PRkNN: Efficient and Privacy-Preserving Reverse kNN Query over Encrypted Data

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

Abstract—The advance of cloud computing has driven an emerging trend of outsourcing the rapidly growing data and query services to a powerful cloud for easing the local storage and computing pressure. Meanwhile, when taking data privacy into account, data are usually outsourced to the cloud in an encrypted form. As a result, query services have to be performed over the encrypted data. Among all kinds of query services, the reverse kNN query is highly popular in various applications, such as taxi dispatching and targeted push of multimedia information, but its privacy has not received sufficient attention. To our best knowledge, many existing privacy-preserving reverse kNN query schemes still have some limitations on the query result accuracy, dataset privacy, and flexible support for the choice of the query object and the parameter k. Aiming at addressing these limitations, in this paper, we propose an efficient and privacy-preserving reverse kNN query scheme over encrypted data, named PRkNN. Specifically, we first design a modified M-tree (MM-tree) to index the dataset and further present an MM-Tree based reverse kNN query algorithm in the filter and refinement framework. Then, we leverage the lightweight matrix encryption to carefully design a filter predicate encryption scheme (FPE) and a refinement predicate encryption scheme (RPE); and propose our PRkNN scheme by applying them to protect the privacy of the MM-Tree based reverse kNN query algorithm. Detailed security analysis shows that FPE and RPE schemes are selectively secure, and our PRkNN scheme can preserve both query privacy and dataset privacy. In addition, we conduct extensive experiments to evaluate the performance of our scheme, and the results demonstrate that our scheme is efficient.

Index Terms—Reverse kNN, Encrypted Data, Privacy Preservation, Modified M-Tree, Filter-Refinement.

1 Introduction

The increasing adoption of Industry 4.0, the advance of wireless network technology, and the ubiquity of Internet of Things have fostered the growing volumes of data generated in a wealth of fields and promoted the boom of the big data market [1]. As reported by Technavio [2], the big data market is predicted to have a progressive rise of over USD 247 Billion at about 18% compound annual growth rate during 2021-2025. As a key benefit, big data has the value of offering a variety of data-based query services.

Due to the large data volume and limited processing capabilities of the local data owner, an effective strategy to house the big data and handle the query services is to resort to the influential cloud computing technology, i.e., outsource the big data and the corresponding query services to a powerful cloud server [3], [4]. However, since the cloud server run by a third party is not fully trusted, the privacy of the outsourced data has become a challenging issue. Although data encryption techniques can naturally tackle the privacy issue, they inevitably degrade and even disable the data utility, preventing the cloud server from performing queries over the outsourced data.

- *Y. Zheng, F. Wang, and H. Zhu are with the State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an, 710071, China (e-mail: zhengyandong@xidian.edu.cn, wangfengwei@xidian.edu.cn, zhuhui@xidian.edu.cn). (Corresponding author: Hui Zhu)*
- *R. Lu, S. Zhang, and Y. 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).*
- *J. Shao is with School of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou, 310018, China (e-mail: chn.junshao@gmail.com).*

Targeting different kinds of queries, various privacy-preserving query schemes over outsourced data have been proposed, but the research on privacy-preserving reverse k nearest query (kNN) is far less than expected. The kNN query is to retrieve top-k objects closest to an intentional query object, while the reverse kNN query is to retrieve objects regarding the query object as its kNN, as defined by Definition 1 in Section 4. Since its query results reflect the influence of the query object, the reverse kNN query has broad applications, e.g., taxi dispatching and targeted push of multimedia information [5].

Although some schemes for reverse kNN queries have been proposed, most of them focus on improving the query efficiency over plaintext without considering privacy issues [5]. Several schemes [6]–[10] over encrypted data have taken the privacy into consideration, but they still have some limitations. Specifically, the anonymizing techniques based schemes in [6], [7] ensure the query privacy in a statistical manner while sacrificing the accuracy of query results. The private information retrieval based scheme [8] also only concerns the query privacy without considering the dataset privacy. Although the schemes [9], [10] can well preserve both the query privacy and the dataset privacy, they suffer from some practicality issues. The scheme [9] was designed for spatial data, and it is not trivial to extend the adopted Delaunay Triangulation structure to index multi-dimensional data. Meanwhile, it limits users' query objects to be exact ones existing in the outsourced dataset. The scheme [10] requires the parameter k to be predefined, in which case users cannot select k according to their personalized needs. Thus, it remains a challenge to design a privacy-preserving reverse kNN query scheme over encrypted multi-dimensional data supporting a flexible choice of the query object and the parameter k.

Aiming at addressing this challenge, in this work, we propose an efficient and privacy-preserving reverse kNN query scheme, named PRkNN. Its core idea is to design a modified M-tree (MM-tree) to index the multi-dimensional dataset for improving the query efficiency and supporting the reverse kNN query with a flexible query object and k. Based on the MM-tree, we present a reverse kNN query algorithm and further design a filter predicate encryption scheme (FPE) and a refinement predicate encryption scheme (RPE) to protect the reverse kNN query privacy over the MM-tree. Our contributions are three folds as follows.

- First, we design an MM-tree to index the dataset and further propose an MM-tree based reverse kNN query algorithm in the filter and refinement framework. The algorithm copes with reverse kNN queries in the order of filter and refinement, where the former filters out candidate query results and the latter further refines the candidate results. Thanks to the filter strategy, the query efficiency can be boosted by pruning some branches of the tree that cannot include the query results.
- Second, to protect the privacy of MM-tree based reverse kNN queries, we leverage the lightweight matrix encryption to carefully propose a filter predicate encryption scheme (FPE) and a refinement predicate encryption scheme (RPE), which can provide the security guarantee for the filter stage and refinement stage of the reverse kNN query algorithm, respectively.
- Third, we propose our PRkNN scheme by applying FPE and RPE schemes to preserve the privacy of the MM-tree based reverse kNN query algorithm. Then, we rigorously prove that both FPE and RPE schemes are selectively secure in the simulation-based real/ideal worlds model and show that our PRkNN scheme can preserve the privacy of the query requests and dataset. We conduct extensive experiments to evaluate its performance, and the experimental results demonstrate that our scheme is efficient.

The remainder of this paper is organized as follows. In Section 2, we present some related work. In Section 3, we introduce our system model and security model. Then, we describe some preliminaries in Section 4. In Section 5, we introduce the building blocks of our scheme, including two reverse kNN query algorithms over M-tree and MM-tree, and detailed FPE and RPE schemes. In Section 6, we present our scheme, followed by security analysis and performance evaluation in Section 7 and Section 8, respectively. Finally, we draw our conclusion in Section 9.

2 Related Work

In this section, we review some existing works on privacy-preserving kNN query and reverse kNN query, respectively.

2.1 Privacy-Preserving kNN Query

The kNN query, which targets retrieving top-k data records having the smallest distances to a query record, has a growing number of applications in various fields, including eHealthcare, location-based services, and signal processing. It has therefore received considerable attention from the industry and academia, and a good many privacy-preserving kNN query schemes have been designed using matrix encryption, homomorphic encryption, private information retrieval, and other privacy preservation techniques. In 2009, Wong et al. [11] designed an asymmetric scalar-product-preserving encryption (ASPE) to realize kNN queries, in which the privacy of both dataset records and query records are considered. Based on the ASPE scheme, many privacy-preserving kNN query and similarity range query schemes [12]–[17] were subsequently proposed. However, such schemes cannot resist known-plaintext attacks as proved in [18], so they only have a weak security. To enhance the security of kNN queries, Zheng et al. [19] proposed a modified ASPE scheme by introducing more random numbers into ciphertexts. Meanwhile, many schemes [20]–[23] leverage homomorphic encryption techniques to protect the data privacy and query privacy, where the schemes [21], [22] can even protect the access pattern privacy of the dataset. In addition, some privacy-preserving kNN query schemes [24]–[27] were designed by employing other privacy preservation techniques, e.g., private information retrieval.

2.2 Privacy-Preserving Reverse kNN Query

Reverse kNN query, which targets retrieving data records regarding the query record as kNN, has attracted a growing interest from the industry and academia. Many researches have been done for processing reverse kNN query [5], but only a limited number of works take the privacy into consideration. Specifically, Du et al. [6] presented a privacy-aware reverse kNN query scheme for location-based services by designing a structure, called Voronoi Cell for Regions (VCR), and anonymizing locations in dataset and query requests into rectangles, where the similarity is measured by the Manhattan distance. Due to the anonymizing strategy, this scheme cannot return inaccurate query results. Lin et al. [7] utilized the anonymization techniques to design a privacy-preserving reverse NN query scheme in the scenario of road network, which can protect the query privacy to some degree. However, since this scheme performs reverse kNN query over uncertain data, the query results are also not accurate. To strengthen the security, Pournajaf et al. [8] designed a privacy-preserving reverse kNN query scheme based on the private information retrieval technique. Although the proposed scheme can provide a strong privacy guarantee for the query records, it also does not take the dataset privacy into consideration. Targeting at protecting both the dataset privacy and query privacy, Li et al. [9] employed Delaunay Triangulation as the index structure and further utilized structure encryption, order-preserving encryption as well as the ASPE scheme to design a privacy-preserving reverse kNN query scheme for encrypted spatial data. The proposed scheme can not only efficiently support reverse kNN queries but also dynamically update the dataset. However, since this work focuses on two-dimensional spatial data and it is not trivial to extend the adopted Delaunay Triangulation structure to index multi-dimensional data, the proposed scheme is unable to support reverse kNN queries over multi-dimensional data. In addition, the scheme limits users' query records to be ones existing in the outsourced dataset. To date, the scheme in [10] is the only scheme that can support privacy-preserving reverse kNN queries over encrypted multi-dimensional data and an arbitrary choice of query records. Nevertheless, the parameter k in this scheme must be fixed and predefined before outsourcing the dataset to the cloud, which constrains its flexibility and application scenarios.

Different from the aforementioned schemes, our PRkNN scheme can support privacy-preserving reverse kNN queries over encrypted multi-dimensional data, an arbitrary choice of query records, and the flexible choice of k.

3 System Model and Security Model

In this section, we define the system model and security model considered in our scheme.

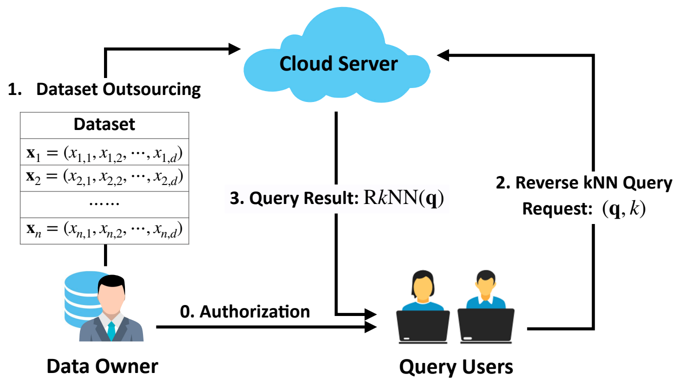

Cloud Server
1. Dataset Outsourcing
Dataset
$\mathbf{x}_1 = (x_{1,1}, x_{1,2}, \cdots, x_{1,d})$
$\mathbf{x}_2 = (x_{2,1}, x_{2,2}, \cdots, x_{2,d})$
……
$\mathbf{x}_n = (x_{n,1}, x_{n,2}, \cdots, x_{n,d})$
3. Query Result: RkNN($\mathbf{q}$)
2. Reverse kNN Query Request: $(\mathbf{q}, k)$
0. Authorization
Data Owner
Query Users

Fig. 1. System model under consideration

3.1 System Model

Our system focuses on a reverse kNN query model under the cloud outsourcing scenario with three kinds of participants: one data owner, one cloud server, and multiple query users, as depicted in Fig. 1.

- Data Owner: The data owner has accumulated a large dataset with n d-dimensional records, denoted by $\mathcal{X} = \{\mathbf{x}_i = (x_{i,1}, x_{i,2}, \cdots, x_{i,d}) | i = 1, 2, \cdots, n\}$. To derive benefits from the data, the data owner exploits $\mathcal{X}$ to offer reverse kNN query services to users in need. Meanwhile, owing to the limited computing power, it outsources $\mathcal{X}$ together with the corresponding reverse kNN query services to the cloud server. To get a fast response on the query action and protect the privacy of $\mathcal{X}$, the data owner indexes the dataset with a customized tree structure and encrypts the index before outsourcing it to the cloud server. Meanwhile, without loss of generality, we assume that all data in the system are integers. If they are not, we can transform them into integers by scaling. Especially, when transforming decimals into integers, the scaling ratio is set based on the precision requirement of real applications. If the precision requirement is high, we can choose a large ratio. Otherwise, we can choose a small ratio.
- Cloud Server: The cloud server with high computing power is delegated by the data owner to offer reverse kNN query services to query users. Suppose that $(\mathbf{q}, k)$ is a reverse kNN query request, where $\mathbf{q} = (q_1, q_2, \cdots, q_d)$ is a d-dimensional query record and k is a positive integer. Upon receiving $(\mathbf{q}, k)$ from a query user, the cloud server traverses the encrypted index of $\mathcal{X}$ for all records regarding $\mathbf{q}$ as one of their k nearest neighbors, i.e., RkNN$(\mathbf{q}) = \{\mathbf{x}_i | \mathbf{x}_i \in \mathcal{X}, d(\mathbf{q}, \mathbf{x}_i) \leq d(k\text{NN}(\mathbf{x}_i), \mathbf{x}_i)\}$, where kNN$(\mathbf{x}_i)$ denotes the k-th closest record to $\mathbf{x}_i$ in $\mathcal{X}$ and $d(\cdot, \cdot)$ denotes the Euclidean distance metric. Finally, the cloud server sends RkNN$(\mathbf{q})$ to the query user as the query result.
- Query Users $\mathcal{U} = \{U_1, U_2, \cdots\}$: Our scheme serves for multiple query users $\mathcal{U} = \{U_1, U_2, \cdots\}$. To ensure that the service is only enjoyed by legitimate users, each U_i is required to be authorized by the data owner when registering in the system. After that, U_i is eligible to launch reverse kNN query requests to the cloud server.

3.2 Security Model

In our security model, since the data owner is in charge of establishing the entire system, we take it for granted that the data owner is *trusted*. For the cloud server, it is managed by a third-party vendor, and as a result, is assumed to be *semi-honest*. To be specific, the cloud server will handle reverse kNN queries in full accordance with our scheme but may be curious to infer the plaintext of the dataset and query requests. Regarding query users, they are authorized by the data owner, and their misbehaviors may result in serious penalties and even the revocation of authorization from the data owner. Hence, we assume that query users are *honest*, fully following our scheme to launch reverse kNN query requests and not colluding with the cloud server. Note that the system may be subject to other passive or active attacks, such as DoS attack. Since this work focuses on privacy preservation, those attacks are beyond the scope of this paper and will be discussed in our future work.

4 Preliminaries

In this section, we recall the formal definition of reverse kNN query and the structure of M-tree that will serve as the preliminaries of our scheme.

- ***Definition of reverse kNN query:*** Let $\mathcal{X} = \{\mathbf{x}_i = (x_{i,1}, x_{i,2}, \cdots, x_{i,d}) | i = 1, 2, \cdots, n\}$ be a dataset and $\mathbf{q} = (q_1, q_2, \cdots, q_d)$ be a d-dimensional query record, and k be a positive integer. The reverse kNN query can be formally defined as the following Definition 1.

Definition 1 (Reverse kNN Query) *A reverse kNN query is in the form of $(\mathbf{q}, k)$ and aims to search on $\mathcal{X}$ for records regarding $\mathbf{q}$ as one of their kNN, and the query result is RkNN$(\mathbf{q}) = \{\mathbf{x}_i | \mathbf{x}_i \in \mathcal{X}, d(\mathbf{q}, \mathbf{x}_i) \leq d(k\text{NN}(\mathbf{x}_i), \mathbf{x}_i)\}$, where kNN$(\mathbf{x}_i)$ denotes the k-th closest record to $\mathbf{x}_i$ in $\mathcal{X}$, and $d(\cdot, \cdot)$ denotes the Euclidean distance metric.*

- ***M-tree:*** M-tree is a tree-based data structure designed for organizing data records based on the distance metric (i.e., Euclidean distance in our work) [28]. Each internal node defines a ball region $\text{B}(\mathbf{p}, \tau)$ with a pivot record $\mathbf{p}$ as the center and a value τ as the radius. Meanwhile, each internal node keeps a set of pointers $(\text{p}_1, \text{p}_2, \cdots)$ pointing to its child nodes in the next level, where all child nodes' ball

Algorithm 1 RkNNQuery(Tree T, Query $(\mathbf{q}, k)$)

```
1: Set $\mathcal{C} = \emptyset$; // Candidate result
2: Set $\mathcal{R} = \emptyset$; // Final result
   // Filter stage
3: $\mathcal{C}$ = Filter(T.$root$, $(\mathbf{q}, k)$);
   // Refinement stage
4: for each $\mathbf{x}_i \in \mathcal{C}$ do
5:     $\tau_{\mathbf{x}_i,k} = d(k\text{NN}(\mathbf{x}_i), \mathbf{x}_i)$;
6:     if $d(\mathbf{x}_i, \mathbf{q}) \le \tau_{\mathbf{x}_i,k}$ then
7:         $\mathcal{R} = \mathcal{R} \cup \{\mathbf{x}_i\}$;
8: return $\mathcal{R}$;
```

Algorithm 2 Filter(Node $node$, Query $(\mathbf{q}, k)$)

```
1: if $node$ is a leaf with $\mathbf{x}_i$ then
2:     $\mathcal{C} = \mathcal{C} \cup \{\mathbf{x}_i\}$;
3: else if $node$ is an internal node with $\text{B}(\mathbf{p}, \tau)$ then
4:     $\tau_{\mathbf{p},k} = d(k\text{NN}(\mathbf{p}), \mathbf{p})$;
5:     if $|\text{Records}(node)| \ge k$ and $d(\mathbf{p}, \mathbf{q}) > 2\tau + \tau_{\mathbf{p},k}$ then
6:         continue;
7:     else
8:         for each child of $\mathbf{p}$, i.e., $node.\text{p}_i$, do
9:             Filter($node.\text{p}_i$, $(\mathbf{q}, k)$);
```

regions reside on the ball region of the current internal node. Particularly, in the original M-tree, to speed up the efficiency of nearest neighbor queries, each internal node keeps a distance value from its ball region center to its parent node's ball region center. Since we focus on the reverse kNN query without using this value, we remove it from our M-tree.

5 Building Blocks

In this section, we propose two reverse kNN query algorithms over M-tree and MM-tree, our FPE scheme, and RPE scheme, which are building blocks of our scheme.

5.1 M-Tree based Reverse kNN Query Algorithm

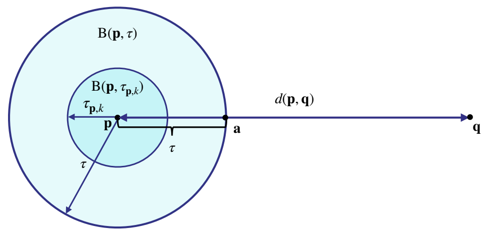
Let T be an M-tree built based on the dataset $\mathcal{X} = \{\mathbf{x}_i | i = 1, 2, \cdots, n\}$ and $(\mathbf{q}, k)$ be a reverse kNN query. The M-tree based reverse kNN query algorithm can be applied to perform the reverse kNN query $(\mathbf{q}, k)$ over the M-tree T. The algorithm is designed based on a filter and refinement framework, which deals with reverse kNN queries through a filter stage and a refinement stage. The former discards some records having no contribution to the query answer and produces a candidate query result, which is then refined by the latter stage as shown in Algorithm 1.

• ***Filter stage:*** As shown in Algorithm 2, the searcher in this stage traverses T in a depth-first manner. When the traversed node is a leaf node with a record $\mathbf{x}_i$, the searcher directly places $\mathbf{x}_i$ into the candidate result, namely, $\mathcal{C} = \mathcal{C} \cup \{\mathbf{x}_i\}$. In case where the traversed one is an internal node with a ball region $\text{B}(\mathbf{p}, \tau)$, if the node satisfies: i) the number of records covered by the node is equal to or larger than k, i.e., $|\text{Records}(node)| \ge k$; and ii) $d(\mathbf{p}, \mathbf{q}) > 2\tau + d(k\text{NN}(\mathbf{p}), \mathbf{p})$, then all records covered by this node cannot contribute to the query answer and can be discarded. Otherwise, the searcher moves forward to filter each child, i.e., $node.\text{p}_i$, of the current node.

• ***Refinement stage:*** As shown in Algorithm 1, the searcher in this stage refines the candidate result $\mathcal{C}$. For each $\mathbf{x}_i \in \mathcal{C}$, if it meets that $d(\mathbf{x}_i, \mathbf{q}) \le d(k\text{NN}(\mathbf{x}_i), \mathbf{x}_i)$, $\mathbf{x}_i$ is a reverse kNN of $\mathbf{q}$ and placed into the final query result, i.e., $\mathcal{R} = \mathcal{R} \cup \{\mathbf{x}_i\}$.

Theorem 1 (Correctness) *The M-tree based reverse kNN query algorithm is correct.*

Proof. We show the correctness of the M-tree based reverse kNN query algorithm by respectively proving the correctness of the filter stage and refinement stage.



$\text{B}(\mathbf{p}, \tau)$
$\text{B}(\mathbf{p}, \tau_{\mathbf{p},k})$
$\tau_{\mathbf{p},k}$
$\mathbf{p}$
$\mathbf{a}$
τ
τ
$d(\mathbf{p}, \mathbf{q})$
$\mathbf{q}$

Fig. 2. Illustration for the correctness of filter stage

In the filter stage, when a node with $\text{B}(\mathbf{p}, \tau)$ satisfies $|\text{Records}(node)| \ge k$ and $d(\mathbf{p}, \mathbf{q}) > 2\tau + d(k\text{NN}(\mathbf{p}), \mathbf{p})$, all records residing on the node will be discarded. The filter strategy is correct because these discarded records cannot be the reverse kNN of $\mathbf{q}$. Specifically, suppose that $\tau_{\mathbf{p},k} = d(k\text{NN}(\mathbf{p}), \mathbf{p})$ is the distance between $\mathbf{p}$ and its k-th closest record $k\text{NN}(\mathbf{p})$. Then, when $|\text{Records}(node)| \ge k$, there is a ball $\text{B}(\mathbf{p}, \tau_{\mathbf{p},k})$ with $\mathbf{p}$ as the center and $\tau_{\mathbf{p},k}$ as the radius inside the ball $\text{B}(\mathbf{p}, \tau)$, as shown in Fig. 2. Meanwhile, $\text{B}(\mathbf{p}, \tau_{\mathbf{p},k})$ contains at least k records. Since the distance from each record $\mathbf{x}_i \in \text{B}(\mathbf{p}, \tau)$ to any record $\mathbf{x}_j \in \text{B}(\mathbf{p}, \tau_{\mathbf{p},k})$ is equal to or less than $\tau + \tau_{\mathbf{p},k}$, i.e., $d(\mathbf{x}_i, \mathbf{x}_j) \le \tau + \tau_{\mathbf{p},k}$, the distance between $\mathbf{x}_i$ to $k\text{NN}(\mathbf{x}_i)$ is equal to or less than $\tau + \tau_{\mathbf{p},k}$, i.e., $d(\mathbf{x}_i, k\text{NN}(\mathbf{x}_i)) \le \tau + \tau_{\mathbf{p},k}$. On the other hand, the distance between $\mathbf{q}$ to $\mathbf{x}_i$ is equal to or greater than $d(\mathbf{p}, \mathbf{q}) - \tau$, namely, $d(\mathbf{q}, \mathbf{x}_i) \ge d(\mathbf{p}, \mathbf{q}) - \tau$. In this case, when $d(\mathbf{p}, \mathbf{q}) - \tau > \tau + \tau_{\mathbf{p},k}$, i.e., $d(\mathbf{p}, \mathbf{q}) > 2\tau + \tau_{\mathbf{p},k}$, we have $d(\mathbf{q}, \mathbf{x}_i) > d(\mathbf{x}_i, k\text{NN}(\mathbf{x}_i))$. It means that $\mathbf{q}$ cannot be one of $\mathbf{x}_i$'s kNN, and as a result, $\mathbf{x}_i$ is not the reverse kNN of $\mathbf{q}$. All discarded records in $\text{B}(\mathbf{p}, \tau)$ cannot be the reverse kNN of $\mathbf{q}$. Therefore, the filter stage is correct. For the refinement stage, the refinement inequality is $d(\mathbf{x}_i, \mathbf{q}) \le d(k\text{NN}(\mathbf{x}_i), \mathbf{x}_i)$, and the correctness is oblivious. □

5.2 MM-tree based Reverse kNN Query Algorithm

In this subsection, we propose a modified M-tree, named MM-tree, and an MM-tree based reverse kNN query algorithm. MM-tree is designed by modifying M-tree. Since the M-tree based reverse kNN query algorithm needs to compute $\tau_{\mathbf{p},k} = d(k\text{NN}(\mathbf{p}), \mathbf{p})$ and $\tau_{\mathbf{x}_i,k} = d(k\text{NN}(\mathbf{x}_i), \mathbf{x}_i)$ when filtering the internal node with $(\mathbf{p}, \tau)$ and refining the leaf node with $\mathbf{x}_i$. To speed up the query efficiency and facilitate the design of the subsequent privacy preservation scheme, we make some modifications to M-tree as follows.

(1) The number of records covered by each internal node must be equal to or greater than a predefined value $k_{\max}$, i.e., $|\text{Records}(node)| \ge k_{\max}$.

(2) In addition to $(\mathbf{p}, \tau)$, each internal node is attached an additional distance list $\mathtt{L}_{\mathbf{p}} = [\tau_{\mathbf{p},1}, \tau_{\mathbf{p},2}, \cdots, \tau_{\mathbf{p},k_{\max}}]$, where $\tau_{\mathbf{p},k} = d(k\text{NN}(\mathbf{p}), \mathbf{p})$.

(3) Besides $\mathbf{x}_i$, each leaf node is attached an additional distance list $\mathtt{L}_{\mathbf{x}_i} = [\tau_{\mathbf{x}_i,1}, \tau_{\mathbf{x}_i,2}, \cdots, \tau_{\mathbf{x}_i,k_{\max}}]$, where $\tau_{\mathbf{x}_i,k} = d(k\text{NN}(\mathbf{x}_i), \mathbf{x}_i)$.

The modified M-tree is our MM-tree, which can efficiently support reverse kNN queries whose k satisfies $k \le k_{\max}$. Since each internal node in MM-tree satisfies $|\text{Records}(node)| \ge k_{\max} \ge k$, we naturally have $|\text{Records}(node)| \ge k$. Thus, the filter condition in Algorithm 2 will become $d(\mathbf{p}, \mathbf{q}) > 2\tau + \tau_{\mathbf{p},k}$. In this case, the basic operations in the filter and refinement stages of reverse kNN queries over MM-Tree can be summarized as:

$$\begin{cases} d(\mathbf{p}, \mathbf{q}) > 2\tau + \tau_{\mathbf{p},k} & \text{Filter stage} \\ d(\mathbf{x}_i, \mathbf{q}) \le \tau_{\mathbf{x}_i,k} & \text{Refinement stage.} \end{cases} \tag{1}$$

Therefore, preserving the privacy of MM-Tree based reverse kNN queries is to protect the computation privacy of two inequalities in Eq. (1), which can be achieved by the FPE scheme and RPE scheme in Section 5.3 and Section 5.4.

5.3 FPE Scheme

On input a reverse kNN query $(\mathbf{q}, k)$ and an internal node with $(\mathbf{p}, \tau, \mathtt{L}_{\mathbf{p}} = [\tau_{\mathbf{p},1}, \tau_{\mathbf{p},2}, \cdots, \tau_{\mathbf{p},k_{\max}}])$, the filter predication encryption, i.e., FPE scheme, is designed to privately determine whether the filter inequality holds or not, i.e., whether $d(\mathbf{p}, \mathbf{q}) > 2\tau + \tau_{\mathbf{p},k}$ or not.

First, we have

$$\begin{aligned} & d(\mathbf{p}, \mathbf{q}) > 2\tau + \tau_{\mathbf{p},k} \\ \Leftrightarrow\ & d(\mathbf{p}, \mathbf{q})^2 > (2\tau + \tau_{\mathbf{p},k})^2 \\ \Leftrightarrow\ & ||\mathbf{p}||^2 - 4\tau^2 - 4\tau\tau_{\mathbf{p},k} - \tau_{\mathbf{p},k}^2 - 2\mathbf{p} \circ \mathbf{q} + ||\mathbf{q}||^2 > 0. \end{aligned} \tag{2}$$

If let

$$\begin{cases} \mathbf{P} = \begin{bmatrix} ||\mathbf{p}||^2 - 4\tau^2 & \tau & \mathbf{p} & 1 \end{bmatrix} \\ \mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -4 & 0 \\ -2\mathbf{q} & 0 & 0 \\ ||\mathbf{q}||^2 & 0 & -1 \end{bmatrix} \\ \mathbf{T} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \tau_{\mathbf{p},1} & \tau_{\mathbf{p},2} & \cdots & \tau_{\mathbf{p},k_{\max}} \\ \tau_{\mathbf{p},1}^2 & \tau_{\mathbf{p},2}^2 & \cdots & \tau_{\mathbf{p},k_{\max}}^2 \end{bmatrix} \\ \mathbf{e}_k = \begin{bmatrix} 0 & \cdots & 1 & \cdots & 0 \end{bmatrix} \end{cases} \tag{3}$$

we can infer that

$$\mathbf{PQTe}_k^T = ||\mathbf{p}||^2 - 4\tau^2 - 4\tau\tau_{\mathbf{p},k} - \tau_{\mathbf{p},k}^2 - 2\mathbf{p} \circ \mathbf{q} + ||\mathbf{q}||^2,$$

where $\mathbf{e}_k$ is a $k_{\max}$-dimensional vector with $\mathbf{e}_{k,k} = 1$ and $\mathbf{e}_{k,i} = 0$ for $i \ne k$. By further combining the equation with Eq. (2), we have

$$d(\mathbf{p}, \mathbf{q}) > 2\tau + \tau_{\mathbf{p},k} \Leftrightarrow \mathbf{PQTe}_k^T > 0. \tag{4}$$

Thus, privately determining "$d(\mathbf{p}, \mathbf{q}) > 2\tau + \tau_{\mathbf{p},k}$" is equivalent to privately determine that of "$\mathbf{PQTe}_k^T > 0$". Considering that $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{T}$, $\mathbf{e}_k$ are either vectors or matrices, we utilize matrix encryption to devise our FPE scheme for protecting the privacy of "$\mathbf{PQTe}_k^T > 0$". The main idea of our scheme is to randomly split each of $\mathbf{P}, \mathbf{Q}, \mathbf{T}, \mathbf{e}_k$ into two parts $\{\mathbf{P}_1, \mathbf{P}_2\}$, $\{\mathbf{Q}_1, \mathbf{Q}_2\}$, $\{\mathbf{T}_1, \mathbf{T}_2\}$, $\{\mathbf{e}_{k,1}, \mathbf{e}_{k,2}\}$ such that

$$\begin{cases} \mathbf{P}_1 + \mathbf{P}_2 = \mathbf{P}; \quad \mathbf{Q}_1 + \mathbf{Q}_2 = \mathbf{Q}; \\ \mathbf{T}_1 + \mathbf{T}_2 = \mathbf{T}; \quad \mathbf{e}_{k,1} + \mathbf{e}_{k,2} = \mathbf{e}_k. \end{cases} \tag{5}$$

For example, we can split $\mathbf{P}$ by 1) randomly choosing a matrix $\mathbf{P}_1$ with the same size as $\mathbf{P}$; and 2) computing $\mathbf{P}_2 = \mathbf{P} - \mathbf{P}_1$. In a similar way, $\mathbf{Q}$, $\mathbf{T}$, and $\mathbf{e}_k$ can be respectively split into $\{\mathbf{Q}_1, \mathbf{Q}_2\}$, $\{\mathbf{T}_1, \mathbf{T}_2\}$, $\{\mathbf{e}_{k,1}, \mathbf{e}_{k,2}\}$. After splitting, we have

$$\begin{aligned} \mathbf{PQTe}_k^T &= (\mathbf{P}_1 + \mathbf{P}_2)(\mathbf{Q}_1 + \mathbf{Q}_2)(\mathbf{T}_1 + \mathbf{T}_2)(\mathbf{e}_{k,1}^T + \mathbf{P}_{k,2}^T) \\ &= \sum_{u=1}^{2}\sum_{v=1}^{2}\sum_{l=1}^{2}\sum_{z=1}^{2} \mathbf{P}_u\mathbf{Q}_v\mathbf{T}_l\mathbf{e}_{k,z}^T. \end{aligned} \tag{6}$$

Then, we use secret matrices to encrypt $\{\mathbf{P}_u, \mathbf{Q}_v, \mathbf{T}_l, \mathbf{e}_{k,z}\}$ for $\{u, v, l, z\} \in \{1, 2\}$. Since $\{\mathbf{P}_u, \mathbf{T}_l\}$ and $\{\mathbf{Q}_v, \mathbf{e}_{k,z}\}$ are respectively held by the data owner and query user, they will be independently encrypted into a ciphertext $\mathtt{C}_{\mathbf{P},\mathbf{T}}$ and a trapdoor $\mathtt{TD}_{\mathbf{Q},\mathbf{e}_k}$ by the data owner and the query user using $\mathsf{FPE.Enc}(sk, \mathbf{P}, \mathbf{T})$ and $\mathsf{FPE.TrapdoorGen}(sk, \mathbf{Q}, \mathbf{e}_k)$ algorithms below. Based on $\mathtt{C}_{\mathbf{P},\mathbf{T}}$ and $\mathtt{TD}_{\mathbf{Q},\mathbf{e}_k}$, we can do the filter using $\mathsf{FPE.FilterQuery}(\mathtt{C}_{\mathbf{P},\mathbf{T}}, \mathtt{TD}_{\mathbf{Q},\mathbf{e}_k})$ algorithm below. Specifically, the FPE scheme has four algorithms, i.e., $\Pi_{\mathtt{FPE}} = (\mathsf{FPE.Setup}, \mathsf{FPE.Enc}, \mathsf{FPE.TrapdoorGen}, \mathsf{FPE.FilterQuery})$.

• $\mathsf{FPE.Setup}(d)$: Given the dimensions of data to be processed, denoted by d, the setup algorithm setups the scheme by randomly selecting a group of invertible matrices as the secret key, namely,

$$\begin{aligned} sk = \{\, & \mathbf{M}_{1,u,v,l,z}, \mathbf{M}_{2,u,v,l,z}, \mathbf{M}_{3,u,v,l,z}, \mathbf{M}'_{1,u,v,l,z}, \\ & \mathbf{M}'_{2,u,v,l,z}, \mathbf{M}'_{3,u,v,l,z} | \{u, v, l, z\} \in \{1, 2\}\}. \end{aligned} \tag{7}$$

where

$$\begin{cases} \mathbf{M}_{1,u,v,l,z}, \mathbf{M}'_{1,u,v,l,z} \in \mathbb{R}^{(d+6)\times(d+6)} \\ \mathbf{M}_{2,u,v,l,z}, \mathbf{M}'_{2,u,v,l,z} \in \mathbb{R}^{6\times 6} \\ \mathbf{M}_{3,u,v,l,z}, \mathbf{M}'_{3,u,v,l,z} \in \mathbb{R}^{(k_{\max}+3)\times(k_{\max}+3)}. \end{cases} \tag{8}$$

and $\mathbb{R}$ is the real domain. It is worth noting that each of $\{\mathbf{M}_{1,u,v,l,z}, \mathbf{M}_{2,u,v,l,z}, \mathbf{M}_{3,u,v,l,z}, \mathbf{M}'_{1,u,v,l,z}, \mathbf{M}'_{2,u,v,l,z}, \mathbf{M}'_{3,u,v,l,z}\}$ is related to 16 matrices, since $\{u, v, l, z\}$ can be 1 or 2, . As a result, sk contains 96 matrices.

• $\mathsf{FPE.Enc}(sk, \mathbf{P}, \mathbf{T})$: The encryption algorithm utilizes sk to encrypt $\mathbf{P}$ and $\mathbf{T}$ in Eq. (3) as follows.

Step 1: The encryptor randomly splits $\mathbf{P}$ into two vectors $\mathbf{P}_1$ and $\mathbf{P}_2$ by 1) randomly choosing a matrix $\mathbf{P}_1 \in \mathbb{R}^{d+3}$; and 2) computing $\mathbf{P}_2 = \mathbf{P} - \mathbf{P}_1$. That is, $\mathbf{P} = \mathbf{P}_1 + \mathbf{P}_2$. Similarly, it randomly splits $\mathbf{T}$ into two matrices $\mathbf{T}_1$ and $\mathbf{T}_2$ such that $\mathbf{T} = \mathbf{T}_1 + \mathbf{T}_2$, where $\{\mathbf{T}_1, \mathbf{T}_2\} \in \mathbb{R}^{3\times k_{\max}}$.

Step 2: The encryptor chooses a set of random numbers in the real domain, including i) random numbers $r_{\mathbf{P}} > 0$, $r_{\mathbf{T}} > 0$, $r'_{\mathbf{P}} > 0$, and $r'_{\mathbf{T}} > 0$; ii) random numbers $\{r_{\mathbf{P}_{u,v,l,z}}, r_{\mathbf{T}_{u,v,l,z}}, r_{\mathbf{P}'_{u,v,l,z}}, r_{\mathbf{T}'_{u,v,l,z}} | \{u, v, l, z\} \in \{1, 2\}\}$ satisfying $\frac{1}{2}r_{\mathbf{P}} > r_{\mathbf{P}_{u,v,l,z}} > 0$, $\frac{1}{2}r_{\mathbf{T}} > r_{\mathbf{T}_{u,v,l,z}} > 0$, $\frac{1}{2}r'_{\mathbf{P}} > r_{\mathbf{P}'_{u,v,l,z}} > 0$, and $\frac{1}{2}r'_{\mathbf{T}} > r_{\mathbf{T}'_{u,v,l,z}} > 0$ for

$\{u,v,l,z\} \in \{1,2\}$; and iii) random numbers $\{\alpha_{\mathbf{P}_{u,v,l,z}}, \alpha_{\mathbf{T}_{u,v,l,z}}, \alpha_{\mathbf{P}'_{u,v,l,z}}, \alpha_{\mathbf{T}'_{u,v,l,z}} | \{u,v,l,z\} \in \{1,2\}\}$ satisfying

$$\sum_{u=1}^{2}\sum_{v=1}^{2}\sum_{l=1}^{2}\sum_{z=1}^{2}(\alpha_{\mathbf{P}_{u,v,l,z}}\alpha_{\mathbf{T}_{u,v,l,z}} + \alpha_{\mathbf{P}'_{u,v,l,z}}\alpha_{\mathbf{T}'_{u,v,l,z}}) = 0. \tag{9}$$

Since $\{\alpha_{\mathbf{P}_{u,v,l,z}}, \alpha_{\mathbf{T}_{u,v,l,z}}, \alpha_{\mathbf{P}'_{u,v,l,z}}, \alpha_{\mathbf{T}'_{u,v,l,z}} | \{u,v,l,z\} \in \{1,2\}\}$ is a set with 64 random numbers, we can first choose 63 random numbers and the last one is computed based on Eq. (9). These random numbers are different for different $\mathbf{P}$'s and $\mathbf{T}$'s.

Step 3: The encryptor encrypts $\mathbf{P}$ into ciphertexts.

(1) Extend $\mathbf{P}$ to a group of $(d+6)$-dimensional vectors $\{\mathbf{P}_{u,v,l,z}, \mathbf{P}'_{u,v,l,z} | \{u,v,l,z\} \in \{1,2\}\}$ as

$$\begin{cases} \mathbf{P}_{u,v,l,z} = \begin{bmatrix} r_{\mathbf{P}} * \mathbf{P}_u & -r_{\mathbf{P}_{u,v,l,z}} & \alpha_{\mathbf{P}_{u,v,l,z}} & 1 \end{bmatrix} \\ \mathbf{P}'_{u,v,l,z} = \begin{bmatrix} r'_{\mathbf{P}} * \mathbf{P}_u & -r_{\mathbf{P}'_{u,v,l,z}} & \alpha_{\mathbf{P}'_{u,v,l,z}} & 1 \end{bmatrix}. \end{cases} \tag{10}$$

(2) Encrypt each $\mathbf{P}_{u,v,l,z}$ and $\mathbf{P}'_{u,v,l,z}$ into ciphertexts as

$$\begin{cases} \mathtt{C}_{\mathbf{P}_{u,v,l,z}} = \mathbf{P}_{u,v,l,z}\mathbf{M}_{1,u,v,l,z} \\ \mathtt{C}_{\mathbf{P}'_{u,v,l,z}} = \mathbf{P}'_{u,v,l,z}\mathbf{M}'_{1,u,v,l,z}. \end{cases} \tag{11}$$

Step 4: The encryptor encrypts $\mathbf{T}$ into ciphertexts.

(1) Extend $\mathbf{T}$ to a group of $6 \times (k_{\max}+3)$ matrices $\{\mathbf{T}_{u,v,l,z}, \mathbf{T}'_{u,v,l,z} | \{u,v,l,z\} \in \{1,2\}\}$ as

$$\begin{cases} \mathbf{T}_{u,v,l,z} = \begin{bmatrix} r_{\mathbf{T}} * \mathbf{T}_l & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & r_{\mathbf{T}_{u,v,l,z}} & 0 & 0 \\ \mathbf{O} & 0 & \alpha_{\mathbf{T}_{u,v,l,z}} & 0 \\ \mathbf{O} & 0 & 0 & 1 \end{bmatrix} \\ \mathbf{T}'_{u,v,l,z} = \begin{bmatrix} r'_{\mathbf{T}} * \mathbf{T}_l & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & r_{\mathbf{T}'_{u,v,l,z}} & 0 & 0 \\ \mathbf{O} & 0 & \alpha_{\mathbf{T}'_{u,v,l,z}} & 0 \\ \mathbf{O} & 0 & 0 & 1 \end{bmatrix}. \end{cases} \tag{12}$$

(2) Encrypt each $\mathbf{T}_{u,v,l,z}$ and $\mathbf{T}'_{u,v,l,z}$ into ciphertexts as

$$\begin{cases} \mathtt{C}_{\mathbf{T}_{u,v,l,z}} = \mathbf{M}_{2,u,v,l,z}^{-1}\mathbf{T}_{u,v,l,z}\mathbf{M}_{3,u,v,l,z}; \\ \mathtt{C}_{\mathbf{T}'_{u,v,l,z}} = \mathbf{M}'^{-1}_{2,u,v,l,z}\mathbf{T}'_{u,v,l,z}\mathbf{M}'_{3,u,v,l,z}. \end{cases} \tag{13}$$

Step 5: The encryption algorithm finally outputs the ciphertext of $\mathbf{P}$ and $\mathbf{T}$, i.e., $\mathtt{C}_{\mathbf{P},\mathbf{T}} = \{\mathtt{C}_{\mathbf{P}_{u,v,l,z}}, \mathtt{C}_{\mathbf{P}'_{u,v,l,z}}, \mathtt{C}_{\mathbf{T}_{u,v,l,z}}, \mathtt{C}_{\mathbf{T}'_{u,v,l,z}} | \{u,v,l,z\} \in \{1,2\}\}$.

• FPE.TrapdoorGen$(sk, \mathbf{Q}, \mathbf{e}_k)$: The trapdoor generation algorithm utilizes the secret key sk to generate trapdoors for $\mathbf{Q}$ and $\mathbf{e}_k$ in Eq. (3) as follows.

Step 1: The generator randomly splits $\mathbf{Q}$ into two matrices $\mathbf{Q}_1$ and $\mathbf{Q}_2$ such that $\mathbf{Q} = \mathbf{Q}_1 + \mathbf{Q}_2$, where $\{\mathbf{Q}_1, \mathbf{Q}_2\} \in \mathbb{R}^{(d+3)\times 3}$. Similarly, it randomly splits $\mathbf{e}_k$ into two vectors $\mathbf{e}_{k,1}$ and $\mathbf{e}_{k,2}$ such that $\mathbf{e}_k = \mathbf{e}_{k,1} + \mathbf{e}_{k,2}$, where $\{\mathbf{e}_{k,1}, \mathbf{e}_{k,2}\} \in \mathbb{R}^{k_{\max}}$.

Step 2: The generator chooses a set of random numbers in the real domain, including i) random numbers $r_{\mathbf{Q}} > 0$, $r_{\mathbf{e}_k} > 0$, $r'_{\mathbf{Q}} > 0$, and $r'_{\mathbf{e}_k} > 0$; ii) random numbers $\{r_{\mathbf{Q}_{u,v,l,z}}, r_{\mathbf{e}_{k,u,v,l,z}}, r_{\mathbf{Q}'_{u,v,l,z}}, r_{\mathbf{e}'_{k,u,v,l,z}} | \{u,v,l,z\} \in \{1,2\}\}$ such that $\frac{1}{2}r_{\mathbf{Q}} > r_{\mathbf{Q}_{u,v,l,z}} > 0$, $\frac{1}{2}r_{\mathbf{e}_k} > r_{\mathbf{e}_{k,u,v,l,z}} > 0$, $\frac{1}{2}r'_{\mathbf{Q}} > r_{\mathbf{Q}'_{u,v,l,z}} > 0$, and $\frac{1}{2}r'_{\mathbf{e}_k} > r_{\mathbf{e}'_{k,u,v,l,z}} > 0$ for $\{u,v,l,z\} \in \{1,2\}$; and iii) random numbers $\{\alpha_{\mathbf{Q}_{u,v,l,z}}, \alpha_{\mathbf{e}_{k,u,v,l,z}}, \alpha_{\mathbf{Q}'_{u,v,l,z}}, \alpha_{\mathbf{e}'_{k,u,v,l,z}} | \{u,v,l,z\} \in \{1,2\}\}$ satisfying $\sum_{u=1}^{2}\sum_{v=1}^{2}\sum_{l=1}^{2}\sum_{z=1}^{2}(\alpha_{\mathbf{Q}_{u,v,l,z}}\alpha_{\mathbf{e}_{k,u,v,l,z}} + \alpha_{\mathbf{Q}'_{u,v,l,z}}\alpha_{\mathbf{e}'_{k,u,v,l,z}}) = 0$, which are generated in a similar way to that of $\{\alpha_{\mathbf{P}_{u,v,l,z}}, \alpha_{\mathbf{T}_{u,v,l,z}}, \alpha_{\mathbf{P}'_{u,v,l,z}}, \alpha_{\mathbf{T}'_{u,v,l,z}} | \{u,v,l,z\} \in \{1,2\}\}$. These random numbers are different for different $\mathbf{Q}$'s and $\mathbf{e}_k$'s.

Step 3: The generator generates trapdoors for $\mathbf{Q}$.

(1) Extend $\mathbf{Q}$ to a group of $(d+6) \times 6$ matrices $\{\mathbf{Q}_{u,v,l,z}, \mathbf{Q}'_{u,v,l,z} | \{u,v,l,z\} \in \{1,2\}\}$ as

$$\begin{cases} \mathbf{Q}_{u,v,l,z} = \begin{bmatrix} r_{\mathbf{Q}} * \mathbf{Q}_v & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & r_{\mathbf{Q}_{u,v,l,z}} & 0 & 0 \\ \mathbf{O} & 0 & 1 & 0 \\ \mathbf{O} & 0 & 0 & \alpha_{\mathbf{Q}_{u,v,l,z}} \end{bmatrix} \\ \mathbf{Q}'_{u,v,l,z} = \begin{bmatrix} r'_{\mathbf{Q}} * \mathbf{Q}_v & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & r_{\mathbf{Q}'_{u,v,l,z}} & 0 & 0 \\ \mathbf{O} & 0 & 1 & 0 \\ \mathbf{O} & 0 & 0 & \alpha_{\mathbf{Q}'_{u,v,l,z}} \end{bmatrix}. \end{cases} \tag{14}$$

(2) Encrypt $\mathbf{Q}_{u,v,l,z}$ and $\mathbf{Q}'_{u,v,l,z}$ into trapdoors as

$$\begin{cases} \mathtt{TD}_{\mathbf{Q}_{u,v,l,z}} = \mathbf{M}_{1,u,v,l,z}^{-1}\mathbf{Q}_{u,v,l,z}\mathbf{M}_{2,u,v,l,z} \\ \mathtt{TD}_{\mathbf{Q}'_{u,v,l,z}} = \mathbf{M}'^{-1}_{1,u,v,l,z}\mathbf{Q}'_{u,v,l,z}\mathbf{M}'_{2,u,v,l,z}. \end{cases} \tag{15}$$

Step 4: The generator generates trapdoors for $\mathbf{e}_k$.

(1) Extend $\mathbf{e}_k$ to a group of $(k_{\max}+3)$-dimensional vectors $\{\mathbf{e}_{k,u,v,l,z}, \mathbf{e}'_{k,u,v,l,z} | \{u,v,l,z\} \in \{1,2\}\}$ as

$$\begin{cases} \mathbf{e}_{k,u,v,l,z} = \begin{bmatrix} r_{\mathbf{e}_k} * \mathbf{e}_{k,z} & r_{\mathbf{e}_{k,u,v,l,z}} & 1 & \alpha_{\mathbf{e}_{k,u,v,l,z}} \end{bmatrix} \\ \mathbf{e}'_{k,u,v,l,z} = \begin{bmatrix} r'_{\mathbf{e}_k} * \mathbf{e}_{k,z} & r_{\mathbf{e}'_{k,u,v,l,z}} & 1 & \alpha_{\mathbf{e}'_{k,u,v,l,z}} \end{bmatrix}. \end{cases} \tag{16}$$

(2) Encrypt each $\mathbf{e}_{k,u,v,l,z}$ and $\mathbf{e}'_{k,u,v,l,z}$ into trapdoors as

$$\begin{cases} \mathtt{TD}_{\mathbf{e}_{k,u,v,l,z}} = \mathbf{M}_{3,u,v,l,z}^{-1}\mathbf{e}_{k,u,v,l,z}^{T}; \\ \mathtt{TD}_{\mathbf{e}'_{k,u,v,l,z}} = \mathbf{M}'^{-1}_{3,u,v,l,z}\mathbf{e}'^{T}_{k,u,v,l,z}. \end{cases} \tag{17}$$

Step 5: The trapdoor generation algorithm outputs the trapdoor of $\mathbf{Q}$ and $\mathbf{e}_k$, i.e., $\mathtt{TD}_{\mathbf{Q},\mathbf{e}_k} = \{\mathtt{TD}_{\mathbf{Q}_{u,v,l,z}}, \mathtt{TD}_{\mathbf{Q}'_{u,v,l,z}}, \mathtt{TD}_{\mathbf{e}_{k,u,v,l,z}}, \mathtt{TD}_{\mathbf{e}'_{k,u,v,l,z}} | \{u,v,l,z\} \in \{1,2\}\}$.

• FPE.FilterQuery$(\mathtt{C}_{\mathbf{P},\mathbf{T}}, \mathtt{TD}_{\mathbf{Q},\mathbf{e}_k})$: On input the ciphertext $\mathtt{C}_{\mathbf{P},\mathbf{T}}$ and the trapdoor $\mathtt{TD}_{\mathbf{Q},\mathbf{e}_k}$, the evaluator first calculates

$$s = \sum_{u=1}^{2}\sum_{v=1}^{2}\sum_{l=1}^{2}\sum_{z=1}^{2}(\mathtt{C}_{\mathbf{P}_{u,v,l,z}}\mathtt{TD}_{\mathbf{Q}_{u,v,l,z}}\mathtt{C}_{\mathbf{T}_{u,v,l,z}}\mathtt{TD}_{\mathbf{e}_{k,u,v,l,z}} + \mathtt{C}_{\mathbf{P}'_{u,v,l,z}}\mathtt{TD}_{\mathbf{Q}'_{u,v,l,z}}\mathtt{C}_{\mathbf{T}'_{u,v,l,z}}\mathtt{TD}_{\mathbf{e}'_{k,u,v,l,z}}). \tag{18}$$

If $s > 0$, the evaluator returns 1 to demonstrate that "$\mathbf{PQTe}_k^T > 0$" and returns 0 to demonstrate that "$\mathbf{PQTe}_k^T \leq 0$" otherwise.

Theorem 2 *The FPE scheme is correct.*

Proof. The FPE scheme is correct *iff* $s > 0 \Leftrightarrow \mathbf{PQTe}_k^T > 0$. First, we have

$$s > 0 \Leftrightarrow \sum_{u=1}^{2}\sum_{v=1}^{2}\sum_{l=1}^{2}\sum_{z=1}^{2}(\mathtt{C}_{\mathbf{P}_{u,v,l,z}}\mathtt{TD}_{\mathbf{Q}_{u,v,l,z}}\mathtt{C}_{\mathbf{T}_{u,v,l,z}}\mathtt{TD}_{\mathbf{e}_{k,u,v,l,z}} + \mathtt{C}_{\mathbf{P}'_{u,v,l,z}}\mathtt{TD}_{\mathbf{Q}'_{u,v,l,z}}\mathtt{C}_{\mathbf{T}'_{u,v,l,z}}\mathtt{TD}_{\mathbf{e}'_{k,u,v,l,z}}) > 0. \tag{19}$$

Meanwhile, we have $\mathtt{C}_{\mathbf{P}_{u,v,l,z}}\mathtt{TD}_{\mathbf{Q}_{u,v,l,z}}\mathtt{C}_{\mathbf{T}_{u,v,l,z}}\mathtt{TD}_{\mathbf{e}_{k,u,v,l,z}} = r_{\mathbf{P}} * r_{\mathbf{Q}} * r_{\mathbf{T}} * r_{\mathbf{e}_k} * \mathbf{P}_u\mathbf{Q}_v\mathbf{T}_l\mathbf{e}_{k,z}^T + (\alpha_{\mathbf{P}_{u,v,l,z}} * \alpha_{\mathbf{T}_{u,v,l,z}} + \alpha_{\mathbf{Q}_{u,v,l,z}} * \alpha_{\mathbf{e}_{k,u,v,l,z}}) - r_{\mathbf{P}_{u,v,l,z}} * r_{\mathbf{T}_{u,v,l,z}} * r_{\mathbf{Q}_{u,v,l,z}} * r_{\mathbf{e}_{k,u,v,l,z}}$ and $\mathtt{C}_{\mathbf{P}'_{u,v,l,z}}\mathtt{TD}_{\mathbf{Q}'_{u,v,l,z}}\mathtt{C}_{\mathbf{T}'_{u,v,l,z}}\mathtt{TD}_{\mathbf{e}'_{k,u,v,l,z}} = r'_{\mathbf{P}} * r'_{\mathbf{Q}} * r'_{\mathbf{T}} * r'_{\mathbf{e}_k} * \mathbf{P}_u\mathbf{Q}_v\mathbf{T}_l\mathbf{e}_{k,z}^T + (\alpha_{\mathbf{P}'_{u,v,l,z}} * \alpha_{\mathbf{T}'_{u,v,l,z}} + \alpha_{\mathbf{Q}'_{u,v,l,z}} * \alpha_{\mathbf{e}'_{k,u,v,l,z}}) - r_{\mathbf{P}'_{u,v,l,z}} * r_{\mathbf{T}'_{u,v,l,z}} * r_{\mathbf{Q}'_{u,v,l,z}} * r_{\mathbf{e}'_{k,u,v,l,z}}$.

Based on the FPE scheme, we further have $\sum_{u=1}^2\sum_{v=1}^2\sum_{l=1}^2\sum_{z=1}^2(\alpha_{\mathbf{P}_{u,v,l,z}} * \alpha_{\mathbf{T}_{u,v,l,z}} + \alpha_{\mathbf{P}'_{u,v,l,z}} * \alpha_{\mathbf{T}'_{u,v,l,z}}) = 0$ and $\sum_{u=1}^2\sum_{v=1}^2\sum_{l=1}^2\sum_{z=1}^2(\alpha_{\mathbf{Q}_{u,v,l,z}} * \alpha_{\mathbf{e}_{k,u,v,l,z}} + \alpha_{\mathbf{Q}'_{u,v,l,z}} * \alpha_{\mathbf{e}'_{k,u,v,l,z}}) = 0$. Then, if we let

$$\begin{cases} r_{\mathbf{P},\mathbf{Q},\mathbf{T},\mathbf{e}_k} = r_{\mathbf{P}} * r_{\mathbf{Q}} * r_{\mathbf{T}} * r_{\mathbf{e}_k} \\ r'_{\mathbf{P},\mathbf{Q},\mathbf{T},\mathbf{e}_k} = r'_{\mathbf{P}} * r'_{\mathbf{Q}} * r'_{\mathbf{T}} * r'_{\mathbf{e}_k} \\ r_{\mathbf{P},\mathbf{Q},\mathbf{T},\mathbf{e}_k,u,v,l,z} = r_{\mathbf{P}_{u,v,l,z}} * r_{\mathbf{T}_{u,v,l,z}} * r_{\mathbf{Q}_{u,v,l,z}} * r_{\mathbf{e}_{k,u,v,l,z}} \\ r'_{\mathbf{P},\mathbf{Q},\mathbf{T},\mathbf{e}_k,u,v,l,z} = r_{\mathbf{P}'_{u,v,l,z}} * r_{\mathbf{T}'_{u,v,l,z}} * r_{\mathbf{Q}'_{u,v,l,z}} * r_{\mathbf{e}'_{k,u,v,l,z}}, \end{cases} \tag{20}$$

we can infer that

$$s = (r_{\mathbf{P},\mathbf{Q},\mathbf{T},\mathbf{e}_k} + r'_{\mathbf{P},\mathbf{Q},\mathbf{T},\mathbf{e}_k})\sum_{u=1}^2\sum_{v=1}^2\sum_{l=1}^2\sum_{z=1}^2\mathbf{P}_u\mathbf{Q}_v\mathbf{T}_l\mathbf{e}_{k,z}^T - \sum_{u=1}^2\sum_{v=1}^2\sum_{l=1}^2\sum_{z=1}^2(r_{\mathbf{P},\mathbf{Q},\mathbf{T},\mathbf{e}_{k,u,v,l,z}} + r'_{\mathbf{P},\mathbf{Q},\mathbf{T},\mathbf{e}_{k,u,v,l,z}}). \tag{21}$$

Based on the FPE scheme, we further have i) $\frac{1}{2}r_{\mathbf{P}} > r_{\mathbf{P}_{u,v,l,z}} > 0$, $\frac{1}{2}r_{\mathbf{T}} > r_{\mathbf{T}_{u,v,l,z}} > 0$, $\frac{1}{2}r'_{\mathbf{P}} > r_{\mathbf{P}'_{u,v,l,z}} > 0$, and $\frac{1}{2}r'_{\mathbf{T}} > r_{\mathbf{T}'_{u,v,l,z}} > 0$ for $\{u,v,l,z\} \in \{1,2\}$; and ii) $\frac{1}{2}r_{\mathbf{Q}} > r_{\mathbf{Q}_{u,v,l,z}} > 0$, $\frac{1}{2}r_{\mathbf{e}_k} > r_{\mathbf{e}_{k,u,v,l,z}} > 0$, $\frac{1}{2}r'_{\mathbf{Q}} > r_{\mathbf{Q}'_{u,v,l,z}} > 0$, and $\frac{1}{2}r'_{\mathbf{e}_k} > r_{\mathbf{e}'_{k,u,v,l,z}} > 0$ for $\{u,v,l,z\} \in \{1,2\}$. We have

$$\begin{cases} \frac{1}{16}r_{\mathbf{P},\mathbf{Q},\mathbf{T},\mathbf{e}_k} > r_{\mathbf{P},\mathbf{Q},\mathbf{T},\mathbf{e}_k,u,v,l,z} \\ \frac{1}{16}r'_{\mathbf{P},\mathbf{Q},\mathbf{T},\mathbf{e}_k} > r'_{\mathbf{P},\mathbf{Q},\mathbf{T},\mathbf{e}_k,u,v,l,z} \end{cases}$$
$$\Leftrightarrow \begin{cases} r_{\mathbf{P},\mathbf{Q},\mathbf{T},\mathbf{e}_k} > \sum_{u=1}^2\sum_{v=1}^2\sum_{l=1}^2\sum_{z=1}^2 r_{\mathbf{P},\mathbf{Q},\mathbf{T},\mathbf{e}_k,u,v,l,z} \\ r'_{\mathbf{P},\mathbf{Q},\mathbf{T},\mathbf{e}_k} > \sum_{u=1}^2\sum_{v=1}^2\sum_{l=1}^2\sum_{z=1}^2 r'_{\mathbf{P},\mathbf{Q},\mathbf{T},\mathbf{e}_k,u,v,l,z}. \end{cases} \tag{22}$$

Furthermore, we have

$$r_{\mathbf{P},\mathbf{Q},\mathbf{T},\mathbf{e}_k} + r'_{\mathbf{P},\mathbf{Q},\mathbf{T},\mathbf{e}_k} > \sum_{u=1}^2\sum_{v=1}^2\sum_{l=1}^2\sum_{z=1}^2(r_{\mathbf{P},\mathbf{Q},\mathbf{T},\mathbf{e}_k,u,v,l,z} + r'_{\mathbf{P},\mathbf{Q},\mathbf{T},\mathbf{e}_k,u,v,l,z}). \tag{23}$$

In addition, since $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{T}$, and $\mathbf{e}_k$ are integer vectors or matrices, according to Eq. (21) and Eq. (23), we have

$$\begin{aligned} s > 0 &\Leftrightarrow \sum_{u=1}^2\sum_{v=1}^2\sum_{l=1}^2\sum_{z=1}^2\mathbf{P}_u\mathbf{Q}_v\mathbf{T}_l\mathbf{e}_{k,z}^T > 0 \\ &\Leftrightarrow (\mathbf{P}_1+\mathbf{P}_2)(\mathbf{Q}_1+\mathbf{Q}_2)(\mathbf{T}_1+\mathbf{T}_2)(\mathbf{e}_{k,1}^T+\mathbf{e}_{k,2}^T) > 0 \\ &\Leftrightarrow \mathbf{P}\mathbf{Q}\mathbf{T}\mathbf{e}_k^T > 0. \end{aligned} \tag{24}$$

Thus, the FPE scheme is correct. □

Remark. The intuition of splitting each of $\{\mathbf{P},\mathbf{Q},\mathbf{T},\mathbf{e}_k\}$ into two parts is to enable our scheme to resist against known-plaintext attacks. We take the encryption of $\mathbf{e}_k$ as an example to show that if $\mathbf{e}_k$ is not split, our scheme will suffer from known-plaintext attacks. Specifically, if $\mathbf{e}_k$ is not split, we will encrypt it as follows.

(1) Extend $\mathbf{e}_k$ to a $(k_{\max}+3)$-dimensional vector as

$$\begin{cases} \mathbf{e}_k = \begin{bmatrix} r_{\mathbf{e}_k,1} * \mathbf{e}_k & r_{\mathbf{e}_k,2} & 1 & \alpha_{\mathbf{e}_k} \end{bmatrix} \\ \mathbf{e}'_k = \begin{bmatrix} r'_{\mathbf{e}_k,1} * \mathbf{e}_k & r'_{\mathbf{e}_k,2} & 1 & \alpha_{\mathbf{e}'_k} \end{bmatrix}. \end{cases} \tag{25}$$

(2) Encrypt each $\mathbf{e}_k$ and $\mathbf{e}'_k$ into trapdoors as

$$\mathtt{TD}_{\mathbf{e}_k} = \mathbf{M}_3^{-1}\mathbf{e}_k^T \text{ and } \mathtt{TD}_{\mathbf{e}'_k} = \mathbf{M'}_3^{-1}\mathbf{e'}_k^T. \tag{26}$$

We regard $\mathbf{M}_3^{-1}$ to be a matrix with $(k_{\max+3})$ columns as

$$\mathbf{M}_3^{-1} = [\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_{k_{\max+3}}].$$

Then, $\mathtt{TD}_{\mathbf{e}_k} = \mathbf{M}_3^{-1}\mathbf{e}_k^T$ can be represented as $\mathbf{M}_3^{-1}\mathbf{e}_k^T = r_{\mathbf{e}_k,1} * \mathbf{m}_k + r_{\mathbf{e}_k,2} * \mathbf{m}_{k_{\max+1}} + \mathbf{m}_{k_{\max+2}} + \alpha_{\mathbf{e}_k} * \mathbf{m}_{k_{\max+3}}$. When $k = 1$, we have $\mathtt{TD}_{\mathbf{e}_k} = \mathbf{M}_3^{-1}\mathbf{e}_1^T = r_{\mathbf{e}_1,1} * \mathbf{m}_1 + r_{\mathbf{e}_1,2} * \mathbf{m}_{k_{\max+1}} + \mathbf{m}_{k_{\max+2}} + \alpha_{\mathbf{e}_1} * \mathbf{m}_{k_{\max+3}}$. That is, $\mathbf{M}_3^{-1}\mathbf{e}_1^T$ is a linear combination of $\{\mathbf{m}_1, \mathbf{m}_{k_{\max+1}}, \mathbf{m}_{k_{\max+2}}, \mathbf{m}_{k_{\max+3}}\}$. When an adversary has many pairs of $\{\mathbf{e}_1^{(i)}, \mathtt{TD}_{\mathbf{e}_1}^{(i)}\}_{i=1}^n$ with $k = 1$. All of $\{\mathtt{TD}_{\mathbf{e}_1}^{(i)}\}_{i=1}^n$ are linear combination of $\{\mathbf{m}_1, \mathbf{m}_{k_{\max+1}}, \mathbf{m}_{k_{\max+2}}, \mathbf{m}_{k_{\max+3}}\}$. Then, the rank of $\{\mathtt{TD}_{\mathbf{e}_1}^{(i)}\}_{i=1}^n$ will be the rank of $\{\mathbf{m}_1, \mathbf{m}_{k_{\max+1}}, \mathbf{m}_{k_{\max+2}}, \mathbf{m}_{k_{\max+3}}\}$, i.e., $\mathsf{rank}(\{\mathtt{TD}_{\mathbf{e}_1}^{(i)}\}_{i=1}^n) = 4$. In this case, given a new ciphertext $\mathtt{TD}_{\mathbf{e}_k}$, we can test whether $\mathsf{rank}(\{\mathtt{TD}_{\mathbf{e}_1}^{(i)}\}_{i=1}^n \cap \mathtt{TD}_{\mathbf{e}_k}) \overset{?}{=} 4$. If yes, $\mathtt{TD}_{\mathbf{e}_k}$ will also be a linear combination of $\{\mathbf{m}_1, \mathbf{m}_{k_{\max+1}}, \mathbf{m}_{k_{\max+2}}, \mathbf{m}_{k_{\max+3}}\}$. Then, $\mathbf{e}_k$ will be $\mathbf{e}_1 = (1,0,0,\cdots,0)$, which violates the privacy of $\mathbf{e}_k$.

Differently, in our scheme, we split $\mathbf{e}_k$ to two random vectors such that any $\mathtt{TD}_{\mathbf{e}_k}$ is a random linear combination of $\{\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_{k_{\max}+3}\}$. Thus, $\mathsf{rank}(\{\mathtt{TD}_{\mathbf{e}_1}^{(i)}\}_{i=1}^n) = k_{\max+3}$ for any $n \geq k_{\max+3}$, which makes our scheme free of the known-plaintext attacks.

5.4 RPE Scheme

On input a reverse kNN query $(\mathbf{q}, k)$ and a leaf node with $(\mathbf{x}_i, \mathtt{L}_{\mathbf{x}_i} = [\tau_{\mathbf{x}_i,1}, \tau_{\mathbf{x}_i,2}, \cdots, \tau_{\mathbf{x}_i,k_{\max}}])$, the RPE scheme is designed to privately determine whether the refinement inequality holds or not, i.e., whether $d(\mathbf{x}_i, \mathbf{q}) \leq \tau_{\mathbf{x}_i,k}$ or not. First, we have

$$\begin{aligned} d(\mathbf{x}_i, \mathbf{q}) \leq \tau_{\mathbf{x}_i,k} &\Leftrightarrow d(\mathbf{x}_i, \mathbf{q})^2 - \tau_{\mathbf{x}_i,k}^2 \leq 0 \\ &\Leftrightarrow ||\mathbf{x}_i||^2 - 2\mathbf{x}_i \circ \mathbf{q} + ||\mathbf{q}||^2 - \tau_{\mathbf{x}_i,k}^2 \leq 0. \end{aligned} \tag{27}$$

If let

$$\begin{cases} \widetilde{\mathbf{X}}_i = \begin{bmatrix} ||\mathbf{x}_i||^2 & \mathbf{x}_i & 1 \end{bmatrix} \\ \widetilde{\mathbf{Q}} = \begin{bmatrix} 1 & 0 \\ -2\mathbf{q} & 0 \\ ||\mathbf{q}||^2 & -1 \end{bmatrix} \\ \widetilde{\mathbf{T}} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \tau_{\mathbf{x}_i,1}^2 & \tau_{\mathbf{x}_i,2}^2 & \cdots & \tau_{\mathbf{x}_i,k_{\max}}^2 \end{bmatrix} \\ \widetilde{\mathbf{e}}_k = \begin{bmatrix} 0 & \cdots & 1 & \cdots & 0 \end{bmatrix} \end{cases} \tag{28}$$

we can infer that

$$\widetilde{\mathbf{X}}_i\widetilde{\mathbf{Q}}\widetilde{\mathbf{T}}\widetilde{\mathbf{e}}_k^T = ||\mathbf{x}_i||^2 - 2\mathbf{x}_i \circ \mathbf{q} + ||\mathbf{q}||^2 - \tau_{\mathbf{x}_i,k}^2. \tag{29}$$

By further combining the equation with Eq. (27), we have

$$d(\mathbf{x}_i, \mathbf{q}) \leq \tau_{\mathbf{x}_i,k} \Leftrightarrow \widetilde{\mathbf{X}}_i\widetilde{\mathbf{Q}}\widetilde{\mathbf{T}}\widetilde{\mathbf{e}}_k^T \leq 0. \tag{30}$$

Thus, privately determining "$d(\mathbf{x}_i, \mathbf{q}) \leq \tau_{\mathbf{x}_i,k}$" is equivalent to privately determine that of "$\widetilde{\mathbf{X}}_i \widetilde{\mathbf{Q}} \widetilde{\mathbf{T}} \widetilde{\mathbf{e}}_k^T \leq 0$". Considering that $\widetilde{\mathbf{X}}_i$, $\widetilde{\mathbf{Q}}$, $\widetilde{\mathbf{T}}$, $\widetilde{\mathbf{e}}_k$ are either vectors or matrices, we utilize matrix encryption to devise our RPE scheme for protecting the privacy of the determination "$\widetilde{\mathbf{X}}_i \widetilde{\mathbf{Q}} \widetilde{\mathbf{T}} \widetilde{\mathbf{e}}_k^T \leq 0$". The key idea is to split each of $\widetilde{\mathbf{X}}_i$, $\widetilde{\mathbf{Q}}$, $\widetilde{\mathbf{T}}$, $\widetilde{\mathbf{e}}_k$ into two parts $\{\widetilde{\mathbf{X}}_{i,1}, \widetilde{\mathbf{X}}_{i,2}\}$, $\{\widetilde{\mathbf{Q}}_1, \widetilde{\mathbf{Q}}_2\}$, $\{\widetilde{\mathbf{T}}_1, \widetilde{\mathbf{T}}_2\}$, $\{\widetilde{\mathbf{e}}_{k,1}, \widetilde{\mathbf{e}}_{k,2}\}$ such that

$$\begin{cases} \widetilde{\mathbf{X}}_{i,1}, \widetilde{\mathbf{X}}_{i,2} = \widetilde{\mathbf{X}}_i; & \widetilde{\mathbf{Q}}_1 + \widetilde{\mathbf{Q}}_2 = \widetilde{\mathbf{Q}}; \\ \widetilde{\mathbf{T}}_1 + \widetilde{\mathbf{T}}_2 = \widetilde{\mathbf{T}}; & \widetilde{\mathbf{e}}_{k,1} + \widetilde{\mathbf{e}}_{k,2} = \widetilde{\mathbf{e}}_k. \end{cases} \tag{31}$$

The splitting method is the same as that of FPE scheme. In this case, we have

$$\widetilde{\mathbf{X}}_i \widetilde{\mathbf{Q}} \widetilde{\mathbf{T}} \widetilde{\mathbf{e}}_k^T = \sum_{u=1}^{2} \sum_{v=1}^{2} \sum_{l=1}^{2} \sum_{z=1}^{2} \widetilde{\mathbf{X}}_{i,u} \widetilde{\mathbf{Q}}_v \widetilde{\mathbf{T}}_l \widetilde{\mathbf{e}}_{k,z}^T. \tag{32}$$

Then, we use matrices to encrypt $\{\widetilde{\mathbf{X}}_{i,u}, \widetilde{\mathbf{Q}}_v, \widetilde{\mathbf{T}}_l, \widetilde{\mathbf{e}}_{k,z} | \{u, v, l, z\} \in \{1, 2\}\}$. Since $\{\widetilde{\mathbf{X}}_{i,u}, \widetilde{\mathbf{T}}_l\}$ and $\{\widetilde{\mathbf{Q}}_v, \widetilde{\mathbf{e}}_{k,z}\}$ are respectively held by the data owner and query user, they will be independently encrypted into a ciphertext $\mathtt{C}_{\widetilde{\mathbf{X}}_i, \widetilde{\mathbf{T}}}$ and a trapdoor $\mathtt{TD}_{\widetilde{\mathbf{Q}}, \widetilde{\mathbf{e}}_k}$ by the data owner and the query user using RPE.Enc$(\widetilde{sk}, \widetilde{\mathbf{X}}_i, \widetilde{\mathbf{T}})$ and RPE.TrapdoorGen$(\widetilde{sk}, \widetilde{\mathbf{Q}}, \widetilde{\mathbf{e}}_k)$ algorithms below. Based on $\mathtt{C}_{\mathbf{P},\mathbf{T}}$ and $\mathtt{TD}_{\mathbf{Q},\mathbf{e}_k}$, we can do the refinement using RPE.RefineQuery$(\mathtt{C}_{\widetilde{\mathbf{X}}_i, \widetilde{\mathbf{T}}}, \mathtt{TD}_{\widetilde{\mathbf{Q}}, \widetilde{\mathbf{e}}_k})$ algorithm below. Specifically, the RPE scheme is made up of four algorithms, i.e., $\Pi_{\mathtt{RPE}}$ = (RPE.Setup, RPE.Enc, RPE.TrapdoorGen, RPE.RefineQuery).

• RPE.Setup(d): Given the dimensions of data, i.e., d, the setup algorithm first randomly selects a group of invertible matrices as the secret key, i.e.,

$$\begin{aligned} \widetilde{sk} = \{&\widetilde{\mathbf{M}}_{1,u,v,l,z}, \widetilde{\mathbf{M}}_{2,u,v,l,z}, \widetilde{\mathbf{M}}_{3,u,v,l,z}, \widetilde{\mathbf{M}'}_{1,u,v,l,z}, \\ &\widetilde{\mathbf{M}'}_{2,u,v,l,z}, \widetilde{\mathbf{M}'}_{3,u,v,l,z} | \{u, v, l, z\} \in \{1, 2\}\}, \end{aligned} \tag{33}$$

where

$$\begin{cases} \{\widetilde{\mathbf{M}}_{1,u,v,l,z}, \widetilde{\mathbf{M}'}_{1,u,v,l,z}\} \in \mathbb{R}^{(d+5)\times(d+5)} \\ \{\widetilde{\mathbf{M}}_{2,u,v,l,z}, \widetilde{\mathbf{M}'}_{2,u,v,l,z}\} \in \mathbb{R}^{5\times 5} \\ \{\widetilde{\mathbf{M}}_{3,u,v,l,z}, \widetilde{\mathbf{M}'}_{3,u,v,l,z}\} \in \mathbb{R}^{(k_{\max}+3)\times(k_{\max}+3)}. \end{cases} \tag{34}$$

• RPE.Enc$(\widetilde{sk}, \widetilde{\mathbf{X}}_i, \widetilde{\mathbf{T}})$: The encryption algorithm utilizes $\widetilde{sk}$ to encrypt $\widetilde{\mathbf{X}}_i$ and $\widetilde{\mathbf{T}}$ in Eq. (28) as follows.

Step 1: The encryptor randomly splits $\widetilde{\mathbf{X}}_i$ into two vectors $\widetilde{\mathbf{X}}_{i,1}$ and $\widetilde{\mathbf{X}}_{i,2}$ such that $\widetilde{\mathbf{X}}_i = \widetilde{\mathbf{X}}_{i,1} + \widetilde{\mathbf{X}}_{i,2}$, where $\{\widetilde{\mathbf{X}}_{i,1}, \widetilde{\mathbf{X}}_{i,2}\} \in \mathbb{R}^{d+2}$. Similarly, it randomly splits $\widetilde{\mathbf{T}}$ into two matrices $\widetilde{\mathbf{T}}_1$ and $\widetilde{\mathbf{T}}_2$ such that $\widetilde{\mathbf{T}} = \widetilde{\mathbf{T}}_1 + \widetilde{\mathbf{T}}_2$, where $\{\widetilde{\mathbf{T}}_1, \widetilde{\mathbf{T}}_2\} \in \mathbb{R}^{2\times k_{\max}}$.

Step 2: The encryptor chooses a set of random numbers in the real domain, including i) random numbers $r_{\widetilde{\mathbf{X}}_i} > 0$, $r_{\widetilde{\mathbf{T}}} > 0$, $r'_{\widetilde{\mathbf{X}}_i} > 0$, and $r'_{\widetilde{\mathbf{T}}} > 0$; ii) random numbers $\{r_{\widetilde{\mathbf{X}}_{i,u,v,l,z}}, r_{\widetilde{\mathbf{T}}_{u,v,l,z}}, r_{\widetilde{\mathbf{X}}'_{i,u,v,l,z}}, r_{\widetilde{\mathbf{T}}'_{u,v,l,z}} | \{u, v, l, z\} \in \{1, 2\}\}$ satisfying $\frac{1}{2} r_{\widetilde{\mathbf{X}}_i} > r_{\widetilde{\mathbf{X}}_{i,u,v,l,z}} > 0$, $\frac{1}{2} r_{\widetilde{\mathbf{X}}_i} > r_{\widetilde{\mathbf{T}}_{u,v,l,z}} > 0$, $\frac{1}{2} r'_{\widetilde{\mathbf{X}}_i} > r_{\widetilde{\mathbf{X}}'_{i,u,v,l,z}} > 0$, and $\frac{1}{2} r'_{\widetilde{\mathbf{X}}_i} > r_{\widetilde{\mathbf{T}}'_{u,v,l,z}} > 0$ for $\{u, v, l, z\} \in \{1, 2\}$; and iii) random numbers $\{\alpha_{\widetilde{\mathbf{X}}_{i,u,v,l,z}}, \alpha_{\widetilde{\mathbf{T}}_{u,v,l,z}}, \alpha_{\widetilde{\mathbf{X}}'_{i,u,v,l,z}}, \alpha_{\widetilde{\mathbf{T}}'_{u,v,l,z}} | \{u, v, l, z\} \in \{1, 2\}\}$ satisfying $\sum_{u=1}^{2} \sum_{v=1}^{2} \sum_{l=1}^{2} \sum_{z=1}^{2} (\alpha_{\widetilde{\mathbf{X}}_{i,u,v,l,z}} \alpha_{\widetilde{\mathbf{T}}_{u,v,l,z}} + \alpha_{\widetilde{\mathbf{X}}'_{i,u,v,l,z}} \alpha_{\widetilde{\mathbf{T}}'_{u,v,l,z}}) = 0$, which are generated in a similar way to that of $\{\alpha_{\mathbf{P}_{u,v,l,z}}, \alpha_{\mathbf{T}_{u,v,l,z}}, \alpha_{\mathbf{P}'_{u,v,l,z}}, \alpha_{\mathbf{T}'_{u,v,l,z}} | \{u, v, l, z\} \in \{1, 2\}\}$. These random numbers are different for different $\widetilde{\mathbf{X}}_i$'s and $\widetilde{\mathbf{T}}$'s.

Step 3: The encryptor encrypts $\widetilde{\mathbf{X}}_i$ into ciphertexts.

(1) Extend $\widetilde{\mathbf{X}}_i$ to a group of $(d+5)$-dimensional vectors $\{\mathbf{P}_{u,v,l,z}, \mathbf{P}'_{u,v,l,z} | \{u, v, l, z\} \in \{1, 2\}\}$ as

$$\begin{cases} \widetilde{\mathbf{X}}_{i,u,v,l,z} = \begin{bmatrix} r_{\widetilde{\mathbf{X}}_i} * \widetilde{\mathbf{X}}_{i,u} & -r_{\widetilde{\mathbf{X}}_{i,u,v,l,z}} & \alpha_{\widetilde{\mathbf{X}}_{i,u,v,l,z}} & 1 \end{bmatrix} \\ \widetilde{\mathbf{X}}'_{i,u,v,l,z} = \begin{bmatrix} r'_{\widetilde{\mathbf{X}}_i} * \widetilde{\mathbf{X}}_{i,u} & -r'_{\widetilde{\mathbf{X}}_{i,u,v,l,z}} & \alpha'_{\widetilde{\mathbf{X}}_{i,u,v,l,z}} & 1 \end{bmatrix}. \end{cases} \tag{35}$$

(2) Encrypt each $\widetilde{\mathbf{X}}_{i,u,v,l,z}$ and $\widetilde{\mathbf{X}}'_{i,u,v,l,z}$ as

$$\begin{cases} \mathtt{C}_{\widetilde{\mathbf{X}}_{i,u,v,l,z}} = \widetilde{\mathbf{X}}_{i,u,v,l,z} \widetilde{\mathbf{M}}_{1,u,v,l,z}; \\ \mathtt{C}_{\widetilde{\mathbf{X}}'_{i,u,v,l,z}} = \widetilde{\mathbf{X}}'_{i,u,v,l,z} \widetilde{\mathbf{M}'}_{1,u,v,l,z}. \end{cases} \tag{36}$$

Step 4: The encryptor encrypts $\widetilde{\mathbf{T}}$ into ciphertexts.

(1) Extend $\widetilde{\mathbf{T}}$ to a group of $5 \times (k_{\max} + 3)$ matrices $\{\widetilde{\mathbf{T}}_{u,v,l,z}, \widetilde{\mathbf{T}}'_{u,v,l,z} | \{u, v, l, z\} \in \{1, 2\}\}$ as

$$\begin{cases} \widetilde{\mathbf{T}}_{u,v,l,z} = \begin{bmatrix} r_{\widetilde{\mathbf{T}}} * \widetilde{\mathbf{T}}_l & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & r_{\widetilde{\mathbf{T}}_{u,v,l,z}} & 0 & 0 \\ \mathbf{O} & 0 & \alpha_{\widetilde{\mathbf{T}}_{u,v,l,z}} & 0 \\ \mathbf{O} & 0 & 0 & 1 \end{bmatrix} \\ \widetilde{\mathbf{T}}'_{u,v,l,z} = \begin{bmatrix} r'_{\widetilde{\mathbf{T}}} * \widetilde{\mathbf{T}}_l & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & r_{\widetilde{\mathbf{T}}'_{u,v,l,z}} & 0 & 0 \\ \mathbf{O} & 0 & \alpha_{\widetilde{\mathbf{T}}'_{u,v,l,z}} & 0 \\ \mathbf{O} & 0 & 0 & 1 \end{bmatrix}. \end{cases} \tag{37}$$

(2) Encrypt each $\widetilde{\mathbf{T}}_{u,v,l,z}$ and $\widetilde{\mathbf{T}}'_{u,v,l,z}$ into ciphertexts as

$$\begin{cases} \mathtt{C}_{\widetilde{\mathbf{T}}_{u,v,l,z}} = \widetilde{\mathbf{M}}_{2,u,v,l,z}^{-1} \widetilde{\mathbf{T}}_{u,v,l,z} \widetilde{\mathbf{M}}_{3,u,v,l,z}; \\ \mathtt{C}_{\widetilde{\mathbf{T}}'_{u,v,l,z}} = \widetilde{\mathbf{M}'}_{2,u,v,l,z}^{-1} \widetilde{\mathbf{T}}'_{u,v,l,z} \widetilde{\mathbf{M}'}_{3,u,v,l,z}. \end{cases} \tag{38}$$

Step 5: The encryption algorithm finally outputs the ciphertext of $\widetilde{\mathbf{X}}_i$ and $\widetilde{\mathbf{T}}$, i.e., $\mathtt{C}_{\widetilde{\mathbf{X}}_i, \widetilde{\mathbf{T}}} = \{\mathtt{C}_{\widetilde{\mathbf{X}}_{i,u,v,l,z}}, \mathtt{C}_{\widetilde{\mathbf{X}}'_{i,u,v,l,z}}, \mathtt{C}_{\widetilde{\mathbf{T}}_{u,v,l,z}}, \mathtt{C}_{\widetilde{\mathbf{T}}'_{u,v,l,z}} | \{u, v, l, z\} \in \{1, 2\}\}$.

• RPE.TrapdoorGen$(\widetilde{sk}, \widetilde{\mathbf{Q}}, \widetilde{\mathbf{e}}_k)$: The trapdoor generation algorithm utilizes the secret key $\widetilde{sk}$ to generate trapdoors for $\widetilde{\mathbf{Q}}$ and $\widetilde{\mathbf{e}}_k$ in Eq. (3) as follows.

Step 1: The generator randomly splits $\widetilde{\mathbf{Q}}$ into two matrices $\widetilde{\mathbf{Q}}_1$ and $\widetilde{\mathbf{Q}}_2$ such that $\widetilde{\mathbf{Q}} = \widetilde{\mathbf{Q}}_1 + \widetilde{\mathbf{Q}}_2$, where $\{\widetilde{\mathbf{Q}}_1, \widetilde{\mathbf{Q}}_2\} \in \mathbb{R}^{(d+2)\times 2}$ Similarly, it randomly splits $\widetilde{\mathbf{e}}_k$ into two vectors $\widetilde{\mathbf{e}}_{k,1}$ and $\widetilde{\mathbf{e}}_{k,2}$ such that $\widetilde{\mathbf{e}}_k = \widetilde{\mathbf{e}}_{k,1} + \widetilde{\mathbf{e}}_{k,2}$, where $\{\widetilde{\mathbf{e}}_{k,1}, \widetilde{\mathbf{e}}_{k,2}\} \in \mathbb{R}^{k_{\max}}$.

Step 2: The generator chooses a set of random numbers in the real domain, including i) random numbers $r_{\widetilde{\mathbf{Q}}} > 0$, $r_{\widetilde{\mathbf{e}}_k} > 0$, $r'_{\widetilde{\mathbf{Q}}} > 0$, and $r'_{\widetilde{\mathbf{e}}_k} > 0$; ii) random numbers $\{r_{\widetilde{\mathbf{Q}}_{u,v,l,z}}, r_{\widetilde{\mathbf{e}}_{k,u,v,l,z}}, r_{\widetilde{\mathbf{Q}}'_{u,v,l,z}}, r_{\widetilde{\mathbf{e}}'_{k,u,v,l,z}} | \{u, v, l, z\} \in \{1, 2\}\}$ such that $\frac{1}{2} r_{\widetilde{\mathbf{Q}}} > r_{\widetilde{\mathbf{Q}}_{u,v,l,z}} > 0$, $\frac{1}{2} r_{\widetilde{\mathbf{e}}_k} > r_{\widetilde{\mathbf{e}}_{k,u,v,l,z}} > 0$, $\frac{1}{2} r'_{\widetilde{\mathbf{Q}}} > r_{\widetilde{\mathbf{Q}}'_{u,v,l,z}} > 0$, and $\frac{1}{2} r'_{\widetilde{\mathbf{e}}_k} > r_{\widetilde{\mathbf{e}}'_{k,u,v,l,z}} > 0$ for $\{u, v, l, z\} \in \{1, 2\}$; and iii) random numbers $\{\alpha_{\widetilde{\mathbf{Q}}_{u,v,l,z}}, \alpha_{\widetilde{\mathbf{e}}_{k,u,v,l,z}}, \alpha_{\widetilde{\mathbf{Q}}'_{u,v,l,z}}, \alpha_{\widetilde{\mathbf{e}}'_{k,u,v,l,z}} | \{u, v, l, z\} \in \{1, 2\}\}$ satisfying $\sum_{u=1}^{2} \sum_{v=1}^{2} \sum_{l=1}^{2} \sum_{z=1}^{2} (\alpha_{\widetilde{\mathbf{Q}}_{u,v,l,z}} \alpha_{\widetilde{\mathbf{e}}_{k,u,v,l,z}} + \alpha_{\widetilde{\mathbf{Q}}'_{u,v,l,z}} \alpha_{\widetilde{\mathbf{e}}'_{k,u,v,l,z}}) = 0$, which are generated in a similar way to that of $\{\alpha_{\mathbf{P}_{u,v,l,z}},$

$\alpha_{\mathbf{T}_{u,v,l,z}}, \alpha_{\mathbf{P}'_{u,v,l,z}}, \alpha_{\mathbf{T}'_{u,v,l,z}} | \{u, v, l, z\} \in \{1, 2\}\}$. These random numbers are different for different $\widetilde{\mathbf{Q}}$'s and $\widetilde{\mathbf{e}}_k$'s.

Step 3: The generator generates trapdoors for $\widetilde{\mathbf{Q}}$.

(1) Extend $\widetilde{\mathbf{Q}}$ to a group of $(d+5) \times 5$ matrices $\{\widetilde{\mathbf{Q}}_{u,v,l,z}, \widetilde{\mathbf{Q}}'_{u,v,l,z} | \{u, v, l, z\} \in \{1, 2\}\}$ as

$$\begin{cases} \widetilde{\mathbf{Q}}_{u,v,l,z} = \begin{bmatrix} r_{\widetilde{\mathbf{Q}}} * \widetilde{\mathbf{Q}}_v & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & r_{\widetilde{\mathbf{Q}}_{u,v,l,z}} & 0 & 0 \\ \mathbf{O} & 0 & 1 & 0 \\ \mathbf{O} & 0 & 0 & \alpha_{\widetilde{\mathbf{Q}}_{u,v,l,z}} \end{bmatrix} \\ \widetilde{\mathbf{Q}}'_{u,v,l,z} = \begin{bmatrix} r'_{\widetilde{\mathbf{Q}}} * \widetilde{\mathbf{Q}}_v & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & r_{\widetilde{\mathbf{Q}}'_{u,v,l,z}} & 0 & 0 \\ \mathbf{O} & 0 & 1 & 0 \\ \mathbf{O} & 0 & 0 & \alpha_{\widetilde{\mathbf{Q}}'_{u,v,l,z}} \end{bmatrix} \end{cases}. \quad (39)$$

(2) Encrypt each $\widetilde{\mathbf{Q}}_{u,v,l,z}$ and $\widetilde{\mathbf{Q}}'_{u,v,l,z}$ into trapdoors as

$$\begin{cases} \mathtt{TD}_{\widetilde{\mathbf{Q}}_{u,v,l,z}} = \widetilde{\mathbf{M}}^{-1}_{1,u,v,l,z} \widetilde{\mathbf{Q}}_{u,v,l,z} \widetilde{\mathbf{M}}_{2,u,v,l,z}; \\ \mathtt{TD}_{\widetilde{\mathbf{Q}}'_{u,v,l,z}} = \widetilde{\mathbf{M}'}^{-1}_{1,u,v,l,z} \widetilde{\mathbf{Q}}'_{u,v,l,z} \widetilde{\mathbf{M}'}_{2,u,v,l,z}. \end{cases} \quad (40)$$

Step 4: The generator generates trapdoors for $\widetilde{\mathbf{e}}_k$.

(1) Extend $\widetilde{\mathbf{e}}_k$ to a group of $(k_{\max}+3)$-dimensional vectors $\{\widetilde{\mathbf{e}}_{k,u,v,l,z}, \widetilde{\mathbf{e}}'_{k,u,v,l,z} | \{u, v, l, z\} \in \{1, 2\}\}$ as

$$\begin{cases} \widetilde{\mathbf{e}}_{k,u,v,l,z} = \begin{bmatrix} r_{\widetilde{\mathbf{e}}_k} * \widetilde{\mathbf{e}}_{k,z} & r_{\widetilde{\mathbf{e}}_{k,u,v,l,z}} & 1 & \alpha_{\widetilde{\mathbf{e}}_{k,u,v,l,z}} \end{bmatrix} \\ \widetilde{\mathbf{e}}'_{k,u,v,l,z} = \begin{bmatrix} r'_{\widetilde{\mathbf{e}}_k} * \widetilde{\mathbf{e}}_{k,z} & r_{\widetilde{\mathbf{e}}'_{k,u,v,l,z}} & 1 & \alpha_{\widetilde{\mathbf{e}}'_{k,u,v,l,z}} \end{bmatrix} \end{cases}. \quad (41)$$

(2) Encrypt each $\widetilde{\mathbf{e}}_{k,u,v,l,z}$ and $\widetilde{\mathbf{e}}'_{k,u,v,l,z}$ into trapdoors as

$$\begin{cases} \mathtt{TD}_{\widetilde{\mathbf{e}}_{k,u,v,l,z}} = \widetilde{\mathbf{M}}^{-1}_{3,u,v,l,z} \widetilde{\mathbf{e}}^{T}_{k,u,v,l,z}; \\ \mathtt{TD}_{\widetilde{\mathbf{e}}'_{k,u,v,l,z}} = \widetilde{\mathbf{M}'}^{-1}_{3,u,v,l,z} \widetilde{\mathbf{e}}'^{T}_{k,u,v,l,z}. \end{cases} \quad (42)$$

Step 5: The trapdoor generation algorithm finally outputs the trapdoor of $\widetilde{\mathbf{Q}}$ and $\widetilde{\mathbf{e}}_k$, i.e., $\mathtt{TD}_{\widetilde{\mathbf{Q}},\widetilde{\mathbf{e}}_k} = \{\mathtt{TD}_{\widetilde{\mathbf{Q}}_{u,v,l,z}}, \mathtt{TD}'_{\widetilde{\mathbf{e}}_{k,u,v,l,z}}, \mathtt{TD}_{\widetilde{\mathbf{Q}}_{u,v,l,z}}, \mathtt{TD}'_{\widetilde{\mathbf{e}}_{k,u,v,l,z}} | \{u, v, l, z\} \in \{1, 2\}\}$.

• $\mathsf{RPE.RefineQuery}(\mathtt{C}_{\widetilde{\mathbf{X}}_i,\widetilde{\mathbf{T}}}, \mathtt{TD}_{\widetilde{\mathbf{Q}},\widetilde{\mathbf{e}}_k})$: On input the ciphertext $\mathtt{C}_{\widetilde{\mathbf{X}}_i,\widetilde{\mathbf{T}}}$ and the trapdoor $\mathtt{TD}_{\widetilde{\mathbf{Q}},\widetilde{\mathbf{e}}_k}$, the evaluator first calculates

$$\widetilde{s} = \sum_{u=1}^{2}\sum_{v=1}^{2}\sum_{l=1}^{2}\sum_{z=1}^{2} (\mathtt{C}_{\widetilde{\mathbf{X}}_{i,u,v,l,z}} \mathtt{TD}_{\widetilde{\mathbf{Q}}_{u,v,l,z}} \mathtt{C}_{\widetilde{\mathbf{T}}_{u,v,l,z}} \mathtt{TD}_{\widetilde{\mathbf{e}}_{k,u,v,l,z}} + \mathtt{C}_{\widetilde{\mathbf{X}}'_{i,u,v,l,z}} \mathtt{TD}_{\widetilde{\mathbf{Q}}'_{u,v,l,z}} \mathtt{C}_{\widetilde{\mathbf{T}}'_{u,v,l,z}} \mathtt{TD}_{\widetilde{\mathbf{e}}'_{k,u,v,l,z}}). \quad (43)$$

If $\widetilde{s} < 0$, the evaluator returns 1 to demonstrate that "$\widetilde{\mathbf{X}}_i \widetilde{\mathbf{Q}} \widetilde{\mathbf{T}} \widetilde{\mathbf{e}}_k^T \leq 0$" and returns 0 to demonstrate that "$\widetilde{\mathbf{X}}_i \widetilde{\mathbf{Q}} \widetilde{\mathbf{T}} \widetilde{\mathbf{e}}_k^T > 0$" otherwise.

Theorem 3 *The RPE scheme is correct.*

Proof. The correctness of the RPE scheme can be proved in a similar way to that of the FPE scheme. Due to page limitation, we omit the detailed proof here.

6 Our PRkNN Scheme

This section introduces our PRkNN scheme, which uses the FPE scheme and RPE scheme to preserve the privacy of the MM-tree based reverse kNN query algorithm. Our scheme is defined as $\Pi_{\mathtt{PRkNN}} = (\mathsf{PRkNN.Setup}, \mathsf{PRkNN.Outsource}, \mathsf{PRkNN.TrapdoorGen}, \mathsf{PRkNN.Query})$.

• $\mathsf{PRkNN.Setup}(d, \kappa)$: In the setup algorithm, on input the dimensions of data d and a security parameter κ, the data owner setups the scheme by generating some secret keys as $sk \longleftarrow \mathsf{FPE.Setup}(d)$, $\widetilde{sk} \longleftarrow \mathsf{RPE.Setup}(d)$ and $K \overset{\$}{\in} \{0,1\}^{\kappa}$. Then, it distributes these keys to query users via a secure channel as the authorized keys.

• $\mathsf{PRkNN.Outsource}(sk, \widetilde{sk}, K, \mathcal{X})$: On input the secret keys $\{sk, \widetilde{sk}, K\}$, the data owner outsources its dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{n}$ to the cloud server by following the steps below.

Step 1: Build an MM-tree T for the dataset $\mathcal{X}$.

Step 2: Encrypt internal nodes of the MM-tree T. For each internal node with $(\mathbf{p}, \tau, \mathtt{L}_{\mathbf{p}} = [\tau_{\mathbf{p},1}, \tau_{\mathbf{p},2}, \cdots, \tau_{\mathbf{p},k_{\max}}])$, the data owner constructs a vector $\mathbf{P}$ and a matrix $\mathbf{T}$ based on its key values as Eq. (3), and further utilizes the FPE scheme to encrypt them as $\mathtt{C}_{\mathbf{P},\mathbf{T}} \longleftarrow \mathsf{FPE.Enc}(sk, \mathbf{P}, \mathbf{T})$.

Step 3: Encrypt leaf nodes of the MM-tree T. For each leaf node with $(\mathbf{x}_i, \mathtt{L}_{\mathbf{x}_i} = [\tau_{\mathbf{x}_i,1}, \tau_{\mathbf{x}_i,2}, \cdots, \tau_{\mathbf{x}_i,k_{\max}}])$, the data owner constructs a matrix $\widetilde{\mathbf{X}}_i$ and a vector $\widetilde{\mathbf{T}}$ based on its key values as Eq. (28), and utilizes the RPE scheme to encrypt them as $\mathtt{C}_{\widetilde{\mathbf{X}}_i,\widetilde{\mathbf{T}}} \longleftarrow \mathsf{RPE.Enc}(\widetilde{sk}, \widetilde{\mathbf{X}}_i, \widetilde{\mathbf{T}})$.

Besides, the data owner leverages the AES encryption algorithm to encrypt $\mathbf{x}_i$ as $\mathtt{AES}_K(\mathbf{x}_i)$. In this case, the ciphertext of the leaf node will be $\{\mathtt{C}_{\widetilde{\mathbf{X}}_i,\widetilde{\mathbf{T}}}, \mathtt{AES}_K(\mathbf{x}_i)\}$.

Step 4: The data owner outsources the encrypted MM-tree, denoted by $E(\mathrm{T})$, to the cloud server.

• $\mathsf{PRkNN.TrapdoorGen}(sk, \widetilde{sk}, \mathbf{q}, k)$: On input the secret keys $\{sk, \widetilde{sk}\}$ and a reverse kNN query request $(\mathbf{q}, k)$, the query user generates query trapdoors for $(\mathbf{q}, k)$, including a filter trapdoor and a refinement trapdoor, as follows.

Step 1: Generate a filter trapdoor. The query user first constructs a matrix $\mathbf{Q}$ and a vector $\mathbf{e}_k$ based on $(\mathbf{q}, k)$ as Eq. (3). Then, it utilizes the FPE scheme to generate a filter trapdoor as $\mathtt{TD}_{\mathbf{Q},\mathbf{e}_k} \longleftarrow \mathsf{FPE.TrapdoorGen}(sk, \mathbf{Q}, \mathbf{e}_k)$.

Step 2: Generate a refinement trapdoor. The query user first constructs a matrix $\widetilde{\mathbf{Q}}$ and a vector $\widetilde{\mathbf{e}}_k$ based on $(\mathbf{q}, k)$ as Eq. (28). Then, it utilizes the RPE scheme to generate a refinement trapdoor as $\mathtt{TD}_{\widetilde{\mathbf{Q}},\widetilde{\mathbf{e}}_k} \longleftarrow \mathsf{RPE.TrapdoorGen}(\widetilde{sk}, \widetilde{\mathbf{Q}}, \widetilde{\mathbf{e}}_k)$.

Step 3: The query user sends query trapdoors $\{\mathtt{TD}_{\mathbf{Q},\mathbf{e}_k}, \mathtt{TD}_{\widetilde{\mathbf{Q}},\widetilde{\mathbf{e}}_k}\}$ to the cloud server.

• $\mathsf{PRkNN.Query}(E(\mathrm{T}), \mathtt{TD}_{\mathbf{Q},\mathbf{e}_k}, \mathtt{TD}_{\widetilde{\mathbf{Q}},\widetilde{\mathbf{e}}_k}, K)$: On receiving the query trapdoors $\{\mathtt{TD}_{\mathbf{Q},\mathbf{e}_k}, \mathtt{TD}_{\widetilde{\mathbf{Q}},\widetilde{\mathbf{e}}_k}\}$, the cloud server searches on $E(\mathrm{T})$ for the query result. Similar to the reverse kNN query algorithm on plaintext data, the search algorithm on encrypted data includes a filter stage and a refinement stage. The only difference is that the inequalities "$d(\mathbf{p}, \mathbf{q}) > 2\tau + \tau_{\mathbf{p},k}$" and "$d(\mathbf{x}_i, \mathbf{q}) \leq \tau_{\mathbf{x}_i,k}$" are replaced with "$\mathsf{FPE.FilterQuery}(\mathtt{C}_{\mathbf{P},\mathbf{T}}, \mathtt{TD}_{\mathbf{Q},\mathbf{e}_k}) == 1$" and "$\mathsf{RPE.RefineQuery}(\mathtt{C}_{\widetilde{\mathbf{X}}_i,\widetilde{\mathbf{T}}}, \mathtt{TD}_{\widetilde{\mathbf{Q}},\widetilde{\mathbf{e}}_k}) == 1$". After running the search algorithm, the cloud server obtains the query result $\mathcal{R} = \{\mathtt{AES}_K(\mathbf{x}_i) | \mathsf{RPE.FilterQuery}(\mathtt{C}_{\widetilde{\mathbf{X}}_i,\widetilde{\mathbf{T}}}, \mathtt{TD}_{\widetilde{\mathbf{Q}},\widetilde{\mathbf{e}}_k}) == 1\}$ and sends it to the query user as the query response.

On receiving the query result $\mathcal{R}$, the query user decrypts each $\mathtt{AES}_K(\mathbf{x}_i) \in \mathcal{R}$ to obtain the plaintext record $\mathbf{x}_i$.

7 Security Analysis

In this section, we first rigorously prove the security of the FPE scheme and the RPE scheme. Then, we prove the security of our PRkNN scheme.

7.1 Security of FPE Scheme

We rigorously prove the selective security of the FPE scheme through a simulation-based real/ideal worlds model, in which ciphertexts and trapdoors in the real world are generated by following the FPE scheme while those in the ideal world are simulated based on its leakage. Since the real world is the same as our FPE scheme, we only concentrate on formalizing the ideal world, before which we first take a look at the leakage of our FPE scheme.

Leakage. Given an internal node with $(\mathbf{p}, \tau, \mathtt{L}_\mathbf{p} = [\tau_{\mathbf{p},1},\ \tau_{\mathbf{p},2}, \cdots, \tau_{\mathbf{p},k_{\max}}])$ and a query request $(\mathbf{q}, k)$, the FPE scheme leaks i) the dimensions of records, ciphertexts and trapdoors: d; ii) the maximum value of k: $k_{\max}$; and iii) the filter predicate evaluation result: $\mathsf{FPE.FilterQuery}(\mathtt{C}_{\mathbf{P},\mathbf{T}}, \mathtt{TD}_{\mathbf{Q},\mathbf{e}_k})$.

Based on the above-mentioned leakage, we formalize the ideal world as follows.

Ideal world. To prove the selective security of the FPE scheme, we define our own ideal model by following that defined in [29], in which a probability polynomial time (PPT) adversary and a simulator with the above leakage sequentially execute the following stages.

• *Setup:* At the beginning of the setup stage, the adversary carefully selects key values of an internal node, namely, $(\mathbf{p}, \tau, \mathtt{L}_\mathbf{p} = [\tau_{\mathbf{p},1}, \tau_{\mathbf{p},2}, \cdots, \tau_{\mathbf{p},k_{\max}}])$, where $\mathbf{p}$ is a d-dimensional integer record, τ is a non-negative integer, and $\mathtt{L}_\mathbf{p}$ is an integer list with $k_{\max}$ elements. Then, the adversary sends them to the simulator. Upon receiving $(\mathbf{p}, \tau, \mathtt{L}_\mathbf{p})$, the simulator randomly selects a group of vectors and matrices $\{\widehat{\mathtt{C}}_{\mathbf{P}_{u,v,l,z}}, \widehat{\mathtt{C}}_{\mathbf{P}'_{u,v,l,z}}, \widehat{\mathtt{C}}_{\mathbf{T}_{u,v,l,z}}, \widehat{\mathtt{C}}_{\mathbf{T}'_{u,v,l,z}}\}$, where $\widehat{\mathtt{C}}_{\mathbf{P}_{u,v,l,z}}, \widehat{\mathtt{C}}_{\mathbf{P}'_{u,v,l,z}} \in \mathbb{R}^{d+6}$ and $\widehat{\mathtt{C}}_{\mathbf{T}_{u,v,l,z}}, \widehat{\mathtt{C}}_{\mathbf{T}'_{u,v,l,z}} \in \mathbb{R}^{6\times k_{\max}}$ for $\{u, v, l, z\} \in \{1, 2\}$. Meanwhile, the simulator employs these random vectors and matrices to construct the ciphertext of the internal node as $\widehat{\mathtt{C}}_{\mathbf{P},\mathbf{T}} = \{\widehat{\mathtt{C}}_{\mathbf{P}_{u,v,l,z}}, \widehat{\mathtt{C}}_{\mathbf{P}'_{u,v,l,z}}, \widehat{\mathtt{C}}_{\mathbf{T}_{u,v,l,z}}, \widehat{\mathtt{C}}_{\mathbf{T}'_{u,v,l,z}} | \{u, v, l, z\} \in \{1, 2\}\}$.

• *First stage of trapdoor generation:* At this stage, the adversary sends γ_1 carefully selected reverse kNN query requests $\{(\mathbf{q}_j, k_j)\}_{j=1}^{\gamma_1}$ to the simulator, where γ_1 is a polynomial number, $\mathbf{q}_j$ is a d-dimensional integer record, and k_j is a positive integer satisfying $k_j \leq k_{\max}$ for $1 \leq j \leq \gamma_1$. After receiving these query requests, the simulator constructs the trapdoor for each $(\mathbf{q}_j, k_j)$ as the following steps.

Step 1: The simulator selects a random number $\widehat{s}_j$ based on the leakage $\mathsf{FPE.FilterQuery}(\mathtt{C}_{\mathbf{P},\mathbf{T}}, \mathtt{TD}_{\mathbf{Q}_j,\mathbf{e}_{j,k}})$, where

$$\begin{cases} \widehat{s}_j > 0 & \mathsf{FPE.FilterQuery}(\mathtt{C}_{\mathbf{P},\mathbf{T}}, \mathtt{TD}_{\mathbf{Q}_j,\mathbf{e}_{j,k}}) == 1 \\ \widehat{s}_j < 0 & \mathsf{FPE.FilterQuery}(\mathtt{C}_{\mathbf{P},\mathbf{T}}, \mathtt{TD}_{\mathbf{Q}_j,\mathbf{e}_{j,k}}) == 0. \end{cases} \tag{44}$$

Step 2: The simulator randomly selects a group of vectors and matrices $\{\widehat{\mathtt{TD}}_{\mathbf{Q}_{j,u,v,l,z}}, \widehat{\mathtt{TD}}_{\mathbf{Q}'_{j,u,v,l,z}}, \widehat{\mathtt{TD}}_{\mathbf{e}_{j,k,u,v,l,z}}, \widehat{\mathtt{TD}}_{\mathbf{e}'_{j,k,u,v,l,z}} | \{u, v, l, z\} \in \{1, 2\}\}$ such that

$$\widehat{s}_j = \sum_{u=1}^{2}\sum_{v=1}^{2}\sum_{l=1}^{2}\sum_{z=1}^{2}(\widehat{\mathtt{C}}_{\mathbf{P}_{j,u,v,l,z}} \widehat{\mathtt{TD}}_{\mathbf{Q}_{j,u,v,l,z}} \widehat{\mathtt{C}}_{\mathbf{T}_{j,u,v,l,z}} \widehat{\mathtt{TD}}_{\mathbf{e}_{j,k,u,v,l,z}} + \widehat{\mathtt{C}}_{\mathbf{P}'_{j,u,v,l,z}} \widehat{\mathtt{TD}}_{\mathbf{Q}'_{j,u,v,l,z}} \widehat{\mathtt{C}}_{\mathbf{T}'_{j,u,v,l,z}} \widehat{\mathtt{TD}}_{\mathbf{e}'_{j,k,u,v,l,z}}), \tag{45}$$

where $\{\widehat{\mathtt{TD}}_{\mathbf{Q}_{j,u,v,l,z}}, \widehat{\mathtt{TD}}_{\mathbf{Q}'_{j,u,v,l,z}}\} \in \mathbb{R}^{(d+6)\times(d+6)}$ and $\{\widehat{\mathtt{TD}}_{\mathbf{e}_{j,k,u,v,l,z}}, \widehat{\mathtt{TD}}_{\mathbf{e}'_{j,k,u,v,l,z}}\} \in \mathbb{R}^{k_{\max}+3}$. The simulator further regards these random vectors and matrices as the trapdoor of $(\mathbf{q}_j, k_j)$, denoted by $\widehat{\mathtt{TD}}_{\mathbf{Q}_j,\mathbf{e}_{j,k}} = \{\widehat{\mathtt{TD}}_{\mathbf{Q}_{j,u,v,l,z}}, \widehat{\mathtt{TD}}_{\mathbf{Q}'_{j,u,v,l,z}}, \widehat{\mathtt{TD}}_{\mathbf{e}_{j,k,u,v,l,z}}, \widehat{\mathtt{TD}}_{\mathbf{e}'_{j,k,u,v,l,z}} | \{u, v, l, z\} \in \{1, 2\}\}$.

Step 3: The simulator returns all constructed trapdoors $\{\widehat{\mathtt{TD}}_{\mathbf{Q}_j,\mathbf{e}_{j,k}}\}_{j=1}^{\gamma_1}$ to the adversary.

• *Challenge stage:* The simulator sends the ciphertext $\widehat{\mathtt{C}}_{\mathbf{P},\mathbf{T}}$ to the adversary.

• *Second stage of trapdoor generation:* At this stage, the adversary launches the second round of trapdoor generation requests. Similar to the first stage, the adversary gets the query trapdoors of γ_2 carefully selected reverse kNN query requests $\{(\mathbf{q}_j, k_j)\}_{j=1}^{\gamma_2}$ from the simulator, denoted by $\{\widehat{\mathtt{TD}}_{\mathbf{Q}_j,\mathbf{e}_{j,k}}\}_{j=1}^{\gamma_2}$, where γ_2 is a polynomial number.

The aforementioned formalization demonstrates the adversary in the ideal world can view the ciphertext $\widehat{\mathtt{C}}_{\mathbf{P},\mathbf{T}}$ and trapdoors $\{\widehat{\mathtt{TD}}_{\mathbf{Q}_j,\mathbf{e}_{j,k}}\}_{j=1}^{\gamma_1+\gamma_2}$. That is, $\text{View}_{\text{Ideal}} = \{\widehat{\mathtt{C}}_{\mathbf{P},\mathbf{T}}, \{\widehat{\mathtt{TD}}_{\mathbf{Q}_j,\mathbf{e}_{j,k}}\}_{j=1}^{\gamma_1+\gamma_2}\}$. While those corresponding ciphertext and trapdoors in the real world will be generated by following the FPE scheme, denoted by $\text{View}_{\text{Real}} = \{\mathtt{C}_{\mathbf{P},\mathbf{T}}, \{\mathtt{TD}_{\mathbf{Q}_j,\mathbf{e}_{j,k}}\}_{j=1}^{\gamma_1+\gamma_2}\}$. The security of the FPE scheme is formalized based on $\text{View}_{\text{Real}}$ and $\text{View}_{\text{Ideal}}$ as shown in Definition 2.

Definition 2 (Security of FPE Scheme) *The FPE scheme is selectively secure iff the adversary only has a negligible probability to distinguish View*$_{\textit{Real}}$ *and View*$_{\textit{Ideal}}$.

Theorem 4 *The FPE scheme is selectively secure.*

Proof. Since the security of the FPE scheme depends on the indistinguishability of $\text{View}_{\text{Real}}$ and $\text{View}_{\text{Ideal}}$, we concentrate on proving that $\text{View}_{\text{Real}}$ is indistinguishable from $\text{View}_{\text{Ideal}}$. From the formalization of the ideal world, it is obvious that the ciphertext and trapdoors in $\text{View}_{\text{Ideal}}$ are random vectors or matrices. As a consequence, we will focus on proving that the ciphertext and trapdoors in $\text{View}_{\text{Real}}$ are indistinguishable from random vectors and matrices. Since $\text{View}_{\text{Real}}$ includes one ciphertext $\mathtt{C}_{\mathbf{P},\mathbf{T}}$ and $\gamma_1 + \gamma_2$ trapdoors $\{\mathtt{TD}_{\mathbf{Q}_j,\mathbf{e}_{j,k}}\}_{j=1}^{\gamma_1+\gamma_2}$, we will prove the indistinguishability between $\text{View}_{\text{Real}}$ with random vectors and matrices from three aspects below.

(1) $\mathtt{C}_{\mathbf{P},\mathbf{T}}$ is indistinguishable from a random ciphertext. In the filter predicate encryption scheme, we have $\mathtt{C}_{\mathbf{P},\mathbf{T}} = \{\mathtt{C}_{\mathbf{P}_{u,v,l,z}}, \mathtt{C}_{\mathbf{P}'_{u,v,l,z}}, \mathtt{C}_{\mathbf{T}_{u,v,l,z}}, \mathtt{C}_{\mathbf{T}'_{u,v,l,z}} | \{u, v, l, z\} \in \{1, 2\}\}$, where

$$\begin{cases} \mathtt{C}_{\mathbf{P}_{u,v,l,z}} = \mathbf{P}_{u,v,l,z}\mathbf{M}_{1,u,v,l,z}; \\ \mathtt{C}_{\mathbf{P}'_{u,v,l,z}} = \mathbf{P}'_{u,v,l,z}\mathbf{M}'_{1,u,v,l,z}; \\ \mathtt{C}_{\mathbf{T}_{u,v,l,z}} = \mathbf{M}_{2,u,v,l,z}^{-1}\mathbf{T}_{u,v,l,z}\mathbf{M}_{3,u,v,l,z}; \\ \mathtt{C}_{\mathbf{T}'_{u,v,l,z}} = \mathbf{M}'^{-1}_{2,u,v,l,z}\mathbf{T}'_{u,v,l,z}\mathbf{M}'_{3,u,v,l,z}. \end{cases} \tag{46}$$

On the one hand, the vector $\mathtt{C}_{\mathbf{P}_{u,v,l,z}}$ is calculated by $\mathbf{P}_{u,v,l,z}\mathbf{M}_{1,u,v,l,z}$, where

$$\mathbf{P}_{u,v,l,z} = \begin{bmatrix} r_{\mathbf{P}} * \mathbf{P}_u & -r_{\mathbf{P}_{u,v,l,z}} & \alpha_{\mathbf{P}_{u,v,l,z}} & 1 \end{bmatrix} \tag{47}$$

and $\mathbf{M}_{1,u,v,l,z}$ is an unknown secret matrix. Since the vector $\mathbf{P}_u$ is an unknown vector and $\{r_{\mathbf{P}}, r_{\mathbf{P}_{u,v,l,z}}, \alpha_{\mathbf{P}_{u,v,l,z}}\}$ are random numbers, $\mathbf{P}_{u,v,l,z}$ can be regarded as an unknown random vector. By combining the unknownness of $\mathbf{M}_{1,u,v,l,z}$, we can deduce that $\mathtt{C}_{\mathbf{P}_{u,v,l,z}}$ is indistinguishable from a random vector. In a similar way, we can easily prove that $\mathtt{C}_{\mathbf{P}'_{u,v,l,z}}$ is also a random vector.

On the other hand, the matrix $\mathtt{C}_{\mathbf{T}_{u,v,l,z}}$ is calculated by $\mathbf{M}^{-1}_{2,u,v,l,z}\mathbf{T}_{u,v,l,z}\mathbf{M}_{3,u,v,l,z}$, where

$$\mathbf{T}_{u,v,l,z} = \begin{bmatrix} r_{\mathbf{T}} * \mathbf{T}_l & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & r_{\mathbf{T}_{u,v,l,z}} & 0 & 0 \\ \mathbf{O} & 0 & \alpha_{\mathbf{T}_{u,v,l,z}} & 0 \\ \mathbf{O} & 0 & 0 & 1 \end{bmatrix} \tag{48}$$

and $\{\mathbf{M}_{2,u,v,l,z}, \mathbf{M}_{3,u,v,l,z}\}$ are two unknown secret matrices. Since the vector $\mathbf{T}_l$ is an unknown matrix and $\{r_{\mathbf{T}}, r_{\mathbf{T}_{u,v,l,z}}, \alpha_{\mathbf{T}_{u,v,l,z}}\}$ are random numbers, $\mathbf{T}_{u,v,l,z}$ can be regarded as an unknown matrix. By combining the unknownness of $\{\mathbf{M}_{2,u,v,l,z}, \mathbf{M}_{3,u,v,l,z}\}$, we can infer that $\mathtt{C}_{\mathbf{T}_{u,v,l,z}}$ is indistinguishable from a random matrix. Similarly, we can prove that $\mathtt{C}_{\mathbf{T}'_{u,v,l,z}}$ is also a random matrix.

(2) $\{\mathtt{TD}_{\mathbf{Q}_j,\mathbf{e}_{j,k}}\}_{j=1}^{\gamma_1+\gamma_2}$ are indistinguishable from random trapdoors. We have $\mathtt{TD}_{\mathbf{Q}_j,\mathbf{e}_{j,k}} = \{\mathtt{TD}_{\mathbf{Q}_{j,u,v,l,z}}, \mathtt{TD}_{\mathbf{Q}'_{j,u,v,l,z}}, \mathtt{TD}_{\mathbf{e}_{j,k,u,v,l,z}}, \mathtt{TD}_{\mathbf{e}'_{j,k,u,v,l,z}} | \mathtt{C}_{\mathbf{P},\mathbf{T}} = \{\mathtt{C}_{\mathbf{P}_{u,v,l,z}}, \mathtt{C}_{\mathbf{P}'_{u,v,l,z}}, \mathtt{C}_{\mathbf{T}_{u,v,l,z}}, \mathtt{C}_{\mathbf{T}'_{u,v,l,z}} | \{u,v,l,z\} \in \{1,2\}\}$, where

$$\begin{cases} \mathtt{TD}_{\mathbf{Q}_{j,u,v,l,z}} = \mathbf{M}^{-1}_{1,u,v,l,z}\mathbf{Q}_{j,u,v,l,z}\mathbf{M}_{2,u,v,l,z}; \\ \mathtt{TD}_{\mathbf{Q}'_{j,u,v,l,z}} = \mathbf{M}'^{-1}_{1,u,v,l,z}\mathbf{Q}'_{j,u,v,l,z}\mathbf{M}'_{2,u,v,l,z}; \\ \mathtt{TD}_{\mathbf{e}_{j,k,u,v,l,z}} = \mathbf{M}^{-1}_{3,u,v,l,z}\mathbf{e}^T_{j,k,u,v,l,z}; \\ \mathtt{TD}_{\mathbf{e}'_{j,k,u,v,l,z}} = \mathbf{M}'^{-1}_{3,u,v,l,z}\mathbf{e}'^T_{j,k,u,v,l,z}. \end{cases} \tag{49}$$

Similar to the proof of indistinguishability between $\mathtt{C}_{\mathbf{P},\mathbf{T}}$ and a random ciphertext, since values in $\{\mathbf{Q}_{j,u,v,l,z}, \mathbf{e}_{j,k,u,v,l,z}\}$ are either unknown or random numbers; and $\{\mathbf{M}_{1,u,v,l,z}, \mathbf{M}_{2,u,v,l,z}, \mathbf{M}_{3,u,v,l,z}\}$ are unknown secret matrices, we can deduce that $\mathtt{TD}_{\mathbf{Q}_{j,u,v,l,z}}$ and $\mathtt{TD}_{\mathbf{e}_{j,k,u,v,l,z}}$ are indistinguishable from a random matrix and a random vector, respectively, which also works for $\mathtt{TD}_{\mathbf{Q}'_{j,u,v,l,z}}$ and $\mathtt{TD}_{\mathbf{e}'_{j,k,u,v,l,z}}$.

(3) The combination of $\mathtt{C}_{\mathbf{P},\mathbf{T}}$ and $\{\mathtt{TD}_{\mathbf{Q}_j,\mathbf{e}_{j,k}}\}_{j=1}^{\gamma_1+\gamma_2}$ is indistinguishable from random numbers. Having both $\mathtt{C}_{\mathbf{P},\mathbf{T}}$ and $\{\mathtt{TD}_{\mathbf{Q}_j,\mathbf{e}_{j,k}}\}_{j=1}^{\gamma_1+\gamma_2}$ enables the adversary to remove some random numbers and unknown secret matrices through computing s_j, i.e., $s_j = \sum_{u=1}^2 \sum_{v=1}^2 \sum_{l=1}^2 \sum_{z=1}^2 (\mathtt{C}_{\mathbf{P}_{j,u,v,l,z}} \mathtt{TD}_{\mathbf{Q}_{j,u,v,l,z}} \mathtt{C}_{\mathbf{T}_{j,u,v,l,z}} \mathtt{TD}_{\mathbf{e}_{j,k,u,v,l,z}} + \mathtt{C}_{\mathbf{P}'_{j,u,v,l,z}} \mathtt{TD}_{\mathbf{Q}'_{j,u,v,l,z}} \mathtt{C}_{\mathbf{T}'_{j,u,v,l,z}} \mathtt{TD}_{\mathbf{e}'_{j,k,u,v,l,z}}) = (r_{\mathbf{P},\mathbf{Q}_j,\mathbf{T},\mathbf{e}_{j,k}} + r'_{\mathbf{P},\mathbf{Q}_j,\mathbf{T},\mathbf{e}_{j,k}})(\mathbf{P}\mathbf{Q}_j\mathbf{T}\mathbf{e}_j^T) - \sum_{u=1}^2 \sum_{v=1}^2 \sum_{l=1}^2 \sum_{z=1}^2 (r_{\mathbf{P},\mathbf{Q}_j,\mathbf{T},\mathbf{e}_{j,k,u,v,l,z}} + r'_{\mathbf{P},\mathbf{Q}_j,\mathbf{T},\mathbf{e}_{j,k,u,v,l,z}})$ for $1 \le j \le \gamma_1 + \gamma_2$. Although some random numbers and unknown secret matrices have been removed, the computed value s_j still contains random numbers $\{r_{\mathbf{P},\mathbf{Q}_j,\mathbf{T},\mathbf{e}_{j,k}}, r'_{\mathbf{P},\mathbf{Q}_j,\mathbf{T},\mathbf{e}_{j,k}}, (r_{\mathbf{P},\mathbf{Q}_j,\mathbf{T},\mathbf{e}_{j,k,u,v,l,z}} + r'_{\mathbf{P},\mathbf{Q}_j,\mathbf{T},\mathbf{e}_{j,k,u,v,l,z}}) | \{u,v,l,z\} \in \{1,2\}\}$, which makes s_j indistinguishable from a random number. Thus, the combination of $\mathtt{C}_{\mathbf{P},\mathbf{T}}$ and $\{\mathtt{TD}_{\mathbf{Q}_j,\mathbf{e}_{j,k}}\}_{j=1}^{\gamma_1+\gamma_2}$ is indistinguishable from random numbers.

In summary, the adversary has a negligible probability to distinguish the ciphertext and trapdoors in $\text{View}_{\text{Ideal}}$ from random ones. Thus, the FPE scheme is selectively secure.

7.2 Security of RPE Scheme

Similar to the proof of the FPE scheme, we can prove that the RPE scheme is selectively secure in the simulation-based real/ideal model, but the RPE scheme has a different leakage, as shown below.

Leakage. Given a leaf node with $(\mathbf{x}_i, \mathtt{L}_{\mathbf{x}_i} = [\tau_{\mathbf{x}_i,1}, \tau_{\mathbf{x}_i,2}, \cdots, \tau_{\mathbf{x}_i,k_{\max}}])$ and a query request $(\mathbf{q}, k)$, the RPE scheme leaks i) the dimensions of records, ciphertexts and trapdoors: d; ii) the maximum value of k: $k_{\max}$; and iii) the refinement predicate evaluation result: $\mathsf{RPE.FilterQuery}(\mathtt{C}_{\widetilde{\mathbf{X}}_i,\widetilde{\mathbf{T}}}, \mathtt{TD}_{\widetilde{\mathbf{Q}},\widetilde{\mathbf{e}}_k})$.

Based on the leakage, we can formalize the ideal world of the RPE scheme and define its selective security. Since the proof is similar to that of the FPE scheme, we omit detailed proof description here.

7.3 Security of PRkNN

Relying on the security of the FPE scheme and RPE scheme, we show that our PRkNN scheme can preserve the privacy of the outsourced dataset and reverse kNN query requests.

- *The outsourced dataset is privacy-preserving.* The dataset $\mathcal{X}$, which is represented by an encrypted MM-tree $E(\mathrm{T})$, is stored and searched in the cloud, and demands to be kept private from the honest-but-curious cloud server. Commonly, there are two ways for the cloud server to infer the dataset $\mathcal{X}$ from $E(\mathrm{T})$. One is to make inference from the static encrypted tree $E(\mathrm{T})$, and the other is from the process of tree search. However, both of them are in vain.

On the one hand, when $E(\mathrm{T})$ is in a static state, internal nodes are encrypted by the FPE scheme, and leaf nodes are encrypted by the RPE scheme and AES encryption. Since the security of AES encryption is well recognized; and that of the FPE and RPE schemes has been proved in Section 7.1 and Section 7.2, the cloud server is clueless to infer the dataset $\mathcal{X}$ from ciphertexts of $E(\mathrm{T})$'s internal nodes and leaf nodes. Hence, the dataset is privacy-preserving in the static state. On the other hand, when the tree is searched by the cloud server for reverse kNN queries, compared with the case in the static tree, the cloud server can additionally obtain the filter and refinement predicate evaluation results on each internal node and leaf node, respectively. However, the security of the FPE and RPE schemes ensures the cloud server is still clueless to infer the dataset $\mathcal{X}$ even with the evaluation results. Thus, the dataset is privacy-preserving.

- *The reverse kNN query requests are privacy-preserving.* Each reverse kNN query $(\mathbf{q}, k)$ is sent to the cloud server in the form of the corresponding trapdoors $\{\mathtt{TD}_{\mathbf{Q},\mathbf{e}_k}, \mathtt{TD}_{\widetilde{\mathbf{Q}},\widetilde{\mathbf{e}}_k}\}$ that are respectively generated by the FPE and RPE schemes. The security of these two schemes proved in Section 7.1 and Section 7.2 contributes to the privacy protection of the query $(\mathbf{q}, k)$ such that the cloud server is clueless to infer the query $(\mathbf{q}, k)$. Same as the above privacy preservation analysis of the dataset, when the cloud server searches this query on $E(\mathrm{T})$, it can obtain the the filter and refinement predicate evaluation results of $\{\mathtt{TD}_{\mathbf{Q},\mathbf{e}_k}, \mathtt{TD}_{\widetilde{\mathbf{Q}},\widetilde{\mathbf{e}}_k}\}$ on each internal node and leaf node. However, the security of the FPE and RPE schemes ensures that the query $(\mathbf{q}, k)$ is privacy-preserving even with the leakage of these evaluation results. Therefore, reverse kNN query requests are privacy-preserving.

8 Performance Evaluation

In this section, we theoretically and experimentally analyze the performance of our scheme.

8.1 Theoretical Analysis

We analyze the computational complexity and storage overhead of our scheme.

8.1.1 Computational Complexity

We analyze the computational complexity of our scheme with respect to data outsourcing, trapdoor generation, and query processing.

• *Computational Complexity of Data Outsourcing.* Our scheme represents a dataset to an MM-tree and encrypts it before being outsourced to the cloud server. Building an MM-tree includes two steps, i.e., build an M-tree based on the dataset and transform the M-tree to an MM-tree by attaching an additional distance list to each of internal and leaf nodes. The former step requires $O(n*d)$ computational complexity, and the later step requires $O(n*\log_2 n*d*k_{\max})$ computational complexity. When encrypting MM-tree, internal and leaf nodes are respectively encrypted using $\mathsf{FPE.Enc}(sk, \mathbf{P}, \mathbf{T})$ and $\mathsf{RPE.Enc}(\widetilde{sk}, \widetilde{\mathbf{X}}_i, \widetilde{\mathbf{T}})$ algorithms. Encrypting an internal node requires $O(32*(d+6)^2+32*36*(k_{\max}+3)+32*6*(k_{\max}+3)^2)$ computational complexity, i.e., $O(d^2+k_{\max}^2)$. Encrypting a leaf node requires $O(32*(d+5)^2+32*25*(k_{\max}+3)+32*5*(k_{\max}+3)^2)$ computational complexity, i.e., $O(d^2+k_{\max}^2)$. Thus, encrypting the entire MM-tree takes $O(n*(d^2+k_{\max}^2))$ computational complexity, and the overall computational complexity of data outsourcing will be $O(n*\log_2 n*d*k_{\max}+n*(d^2+k_{\max}^2))$.

• *Computational Complexity of Trapdoor Generation.* A query trapdoor includes a filter trapdoor and a refinement trapdoor. Generating a filter trapdoor requires $O(32*6*(d+6)^2+32*36*(d+6)+32*(k_{\max}+3)^2)$ computational complexity, i.e., $O(d^2+k_{\max}^2)$. Generating a refinement trapdoor requires $O(32*5*(d+5)^2+32*25*(d+6)+32*(k_{\max}+3)^2)$ computational complexity, i.e., $O(d^2+k_{\max}^2)$.

• *Computational Complexity of Query Processing.* Processing a reverse kNN query request is to search on an encrypted MM-tree for the query result. Searching an internal node takes $O(32*6*(d+6)+32*6*(d+6)*(k_{\max}+3)+32*6*(k_{\max}+3))$ computational complexity, i.e., $O(d*k_{\max})$. Searching a leaf node takes $O(32*5*(d+5)+32*5*(d+5)*(k_{\max}+3)+32*5*(k_{\max}+3))$ computational complexity, i.e., $O(d*k_{\max})$. The computational complexity of a reverse kNN query will be $O(k*\log_2 n*d*k_{\max})$.

8.1.2 Storage Overhead

We analyze the storage overhead of encrypted MM-tree and query trapdoor in our scheme.

• *Storage Overhead of Encrypted MM-tree.* The encrypted MM-tree contains internal nodes and leaf nodes. The ciphertext of each internal node is in the form of $\mathtt{C}_{\mathbf{P},\mathbf{T}} = \{\mathtt{C}_{\mathbf{P}_{u,v,l,z}}, \mathtt{C}_{\mathbf{P}'_{u,v,l,z}}, \mathtt{C}_{\mathbf{T}_{u,v,l,z}}, \mathtt{C}_{\mathbf{T}'_{u,v,l,z}} | \{u,v,l,z\} \in \{1,2\}\}$. The corresponding storage overhead includes $32*(d+6)+32*6*(k_{\max}+3)$ real numbers, i.e., $O(d+k_{\max})$. The ciphertext of each leaf node is in the form of $\{\mathtt{C}_{\widetilde{\mathbf{X}}_i,\widetilde{\mathbf{T}}}, \mathtt{AES}_K(\mathbf{x}_i)\}$, where $\mathtt{C}_{\widetilde{\mathbf{X}}_i,\widetilde{\mathbf{T}}} = \{\mathtt{C}_{\widetilde{\mathbf{X}}_{i,u,v,l,z}}, \mathtt{C}_{\widetilde{\mathbf{X}}'_{i,u,v,l,z}}, \mathtt{C}_{\widetilde{\mathbf{T}}_{u,v,l,z}}, \mathtt{C}_{\widetilde{\mathbf{T}}'_{u,v,l,z}} | \{u,v,l,z\} \in \{1,2\}\}$. The corresponding storage overhead includes $32*(d+5)+32*5*(k_{\max}+3)$ real numbers and an AES ciphertext. Thus, the storage overhead of the entire MM-tree will be $O(n*(d+k_{\max}))$ and n AES ciphertexts.

• *Storage Overhead of Query Trapdoor.* A query trapdoor in our scheme includes a filter trapdoor and a refinement trapdoor. The filter trapdoor is in the form of $\mathtt{TD}_{\widetilde{\mathbf{Q}},\widetilde{\mathbf{e}}_k} = \{\mathtt{TD}_{\widetilde{\mathbf{Q}}_{u,v,l,z}}, \mathtt{TD}'_{\widetilde{\mathbf{e}}_{k,u,v,l,z}}, \mathtt{TD}_{\widetilde{\mathbf{Q}}_{u,v,l,z}}, \mathtt{TD}'_{\widetilde{\mathbf{e}}_{k,u,v,l,z}} | \{u,v,l,z\} \in \{1,2\}\}$. The storage overhead includes $32*6*(d+6)+32*(k_{\max}+3)$ real numbers. The refinement trapdoor is in the form of $\mathtt{TD}_{\widetilde{\mathbf{Q}},\widetilde{\mathbf{e}}_k} = \{\mathtt{TD}_{\widetilde{\mathbf{Q}}_{u,v,l,z}}, \mathtt{TD}'_{\widetilde{\mathbf{e}}_{k,u,v,l,z}}, \mathtt{TD}_{\widetilde{\mathbf{Q}}_{u,v,l,z}}, \mathtt{TD}'_{\widetilde{\mathbf{e}}_{k,u,v,l,z}} | \{u,v,l,z\} \in \{1,2\}\}$. The storage overhead includes $32*5*(d+5)+32*(k_{\max}+3)$ real numbers. Thus, the storage overhead of the entire trapdoor is $O(d+k_{\max})$.

8.2 Experimental Analysis

In this section, we experimentally analyze the computational costs of data outsourcing, trapdoor generation, and query processing in our scheme. For the existing schemes [6]–[10], the schemes in [6], [7] cannot return accurate query results; the scheme in [8] cannot protect the dataset privacy; the scheme in [9] is not applicable to multi-dimensional data; and the scheme in [10] requires to predefine the parameter k, as shown in Table 1. In this case, our scheme is the first privacy-preserving reverse kNN query scheme that can simultaneously return accurate query results and support multi-dimensional data and a flexible choice of k only with the constraint of $k \le k_{\max}$. As a result, existing schemes are not competitors of our scheme.

Experimental Setting. We implemented our scheme in Java and conducted experiments on a machine with Apple M1 chip, 16 GB RAM, and macOS Big Sur operating system. In our implementation, we set the security parameter as $\kappa = 256$, and the fanout of the MM-tree is between 2 and 4. We perform the evaluation on a real forest Covertype dataset from the UCI machine learning repository [30]. In our experiment, we take 60000 records from the Covertype dataset to be used for our performance evaluation, and each one has 4 attributes that are chosen from the first ten attributes of all attributes. To have an accurate experimental results, each experiment is conducted 100 times repeatedly, and the average computational costs are reported. Detailed experimental results are described as follows.

8.2.1 Computational Cost

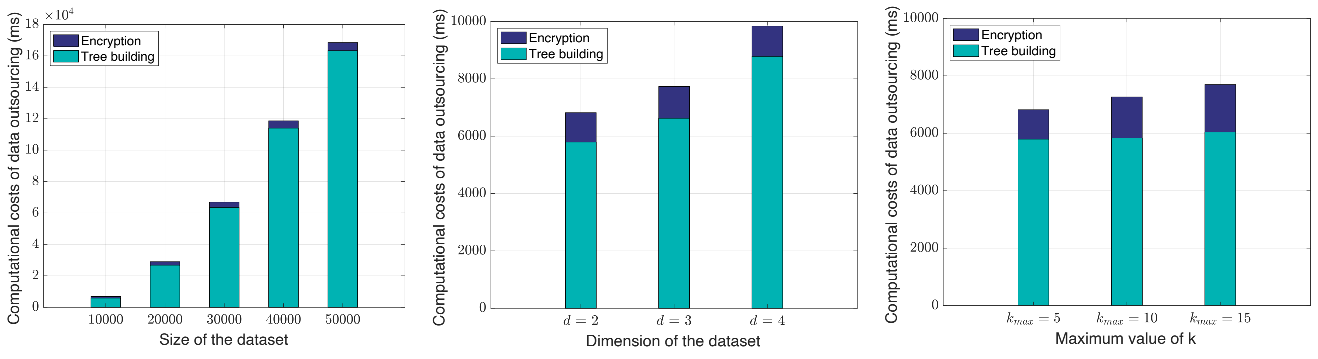
We evaluate the computations costs of data outsourcing, trapdoor generation, and query processing.

• *Data Outsourcing.* The computational costs of tree building and encryption are affected by three parameters, including i) n: the size of the dataset; ii) d: the dimensions of the dataset; iii) $k_{\max}$: the maximum number of k. Thus, the computational costs of data outsourcing depend on the parameters $\{n, d, k_{\max}\}$.

TABLE 1
Comparison among our PRkNN scheme and existing schemes

Scheme	Query Result	Dataset Privacy	#Data Dimension	Parameter k
Du in [6]	Approximate	$\surd$	Two	Predefined ($k = 1$)
Lin in [7]	Approximate	$\surd$	Two	Flexible
Pournajaf in [8]	Accurate	$\times$	Two	Flexible
Li in [9]	Accurate	$\surd$	Two	Predefined ($k = 1$)
Tzouramanis in [10]	Accurate	$\surd$	Multi-dimensional	Predefined (any $k > 0$)
Our PRkNN scheme	Accurate	$\surd$	Multi-dimensional	Flexible ($k \leq k_{max}$)

"$\surd$" denotes the scheme satisfies the objective.
"$\times$" denotes the scheme does not satisfy the objective.



(a) Dataset outsourcing with n (b) Dataset outsourcing with d (c) Dataset outsourcing with $k_{\max}$

Fig. 3. Computational costs of data outsourcing

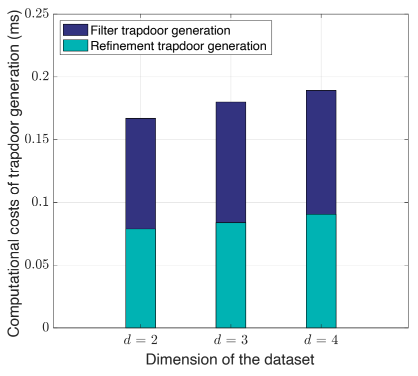
Fig. 3(a), Fig. 3(b), and Fig. 3(c) respectively depict the computational costs of data outsourcing varying with n, d, and $k_{\max}$, including the computational costs of tree building and encryption. Overall, these figures show that the computational costs of tree building (on cyan in figures) account for a much larger proportion than that of tree encryption (on dark blue in figures) during the process of data outsourcing. This is because building an MM-tree requires to compute $k_{\max}$ nearest neighbors for the center $\mathbf{p}$ of each internal node and the data record $\mathbf{x}_i$ of each leaf node, which is computationally expensive even if over plaintext data. In contrary, the computational costs of tree encryption are much smaller, which demonstrates that both FPE and RPE schemes are efficient in the data encryption.

Specifically, Fig. 3(a) shows how the computational costs of data outsourcing vary with n. In this experiment, we set $n \in \{10000, 20000, 30000, 40000, 50000\}$, $d = 2$, $k_{\max} = 5$. The results show that the computational costs of data outsourcing greatly increase with n, which is mainly caused by the increasing computational costs of tree building. It is reasonable because the MM-tree becomes large with the increase of n. Similarly, Fig. 3(b) shows the computational costs of data outsourcing increase with d due to the increasing computational costs of tree building, where the parameters are set as $n = 10000$, $d \in \{2, 3, 4\}$, $k_{\max} = 5$. Fig. 3(c) shows a different trend, i.e., the computational costs of data outsourcing slowly increase with $k_{\max}$ due to the increasing computational costs of tree encryption rather than tree building, where the parameters are set as $n = 10000, d = 2, k_{\max} = \{5, 10, 15\}$.

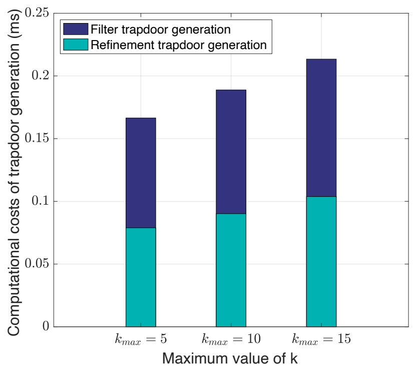
• *Trapdoor Generation.* As described in theoretical analysis, the computational costs of trapdoor generation are affected by the parameters d and $k_{\max}$. In Fig. 4, we respectively depict the computational costs of trapdoor generation varying with d and $k_{\max}$, including filter trapdoor generation and refinement trapdoor generation. From Fig. 4(a) and Fig 4(b), we can see that the computational costs of filter trapdoor generation are slightly higher than that of refinement trapdoor generation, and the overall computational costs of trapdoor generation are low. For example, generating a query trapdoor for a 2-dimensional data with $k_{\max} = 10$ only takes 0.189 ms. Meanwhile, Fig. 4(a) shows the computational costs of trapdoor generation have an increasing trend with the increase of d under the setting of $k_{\max} = 10$. Fig. 4(b) shows the computational costs of trapdoor generation have an increasing trend with the increase of $k_{\max}$ under the setting of $d = 4$. Thus, the computational costs of trapdoor generation increase with $k_{\max}$ and d, which is reasonable because the size of trapdoors increases with $k_{\max}$ and d.

• *Query Processing.* As described in theoretical analysis, the computational costs of reverse kNN query processing is affected by the parameters $\{n, d, k_{\max}, k\}$. In the following, we respectively evaluate how these parameters affect the computational costs of query processing, respectively.

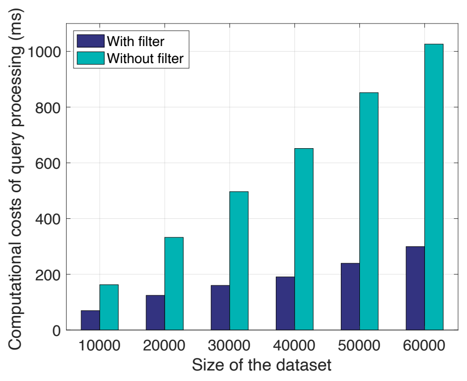
Fig. 4(c) describes the computational costs of query processing varying with n, where the parameters of this experiment are set as $n \in \{10000, 20000, 30000, 40000, 50000, 60000\}$, $d = 2$, $k_{\max} = 5$, and $k = 1$. Meanwhile, to validate the efficiency of our filter strategy, we compare the computational costs of our scheme with filter and without filter. According to this figure, we can observe that there is an growing trend for the computational costs of query processing with the increase of n, and the filter strategy



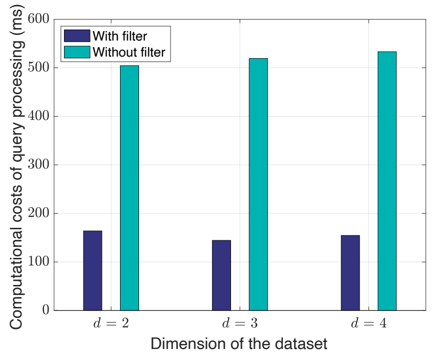
(a) Trapdoor generation with d



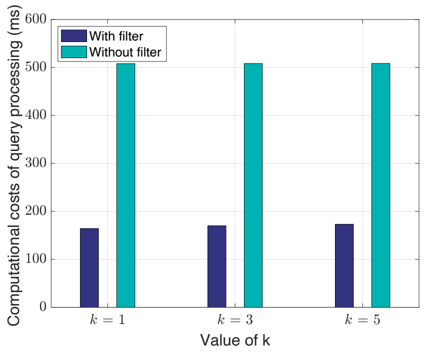
(b) Trapdoor generation with $k_{\max}$



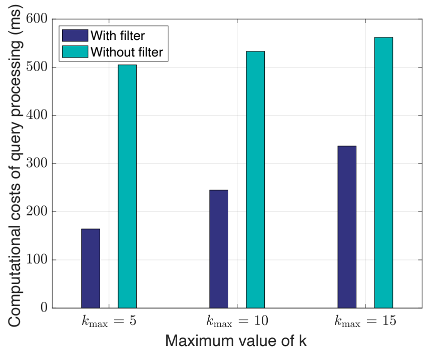
(c) Query processing with n



(d) Query processing with d



(e) Query processing with k



(f) Query processing with $k_{\max}$

Fig. 4. Computational costs of trapdoor generation and query processing

can greatly improve the query efficiency. When n becomes larger, our scheme with filter shows a more significant advantage over that without filter, which demonstrates the effectiveness of our filter strategy.

Fig. 4(d) describes the computational costs of query processing varying with d, where the parameters of this experiment are set as $n = 30000$, $d \in \{2, 3, 4\}$, $k_{\max} = 5$, and $k = 1$. Also, we compare the computational costs of our scheme with filter and without filter to validate the effectiveness of our filter strategy. This figure shows that the computational costs of our scheme without filter slightly grow with d. This is because the increase of d will lead to the slight increase of query processing in the FPE and RPE schemes, which are respectively used for filtering internal nodes and refining leaf nodes during the process of query processing. Meanwhile, the computational costs of our scheme with filter do not have an obvious increasing trend with d. When $d = 3$, the computational costs are even lower than that when $d = 2$. This is because the change of d will affect the structure of the MM-tree, which will lead to the uncertain influence on the computational costs of query processing. Luckily, the overall computational costs with filter are much smaller than that without filter.

Fig. 4(e) describes the computational costs of query processing varying with k, where the parameters of this experiment are set as $n = 30000$, $d = 2$, $k_{\max} = 5$, and $k \in \{1, 3, 5\}$. This figure shows that the computational costs of our scheme with filter slightly grow with k. This is because the increase of k will lead to a larger number of data records in the query results, which requires more computational costs to filter internal nodes and refine leaf nodes of the MM-tree, respectively. The computational costs of our scheme without filter keep unchanged because the query in this case is processed by linear traversing and the computational costs are not affected by k.

Fig. 4(f) describes the computational costs of query processing varying with $k_{\max}$, where the parameters of this experiment are set as $n = 30000$, $d = 2$, $k_{\max} \in \{5, 10, 15\}$, and $k = 2$. This figure shows that the computational costs of our scheme with filter grow with $k_{\max}$, and the advantage of the filter strategy will degrade with the increase of $k_{\max}$. Similarly, the computational costs of our scheme without filter also grow with $k_{\max}$, but the increasing rate is slower. Luckily, our scheme with filter is still more efficient than that without filter.

TABLE 2
Storage overhead of encrypted MM-tree with n

Size of Dataset	$n = 10000$	$n = 20000$	$n = 30000$	$n = 40000$	$n = 50000$
Storage Overhead	43.51 MB	87.24 MB	130.83 MB	174.65 MB	218.27 MB

*We set $d = 2$ and $k_{\max} = 5$.

8.2.2 Storage Overhead

We evaluate the storage overhead of encrypted MM-tree and query trapdoor.

• *Storage Overhead of Encrypted MM-tree.* As described in theoretical analysis, the storage overhead of encrypted MM-

TABLE 3
Storage overhead of encrypted MM-tree with d

#Dimensions	$d = 2$	$d = 3$	$d = 4$	$d = 5$	$d = 6$
Storage Overhead	43.51 MB	44.51 MB	45.14 MB	45.60 MB	46.06 MB

* We set $n = 10000$ and $k_{\max} = 5$.

TABLE 4
Storage overhead of encrypted MM-tree with $k_{\max}$

$k_{\max}$	$k_{\max} = 5$	$k_{\max} = 10$	$k_{\max} = 15$	$k_{\max} = 20$	$k_{\max} = 25$
Storage Overhead	43.51 MB	58.25 MB	72.98 MB	87.72 MB	102.45 MB

* We set $n = 10000$ and $d = 2$.

TABLE 5
Storage overhead of query trapdoor with d

#Dimensions	$d = 2$	$d = 3$	$d = 4$	$d = 5$	$d = 6$
Storage Overhead	7.42 KB	7.92 KB	8.42 KB	8.92 KB	9.42 KB

* We set $k_{\max} = 10$.

TABLE 6
Storage overhead of query trapdoor with $k_{\max}$

$k_{\max}$	$k_{\max} = 5$	$k_{\max} = 10$	$k_{\max} = 15$	$k_{\max} = 20$	$k_{\max} = 25$
Storage Overhead	7.29 KB	8.42 KB	9.48 KB	10.61 KB	11.67 KB

* We set $d = 4$.

tree is affected by n, d, and $k_{\max}$. In Table 2, Table 3, and Table 4, we depict the storage overhead of encrypted MM-tree varying with n, d, and $k_{\max}$, respectively. From these tables, we can see that the storage overhead increases with n, d, and $k_{\max}$.

• *Storage Overhead of Query Trapdoor.* As described in the theoretical analysis, the storage overhead of query trapdoor is affected by d and $k_{\max}$. In Table 5 and Table 6, we depict the storage overhead of query trapdoor varying with d and $k_{\max}$, respectively. From these tables, we can see that the storage overhead increases with d and $k_{\max}$.

9 Conclusion

In this paper, we have proposed an efficient and privacy-preserving reverse kNN query scheme in the outsourced scenario, which can preserve the data privacy and support the flexible choice of query records and the parameter k only with constraint of $k \leq k_{\max}$. Specifically, we first designed an MM-tree to index the dataset, and based on it, we introduced an MM-tree based reverse kNN query algorithm that improves the query efficiency through a filter strategy. Then, we employed the lightweight matrix encryption to carefully design the FPE and RPE schemes. After that, we proposed our PRkNN scheme by applying the FPE and RPE schemes to preserve the privacy of the reverse kNN query algorithm. In our future work, we plan to design a privacy-preserving reverse kNN query scheme supporting the dynamic update of the dataset.

Acknowledgements

This research was supported in part by NSERC Discovery Grants (04009, RGPIN-2022-03244), and NSFC (61972304).

References

[1] F. Wang, H. Zhu, R. Lu, Y. Zheng, and H. Li, "A privacy-preserving and non-interactive federated learning scheme for regression training with gradient descent," *Inf. Sci.*, vol. 552, pp. 183–200, 2021.

[2] "Big data market: Increasing data generation to drive growth post the crisis," online: https://www.prnewswire.com/news-releases/big-data-market-increasing-data-generation-to-drive-growth-post-the-crisis-301315941.html.

[3] S. Zhang, S. Ray, R. Lu, Y. Zheng, and J. Shao, "Preserving location privacy for outsourced most-frequent item query in mobile crowdsensing," *IEEE Internet Things J.*, vol. 8, no. 11, pp. 9139–9150, 2021.

[4] Y. Zheng, R. Lu, Y. Guan, J. Shao, and H. Zhu, "Towards practical and privacy-preserving multi-dimensional range query over cloud," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2021, doi=10.1109/TDSC.2021.3101120.

[5] S. Yang, M. A. Cheema, X. Lin, and W. Wang, "Reverse k nearest neighbors query processing: Experiments and analysis," *Proc. VLDB Endow.*, vol. 8, no. 5, pp. 605–616, 2015.

[6] Y. Du, "Privacy-aware RNN query processing on location-based services," in *(MDM 2007)*, C. Becker, C. S. Jensen, J. Su, and D. Nicklas, Eds., 2007, pp. 253–257.

[7] X. Lin, L. Zhou, P. Chen, and J. Gu, "Privacy preserving reverse nearest-neighbor queries processing on road network," in *WAIM 2012*, ser. Lecture Notes in Computer Science, vol. 7419, 2012, pp. 19–28.

[8] L. Pournajaf, F. Tahmasebian, L. Xiong, V. S. Sunderam, and C. Shahabi, "Privacy preserving reverse k-nearest neighbor queries," in *MDM 2018*, 2018, pp. 177–186.

[9] X. Li, T. Xiang, S. Guo, H. Li, and Y. Mu, "Privacy-preserving reverse nearest neighbor query over encrypted spatial data," *IEEE Transactions on Services Computing*, pp. 1–1, 2021, doi=10.1109/TSC.2021.3065356.

[10] T. Tzouramanis and Y. Manolopoulos, "Secure reverse k-nearest neighbours search over encrypted multi-dimensional databases," in *IDEAS 2018*, 2018, pp. 84–94.

[11] W. K. Wong, D. W. Cheung, B. Kao, and N. Mamoulis, "Secure knn computation on encrypted databases," in *SIGMOD*, 2009, pp. 139–152.

[12] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y. T. Hou, and H. Li, "Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," in *ASIA CCS '13*, 2013, pp. 71–82.

[13] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Trans. Parallel Distributed Syst.*, vol. 25, no. 1, pp. 222–233, 2014.

[14] B. Wang, S. Yu, W. Lou, and Y. T. Hou, "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud," in *INFOCOM 2014*, 2014.

[15] H. Li, Y. Yang, T. H. Luan, X. Liang, L. Zhou, and X. S. Shen, "Enabling fine-grained multi-keyword search supporting classified sub-dictionaries over encrypted cloud data," *IEEE Trans. Dependable Secur. Comput.*, vol. 13, no. 3, pp. 312–325, 2016.

[16] J. Yu, P. Lu, Y. Zhu, G. Xue, and M. Li, "Toward secure multi-keyword top-k retrieval over encrypted cloud data," *IEEE Trans. Dependable Secur. Comput.*, vol. 10, no. 4, pp. 239–250, 2013.

[17] Y. Zhu, R. Xu, and T. Takagi, "Secure k-nn computation on encrypted cloud data without sharing key with query users," in *ASIACCS 2013*, 2013, pp. 55–60.

[18] W. Lin, K. Wang, Z. Zhang, and H. Chen, "Revisiting security risks of asymmetric scalar product preserving encryption and its variants," in *ICDCS 2017*, 2017, pp. 1116–1125.

[19] Y. Zheng, R. Lu, Y. Guan, J. Shao, and H. Zhu, "Efficient privacy-preserving similarity range query with quadsector tree in ehealthcare," *IEEE Transactions on Services Computing*, pp. 1–1, 2021, doi=10.1109/TSC.2021.3081350.

[20] 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, 2013.

[21] Y. Elmehdwi, B. K. Samanthula, and W. Jiang, "Secure k-nearest

neighbor query over encrypted data in outsourced environments," in *ICDE 2014*, 2014, pp. 664–675.

[22] Y. Guan, R. Lu, Y. Zheng, J. Shao, and G. Wei, "Toward oblivious location-based k-nearest neighbor query in smart cities," *IEEE Internet of Things Journal*, pp. 1–1, 2021, doi=10.1109/JIOT.2021.3068859.

[23] Y. Zheng, R. Lu, and J. Shao, "Achieving efficient and privacy-preserving k-nn query for outsourced ehealthcare data," *J. Medical Systems*, vol. 43, no. 5, pp. 123:1–123:13, 2019.

[24] S. Choi, G. Ghinita, H. Lim, and E. Bertino, "Secure knn query processing in untrusted cloud environments," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 11, pp. 2818–2831, 2014.

[25] B. Wang, Y. Hou, and M. Li, "Practical and secure nearest neighbor search on encrypted large-scale data," in *INFOCOM 2016*, 2016, pp. 1–9.

[26] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K. Tan, "Private queries in location based services: anonymizers are not necessary," in *SIGMOD 2008*, 2008, pp. 121–132.

[27] S. Papadopoulos, S. Bakiras, and D. Papadias, "Nearest neighbor search with strong location privacy," *Proc. VLDB Endow.*, vol. 3, no. 1, pp. 619–629, 2010.

[28] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: An efficient access method for similarity search in metric spaces," in *VLDB'97*, 1997, pp. 426–435.

[29] Y. Zheng, R. Lu, Y. Guan, J. Shao, and H. Zhu, "Towards practical and privacy-preserving multi-dimensional range query over cloud," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2021, doi=10.1109/TDSC.2021.3101120.

[30] "Forest CoverType Dataset," online: http://archive.ics.uci.edu/ml/datasets/Covertype.

Yandong Zheng (Member, IEEE) received the M.S. degree from the Department of Computer Science, Beihang University, China, in 2017, and received 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 interest includes 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 with the University of Waterloo from May 2012 to April 2013. He is currently a 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 with 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 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).

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

Yunguo Guan is a PhD student of the Faculty of Computer Science, University of New Brunswick, Canada. His research interests include applied cryptography and game theory.

Fengwei Wang (Member, IEEE) received his B.Sc. degree from Xidian University in 2016 and Ph.D. degree from Xidian University in 2021. In 2019, he was with the Faculty of Computer Science, University of New Brunswick as a visiting scholar.

Since 2021, he has been the 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.

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 with 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.

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 with 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.