulate the ideal world. Then, we define and prove the security of our scheme.

Our scheme has two adversaries, who corrupt S1 and S2, respectively. The information leaked to them is $\mathcal{L}_1$ and $\mathcal{L}_2$.

- $\mathcal{L}_1$: Leakages to S1 include 1) Public parameters: $\mathtt{pk_f}$, $\mathtt{pk_r}$, and $\mathcal{M}$; 2) The secret key: $\mathtt{sk_r}$; 3) The number of elements in $\mathcal{E}$, i.e., $|\mathcal{E}|$; 4) The number of elements in each $\mathcal{W}_j$, i.e., $|\mathcal{W}_j|$, where $e_j \in \mathcal{W}$; 5) The access pattern of the index $[\![\mathcal{I}]\!]$. Note that the access pattern is not the real access pattern, because each data record is accessed with $(t - 1)$ dummy records; 6) The values of $\{|Q|, k_{\max}, k\}$; and 7) Intermediate values, including n_q and $|\mathcal{R}_1|$.
- $\mathcal{L}_2$: Leakages to S2 include 1) Public parameters: $\mathtt{pk_f}$, $\mathtt{pk_r}$, and $\mathcal{M}$; 2) The secret key: $\mathtt{sk_f}$; 3) The number of sets in $\mathcal{S}$, i.e., $|\mathcal{S}|$; 4) The number of elements in each set $S_i \in \mathcal{S}$, i.e., $|S_i|$; 5) The access pattern of $[\![\mathcal{S}]\!]$, which is also not the real access pattern due to dummy records; 6) The value of $\{|Q|, k_{\max}, k\}$; and 7) Intermediate values, including the sign of $g_{j,l}$, i.e., $\mathrm{sign}(g_{j,l})$ and $|[\![\mathcal{C}]\!]|$.

Based on $\mathcal{L}_1$ and $\mathcal{L}_2$, we simulate the ideal world of our SetRkNN query scheme.

Ideal World: The ideal world has four participants, including two probabilistic polynomial-time (PPT) adversaries $\{Adv_1, Adv_2\}$, and two PPT simulators $\{Sim_1, Sim_2\}$, where Sim_1 has $\mathcal{L}_1$ and Sim_2 has $\mathcal{L}_2$. Sim_1 (resp. Sim_2) will intercept messages to Adv_1 (resp. Adv_2), and replace these messages with values that are simulated based on $\mathcal{L}_1$ (resp. $\mathcal{L}_2$). Two adversaries $\{Adv_1, Adv_2\}$ interact with two simulators $\{Sim_1, Sim_2\}$ as follows.

Phase I: Simulated System Initialization: In the system initialization phase, Sim_1 sends $\mathtt{sk_r}$ to Adv_1, and Sim_2 sends $\mathtt{sk_f}$ to Adv_2.

Phase II: Simulated Data Outsourcing: In the data outsourcing phase, Adv_1/Adv_2 chooses a set dataset $\mathcal{S} = \{(id_i, S_i)\}_{i=1}^{n}$ and sends it to Sim_1 and Sim_2. Then, Sim_1 and Sim_2 do the simulation of data outsourcing as follows.

- Sim_1 constructs an inverted prefix index $[\![\mathcal{I}]\!]^{\mathtt{sim}}$ as the simulation of $[\![\mathcal{I}]\!]$. Specifically, Sim_1 chooses $|\mathcal{E}|$ random integers in $Z_{\mathtt{N}}$, denoted by $\{[\![e_j]\!]^{\mathtt{sim}} | 1 \le j \le |\mathcal{E}|\}$, as the simulation of $[\![e_j]\!]$. Then, it constructs $[\![\mathcal{W}_j]\!]^{\mathtt{sim}} \leftarrow \{[\![\mathbf{w}_{j,i}]\!]^{\mathtt{sim}} | 1 \le i \le |\mathcal{W}_j|\}$ as the simulation of $[\![\mathcal{W}_j]\!]$, where $[\![\mathbf{w}_{j,i}]\!]^{\mathtt{sim}} = ([\![id_i]\!]^{\mathtt{sim}}, [\![\boldsymbol{\alpha}_i]\!]^{\mathtt{sim}}, [\![-\boldsymbol{\beta}_i]\!]^{\mathtt{sim}}, [\![\mathbf{b}_{j,i}]\!]^{\mathtt{sim}}, [\![|S_i|]\!]^{\mathtt{sim}})$. Meanwhile, $\{[\![\boldsymbol{\alpha}_i]\!]^{\mathtt{sim}}, [\![-\boldsymbol{\beta}_i]\!]^{\mathtt{sim}}, [\![\mathbf{b}_{j,i}]\!]^{\mathtt{sim}}\}$ are three $k_{\max}$-dimensional random vectors in $Z_{\mathtt{N}}$, and $\{[\![id_i]\!]^{\mathtt{sim}}, [\![|S_i|]\!]^{\mathtt{sim}}\}$ are two random numbers in $Z_{\mathtt{N}}$. Then, Sim_1 sends $\{([\![e_j]\!]^{\mathtt{sim}}, [\![\mathcal{W}_j]\!]^{\mathtt{sim}}) | 1 \le j \le |\mathcal{E}|\}$ to Adv_1 as the simulation of $[\![\mathcal{I}]\!]$.
- Sim_2 constructs the encrypted dataset $[\![\mathcal{S}]\!]^{\mathtt{sim}} = \{[\![(S_i, \mathbf{s}_i)]\!]^{\mathtt{sim}} | 1 \le i \le |\mathcal{S}|\}$ as the simulation of $[\![\mathcal{S}]\!]$, where

$$[\![(S_i, \mathbf{s}_i)]\!]^{\mathtt{sim}} \leftarrow ([\![S_i]\!]^{\mathtt{sim}}, [\![|S_i|]\!]^{\mathtt{sim}}, [\![-\boldsymbol{\alpha}_i]\!]^{\mathtt{sim}}, [\![\boldsymbol{\alpha}_i + \boldsymbol{\beta}_i]\!]^{\mathtt{sim}}).$$

Meanwhile, $[\![S_i]\!]^{\mathtt{sim}}$ is a set with $2 * |S_i|$ random numbers in $Z_{\mathtt{N}}$, $[\![|S_i|]\!]^{\mathtt{sim}}$ is a random number in $Z_{\mathtt{N}}$, and $\{[\![-\boldsymbol{\alpha}_i]\!]^{\mathtt{sim}}, [\![\boldsymbol{\alpha}_i + \boldsymbol{\beta}_i]\!]^{\mathtt{sim}}\}$ are two $k_{\max}$-dimensional random vectors in $Z_{\mathtt{N}}$. Then, Sim_2 sends $[\![\mathcal{S}]\!]^{\mathtt{sim}}$ to Adv_2 as the simulation of $[\![\mathcal{S}]\!]$.

Phase III: Simulated Token Generation: In the token generation phase, Adv_1/Adv_2 chooses a set RkNN query request (Q, k) and sends it to Sim_1 and Sim_2. Then, Sim_1 and Sim_2 do the simulation of token generation.

• Sim_1 constructs a filter token $[\![F_Q]\!]^{\mathtt{sim}}$ as the simulation of $[\![F_Q]\!]$, where $[\![F_Q]\!]^{\mathtt{sim}} \leftarrow \{[\![\mathbf{w}_{q,j}]\!]^{\mathtt{sim}} = ([\![e_j]\!]^{\mathtt{sim}}, -j) | 1 \le j \le |Q|\}$ and $[\![e_j]\!]^{\mathtt{sim}}$ is a random number in $Z_{\mathbb{N}}$. Then, Sim_1 sends $([\![F_Q]\!]^{\mathtt{sim}}, |Q|, k)$ to Adv_1.

• Sim_2 constructs a refinement token $[\![Q]\!]^{\mathtt{sim}}$ as the simulation of $[\![Q]\!]$, where $[\![Q]\!]^{\mathtt{sim}}$ contains $2*|Q|$ random numbers in $Z_{\mathbb{N}}$. Then, Sim_2 sends $([\![Q]\!]^{\mathtt{sim}}, |Q|, k)$ to Adv_2.

Phase IV: Simulated Set RkNN Query Processing: In the query processing phase, Sim_1 and Sim_2 simulate the views of S1 and S2 in the filter/refinement stages.

• *Simulated Filter Stage.* Sim_1 and Sim_2 simulate the views of S1 and S2 in the filter stage. Specifically, the following simulated values will be used for replacing those corresponding values in the filter stage.

Step 1. Simulate $[\![\mathcal{G}_j]\!]$. In the filter stage, Sim_2 constructs a set $[\![\mathcal{G}_j^{\mathtt{sim}}]\!]$ as the simulation of $[\![\mathcal{G}_j]\!]$. Specifically, Sim_2 first constructs a set $\mathcal{G}_j^{\mathtt{sim}} = \{g_{j,l}^{\mathtt{sim}} | 1 \le l \le |\mathcal{E}|\}$ satisfying

$$\begin{cases} g_{j,l}^{\mathtt{sim}} \in \mathcal{M}^+ & \text{If } \mathrm{sign}(g_{j,l}) = \text{“}+\text{”} \\ g_{j,l}^{\mathtt{sim}} \in \mathcal{M}^- & \text{If } \mathrm{sign}(g_{j,l}) = \text{“}-\text{”}, \end{cases} \tag{50}$$

where $\{\mathcal{M}^+, \mathcal{M}^-\}$ respectively denote the collection of positive numbers and negative numbers in $\mathcal{M}$. Then, Sim_2 uses $\mathtt{pk_f}$ to encrypt $\mathcal{G}_j^{\mathtt{sim}}$ as $[\![\mathcal{G}_j^{\mathtt{sim}}]\!] = \{[\![g_{j,l}^{\mathtt{sim}}]\!] | 1 \le l \le |\mathcal{E}|\}$ and sends $[\![\mathcal{G}_j^{\mathtt{sim}}]\!]$ to Adv_2.

Step 2. Simulate $\mathcal{P}_j$. Based on the access pattern of $[\![\mathcal{I}]\!]$, Sim_1 can identify $\mathcal{P}_j$ and then sends it to Adv_1.

Step 3. Simulate for the filter protocol. In the filter protocol, the core computation is to run the OLTC and OGTC protocols to compute $\{[\![b_u]\!], [\![b_v]\!]\}$. We focus on simulating the OLTC and OGTC protocols. In both protocol, Sim_1 needs to simulate $[\![b']\!]$ received by Adv_1, and Sim_2 needs to simulate $[\![m']\!]$ received by Adv_2. Specifically, Sim_1 chooses a random number $[\![b']\!]^{\mathtt{sim}} \in Z_{\mathbb{N}}$ and sends it to Adv_1 as the simulation of $[\![b']\!]$. For the simulation of $[\![m']\!]$, Sim_2 chooses a random number $m'^{\mathtt{sim}} \in \mathcal{M}$ and sends $[\![m'^{\mathtt{sim}}]\!]$ to Adv_2.

Step 4. Simulate $[\![n'_q]\!]$ received by Adv_2 and n'_q received by Adv_1. Sim_2 chooses a random number $1 \le n_q'^{\mathtt{sim}} \le |[\![\mathcal{C}]\!]|/t$ and uses $\mathtt{pk_f}$ to encrypt it into a ciphertext $[\![n_q'^{\mathtt{sim}}]\!]$. Then, Sim_2 sends $[\![n_q'^{\mathtt{sim}}]\!]$ to Adv_2. Then, Sim_1 can pick out n'_q from the leakage $\mathcal{L}_1$ and sends it to Adv_1.

Step 5. Simulate the candidate result $[\![\mathcal{C}]\!]$. Specifically, based on the access pattern of $\mathcal{S}$ in $\mathcal{L}_2$, Sim_2 identifies the set $\mathcal{C}'$ and the number of elements in $[\![\mathcal{C}]\!]$, i.e., $|[\![\mathcal{C}]\!]|$. Then, Sim_2 constructs a set $\mathcal{C}^{\mathtt{sim}} = \mathcal{C}' \cup \{\underbrace{0, 0, \cdots, 0}_{|[\![\mathcal{C}]\!]| - |\mathcal{C}'|}\}$.

Then, Sim_2 uses $\mathtt{pk_f}$ to encrypt each element of $\mathcal{C}^{\mathtt{sim}}$ and sends the encrypted set $[\![\mathcal{C}^{\mathtt{sim}}]\!]$ to Adv_2.

• *Simulated Refinement Stage.* Sim_1 and Sim_2 simulate the views of Adv_1 and Adv_2 in the refinement stage as follows.

Step 1. Simulate for the refinement protocol. The core operations of the refinement protocols are to run OLTC and OGTC protocols. Sim_1 and Sim_2 can do the simulation in a similar way as ***Step 3*** of the simulated filter sage.

Step 2. Simulate $\mathcal{R}_1$. Based on $\mathcal{L}_1$, Sim_1 can obtain $|\mathcal{R}_1|$. Then, Sim_1 generates a set $\mathcal{R}_1^{\mathtt{sim}}$ with $|\mathcal{R}_1|$ random numbers in $\mathcal{M}$. Then, Sim_1 uses $\mathtt{pk_r}$ to encrypt each element of $\mathcal{R}_1^{\mathtt{sim}}$ and sends the encrypted set $[\![\mathcal{R}_1^{\mathtt{sim}}]\!]$ to Adv_1.

In the ideal world, the views of Adv_1 are denoted by $\mathrm{View}_{Adv_1}^{\mathrm{Ideal},\mathcal{L}_1} = \{[\![\mathcal{I}]\!]^{\mathtt{sim}}, [\![F_Q]\!]^{\mathtt{sim}}, |Q|, k, \mathcal{P}_j, [\![b']\!]^{\mathtt{sim}}, n'_q, [\![\mathcal{R}_1^{\mathtt{sim}}]\!]\}$. The views of Adv_2 are denoted by $\mathrm{View}_{Adv_2}^{\mathrm{Ideal},\mathcal{L}_2} = \{[\![\mathcal{S}]\!]^{\mathtt{sim}}, [\![Q]\!]^{\mathtt{sim}}, |Q|, k, [\![\mathcal{G}_j^{\mathtt{sim}}]\!], [\![n_q'^{\mathtt{sim}}]\!], [\![m'^{\mathtt{sim}}]\!], [\![\mathcal{C}^{\mathtt{sim}}]\!]\}$. Meanwhile, the views of Adv_1 and Adv_2 in the real world, denoted by $\mathrm{View}_{Adv_1}^{\mathrm{Real}}$ and $\mathrm{View}_{Adv_2}^{\mathrm{Real}}$, are those generated in our SetRkNN scheme. Based on $\{\mathrm{View}_{Adv_1}^{\mathrm{Ideal},\mathcal{L}_1}, \mathrm{View}_{Adv_2}^{\mathrm{Ideal},\mathcal{L}_2}\}$ and $\{\mathrm{View}_{Adv_1}^{\mathrm{Real}}, \mathrm{View}_{Adv_2}^{\mathrm{Real}}\}$, we can define the security of our scheme.

Definition 6 (Security of SetRkNN): Our SetRkNN scheme is selectively secure with leakages $\mathcal{L}_1$ and $\mathcal{L}_2$ to S1 and S2 *iff* for any two PPT adversaries Adv_1 and Adv_2, there exists two simulators Sim_1 and Sim_2 such that the probability that Adv_1 and Adv_2 can distinguish the views of real world and ideal world (simulated by Sim_1 and Sim_2) is negligible, i.e., both $|\Pr(\mathrm{View}_{Adv_1}^{\mathrm{Ideal},\mathcal{L}_1} = 1) - \Pr(\mathrm{View}_{Adv_1}^{\mathrm{Real}} = 1)|$ and $|\Pr(\mathrm{View}_{Adv_2}^{\mathrm{Ideal},\mathcal{L}_2} = 1) - \Pr(\mathrm{View}_{Adv_2}^{\mathrm{Real}} = 1)|$ are negligible.

Theorem 6: If the SHE scheme is IND-CPA secure, our SetRkNN scheme is selectively secure with $\mathcal{L}_1$ and $\mathcal{L}_2$.

Proof: Based on Definition 6, we prove the selective security of our scheme by showing that Adv_1 and Adv_2 cannot distinguish their views in real and ideal worlds.

For Adv_1, its views in the ideal world are $\mathrm{View}_{Adv_1}^{\mathrm{Ideal},\mathcal{L}_1} = \{[\![\mathcal{I}]\!]^{\mathtt{sim}}, [\![F_Q]\!]^{\mathtt{sim}}, |Q|, k, \mathcal{P}_j, [\![b']\!]^{\mathtt{sim}}, n'_q, [\![\mathcal{R}_1^{\mathtt{sim}}]\!]\}$. Its views in the real world are $\mathrm{View}_{Adv_1}^{\mathrm{Real}} = \{[\![\mathcal{I}]\!], [\![F_Q]\!], |Q|, k, \mathcal{P}_j, [\![b']\!], n'_q, [\![\mathcal{R}_1]\!]\}$, where all values in $\mathrm{View}_{Adv_1}^{\mathrm{Real}}$ are generated based on our SetRkNN scheme. First, $\{[\![\mathcal{I}]\!]^{\mathtt{sim}}, [\![F_Q]\!]^{\mathtt{sim}}, [\![b']\!]^{\mathtt{sim}}\}$ in the ideal world are random numbers in $Z_{\mathbb{N}}$. $\{[\![\mathcal{I}]\!], [\![F_Q]\!], [\![b']\!]\}$ are SHE ciphertexts that are encrypted by $\mathtt{pk_f}$. Since Adv_1 has no access to the secret key of $\mathtt{sk_f}$, the IND-CPA security of SHE scheme can guarantee that Adv_1 cannot distinguish $\{[\![\mathcal{I}]\!]^{\mathtt{sim}}, [\![F_Q]\!]^{\mathtt{sim}}, [\![b']\!]^{\mathtt{sim}}\}$ and $\{[\![\mathcal{I}]\!], [\![F_Q]\!], [\![b']\!]\}$. Second, $\{|Q|, k, \mathcal{P}_j, n'_q\}$ are the same in real and ideal worlds. Adv_1 also cannot use them to distinguish real and ideal worlds. Third, $[\![\mathcal{R}_1^{\mathtt{sim}}]\!]$ in the ideal world is a set with random numbers. $[\![\mathcal{R}_1]\!]$ in the real world contains ciphertexts generated based on our scheme. Since Adv_1 has a secret key $\mathtt{sk_r}$, Adv_1 can recover $\mathcal{R}_1^{\mathtt{sim}}$ and $\mathcal{R}_1$. For $\mathcal{R}_1^{\mathtt{sim}}$ and $\mathcal{R}_1$, both of them contain $|\mathcal{R}_1|$ random numbers, Adv_1 also cannot distinguish them. Therefore, Adv_1 cannot distinguish $\mathrm{View}_{Adv_1}^{\mathrm{Ideal},\mathcal{L}_1}$ and $\mathrm{View}_{Adv_1}^{\mathrm{Real}}$.

For Adv_2, its views in the ideal world are $\mathrm{View}_{Adv_2}^{\mathrm{Ideal},\mathcal{L}_2} = \{[\![\mathcal{S}]\!]^{\mathtt{sim}}, [\![Q]\!]^{\mathtt{sim}}, |Q|, k, [\![\mathcal{G}_j^{\mathtt{sim}}]\!], [\![n_q'^{\mathtt{sim}}]\!], [\![m'^{\mathtt{sim}}]\!], [\![\mathcal{C}^{\mathtt{sim}}]\!]\}$. Its views in the real world are $\mathrm{View}_{Adv_2}^{\mathrm{Real}} = \{[\![\mathcal{S}]\!], [\![Q]\!], |Q|, k, [\![\mathcal{G}_j]\!], [\![n'_q]\!], [\![m']\!], [\![\mathcal{C}]\!]\}$, where all values in $\mathrm{View}_{Adv_1}^{\mathrm{Real}}$ are generated based on our SetRkNN scheme. First, $\{[\![\mathcal{S}]\!]^{\mathtt{sim}}, [\![Q]\!]^{\mathtt{sim}}\}$ are random numbers in $Z_{\mathbb{N}}$, and $\{[\![\mathcal{S}]\!], [\![Q]\!]\}$ are SHE ciphertexts that are encrypted by $\mathtt{pk_r}$. Since Adv_2 does not have access to the secret key $\mathtt{sk_r}$, the IND-CPA security of SHE scheme can guarantee that Adv_2 cannot distinguish $\{[\![\mathcal{S}]\!]^{\mathtt{sim}}, [\![Q]\!]^{\mathtt{sim}}\}$ and $\{[\![\mathcal{S}]\!], [\![Q]\!]\}$. Second, $\{|Q|, k\}$ are the same in real and ideal worlds, so Adv_1 cannot distinguish them. Third, $\{[\![\mathcal{G}_j^{\mathtt{sim}}]\!], [\![m'^{\mathtt{sim}}]\!], [\![n_q'^{\mathtt{sim}}]\!], [\![\mathcal{C}^{\mathtt{sim}}]\!]\}$ and $\{[\![\mathcal{G}_j]\!], [\![m']\!], [\![n'_q]\!], [\![\mathcal{C}]\!]\}$ are SHE ciphertexts encrypted based on our scheme. Since Adv_2 has the secret key $\mathtt{sk_r}$, it can recover the plaintexts $\{\mathcal{G}_j^{\mathtt{sim}}, m'^{\mathtt{sim}}, n_q'^{\mathtt{sim}}, \mathcal{C}^{\mathtt{sim}}\}$ and

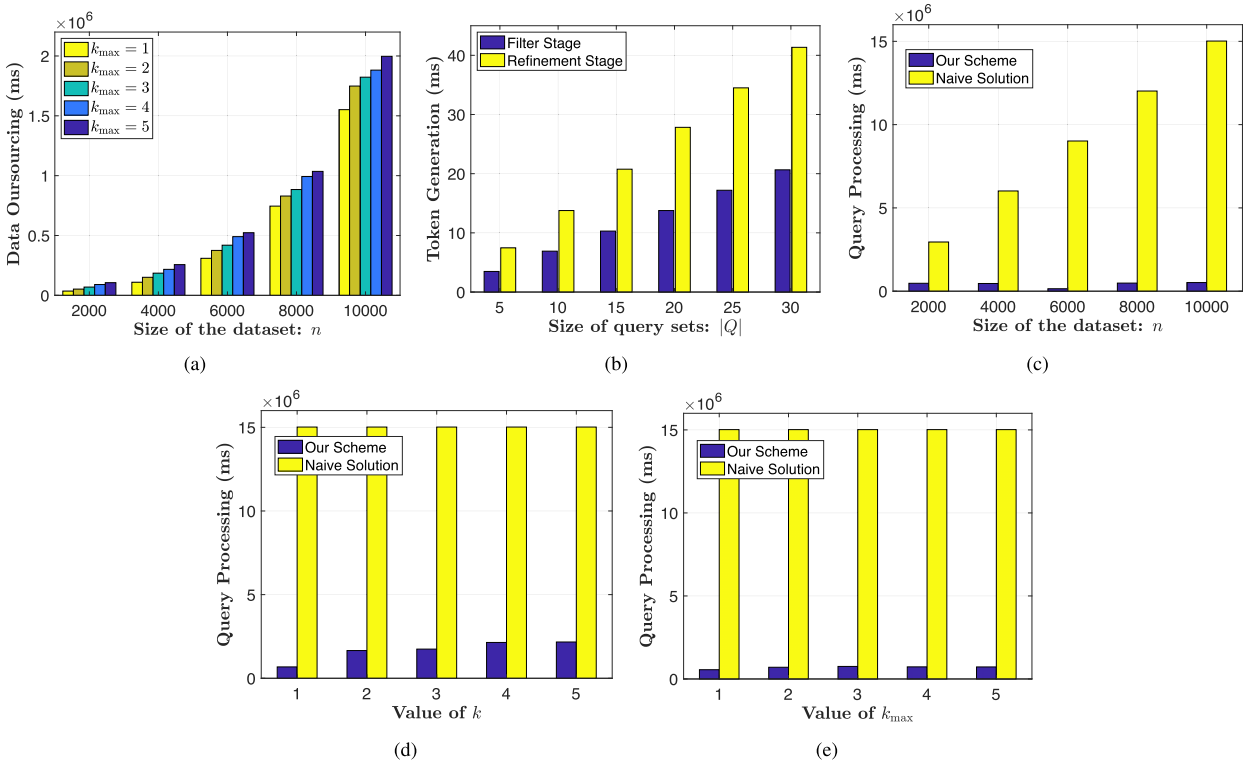

Fig. 2. Computational costs evaluation. (a) Data outsourcing with n and $k_{\max}$; (b) Token generation with $|Q|$; (c) Query processing with n, where $k = k_{\max} = 1$; (d) Query processing with k, where $n = 10000$ and $k_{\max} = 5$; (e) Query processing with $k_{\max}$, where $n = 10000$ and $k = 1$.

$\{\mathcal{G}_j, m', n'_q, \mathcal{C}\}$, where

$$\mathcal{G}_j^{\mathtt{sim}} = \{g_{j,l}^{\mathtt{sim}} | 1 \le l \le |\mathcal{E}|\} \text{ and } \mathcal{G}_j = \{g_{j,l} | e_l \in \mathcal{I}\}. \quad (51)$$

For each pair of $\{g_{j,l}^{\mathtt{sim}}, g_{j,l}\}$, they have the same sign based on Eq. (50). Meanwhile, $g_{j,l}^{\mathtt{sim}}$ is a random number, and $[\![g_{j,l}]\!] \leftarrow r_1 * ([\![e_j]\!]^2 + [\![-1]\!] * 2 * [\![e_j]\!] * [\![e_l]\!] + [\![e_l]\!]^2) - r_2$. Since $[\![g_{j,l}]\!]$ contains two random numbers, it is also a random number. In this case, Adv_1 cannot distinguish $g_{j,l}^{\mathtt{sim}}$ and $g_{j,l}$ for $1 \le l \le |\mathcal{E}|$, i.e., it cannot distinguish $\mathcal{G}_j^{\mathtt{sim}}$ and $\mathcal{G}_j$. Similarly, it also cannot distinguish $m'^{\mathtt{sim}}$ and m'. For $\{n'_q, \mathcal{C}\}$, both of them are random numbers in real and ideal worlds, and Adv_2 cannot distinguish them. Therefore, Adv_2 cannot distinguish $\text{View}_{Adv_2}^{\text{Ideal},\mathcal{L}_2}$ and $\text{View}_{Adv_2}^{\text{Real}}$. In summary, our scheme is selectively secure. □

Theorem 7: Our scheme can preserve the privacy of dataset and query requests, and achieve t-access pattern unlinkability.

Proof: Based on Theorem 6, our scheme only leaks the information in $\mathcal{L}_1$ and $\mathcal{L}_2$. It is obvious that our scheme can protect the plaintexts of dataset $\mathcal{S}$ and query requests (Q, k)'s. Next, we focus on showing that our scheme achieves t-access pattern unlinkability. First, when the server S1 searches on $[\![\mathcal{I}]\!]$ to find $[\![\mathcal{W}_j]\!]$ for $[\![\mathbf{w}_{q,j}]\!] \in [\![F_Q]\!]$. S2 will additionally choose $t-1$ elements and sends their identifies together with j to S1. In this case, $[\![\mathcal{W}_j]\!]$ will be accessed with other $(t-1)$ $[\![\mathcal{W}_i]\!]$'s, where $i \neq j$. Meanwhile, in the refinement stage, S1 will randomly choose $(t-1) * n_q$ dummy identifies $\mathcal{D} = \{id_{q,i}\}_{i=1}^{(t-1)*n_q}$ and sends them to S2 with $[\![\mathcal{B}']\!]$. In this case, each candidate set is accessed with $(t-1)$ sets. Our scheme achieves t-access pattern unlinkability. □

VII. Performance Evaluation

This section shows the experimental results of our scheme in computational costs and communication overheads.

A. Experiment Setting

The scheme prototype was implemented in Java and evaluated on a workstation with Intel(R) Xeon(R) Gold 6226R CPU, 251GB RAM, and Ubuntu 20.04 operating system. Parameters are set as $\mathtt{k}_{\mathcal{M}} = 13$, $\mathtt{k}_{\mathtt{r}} = 80$, $\mathtt{k}_{\mathtt{L}} = 80$, $\mathtt{k}_{\mathtt{p}} = 1024$, $\mathtt{k}_{\mathtt{q}} = 236896$, and $t = 2$. When the multiplicative depth in our scheme is larger than the maximum multiplicative depth of the SHE scheme, we will call the bootstrapping protocol in [19] to remove noise in the encrypted data. The evaluation was performed utilizing a real Jeter dataset [37] that has ratings of 100 jokes from 24,938 users. Each rating ranges from -10 to 10. We transform each user's ratings into a joke set containing all jokes with the user's ratings greater than 7. In the following, we describe the experimental results of computational costs and communication overheads.

B. Computational Costs

We evaluate the computational costs of SetRkNN with respect to data outsourcing, token generation, and set RkNN query processing.

1) Data Outsourcing: The graph in Fig. 2(a) investigates the impact of n and $k_{\max}$ on the computational costs of data outsourcing. As the increase of n, the data outsourcing time sharply grows. In contrast, the increase of $k_{\max}$ only leads to a slight gradual growth of the data outsourcing time. The experimental results are expected because n affects both the

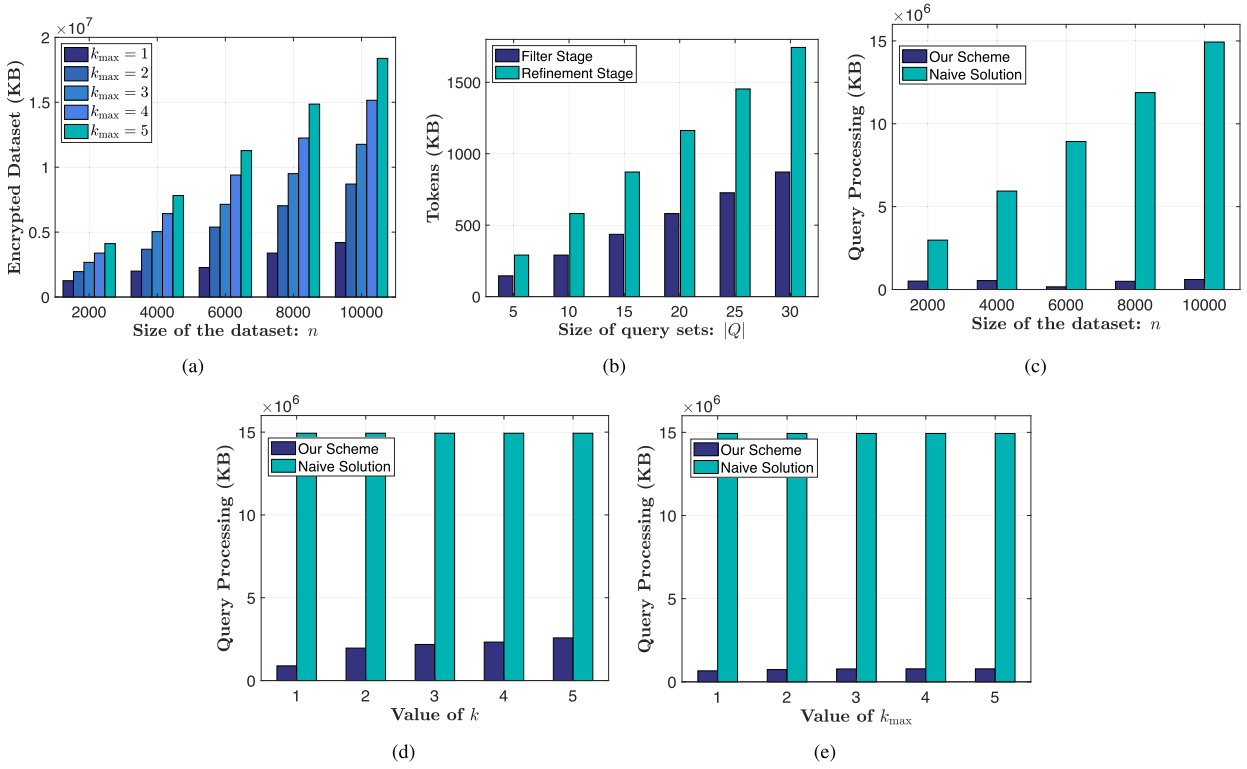

Fig. 3. Communication overheads evaluation. (a) Encrypted dataset transmission with n and $k_{\max}$; (b) Tokens transmission with $|Q|$; (c) Query processing transmission with n, where $k = k_{\max} = 1$; (d) Query processing transmission with k, where $n = 10000$ and $k_{\max} = 5$; (e) Query processing transmission with $k_{\max}$, where $n = 10000$ and $k = 1$.

index building time and dataset encryption time while $k_{\max}$ only affects index building time.

2) Token Generation: The graph in Fig. 2(b) explores the impact of $|Q|$ on the computational costs of token generation in the filter stage and refinement stage, respectively. As shown in Fig. 2(b), the filter and refinement tokens generation time has an expected increase when $|Q|$ becomes larger. The filter token generation is about $1\times$ faster than the refinement token generation because each element of Q corresponds to one SHE ciphertext in the filter token and two SHE ciphertexts in the refinement token.

3) Set RkNN Query Processing: The graph in Fig. 2 studies the impact of n, k, and $k_{\max}$ on the computational costs of query processing under the setting of $|Q| = 5$. Meanwhile, we compare the query processing time of our scheme with that of a naive solution that handles set RkNN queries by accessing all sets in the dataset and using our refinement protocol to do the verification. The graph in Fig. 2(c) indicates that the query processing time generally increases as n gets larger, but it fluctuates. For example, the query processing time in $n = 6000$ is shorter than that in $n = 10000$. The logic behind the fluctuation is that the query time heavily relies on the index of the dataset. However, the size and structure of the index are closely related to the dataset's features, e.g., the similarity values between set records will affect the resulting set $\psi(S_i)$. The graph in Fig. 2(d) shows a definitely increasing trend in the query processing time with the growth of k. For the same query request, when k increases, the number of records in the query answer will become larger. As a result, the filter and refinement costs will increase. The graph in Fig. 2(e) tells us the query processing time increases with the growth of $k_{\max}$. For the same dataset, when $k_{\max}$ becomes larger, the index will also expand. Each element will be associated with more entries, which will lead to an increase in filter costs. The most notable observation is that Fig. 2 indicates the definite superiority of our scheme in comparison to the naive solution in terms of computational costs.

C. Communication Overheads

We evaluate the communication overheads of our scheme in encrypted dataset transmission, tokens transmission, and query processing transmission between two servers.

1) Encrypted Dataset Transmission: The graph in Fig. 3(a) investigates the impact of n and $k_{\max}$ on the transmission overhead of the encrypted dataset. As the increase of n, the size of the encrypted dataset sharply grows, and the increase of $k_{\max}$ only leads to a slight gradual growth of the dataset transmission overhead. The experimental results are expected because n affects the size of the index and dataset while $k_{\max}$ only affects the size of the index.

2) Tokens Transmission: The graph in Fig. 3(b) explores the impact of $|Q|$ on the transmission overhead of filter tokens and refinement tokens. As shown in Fig. 3(b), the filter and refinement tokens transmission overheads increase when $|Q|$ becomes larger. The size of the filter token is half of that of the refinement token because each element of Q corresponds to one SHE ciphertext in the filter token and two SHE ciphertexts in the refinement token.

3) Query Processing Transmission: The graph in Fig. 3 studies the impact of n, k, and $k_{\max}$ on the transmission

overhead of query processing between two servers under the setting of $|Q| = 5$. Meanwhile, we compare the transmission overhead of our scheme with that of the naive solution. The graph in Fig. 3(c) indicates that the query processing time generally increases as n gets larger, but it fluctuates. For example, the query transmission overhead in $n = 6000$ is smaller than that in $n = 10000$. The logic behind the fluctuation is the same as that in the query processing time with n. The graph in Fig. 3(d) shows an increasing trend of the transmission overhead with the growth of k. For the same query request, when k increases, the number of records in the query answer will become larger. As a result, the transmission overheads in the filter and refinement stages will increase. The graph in Fig. 3(e) tells us the transmission overhead increases with the growth of $k_{\max}$. For the same dataset, when $k_{\max}$ becomes larger, the index will also expand. Each element will be associated with more entries, leading to an increase in transmission overheads in the filter stage. The most notable observation is that Fig. 3 indicates the superiority of our scheme over the naive solution in communication overheads.

VIII. Related Works

In this work, we focus on studying privacy-preserving set RkNN queries over encrypted data. In the following, we review some works that are closely related to our work.

A. Privacy-Preserving RkNN Query

Du et al. [9] proposed a privacy-preserving reverse NN query scheme in location-based services over a bichromatic dataset. The scheme translates both dataset records and query records into rectangles for data privacy and introduced the concept of "Voronoi cell for regions" to improve query efficiency. However, the "Voronoi cell for regions" index is only applicable to two-dimensional datasets. Meanwhile, since both dataset and query requests are anonymized into rectangles, the query results are approximate. Lin et al. [12] proposed a privacy-preserving reverse NN query scheme in the road network. This scheme utilizes a network-based Voronoi cell to index the dataset for improving query efficiency and anonymizes query points into related regions to protect query privacy. However, this scheme is only applicable to location data. Pournajaf et al. [10] employed a private information retrieval protocol to design a privacy-preserving RkNN query scheme, which retrieves query results through multiple rounds of PIR protocols. This scheme designs an RkNN-HG index to reduce the number of rounds. However, it does not protect dataset privacy and is only applicable to location data. Li et al. [11] proposed a privacy-preserving reverse NN query scheme based on Delaunay Triangulation, structure encryption, and reference-locked order-preserving encryption. It can support dynamic updates of the dataset. Although it can protect data privacy and query privacy, it is only suitable for two-dimensional data. Tzouramanis and Manolopoulos [13] presented an RkNN query scheme over encrypted multi-dimensional data. This scheme uses a secure SS-tree to index the data for improving query efficiency, but it only supports low-dimensional data and cannot support set data.

B. Privacy-Preserving Jaccard Based Set kNN Query

Singh et al. [14] proposed a secure Jaccard similarity calculation method. This scheme randomizes all elements of set records by multiplying them with the same random number r. Then, it uses a public-key encryption scheme to encrypt all set records and outsources the ciphertexts to the cloud. The cloud server with a secret key decrypts all ciphertexts and performs Jaccard similarity computation over randomized set records. However, this scheme leaks the relationship between set records and is not secure. Blundo et al. [15] utilized a private set intersection technique to design a secure two-party Jaccard similarity calculation protocol, but it is not suitable for the outsourced scenario. Le and Phuong [16] proposed a secure Jaccard similarity based kNN query scheme based on a somewhat homomorphic encryption scheme [38] in a two-server model. Using homomorphic properties, one server computes the intersection and cardinality sum of any two set records. The other server with a secret key decrypts these data and obtains the query result. Nevertheless, this scheme leaks the intersection and cardinality values of set records to the cloud server, which will violate data privacy.

Besides, set records can be transformed into binary vectors, and set similarity computation can be achieved using secure Euclidean distance computation based protocols [27]. As a result, some privacy-preserving Euclidean distance based kNN query schemes are applicable to implement set kNN queries. Such schemes should not introduce data structures because data structures for Euclidean distance are not suitable for sets. Specifically, Wong et al. [17] designed an asymmetric scalar-product-preserving encryption (ASPE) to achieve Euclidean distance based kNN query, but it is not KPA secure. Zhang et al. [18] integrated Paillier homomorphic encryption to enhance the security of the ASPE scheme. Still, it is secure when the adversary cannot access the plaintext of any encrypted query. Thus, it is not a real KPA secure scheme. Zheng et al. [19] proposed a modified ASPE (MASPE) scheme to strengthen the ASPE scheme to be KPA secure. Elmehdwi et al. [20] proposed a secure kNN query scheme by using homomorphic encryption to design bit-wise secure multiplication, secure minimum, etc. The scheme has strong security but is computationally inefficient. Wu et al. [21] proposed integer-based Euclidean distance computation and data comparison protocols. It is more computationally efficient than the scheme in [20]. Unfortunately, when set records are transformed into vectors, the size of vectors is equal to the number of elements in all set records, which is a large number. Meanwhile, set kNN queries have to be processed by linear traversing. Thus, the solutions above are computationally inefficient.

In addition, Zhan et al. [4] proposed a secure kNN classification scheme over the vertically distributed dataset. The scheme is a multi-party computation scheme and designed based on the Paillier homomorphic encryption scheme [39]. Shaneck et al. [3] introduced a secure nearest neighbor search scheme over the horizontally distributed dataset. The scheme is also a multi-party computation scheme and designed based on secure multi-party computation primitives, including secure Euclidean distance computation, secure comparison, and secure division. However, the schemes in [4]

and [3] are multi-party computation schemes with vertically/horizontally distributed datasets and are not applicable to the outsourced scenario considered in our scheme. Meanwhile, some privacy-preserving query schemes have been proposed for other query types, including predication query [8], skyline query [6], and aggregation query [7]. Specifically, Gilad-Bachrach et al. [8] proposed a CryptoNets framework to protect the privacy of neural networks based classification using the leveled homomorphic encryption technique. Benefiting from the homomorphic properties, the proposed scheme is not only privacy-preserving but also computationally accurate. Liu et al. [6] proposed a secure skyline query scheme over encrypted data in the cloud. In this scheme, the authors first introduced a secure dominance protocol based on the Paillier homomorphic encryption [39] and then designed data partitioning and lazy merging optimization methods to reduce the computational costs of skyline query. Zhang et al. [7] proposed a privacy-preserving aggregate query scheme to select an optimal location for query users in road networks. In this scheme, the authors first designed two secure and efficient bit-wise addition and comparison circuits and then introduced a privacy-preserving aggregate query scheme based on these two circuits. However, since the schemes in [6], [8], and [7] were designed for specific query types and are not applicable to the RkNN query over encrypted data.

Different from existing schemes, our scheme aims to achieve efficient and privacy-preserving set RkNN queries over encrypted data, which is the first work for set RkNN queries.

IX. Conclusion

In this paper, we have proposed an efficient and private set RkNN query scheme in the outsourced scenario. We first leveraged the prefix and length filters of Jaccard similarity to design an inverted prefix index and utilized this index to support set RkNN queries with sublinear search efficiency. Then, based on SHE and PHE schemes, we introduced OLTC and OGTC protocols. Based on them, we further designed our private filter/refinement protocols to protect the privacy of index searching and candidate records refinement. After that, we proposed our scheme based on the private filter/refinement protocols and elaborately designed our scheme to guarantee t-access pattern unlinkability. The simulation-based security proof shows that our scheme can protect data privacy and access pattern privacy. Experimental results indicate that our scheme is more efficient in computational costs and communication overheads than the naive solution.

References

[1] Y. Zheng, R. Lu, Y. Guan, S. Zhang, J. Shao, and H. Zhu, "Efficient and privacy-preserving similarity query with access control in eHealthcare," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 880–893, 2022.

[2] Y. Zheng et al., "PMRQ: Achieving efficient and privacy-preserving multidimensional range query in eHealthcare," *IEEE Internet Things J.*, vol. 9, no. 18, pp. 17468–17479, Sep. 2022.

[3] M. Shaneck, Y. Kim, and V. Kumar, "Privacy preserving nearest neighbor search," in *Proc. 6th IEEE Int. Conf. Data Mining Workshops (ICDMW)*, Jan. 2006, pp. 247–276.

[4] J. Z. Zhan, L. Chang, and S. Matwin, "Privacy preserving K-nearest neighbor classification," *Int. J. Netw. Secur.*, vol. 1, no. 1, pp. 46–51, 2005.

[5] Y. Zheng, R. Lu, and J. Shao, "Achieving efficient and privacy-preserving k-NN query for outsourced eHealthcare data," *J. Med. Syst.*, vol. 43, no. 5, p. 123, May 2019.

[6] J. Liu, J. Yang, L. Xiong, and J. Pei, "Secure and efficient skyline queries on encrypted data," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 7, pp. 1397–1411, Jul. 2019.

[7] S. Zhang, S. Ray, R. Lu, Y. Zheng, Y. Guan, and J. Shao, "PPAQ: Privacy-preserving aggregate queries for optimal location selection in road networks," *IEEE Internet Things J.*, vol. 9, no. 20, pp. 20178–20188, Oct. 2022.

[8] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proc. Int. Conf. Mach. Learn. (ICML)*, in (JMLR Workshop and Conference Proceedings), vol. 48, 2016, pp. 201–210.

[9] Y. Du, "Privacy-aware RNN query processing on location-based services," in *Proc. Int. Conf. Mobile Data Manage.*, May 2007, pp. 253–257.

[10] L. Pournajaf, F. Tahmasebian, L. Xiong, V. Sunderam, and C. Shahabi, "Privacy preserving reverse k-nearest neighbor queries," in *Proc. 19th IEEE Int. Conf. Mobile Data Manage. (MDM)*, Jun. 2018, pp. 177–186.

[11] X. Li, T. Xiang, S. Guo, H. Li, and Y. Mu, "Privacy-preserving reverse nearest neighbor query over encrypted spatial data," *IEEE Trans. Services Comput.*, vol. 15, no. 5, pp. 2954–2968, Sep. 2022.

[12] X. Lin, L. Zhou, P. Chen, and J. Gu, "Privacy preserving reverse nearest-neighbor queries processing on road network," in *Proc. XMLDM*, vol. 7419, 2012, pp. 19–28.

[13] T. Tzouramanis and Y. Manolopoulos, "Secure reverse k-nearest neighbours search over encrypted multi-dimensional databases," in *Proc. 22nd Int. Database Eng. Appl. Symp. (IDEAS)*, 2018, pp. 84–94.

[14] M. D. Singh, P. R. Krishna, and A. Saxena, "A privacy preserving Jaccard similarity function for mining encrypted data," in *Proc. TENCON IEEE Region 10 Conf.*, Nov. 2009, pp. 1–4.

[15] C. Blundo, E. D. Cristofaro, and P. Gasti, "EsPRESSo: Efficient privacy-preserving evaluation of sample set similarity," in *Proc. SETOP*, vol. 7731, 2012, pp. 89–103.

[16] T. T. N. Le and T. V. X. Phuong, "Privacy preserving Jaccard similarity by cloud-assisted for classification," *Wireless Pers. Commun.*, vol. 112, no. 3, pp. 1875–1892, Jun. 2020.

[17] W. K. Wong, D. W.-L. Cheung, B. Kao, and N. Mamoulis, "Secure kNN computation on encrypted databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 2009, pp. 139–152.

[18] Z. Zhang, K. Wang, C. Lin, and W. Lin, "Secure top-k inner product retrieval," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2018, pp. 77–86.

[19] Y. Zheng, R. Lu, Y. Guan, J. Shao, and H. Zhu, "Efficient and privacy-preserving similarity range query over encrypted time series data," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 4, pp. 2501–2516, Jul. 2022.

[20] Y. Elmehdwi, B. K. Samanthula, and W. Jiang, "Secure k-nearest neighbor query over encrypted data in outsourced environments," in *Proc. IEEE 30th Int. Conf. Data Eng.*, Mar. 2014, pp. 664–675.

[21] W. Wu, J. Liu, H. Rong, H. Wang, and M. Xian, "Efficient k-nearest neighbor classification over semantically secure hybrid encrypted cloud database," *IEEE Access*, vol. 6, pp. 41771–41784, 2018.

[22] H. Mahdikhani, R. Lu, Y. Zheng, J. Shao, and A. A. Ghorbani, "Achieving $O(\log^3 n)$ communication-efficient privacy-preserving range query in fog-based IoT," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5220–5232, Jun. 2020.

[23] Y. Guan, R. Lu, Y. Zheng, S. Zhang, J. Shao, and G. Wei, "Toward privacy-preserving cybertwin-based spatiotemporal keyword query for ITS in 6G era," *IEEE Internet Things J.*, vol. 8, no. 22, pp. 16243–16255, Nov. 2021, doi: 10.1109/JIOT.2021.3096674.

[24] M.-S. Lacharité, B. Minaud, and K. G. Paterson, "Improved reconstruction attacks on encrypted data using range query leakage," in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Francisco, CA, USA, May 2018, pp. 297–314.

[25] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart, "Leakage-abuse attacks against searchable encryption," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Security*, 2015, pp. 668–679.

[26] Z. Zhang, K. Wang, W. Lin, A. W.-C. Fu, and R. C.-W. Wong, "Practical access pattern privacy by combining PIR and oblivious shuffle," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, Nov. 2019, pp. 1331–1340.

[27] Y. Zheng, R. Lu, Y. Guan, J. Shao, and H. Zhu, "Achieving efficient and privacy-preserving exact set similarity search over encrypted data," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 2, pp. 1090–1103, Mar. 2022.
[28] V. Nikolaenko, S. Ioannidis, U. Weinsberg, M. Joye, N. Taft, and D. Boneh, "Privacy-preserving matrix factorization," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2013, pp. 801–812.
[29] Y. Zheng, H. Duan, and C. Wang, "Learning the truth privately and confidently: Encrypted confidence-aware truth discovery in mobile crowdsensing," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 10, pp. 2475–2489, Oct. 2018.
[30] S. Rane and P. T. Boufounos, "Privacy-preserving nearest neighbor methods: Comparing signals without revealing them," *IEEE Signal Process. Mag.*, vol. 30, no. 2, pp. 18–28, Mar. 2013.
[31] D. Amagata, T. Hara, and C. Xiao, "Dynamic set kNN self-join," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Apr. 2019, pp. 818–829.
[32] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," in *Innovations in Theoretical Computer Science*. Cambridge, MA, USA: ACM, Jan. 2012, pp. 309–325.
[33] J. H. Cheon, A. Kim, M. Kim, and Y. S. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Proc. Adv. Cryptol. ASIACRYPT 23rd Int. Conf. Theory Appl. Cryptol. Inf. Security*, in Lecture Notes in Computer Science, vol. 10624, Dec. 2017, pp. 409–437.
[34] S. D. Galbraith, S. W. Gebregiyorgis, and S. Murphy, "Algorithms for the approximate common divisor problem," *IACR Cryptol. ePrint Arch.*, vol. 19, p. 215, Aug. 2016.
[35] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 6110. Monaco: Springer, 2010, pp. 24–43.
[36] A. Becker, L. Ducas, N. Gama, and T. Laarhoven, "New directions in nearest neighbor searching with applications to lattice sieving," in *Proc. 27th Annu. ACM-SIAM Symp. Discrete Algorithms*, Jan. 2016, pp. 10–24.
[37] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: A constant time collaborative filtering algorithm," *Inf. Retr.*, vol. 4, no. 2, pp. 133–151, Jul. 2001.
[38] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," IACR Cryptol. ePrint Arch., May 2012, p. 144.
[39] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 1592. Prague, Czech Republic: Springer, May 1999, pp. 223–238.

Yandong Zheng (Member, IEEE) received the M.S. degree from the Department of Computer Science, Beihang University, China, in 2017, and the Ph.D. degree from the Department of Computer Science, University of New Brunswick, Canada, in 2022.

Since 2022, she has been an Associate Professor with the School of Cyber Engineering, Xidian University. Her research interests include cloud computing security, big data privacy, and applied privacy.

Rongxing Lu (Fellow, IEEE) received the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Waterloo, Canada, in 2012. He worked as a Post-Doctoral Fellow at the University of Waterloo from May 2012 to April 2013. He is currently the Mastercard IoT Research Chair, a University Research Scholar, and an Associate Professor with the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Canada. Before that, he worked as an Assistant Professor at the School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore, from April 2013 to August 2016. His research interests include applied cryptography, privacy enhancing technologies, and the IoT-big data security and privacy. He also serves as the Chair of IEEE Communications and Information Security Technical Committee (ComSoc CISTC) and the Founding Co-Chair of IEEE TEMS Blockchain and Distributed Ledgers Technologies Technical Committee (BDLT-TC).

Hui Zhu (Senior Member, IEEE) received the B.Sc. degree from Xidian University, Xian, China, in 2003, the M.Sc. degree from Wuhan University, Wuhan, China, in 2005, and the Ph.D. degree from Xidian University, in 2009.

He was a Research Fellow at the School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore, in 2013. Since 2016, he has been a Professor with the School of Cyber Engineering, Xidian University. His current research interests include applied cryptography, data security, and privacy.

Songnian Zhang received the M.S. degree from Xidian University, China, in 2016. He is currently pursuing the Ph.D. degree with the Faculty of Computer Science, University of New Brunswick, Canada. His research interests include cloud computing security, big data query, and query privacy.

Yunguo Guan is currently pursuing the Ph.D. degree with the Faculty of Computer Science, University of New Brunswick, Canada. His research interests include applied cryptography and game theory.

Jun Shao (Senior Member, IEEE) received the Ph.D. degree from the Department of Computer and Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2008.

He was a Post-Doctoral Fellow at the School of Information Sciences and Technology, Pennsylvania State University, Pennsylvania, PA, USA, from 2008 to 2010. He is currently a Professor with the School of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou, China. His current research interests include network security and applied cryptography.

Fengwei Wang (Member, IEEE) received the B.Sc. degree from Xidian University in 2016 and the Ph.D. degree from Xidian University in 2021.

In 2019, he was at the Faculty of Computer Science, University of New Brunswick, as a Visiting Scholar. Since 2021, he has been a Lecturer with the School of Cyber Engineering, Xidian University, Xi'an, China. His research interests include the areas of applied cryptography, cyber security, and privacy.

Hui Li (Member, IEEE) received the B.Sc. degree from Fudan University in 1990, and the M.Sc. and Ph.D. degrees from Xidian University, China, in 1993 and 1998, respectively.

Since 2005, he has been a Professor with the School of Telecommunication Engineering, Xidian University. His research interests are in the areas of cryptography, wireless network security, information theory, and network coding.

Dr. Li served as the TPC Co-Chair of ISPEC 2009 and IAS 2009, the General Co-Chair of E-Forensic 2010, ProvSec 2011, and ISC 2011, and the Honorary Chair of NSS 2014 and ASIACCS 2016.