Towards Efficient and Privacy-Preserving User-Defined Skyline Query over Single Cloud

Songnian Zhang, Suprio Ray, *Member, IEEE,* Rongxing Lu, *Fellow, IEEE,* Yandong Zheng, Yunguo Guan, and Jun Shao *Member, IEEE*

Abstract—Skyline queries, especially those variants that allow users to define their own query criteria, are very promising and practical techniques in multi-criteria decision making applications. Meanwhile, the growing data volume drives the service providers to outsource their data to the cloud for reaping economic benefits. However, privacy concerns compel the outsourced data to be encrypted and require to perform the skyline queries over encrypted data. To achieve the privacy-preserving skyline queries, many schemes were proposed in the literature. However, those existing solutions cannot fully support the user-defined query criteria in skyline queries, and most of them employ a two-server model to support skyline queries over ciphertexts, which needs multi-round communications between the deployed two servers. In this paper, we propose a privacy-preserving user-defined skyline query scheme in a single-server model, which eliminates extra communications. Specifically, we first formally define the user-defined skyline query. Then, based on the idea of converting order relations into computing the inner products of two multi-dimensional points, we design three predicate encryption schemes. Finally, we adopt these predicate encryption schemes to construct our proposed scheme. Detailed security analysis shows that these predicate encryption schemes are selectively secure, and the proposed user-defined skyline query scheme is privacy-preserving. In addition, extensive experiments are conducted, and the results show that our proposed scheme outperforms the alternative scheme by up to an order of magnitude in terms of computational costs when performing user-defined skyline queries.

Index Terms—Skyline query, Use-defined skyline, Privacy preservation, Predicate encryption, Single server model.

1 Introduction

As an important multi-criteria analysis with diverse applications in practice, skyline queries have attracted considerable interest in academic and industrial communities [1]–[7]. Traditional skyline queries need to check expensive dominating relations over the whole dataset. However, it is impractical and inefficient, since a query user may only be interested in some attributes instead of all attributes [8] and have his/her own preferences [9]. Besides, in real-world applications, query users may only focus on skyline points that fall within constrained regions [4]. Therefore, the skyline queries that allow users to select attributes, preferences, and constrained regions are more scalable and practical, thus have more promising application prospects.

An illustrative example is that users would like to obtain a set of interesting hotels around a beach from a potentially large number of hotels in lodging reservation applications, e.g., Priceline and Booking. Assume that there are three attributes for a hotel, (*price*, *distance*, *score*), where *distance* indicates the distance from the hotel to the beach, and *score* represents the service quality. In reality, some users may be interested in the *price* and *distance* attributes, thus select them to determine skyline points, while others may pay close attention to the *price* and *score* attributes. Further, for an attribute, different users may have different preferences. For example, a user may like hotels close to the beach. However, someone would have hotels that are remote from the beach to avoid being too noisy. Besides, users would like the retrieved skyline points to lie inside a specific range. For instance, a user may expect the price to be from \$300 to \$400, while someone may only have a budget from \$50 to \$100. Such user-defined skyline queries are useful in real-world applications since they not only enable skyline points to be more in line with users' demands, but also facilitate efficiency for the service provider due to avoiding searching skyline points over the whole dataset. In this paper, we consider such user-defined skyline queries, i.e., users can select attributes, choose preferences, and define constrained regions, which will be formally defined in Section 3.1.

With the rapidly growing data volume, some skyline query service providers have to outsource their data and skyline query services to an external cloud, e.g., Azure, due to performance considerations. However, it incurs serious privacy concerns for both the data owner and query users. On the one hand, the data owner is unwilling to disclose such valuable business data assets to the cloud. On the other hand, the query users may be concerned regarding whether the cloud would collect the query data to accumulate user profiles. To address these privacy issues, a promising solution is to encrypt outsourced data and query requests and perform skyline queries over ciphertexts. In recent years, there have been a plethora of research work done in the broad area of privacy-preserving skyline queries [10]–[14]. Unfortunately, they cannot be directly applied to our user-defined skyline query scenario, which requires the cloud to search skyline points over encrypted data and meet dynam-

- *S. Zhang, S. Ray, R. Lu, Y. Zheng and Y. Guan are with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB E3B 5A3, Canada (e-mail: szhang17@unb.ca, sray@unb.ca, rlu1@unb.ca, yzheng8@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).*

ically defined query criteria. Existing solutions are either inefficient or unable to deal with the skyline queries that allow users to have selections in attributes, preferences, and constrained regions over ciphertexts. Thus, achieving efficient and privacy-preserving user-defined skyline queries is still challenging.

Besides, most of the existing privacy-preserving skyline query schemes require communications between the cloud and either the query user or an additional cloud to perform skyline computations [11]–[14]. It incurs extra communication costs and limits their applications. This two-party computation is introduced since determining dominating relations over ciphertexts needs to obtain each attribute's order relation while ensuring these order relations without leaking. Thus, it is significantly challenging to achieve the privacy-preserving skyline queries only in a single cloud server and without extra communications to query users.

To solve the above challenges, in this paper, we propose an efficient and privacy-preserving user-defined skyline query scheme without an additional server. The core idea of our scheme is to convert determining order relations into computing the inner products of two points. Based on this idea, we propose predicate encryption schemes to achieve user-defined skyline queries over encrypted data in the single-server model. First, we propose a multi-dimensional point-inside-rectangle predicate encryption (MPRPE) to check whether a point falls within a constrained region. Then, we design a multi-dimensional point dominating predicate encryption (MPDPE) to determine skyline points, in which query users can select attributes of interest and define their preferences. In order to improve the efficiency in checking points inside a constrained region, we introduce R-tree and design a hyper-rectangle intersection predicate encryption (HRIPE) to retrieve the points inside a constrained region over an encrypted R-tree. Specifically, the main contributions of this paper are three folds as follows.

- First, we design three novel predicate encryption schemes to determine point-inside-rectangle, rectangle intersection, and dominating relations over ciphertexts. Among them, MPRPE can determine whether points lie inside a constrained region, while HRIPE can check whether two rectangles intersect. Although MPDPE is designed to determine dominating relations under user-defined attributes and preferences, it can be applied to other privacy-preserving skyline queries as a component.
- Second, based on these three predicate encryption, we propose our privacy-preserving user-defined skyline query scheme in the single-server model. In our scheme, the lightweight predicate encryption ensures the performance of our proposed scheme. To further improve efficiency, we integrate HRIPE to enable searching on encrypted R-tree. Since we determine dominating relations by computing inner products, our scheme naturally avoids the order relation leakage on each attribute.
- Finally, we analyze the security of these predicate encryption schemes and demonstrate that our user-defined skyline query scheme is indeed privacy-preserving. Furthermore, we conduct extensive experiments to evaluate the performance of the proposed scheme and compare it with the existing privacy-preserving skyline query scheme. The results show that our scheme is computationally efficient and outperforms the competition scheme by up to an order of magnitude in terms of computational costs when performing skyline queries.

The remainder of this paper is organized as follows. In Section 2, we introduce our system model, security model, and design goal. Then, we review the preliminaries in Section 3. After that, we present our privacy-preserving user-defined skyline query scheme in Section 4, followed by security analysis and performance evaluation in Section 5 and Section 6, respectively. Finally, we discuss some related works in Section 7 and draw our conclusion in Section 8.

2 Models and Design Goal

In this section, we formalize our system model, security model, and identify our design goal.

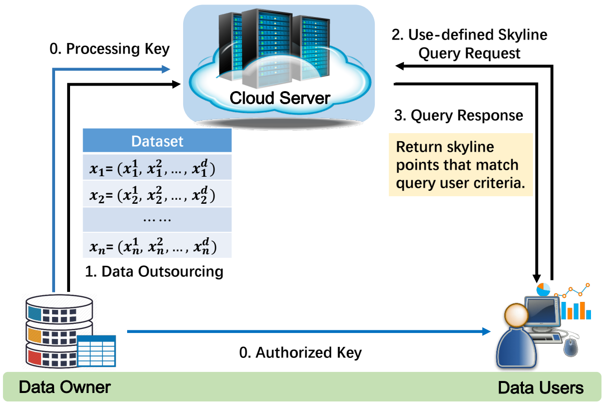

0. Processing Key
Cloud Server
2. Use-defined Skyline Query Request
3. Query Response
Return skyline points that match query user criteria.
Dataset
$x_1 = (x_1^1, x_1^2, \dots, x_1^d)$
$x_2 = (x_2^1, x_2^2, \dots, x_2^d)$
……
$x_n = (x_n^1, x_n^2, \dots, x_n^d)$
1. Data Outsourcing
0. Authorized Key
Data Owner
Data Users

Fig. 1. System model under consideration

2.1 System Model

In our system model, we consider a typical cloud-based user-defined skyline query model, which is comprised of a data owner $\mathcal{O}$, a cloud server $\mathcal{S}$, and multiple query users $\mathcal{U}=\{u_1, u_2, \cdots\}$, as shown in Fig. 1.

Data Owner $\mathcal{O}$: In our system model, the data owner $\mathcal{O}$ holds a dataset $\mathcal{X}$ with n records, and each record has d attributes. In this paper, we will use "record" and "point" interchangeably. For ease of description, we say each record is a d-dimensional vector, i.e., $\mathcal{X} = \{\mathrm{x}_i = (\mathrm{x}_i^1, \mathrm{x}_i^2, \cdots, \mathrm{x}_i^d) \mid 1 \leq i \leq n\}$. We assume that all dimensions are integers. It is reasonable since we can easily convert non-integer data into integers [15]. To make full use of these data, the data owner would like to offer the user-defined skyline query services to query users. However, since $\mathcal{O}$ may not be powerful in terms of storage and computing resources, he/she tends to outsource the dataset $\mathcal{X}$ and the query services to a cloud.

Cloud Server $\mathcal{S}$: The cloud server is considered as powerful in both storage and computing resources, which serves as a bridge between the data owner and query users. On the one hand, it provides storage services to the data owner. On the other hand, it offers reliable user-defined skyline query services to the query users. Upon receiving a query request, i.e., finding skyline points matching query user's criteria, $\mathcal{S}$ will search on the outsourced dataset $\mathcal{X}$ and return the desired skylines.

Query users $\mathcal{U}=\{u_1, u_2, \cdots\}$: In our system, query users can enjoy the user-defined skyline query services from the cloud server $\mathcal{S}$. Before launching a skyline query request, they must be authorized by the data owner $\mathcal{O}$ with an authorized key, as shown in Fig. 1. That is, only the authorized users can receive responses from the cloud server $\mathcal{S}$.

2.2 Security Model

In our security model, since the data owner $\mathcal{O}$ is an initiator of the whole system, $\mathcal{O}$ is considered to be fully trusted. However, the cloud server $\mathcal{S}$ is considered to be honest-but-curious, which means it will honestly follow the underlying scheme but may be curious to learn some private information. For query users $\mathcal{U}$, we consider the authorized users to be honest. That is, they will sincerely follow the protocol to issue the user-defined skyline queries. Since the cloud server is not fully trusted, the data owner will encrypt the dataset before outsourcing it to the cloud server. Therefore, in our model, the cloud server stores the encrypted dataset and provides the user-defined skyline query services over these encrypted data to the query users. Since the cloud server is honest-but-curious, it may attempt to obtain the private information, including the plaintexts of the encrypted dataset, query requests, and query results, based on the stored dataset and the process of user-defined skyline queries. In addition, similar to [16], [17], we assume that the cloud server would not collude with any query user. This is because our work focuses on the privacy computation technique, which is orthogonal to the current researches [18], [19] for the collusion issue. Our proposed scheme can also integrate the approaches used in [18], [19] to avoid key exposure due to the collusion issue. It is worth noting that there may be other active attacks, e.g., Denial of Service (DoS) attacks, to the system. Since we focus on privacy preservation, those attacks are beyond the scope of this paper and will be discussed in our future work.

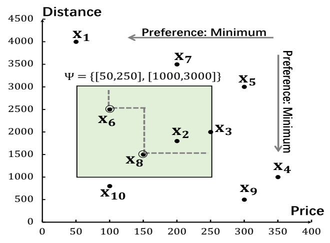
2.3 Design Goal

Under the aforementioned system model and security model, we aim to present a privacy-preserving and efficient user-defined skyline query scheme. In particular, the following objectives should be attained.

- Privacy Preservation: The fundamental requirement of the proposed scheme is privacy preservation. First, the outsourced dataset should be kept secret from the cloud server. Second, the query requests and query results should be kept secret from the cloud server.
- Efficiency: It is inevitable that the privacy requirements will incur extra computational costs. For practicality considerations, we need to minimize the computational costs of querying user-defined skyline over encrypted data.

3 Preliminaries

In this section, we first formally define the user-defined skyline queries, and then introduce the Block Nested Loop (BNL) algorithm for skyline search [1], which will be used in our proposed scheme.



Record	Price ($)	Distance (m)	Score
$\mathbf{x}_1$	50	4000	8.6
$\mathbf{x}_2$	200	1800	8.5
$\mathbf{x}_3$	250	2000	6.8
$\mathbf{x}_4$	350	1000	9.6
$\mathbf{x}_5$	300	3000	9.0
$\mathbf{x}_6$	100	2500	7.1
$\mathbf{x}_7$	200	3500	8.3
$\mathbf{x}_8$	150	1500	6.9
$\mathbf{x}_9$	300	500	8.9
$\mathbf{x}_{10}$	100	800	7.2

Fig. 2. An example of the user-defined skyline query, in which $\mathcal{D} = \{$*price*, *distance*, *score*$\}$ The desired skyline points fall within $50 \leq$ *price* $\leq 250, 1000 \leq$ *distance* ≤ 3000.

3.1 User-defined Skyline Queries

In the skyline query realm, the subspace skyline [20], [21] has attracted extensive attention due to the fact that different users may be interested in different dimensions of data. By integrating the preference selection into the subspace skyline query, Liu et al. [14] proposed the first version user-defined skyline query. Meanwhile, Dellis et al. [22] showed that the constrained subspace skyline is more efficient than the subspace skyline query since the former reduces the dataset before feeding it to the skyline operator. Besides, the constrained subspace skyline query is more practical, as query users may only be interested in the points that fall within a constrained region.

Definition 1 (Constrained Region)**.** *Given a d-dimensional dataset $\mathcal{X}$, a constrained region Ψ=$\{\psi^1, \psi^2, \cdots, \psi^d\}$ is determined by d constraints, where each constraints $\psi^i (1 \leq i \leq d)$ is expressed as a range along a dimension of the dataset, i.e, ψ^i=$[\psi^i_{min}, \psi^i_{max}]$. We define that ψ^i_{min} and ψ^i_{max} are the minimum and maximum restriction values on the i-th dimension.*

In this work, we integrate the preference selection into the constrained subspace skyline and formally define a practical version of the user-defined skyline query as follows.

Definition 2 (User-defined Dominance)**.** *Given a d-dimensional dataset $\mathcal{X}$, we suppose $\mathtt{p} = (\mathtt{p}^1, \mathtt{p}^2, \cdots, \mathtt{p}^d)$ and $\mathtt{r} = (\mathtt{r}^1, \mathtt{r}^2, \cdots, \mathtt{r}^d)$ are two d-dimensional data points in $\mathcal{X}$. Let $\mathcal{D}$ be the dimension set consisting of all the d dimensions, i.e., $|\mathcal{D}| = d$. For any user-defined dimension set $\mathcal{B}$, where $\mathcal{B} \subseteq \mathcal{D}$, $\mathtt{p}$ dominates $\mathtt{r}$, denoted as $\mathtt{p} \prec_{\mathcal{B}} \mathtt{r}$, if it satisfies: 1) $\forall\ \mathtt{p}^i, \mathtt{r}^i \in \mathcal{B}$, a user prefers the minimum (maximum) value in the dimension, that is $\mathtt{p}^i \leq \mathtt{r}^i$ ($\mathtt{p}^i \geq \mathtt{r}^i$); 2) $\exists\ \mathtt{p}^j, \mathtt{r}^j \in \mathcal{B}$ such that $\mathtt{p}^j < \mathtt{r}^j$ ($\mathtt{p}^j > \mathtt{r}^j$), where $1 \leq i, j \leq d$.*

Definition 3 (User-defined Skyline Query)**.** *Given a dataset $\mathcal{X}$, a user-defined dimension set $\mathcal{B}$ ($2 \leq |\mathcal{B}| \leq d$), and a constrained region Ψ, the user-defined skyline query returns a dataset SK=$\{\mathtt{x}_1, \mathtt{x}_2, \cdots\} \subseteq \mathcal{X}$, where $\mathtt{x}_i \in SK$ is a d-dimensional vector and satisfies the following conditions:*
1) $\mathtt{x}_i \in \Psi$, i.e., $\mathtt{x}^j_i \in \psi^j\ \ \forall\ \mathtt{x}^j_i \in \mathcal{B}$.
2) $\nexists \mathtt{r} \in \mathcal{X} \cap \Psi$ such that $\mathtt{r} \prec_{\mathcal{B}} \mathtt{x}_i$.

A simple user-defined skyline query example is shown in Fig. 2. In the example, $\mathcal{X} = \{\mathtt{x}_1, \mathtt{x}_2, \cdots, \mathtt{x}_{10}\}$, $\mathcal{B} = \{$*price*, *distance*$\}$, and $\Psi = \{[50, 250], [1000, 3000]\}$. If the query user prefers the minimum value for both selected attributes, we have *SK*=$\{\mathtt{x}_6, \mathtt{x}_8\}$.

3.2 Block Nested Loop

The Block Nested Loop (BNL) algorithm was proposed in [1] to compute skylines. BNL builds on the concept by scanning the input data, e.g., $\mathcal{X}$, and keeping a list of candidate skyline points in a window $\mathcal{W}$. The first data point is directly inserted into the window. For each subsequent point $\mathtt{x} \in \mathcal{X}$, there are three cases:

- **Case 1:** $\exists \mathtt{r} \in \mathcal{W}$, if $\mathtt{r} \prec \mathtt{x}$, then $\mathtt{x}$ is discarded;
- **Case 2:** $\exists \mathtt{r} \in \mathcal{W}$, if $\mathtt{x} \prec \mathtt{r}$, then $\mathtt{r}$ is removed from the window;
- **Case 3:** $\forall \mathtt{r} \in \mathcal{W}$, if $\mathtt{x}$ is incomparable with $\mathtt{r}$, then $\mathtt{x}$ is inserted into $\mathcal{W}$;

where the symbol '$\prec$' indicates the dominating relation, e.g., $\mathtt{r} \prec \mathtt{x}$ means that $\mathtt{r}$ dominates $\mathtt{x}$. The advantage of BNL is its wide applicability, since it can be used for any dimensionality without indexing or sorting the input data.

In this work, we consider the user-defined skyline, which leads the input data to be dynamically selected. Since the BNL algorithm does not need to preprocess the input data, we will employ the BNL algorithm in our proposed scheme.

4 Our Proposed Scheme

In this section, we will present our privacy-preserving user-defined skyline query scheme. Before delving into the details of our proposed scheme, we first present three predicate encryption schemes: namely 1) multi-dimensional point-inside-rectangle predicate encryption (MPRPE); 2) hyper-rectangle intersection predicate encryption (HRIPE); and 3) multi-dimensional point dominating predicate encryption (MPDPE), which serve as the building blocks of our proposed scheme.

4.1 MPRPE

To achieve the privacy-preserving user-defined skyline query, one of the critical operations is to securely check whether the data record $\mathtt{x}$ lies inside a constrained region Ψ. In order to deal with this challenge, we construct a Multi-dimensional Point-inside-Rectangle Predicate Encryption (MPRPE) to perform the operation over ciphertexts.

First, we would like to introduce two basic definitions [23], which will be used in the MPRPE scheme, as shown in the following.

Definition 4 (Multi-Dimensional Lattice). *Assume the ith-dimension values can be encoded into discrete integers falling into* $[1, \mathtt{T}_i]$, *where* $\mathtt{T}_i$ *is the domain of the ith-dimension. Let* $\Delta = \{\mathtt{T}_1, \mathtt{T}_2, \cdots, \mathtt{T}_d\}$, *a* ***multi-dimensional lattice*** $\mathbb{L}_\Delta = [\mathtt{T}_1] \times [\mathtt{T}_2] \times \cdots \times [\mathtt{T}_d]$, *where* $[\mathtt{T}_i] = \{1, 2, \cdots, \mathtt{T}_i\}$ *and* $1 \leq i \leq d$.

Definition 5 (Hyper-rectangle). *A* ***hyper-rectangle*** $\mathbf{R}$ *in* $\mathbb{L}_\Delta$ *is defined as* $\mathbf{R} = (\mathtt{R}_1, \mathtt{R}_2, \cdots, \mathtt{R}_d)$, *where* $\mathtt{R}_i$ *is a sub-range in the ith-dimension, i.e.,* $\mathtt{R}_i \subseteq [\mathtt{T}_i]$, $\forall i \in [1, d]$.

Obviously, the constrained region Ψ defined in Section 3.1 is equivalent to the hyper-rectangle $\mathbf{R}$. Therefore, the first query condition in the user-defined skyline can be addressed by checking whether points fall within a hyper-rectangle. Considering privacy, we enable our MPRPE to check whether a point $\mathtt{x}$ falls within $\mathbf{R}$ without revealing the point and the hyper-rectangle. Typically, our MPRPE scheme has four algorithms $\prod_{\mathsf{MPRPE}}$={MPRPE.Setup, MPRPE.Enc, MPRPE.TokenGen, MPRPE.Query} and works as follows:

- MPRPE.Setup(Δ, w): Given $\Delta = \{\mathtt{T}_1, \mathtt{T}_2, \cdots, \mathtt{T}_d\}$ and the number of dummy dimensions w ($w \geq 2$), the setup algorithm outputs a random invertible matrix $\mathrm{M} \in \mathbb{R}^{\theta \times \theta}$ as a secret key, where $\theta = \sum_{i=1}^{d} \mathtt{T}_i + 1 + w$. In addition, the algorithm generates the d-dimensional lattice $\mathbb{L}_\Delta$.
- MPRPE.Enc($\mathtt{x}$, M, $\mathbb{L}_\Delta$): The encryption algorithm takes a d-dimensional point $\mathtt{x}$, the secret key M, and the lattice $\mathbb{L}_\Delta$ as inputs. First, this algorithm encodes the point $\mathtt{x} = (\mathtt{x}^1, \mathtt{x}^2, \cdots, \mathtt{x}^d)$ into a new θ-dimensional vector $\widetilde{\mathtt{x}}$:

$$\widetilde{\mathtt{x}} = (\mathtt{r_e}\alpha(\mathtt{x}^1), \mathtt{r_e}\alpha(\mathtt{x}^2), \cdots, \mathtt{r_e}\alpha(\mathtt{x}^d), -\mathtt{r_e}, \mathtt{r}_\mathtt{x}^1, \cdots, \mathtt{r}_\mathtt{x}^w),$$

where $\mathtt{r_e}$ and $\mathtt{r}_\mathtt{x}^p$ ($1 \leq p \leq w$) $\in \mathbb{R}^+$ are random real numbers satisfying $\mathtt{r_e} > \sum_{p=1}^{w} \mathtt{r}_\mathtt{x}^p$, and $\alpha(\mathtt{x}^i) = (\underbrace{0, 0, \cdots, 0}_{\mathtt{x}^i - 1}, 1, \underbrace{0, \cdots, 0}_{\mathtt{T}_i - \mathtt{x}^i})$ is a $\mathtt{T}_i$-dimensional vector, i.e.,

$$\alpha(\mathtt{x}^i)_{j \in [1, \mathtt{T}_i]} = \begin{cases} 1 & \text{if } j = \mathtt{x}^i \\ 0 & \text{Otherwise,} \end{cases} \tag{1}$$

Then, the algorithm outputs the ciphertext of $\mathtt{x}$, denoted as $||\mathtt{x}||$, by encrypting $\widetilde{\mathtt{x}}$ with the secret key M, i.e., $||\mathtt{x}||$ = MPRPE.Enc($\mathtt{x}$, M, $\mathbb{L}_\Delta$) = $\widetilde{\mathtt{x}}$ M.

- MPRPE.TokenGen($\mathbf{R}$, M, $\mathbb{L}_\Delta$): On input of a hyper-rectangle $\mathbf{R} = (\mathtt{R}_1, \mathtt{R}_2, \cdots, \mathtt{R}_d)$, the secret key M, and the lattice $\mathbb{L}_\Delta$, the token generation algorithm can output a query token $\langle \mathtt{q} \rangle$. First, the algorithm encodes $\mathbf{R}$ into a query vector $\widetilde{\mathtt{q}}$ that has θ dimensions:

$$\widetilde{\mathtt{q}} = (\mathtt{r_t}\beta(\mathtt{R}_1), \mathtt{r_t}\beta(\mathtt{R}_2), \cdots, \mathtt{r_t}\beta(\mathtt{R}_d), d \cdot \mathtt{r_t}, \mathtt{r}_\mathtt{q}^1, \cdots, \mathtt{r}_\mathtt{q}^w),$$

where $\mathtt{r_t}$, $\mathtt{r}_\mathtt{q}^p$ ($1 \leq p \leq w$) $\in \mathbb{R}^+$ are random real numbers satisfying $\mathtt{r_t} > \sum_{p=1}^{w} \mathtt{r}_\mathtt{q}^p$, and $\beta(\mathtt{R}_i) = (0, 0, \cdots, 0, \underbrace{1, 1, \cdots, 1}_{\mathtt{R}_i}, 0, \cdots, 0)$, i.e.,

$$\beta(\mathtt{R}_i)_{j \in [1, \mathtt{T}_i]} = \begin{cases} 1 & \text{if } j \in \mathtt{R}_i \\ 0 & \text{Otherwise,} \end{cases} \tag{2}$$

Then, the algorithm outputs a query token $\langle \mathtt{q} \rangle$ = MPRPE.TokenGen($\mathbf{R}$, M, $\mathbb{L}_\Delta$) = $\widetilde{\mathtt{q}}(\mathrm{M}^{-1})^T$.

- MPRPE.Query($||\mathtt{x}||$, $\langle \mathtt{q} \rangle$): On input of an encrypted point $||\mathtt{x}||$ and a query token $\langle \mathtt{q} \rangle$, this algorithm calculates the inner product between $||\mathtt{x}||$ and $\langle \mathtt{q} \rangle$ and outputs *true iff* $||\mathtt{x}|| \circ \langle \mathtt{q} \rangle > 0$, and $false$ otherwise.

Correctness. We say the MPRPE scheme is correct if one can determine whether $\mathtt{x} \in \mathbf{R}$ by observing the result of the MPRPE.Query algorithm, namely, if the result is $true$, $\mathtt{x} \in \mathbf{R}$, otherwise $\mathtt{x} \notin \mathbf{R}$. We show the proof as follows.

Proof. Given an encrypted point $||\mathtt{x}||$ and a query token $\langle \mathtt{q} \rangle$, the MPRPE.Query algorithm runs $||\mathtt{x}|| \circ \langle \mathtt{q} \rangle$:

$$\begin{aligned} ||\mathtt{x}|| \circ \langle \mathtt{q} \rangle &= ||\mathtt{x}|| \cdot \langle \mathtt{q} \rangle^T = \widetilde{\mathtt{x}}\mathrm{M}\mathrm{M}^{-1}\widetilde{\mathtt{q}}^T = \widetilde{\mathtt{x}} \cdot \widetilde{\mathtt{q}}^T \\ &= \mathtt{r_e}\mathtt{r_t}(\sum_{i=1}^{d}(\alpha(\mathtt{x}^i)\beta(\mathtt{R}_i)^T) - d) + \sum_{p=1}^{w} \mathtt{r}_\mathtt{x}^p \mathtt{r}_\mathtt{q}^p. \end{aligned} \tag{3}$$

From Eq. (1) and Eq. (2), we can see that, for the i-th dimension, if $\alpha(\mathtt{x}^i)\beta(\mathtt{R}_i)^T = 1$, $\mathtt{x}^i \in \mathtt{R}_i$, and $\mathtt{x}^i \notin \mathtt{R}_i$ otherwise. Therefore, only when all dimensions satisfy $\alpha(\mathtt{x}^i)\beta(\mathtt{R}_i)^T = 1$, i.e., $\sum_{i=1}^{d}(\alpha(\mathtt{x}^i)\beta(\mathtt{R}_i)^T) = d$, we have $\mathtt{x} \in \mathbf{R}$. Thus $||\mathtt{x}|| \circ \langle \mathtt{q} \rangle = \sum_{p=1}^{w} \mathtt{r}_\mathtt{x}^p \mathtt{r}_\mathtt{q}^p > 0 \Rightarrow \mathtt{x} \in \mathbf{R}$. If $\mathtt{x} \notin \mathbf{R}$, i.e., $\exists \mathtt{x}^i \notin \mathtt{R}_i$, it indicates $\sum_{i=1}^{d}(\alpha(\mathtt{x}^i)\beta(\mathtt{R}_i)^T) \le d-1 \Rightarrow ||\mathtt{x}|| \circ \langle \mathtt{q} \rangle < 0$ due to $\mathtt{r_e r_t} > \sum_{p=1}^{w} \mathtt{r}_{\mathtt{x}}^p \mathtt{r}_{\mathtt{q}}^p$. Thus, $||\mathtt{x}|| \circ \langle \mathtt{q} \rangle > 0 \Leftrightarrow \mathtt{x} \in \mathbf{R}$. □

Since the MPRPE scheme only reveals the inner product's result of all dimensions, it can naturally preserve the single dimensional privacy, i.e., whether $\mathtt{x^i} \in \mathtt{R_i}$ when $\mathtt{x} \notin \mathbf{R}$. Besides, this scheme preserves the privacy about how many dimensions are inside or outside the hyper-rectangle $\mathbf{R}$ when $\mathtt{x} \notin \mathbf{R}$.

4.2 HRIPE

By applying MPRPE scheme, we can build a basic solution to the first query condition of the user-defined skyline query in a privacy-preserving manner. However, it is not efficient if we directly use the MPRPE scheme, since the operator needs to search on all the encrypted points one by one with the given query token. In order to improve the search efficiency, we exploit the multi-dimensional tree based structures to index the data records. Since the space-oriented indexes, such as k-d tree and Quadtree, may leak the single dimensional privacy, we employ the data-oriented indexes, specifically R-tree, which can preserve the single dimensional privacy [24]. This is because the space-oriented indexes need to make search decisions on each dimension separately, while the data-oriented indexes can take the space as a whole to make search decisions. However, it incurs a challenge that we need to check whether two hyper-rectangles are intersected on the non-leaf nodes of R-tree without leaking the underlying values. To tackle it, we construct a Hyper-Rectangle Intersection Predicate Encryption (HRIPE) scheme. Motivated by [24], we convert the hyper-rectangle intersection operation into a $2d$-dimensional point-inside-rectangle operation and then extend the MPRPE scheme to build our HRIPE scheme.

Assume that there are two hyper-rectangles, one is query rectangle $\mathbf{R} = \{(\mathtt{R}_1, \mathtt{R}_2, \cdots, \mathtt{R}_d) | \mathtt{R}_i = [\mathtt{R}_{il}, \mathtt{R}_{iu}]\}$, the other is the minimum bounding rectangle (MBR) of R-tree, denoted as $\mathbf{R}' = \{(\mathtt{R}'_1, \mathtt{R}'_2, \cdots, \mathtt{R}'_d) | \mathtt{R}'_i = [\mathtt{R}'_{il}, \mathtt{R}'_{iu}]\}$. For the i-th dimension, the intersection relation between two ranges ($\mathtt{R}_i$ and $\mathtt{R}'_i$) can be transformed into checking whether a 2-dimensional point lies inside a rectangle, i.e.,

$$\mathtt{R}_i \cap \mathtt{R}'_i \neq \varnothing \Leftrightarrow \begin{cases} \mathtt{R}'_{il} \in [1, \mathtt{R}_{iu}] \\ \mathtt{R}'_{iu} \in [\mathtt{R}_{il}, \mathtt{T}_i]. \end{cases}$$

Since $\mathtt{R} \cap \mathtt{R}' \neq \varnothing \Leftrightarrow$ all dimensions hold $\mathtt{R}_i \cap \mathtt{R}'_i \neq \varnothing$, for i=$1, 2, \cdots, d$, we can transform the hyper-rectangle intersection problem into the point-inside-rectangle problem and build our HRIPE scheme based on the MPRPE scheme. Similarly, the HRIPE scheme also involves four algorithms $\prod_{\mathsf{HRIPE}}$={HRIPE.Setup, HRIPE.Enc, HRIPE.TokenGen, HRIPE.Query} and is shown as follows.

- HRIPE.Setup(Δ, w): Given $\Delta = \{\mathtt{T}_1, \mathtt{T}_2, \cdots, \mathtt{T}_d\}$ and the number of dummy dimensions w ($w \ge 2$), the setup algorithm outputs a random invertible matrix $\mathrm{M} \in \mathbb{R}^{\theta \times \theta}$ as a secret key, where $\theta = 2\sum_{i=1}^{d} \mathtt{T}_i + 1 + w$, and a d-dimensional lattice $\mathbb{L}_\Delta$.
- HRIPE.Enc($\mathbf{R}'$, M, $\mathbb{L}_\Delta$): On input of a hyper-rectangle $\mathbf{R}'$=$\{(\mathtt{R}'_1, \mathtt{R}'_2, \cdots, \mathtt{R}'_d) | \mathtt{R}'_i = [\mathtt{R}'_{il}, \mathtt{R}'_{iu}]\}$, the secret key M, and the lattice $\mathbb{L}_\Delta$, the encryption algorithm first encodes $\mathbf{R}'$ into a new θ-dimensional vector:

$$\widetilde{\mathbf{R}'} = (\mathtt{r_e}\gamma(\mathtt{R}'_1), \mathtt{r_e}\gamma(\mathtt{R}'_2), \cdots, \mathtt{r_e}\gamma(\mathtt{R}'_d), -\mathtt{r_e}, \mathtt{r}_{\mathtt{R}}^1, \cdots, \mathtt{r}_{\mathtt{R}}^w),$$

where $\mathtt{r_e}$ and $\mathtt{r}_{\mathtt{R}}^p$ $(1 \le p \le w) \in \mathbb{R}^+$ are random real numbers satisfying $\mathtt{r_e} > \sum_{p=1}^{w} \mathtt{r}_{\mathtt{R}}^p$, and $\gamma(\mathtt{R}'_i) = (\underbrace{0, \cdots, 0}_{\mathtt{R}'_{il}-1}, 1, \underbrace{0, \cdots, 0}_{\mathtt{T}_i - \mathtt{R}'_{il}}, \underbrace{0, \cdots, 0}_{\mathtt{R}'_{iu}-1}, 1, \underbrace{0, \cdots, 0}_{\mathtt{T}_i - \mathtt{R}'_{iu}})$ is a $2\mathtt{T}_i$-dimensional vector, i.e.,

$$\gamma(\mathtt{R}'_i)_{j \in [1, 2\mathtt{T}_i]} = \begin{cases} 1 & \text{if } j = \mathtt{R}'_{il} \text{ or } j = \mathtt{R}'_{iu} + \mathtt{T}_i \\ 0 & \text{Otherwise,} \end{cases} \tag{4}$$

Then, the algorithm outputs the ciphertext of $\mathbf{R}'$, denoted as $||\mathbf{R}'||$, where $||\mathbf{R}'|| = \mathsf{HRIPE.Enc}(\mathbf{R}', \mathrm{M}, \mathbb{L}_\Delta) = \widetilde{\mathbf{R}'}\,\mathrm{M}$.
- HRIPE.TokenGen($\mathbf{R}$, M, $\mathbb{L}_\Delta$): Taking a query hyper-rectangle $\mathbf{R} = \{(\mathtt{R}_1, \mathtt{R}_2, \cdots, \mathtt{R}_d) \mid \mathtt{R}_i = [\mathtt{R}_{il}, \mathtt{R}_{iu}]\}$, the secret key M, and the lattice $\mathbb{L}_\Delta$ as inputs, the token generation algorithm can output a query token $\langle \mathtt{q_h} \rangle$. First, the algorithm encodes $\mathbf{R}$ into a θ-dimensional query vector:

$$\widetilde{\mathtt{q_h}} = (\mathtt{r_t}\delta(\mathtt{R}_1), \mathtt{r_t}\delta(\mathtt{R}_2), \cdots, \mathtt{r_t}\delta(\mathtt{R}_d), 2d \cdot \mathtt{r_t}, \mathtt{r}_{\mathtt{q}}^1, \cdots, \mathtt{r}_{\mathtt{q}}^w),$$

where $\mathtt{r_t}$ and $\mathtt{r}_{\mathtt{q}}^p$ $(1 \le p \le w) \in \mathbb{R}^+$ are random real numbers satisfying $\mathtt{r_t} > \sum_{p=1}^{w} \mathtt{r}_{\mathtt{q}}^p$, and $\delta(\mathtt{R}_i) = (\underbrace{1, 1, \cdots, 1}_{\mathtt{R}_{iu}}, 0, 0, \cdots, 0, \underbrace{1, 1, \cdots, 1}_{\mathtt{T}_i - \mathtt{R}_{il} + 1})$ is a $2\mathtt{T}_i$-dimensional vector, i.e.,

$$\delta(\mathtt{R}_i)_{j \in [1, 2\mathtt{T}_i]} = \begin{cases} 1 & \text{if } j \in [1, \mathtt{R}_{iu}] \\ & \quad \text{or } j \in [\mathtt{R}_{il} + \mathtt{T}_i, 2\mathtt{T}_i] \\ 0 & \text{Otherwise} \end{cases} \tag{5}$$

Then, the algorithm outputs a query token $\langle \mathtt{q_h} \rangle$ = HRIPE.TokenGen($\mathbf{R}$, M, $\mathbb{L}_\Delta$) = $\widetilde{\mathtt{q_h}}(\mathrm{M}^{-1})^T$.
- HRIPE.Query($||\mathbf{R}'||$, $\langle \mathtt{q_h} \rangle$): On input of an encrypted hyper-rectangle $||\mathbf{R}'||$ and a query token $\langle \mathtt{q_h} \rangle$, this algorithm calculates the inner product between $||\mathbf{R}'||$ and $\langle \mathtt{q_h} \rangle$ and outputs *true iff* $||\mathbf{R}'|| \circ \langle \mathtt{q_h} \rangle > 0$, and *false* otherwise.

Correctness. We say the HRIPE scheme is correct if one can determine whether two hyper-rectangles are intersected by observing the result of the HRIPE.Query algorithm, namely, if the result is *true*, $\mathtt{R} \cap \mathtt{R}' \neq \varnothing$, otherwise $\mathtt{R} \cap \mathtt{R}' = \varnothing$. To save space, we do not show the detailed proof since it is similar to the MPRPE scheme.

4.3 MPDPE

To provide the user-defined skyline query services, the other essential operation is to check whether a point is dominated by another point. Although it is simple to determine the dominating relations over plaintexts, checking the dominating relations over encrypted data is still challenging. To address it, we propose a Multi-dimensional Point Dominating Predicate Encryption (MPDPE) scheme to check dominance over encrypted data points. Considering the second query condition in the user-defined skyline query (Definition 3), we construct a query vector $\mathtt{q_s} = (\mathtt{q}_{\mathtt{s}}^1, \mathtt{q}_{\mathtt{s}}^2, \cdots, \mathtt{q}_{\mathtt{s}}^d)$ to allow query users to dynamically determine which dimensions are selected and the corresponding preferences, where

$$\mathtt{q}_{\mathtt{s}}^{i \in [1,d]} = \begin{cases} 1 & \text{if } i\text{th-dimension is chosen and prefer } \mathit{max} \\ -1 & \text{if } i\text{th-dimension is chosen and prefer } \mathit{min} \\ 0 & \text{if } i\text{th-dimension is not chosen.} \end{cases}$$

With the constructed query vector $\mathtt{q_s}$, we present our MPDPE scheme, $\prod_{\mathsf{MPDPE}}$={$\mathsf{MPDPE.Setup}$, $\mathsf{MPDPE.Enc}$, $\mathsf{MPDPE.TokenGen}$, $\mathsf{MPDPE.Query}$}, as follows.

• $\mathsf{MPDPE.Setup}(\Delta, w)$: Given $\Delta = \{\mathtt{T}_1, \mathtt{T}_2, \cdots, \mathtt{T}_d\}$ and the number of dummy dimensions w ($w \geq 2$), the setup algorithm outputs two random invertible matrices $\mathrm{M} \in \mathbb{R}^{\theta\times\theta}$ and $\widetilde{\mathrm{M}} \in \mathbb{R}^{\theta\times\theta}$ as secret keys, where $\theta = \sum_{i=1}^{d} \mathtt{T}_i + 3d + 1 + w$, and a d-dimensional lattice $\mathbb{L}_\Delta$.

• $\mathsf{MPDPE.Enc}(\mathtt{x}, \mathrm{M}, \widetilde{\mathrm{M}}, \mathbb{L}_\Delta)$: The encryption algorithm takes a d-dimensional point $\mathtt{x}$, the secret keys $\{\mathrm{M}, \widetilde{\mathrm{M}}\}$, and the lattice $\mathbb{L}_\Delta$ as inputs. First, the algorithm encodes the point $\mathtt{x} = (\mathtt{x}^1, \mathtt{x}^2, \cdots, \mathtt{x}^d)$ into two new θ-dimensional vectors: $\widehat{\mathtt{x}}$ and $\widetilde{\mathtt{x}}$.

$$\begin{aligned}\widehat{\mathtt{x}} =&(\mathtt{r_e}\alpha(\mathtt{x}^1), \mathtt{r_e}\alpha(\mathtt{x}^2), \cdots, \mathtt{r_e}\alpha(\mathtt{x}^d),\\ &\mathtt{r_{e'}}\zeta(\mathtt{x}^1), \mathtt{r_{e'}}\zeta(\mathtt{x}^2), \cdots, \mathtt{r_{e'}}\zeta(\mathtt{x}^d), -\mathtt{r_e}, -\mathtt{r}_\mathtt{x}^1, \cdots, -\mathtt{r}_\mathtt{x}^w),\end{aligned}$$

where $\mathtt{r_e}$, $\mathtt{r_{e'}}$, and $\mathtt{r}_\mathtt{x}^p$ $(1 \leq p \leq w) \in \mathbb{R}^+$ are random real numbers satisfying $\mathtt{r_e} > \mathtt{r_{e'}} \cdot \sum_{i=1}^{d} \mathtt{T}_i{}^2$ and $\mathtt{r_{e'}} > \sum_{p=1}^{w} \mathtt{r}_\mathtt{x}^p$. Meanwhile, $\alpha(\mathtt{x}^i)$ is defined as Eq. (1), and $\zeta(\mathtt{x}^i)$ is a 3-dimensional vector, i.e., $\zeta(\mathtt{x}^i) = (\mathtt{x}^{i^2}, -2\mathtt{x}^i, 1)$. Note that $\mathtt{x}^{i^2} = \mathtt{x}^i \cdot \mathtt{x}^i$.

$$\begin{aligned}\widetilde{\mathtt{x}} =&(\mathtt{r_s}\eta(\mathtt{x}^1), \mathtt{r_s}\eta(\mathtt{x}^2), \cdots, \mathtt{r_s}\eta(\mathtt{x}^d),\\ &\mathtt{r_s}\xi(\mathtt{x}^1), \mathtt{r_s}\xi(\mathtt{x}^2), \cdots, \mathtt{r_s}\xi(\mathtt{x}^d), -\mathtt{r_z}, \underbrace{\mathtt{r_z}, \cdots, \mathtt{r_z}}_{w}),\end{aligned}$$

where $\mathtt{r_s}$ and $\mathtt{r_z} \in \mathbb{R}^+$ are random real numbers, $\eta(\mathtt{x}^i) = (\underbrace{0, 0, \cdots, 0}_{\mathtt{x}^i-1}, \underbrace{1, 1, \cdots, 1}_{\mathtt{T}_i-\mathtt{x}^i+1})$ is a $\mathtt{T}_i$-dimensional vector, i.e.,

$$\eta(\mathtt{x}^i)_{j\in[1,\mathtt{T}_i]} = \begin{cases} 1 & \text{if } j \in [\mathtt{x}^i, \mathtt{T}_i] \\ 0 & \text{Otherwise,} \end{cases} \tag{6}$$

and $\xi(\mathtt{x}^i) = (1, \mathtt{x}^i, \mathtt{x}^{i^2})$. Then, the algorithm outputs two ciphertexts of $\mathtt{x}$, denoted as $||\mathtt{x}||_\mathrm{L}$ (left ciphertext) and $||\mathtt{x}||_\mathrm{R}$ (right ciphertext), by respectively encrypting $\widehat{\mathtt{x}}$ and $\widetilde{\mathtt{x}}$ with the secret keys M and $\widetilde{\mathrm{M}}$, i.e., ($||\mathtt{x}||_\mathrm{L}$, $||\mathtt{x}||_\mathrm{R}$) = $\mathsf{MPDPE.Enc}(\mathtt{x}, \mathrm{M}, \widetilde{\mathrm{M}}, \mathbb{L}_\Delta)$. Specifically, $||\mathtt{x}||_\mathrm{L} = \widehat{\mathtt{x}}\,\mathrm{M}$ and $||\mathtt{x}||_\mathrm{R} = \widetilde{\mathtt{x}}\,\widetilde{\mathrm{M}}$.

• $\mathsf{MPDPE.TokenGen}(\mathtt{q_s}, \mathrm{M}, \widetilde{\mathrm{M}}, \mathbb{L}_\Delta)$: The token generation algorithm takes a preference query vector $\mathtt{q_s}$, the secret keys $\{\widetilde{\mathrm{M}}, \mathrm{M}\}$, and the lattice $\mathbb{L}_\Delta$ as inputs. First, the algorithm encodes the query vector $\mathtt{q_s}$ into a $\theta \times \theta$ query matrix Q as follows.

Step-1. Generate a $(w+1) \times \theta$ random matrix RM that satisfies:

$$\mathrm{RM}_{i,j} = \begin{cases} \mathtt{r}_{i,j} & i \in [2, w+1], j \in [1, \theta]; \\ \sum\limits_{i=2}^{w+1} \mathtt{r}_{i,j} & i = 1, j \in [1, \theta], \end{cases} \tag{7}$$

where $\mathtt{r}_{i,j} \in \mathbb{R}$ are random numbers.

Step-2. Generate a $(\theta - w - 1) \times (w+1)$ padding matrix P that is constructed by

$$\mathrm{P}_{i,j} = \begin{cases} k \cdot \mathtt{r_p} & \text{if } i = \theta - 3d - w - 1, j = 1; \\ \mathtt{r}_\mathtt{q}^j & \text{if } i = \theta - 3d - w - 1, j \in [2, w+1]; \\ 0 & \text{Otherwise,} \end{cases}$$

where $k = \sum_{i=1}^{d} |\mathtt{q}_\mathtt{s}^i|$, and $\mathtt{r_p}$, $\mathtt{r}_\mathtt{q}^j \in \mathbb{R}^+$ are random real numbers satisfying $\mathtt{r_p} > \sum_{j=2}^{w+1} \mathtt{r}_\mathtt{q}^j$.

Step-3. Extend the vector $\mathtt{q_s}$=$(\mathtt{q}_\mathtt{s}^1, \mathtt{q}_\mathtt{s}^2, \cdots, \mathtt{q}_\mathtt{s}^d)$ to a set of mask matrices $\mathsf{Q_s}$=$(\mathsf{Q}_\mathsf{s}^1, \mathsf{Q}_\mathsf{s}^2, \cdots, \mathsf{Q}_\mathsf{s}^d)$. For any matrix Q_s^i $(1 \leq i \leq d)$, it has the size of $\mathtt{T}_i \times \mathtt{T}_i$ and can be constructed as follows:

$$\mathsf{Q}_\mathsf{s}^i = \begin{cases} \mathtt{I} & \text{if } \mathtt{q}_\mathtt{s}^i = 1 \\ \mathtt{V}^i & \text{if } \mathtt{q}_\mathtt{s}^i = -1 \\ \mathsf{O} & \text{if } \mathtt{q}_\mathtt{s}^i = 0, \end{cases}$$

where $\mathtt{I}$ is an identity matrix, O is a zero matrix, and $\mathtt{V}^i$ has the size of $\mathtt{T}_i \times \mathtt{T}_i$ and satisfies:

$$\mathtt{V}^i_{m,n} = \begin{cases} -1 & \text{if } m \in [1, \mathtt{T}_i - 1], n = m + 1 \\ 1 & \text{if } m = \mathtt{T}_i, n \in [1, \mathtt{T}_i] \\ 0 & \text{Otherwise,} \end{cases}$$

Step-4. Extend the vector $\mathtt{q_s}$=$(\mathtt{q}_\mathtt{s}^1, \mathtt{q}_\mathtt{s}^2, \cdots, \mathtt{q}_\mathtt{s}^d)$ to a set of mask matrices $\mathsf{Q_{s'}}$=$(\mathsf{Q}_{\mathsf{s'}}^1, \mathsf{Q}_{\mathsf{s'}}^2, \cdots, \mathsf{Q}_{\mathsf{s'}}^d)$. For any matrix $\mathsf{Q}_{\mathsf{s'}}^i$ $(1 \leq i \leq d)$, it has the size of 3×3 and can be constructed as follows:

$$\mathsf{Q}_{\mathsf{s'}}^i = \begin{cases} \mathtt{I} & \text{if } \mathtt{q}_\mathtt{s}^i = 1 \text{ or } -1 \\ \mathsf{O} & \text{if } \mathtt{q}_\mathtt{s}^i = 0, \end{cases}$$

Step-5. Construct the $\theta \times \theta$ query matrix Q with the aforementioned RM, P, $\mathsf{Q_s}$, and $\mathsf{Q_{s'}}$:

$$\mathsf{Q} = \left[\begin{array}{cccccc|c} \mathtt{r_p}\mathsf{Q}_\mathsf{s}^1 \cdots & \mathsf{O} & \mathsf{O} & \cdots & \mathsf{O} & & \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \\ \mathsf{O} & \cdots & \mathtt{r_p}\mathsf{Q}_\mathsf{s}^\mathsf{d} & \mathsf{O} & \cdots & \mathsf{O} & \\ \mathsf{O} & \cdots & \mathsf{O} & \mathtt{r_{p'}}\mathsf{Q}_{\mathsf{s'}}^1 & \cdots & \mathsf{O} & \mathrm{P} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \\ \mathsf{O} & \cdots & \mathsf{O} & \mathsf{O} & \cdots & \mathtt{r_{p'}}\mathsf{Q}_{\mathsf{s'}}^\mathsf{d} & \\ \hline \multicolumn{6}{c|}{\mathrm{RM}_{i,[1,\theta-w-1]}} & \mathrm{RM}_{i,[\theta-w,\theta]} \end{array}\right] \tag{8}$$

where $\mathtt{r_p}$ is a random number generated in *step-2*, and $\mathtt{r_{p'}}$ is a random real number satisfying $\mathtt{r_p} > \mathtt{r_{p'}} \cdot \sum_{i=1}^{d} \mathtt{T}_i{}^2$ and $\mathtt{r_{p'}} > \sum_{j=2}^{w+1} \mathtt{r}_\mathtt{q}^j$. Then, the token generation algorithm outputs a query token $\langle \mathtt{q_s} \rangle$ = $\mathsf{MPDPE.TokenGen}(\mathtt{q_s}, \mathrm{M}, \widetilde{\mathrm{M}}, \mathbb{L}_\Delta) = \widetilde{\mathrm{M}}^{-1}\mathsf{Q}(\mathrm{M}^{-1})^T$.

• $\mathsf{MPDPE.Query}(||\mathtt{x_t}||_\mathrm{L}, ||\mathtt{x_s}||_\mathrm{R}, \langle \mathtt{q_s} \rangle)$: On input of $\mathtt{x_t}$'s left ciphertext $||\mathtt{x_t}||_\mathrm{L}$, $\mathtt{x_s}$'s right ciphertext $||\mathtt{x_s}||_\mathrm{R}$ $(t \neq s)$, and a query token $\langle \mathtt{q_s} \rangle$, the query algorithm outputs *true iff* $||\mathtt{x_t}||_\mathrm{L} \circ (||\mathtt{x_s}||_\mathrm{R} \cdot \langle \mathtt{q_s} \rangle) > 0$, and $false$ otherwise.

Correctness. We say the MPDPE scheme is correct if one can determine whether $\mathtt{x_t} \prec_\mathcal{B} \mathtt{x_s}$ by observing the result of the $\mathsf{MPDPE.Query}$ algorithm: if the result is $true$, $\mathtt{x_t} \prec_\mathcal{B} \mathtt{x_s}$, otherwise $\mathtt{x_t} \not\prec_\mathcal{B} \mathtt{x_s}$. We show the proof as follows.

Proof. Given $||\mathtt{x_t}||_\mathrm{L}$, $||\mathtt{x_s}||_\mathrm{R}$ and $\langle \mathtt{q_s} \rangle$, the $\mathsf{MPDPE.Query}$ algorithm runs $||\mathtt{x_t}||_\mathrm{L} \circ (||\mathtt{x_s}||_\mathrm{R} \cdot \langle \mathtt{q_s} \rangle)$:

$$\begin{aligned}||\mathtt{x_t}||_\mathrm{L} \circ (||\mathtt{x_s}||_\mathrm{R} \cdot \langle \mathtt{q_s} \rangle) &= (\widehat{\mathtt{x_t}}\mathrm{M}) \circ (\widetilde{\mathtt{x_s}}\widetilde{\mathrm{M}}\widetilde{\mathrm{M}}^{-1}\mathsf{Q}(\mathrm{M}^{-1})^T) \\ &= \widehat{\mathtt{x_t}}\mathrm{M}\mathrm{M}^{-1}(\widetilde{\mathtt{x_s}}\mathsf{Q})^T = \widehat{\mathtt{x_t}}(\widetilde{\mathtt{x_s}}\mathsf{Q})^T\end{aligned} \tag{9}$$

Let $\widetilde{\mathtt{x_s}}\mathrm{Q} = \widetilde{\mathtt{x_{sq}}} = (\varepsilon^1, \varepsilon^2, \cdots, \varepsilon^d, \tau^1, \tau^2, \cdots, \tau^d, \rho_0, \rho_1, \cdots, \rho_w)$. Since $(-\mathtt{r_z}, \underbrace{\mathtt{r_z}, \cdots, \mathtt{r_z}}_{w}) \cdot \mathrm{RM} = (0, 0, ..., 0)$, we have:

$$\begin{cases} \varepsilon^\mu = \mathtt{r_s r_p} \cdot \eta(\mathtt{x_s^\mu}) \cdot \mathrm{Q_s^\mu} \\ \tau^\mu = \mathtt{r_s r_{p'}} \cdot \xi(\mathtt{x_s^\mu}) \cdot \mathrm{Q_{s'}^\mu} \\ \rho_0 = \mathtt{r_s r_p} \cdot k \cdot \eta(\mathtt{x_s^d})_{\mathtt{T}_d} \\ \rho_1 = \mathtt{r_s r_q^2} \cdot \eta(\mathtt{x_s^d})_{\mathtt{T}_d} \\ \vdots \qquad \vdots \\ \rho_w = \mathtt{r_s r_q^{w+1}} \cdot \eta(\mathtt{x_s^d})_{\mathtt{T}_d}, \end{cases} \tag{10}$$

where $1 \leq \mu \leq d$. Since $\mathtt{x_s^\mu} \leq \mathtt{T}_\mu$ (see Definition 4), we always have $\eta(\mathtt{x_s^\mu})_{\mathtt{T}_\mu} = 1$ according to Eq. (6). Thus, $\eta(\mathtt{x_s^d})_{\mathtt{T}_d} = 1$. Consequently, from Eq. (10), we have:

$$\begin{aligned} \widetilde{\mathtt{x_{sq}}} &= (\varepsilon^1, \varepsilon^2, \cdots, \varepsilon^d, \tau^1, \tau^2, \cdots, \tau^d, \rho_0, \rho_1, \cdots, \rho_w) \\ &= \mathtt{r_s}(\mathtt{r_p} \cdot \eta(\mathtt{x_s^1}) \cdot \mathrm{Q_s^1}, \cdots, \mathtt{r_p} \cdot \eta(\mathtt{x_s^d}) \cdot \mathrm{Q_s^d}, \mathtt{r_{p'}} \cdot \xi(\mathtt{x_s^1}) \cdot \mathrm{Q_{s'}^1}, \\ &\quad \cdots, \mathtt{r_{p'}} \cdot \xi(\mathtt{x_s^d}) \cdot \mathrm{Q_{s'}^d}, k \cdot \mathtt{r_p}, \mathtt{r_q^2}, \cdots, \mathtt{r_q^{w+1}}), \end{aligned}$$

where

$$\eta(\mathtt{x_s^\mu}) \cdot \mathrm{Q_s^\mu} = \begin{cases} \eta(\mathtt{x_s^\mu}) & \text{if } \mathtt{q_s^\mu} = 1 \\ \eta(\mathtt{x_s^\mu}) \cdot \mathtt{V}^\mu & \text{if } \mathtt{q_s^\mu} = -1 \\ \mathsf{O} & \text{if } \mathtt{q_s^\mu} = 0, \end{cases} \tag{11}$$

and

$$\xi(\mathtt{x_s^\mu}) \cdot \mathrm{Q_{s'}^\mu} = \begin{cases} \xi(\mathtt{x_s^\mu}) & \text{if } \mathtt{q_s^\mu} = 1 \text{ or } -1 \\ \mathsf{O} & \text{if } \mathtt{q_s^\mu} = 0. \end{cases} \tag{12}$$

Furthermore, we can obtain:

$$\begin{aligned} \eta(\mathtt{x_s^\mu}) \cdot \mathtt{V}^\mu &= (\eta(\mathtt{x_s^\mu})_{\mathtt{T}_\mu}, \eta(\mathtt{x_s^\mu})_{\mathtt{T}_\mu} - \eta(\mathtt{x_s^\mu})_1, \\ &\quad \eta(\mathtt{x_s^\mu})_{\mathtt{T}_\mu} - \eta(\mathtt{x_s^\mu})_2, \cdots, \eta(\mathtt{x_s^\mu})_{\mathtt{T}_\mu} - \eta(\mathtt{x_s^\mu})_{(\mathtt{T}_\mu - 1)}) \\ &= (1, 1 - \eta(\mathtt{x_s^\mu})_1, 1 - \eta(\mathtt{x_s^\mu})_2, \cdots, 1 - \eta(\mathtt{x_s^\mu})_{(\mathtt{T}_\mu - 1)}). \end{aligned}$$

Let $\eta'(\mathtt{x_s^\mu}) = \eta(\mathtt{x_s^\mu}) \cdot \mathtt{V}^\mu$, we have the following equation according to Eq. (6):

$$\eta'(\mathtt{x_s^\mu})_{j \in [1, \mathtt{T}_i]} = \begin{cases} 1 & \text{if } j \in [1, \mathtt{x_s^\mu}] \\ 0 & \text{Otherwise.} \end{cases} \tag{13}$$

Recall Eq. (9), we can obtain:

$$\begin{aligned} ||\mathtt{x_t}||_{\mathrm{L}} \circ (||\mathtt{x_s}||_{\mathrm{R}} \cdot \langle \mathtt{q_s} \rangle) &= \widehat{\mathtt{x_t}}(\widetilde{\mathtt{x_s}}\mathrm{Q})^T = \widehat{\mathtt{x_t}}(\widetilde{\mathtt{x_{sq}}})^T \\ &= \mathtt{r_s r_e r_p}(\sum_{\mu=1}^{d} \alpha(\mathtt{x_t^\mu})(\eta(\mathtt{x_s^\mu})\mathrm{Q_s^\mu})^T - k) \\ &\quad + \mathtt{r_s r_{e'} r_{p'}} \sum_{\mu=1}^{d} \zeta(\mathtt{x_t^\mu})(\xi(\mathtt{x_s^\mu})\mathrm{Q_{s'}^\mu})^T \\ &\quad - \mathtt{r_s} \sum_{i=1}^{w} \mathtt{r_x^i r_q^{i+1}}. \end{aligned} \tag{14}$$

From Eq. (1), Eq. (11), and Eq. (13), we have $\alpha(\mathtt{x_t^\mu})(\eta(\mathtt{x_s^\mu})\mathrm{Q_s^\mu})^T = 1$, only when the μ-th dimension is chosen, and $\mathtt{x_t^\mu} \leq \mathtt{x_s^\mu}$ $(\mathtt{x_t^\mu} \geq \mathtt{x_s^\mu})$ satisfies the user-defined preference, otherwise $\alpha(\mathtt{x_t^\mu})(\eta(\mathtt{x_s^\mu})\mathrm{Q_s^\mu})^T = 0$. If $\sum_{\mu=1}^{d} \alpha(\mathtt{x_t^\mu})(\eta(\mathtt{x_s^\mu})\mathrm{Q_s^\mu})^T \leq k - 1$, it means $\exists \mathtt{x_t^\mu}$ does not satisfy the user-defined preference. Since $\mathtt{r_e} > \mathtt{r_{e'}} \cdot \sum_{i=1}^{d} \mathtt{T}_i{}^2$ and $\mathtt{r_p} > \mathtt{r_{p'}} \cdot \sum_{i=1}^{d} \mathtt{T}_i{}^2$, we have $||\mathtt{x_t}||_{\mathrm{L}} \circ (||\mathtt{x_s}||_{\mathrm{R}} \cdot \langle \mathtt{q_s} \rangle) < 0 \Rightarrow \mathtt{x_t} \not\prec_{\mathcal{B}} \mathtt{x_s}$. Furthermore, If each dimension satisfies $\mathtt{x_t^\mu} \leq \mathtt{x_s^\mu}$ $(\mathtt{x_t^\mu} \geq \mathtt{x_s^\mu})$, $\mathtt{r_s r_e r_p}(\sum_{\mu=1}^{d} \alpha(\mathtt{x_t^\mu})(\eta(\mathtt{x_s^\mu})\mathrm{Q_s^\mu})^T - k) = 0$. In this case, if $\exists \mathtt{x_t^\mu}$ satisfies $\mathtt{x_t^\mu} < \mathtt{x_s^\mu}$ $(\mathtt{x_t^\mu} > \mathtt{x_s^\mu})$, $\mathtt{r_s r_{e'} r_{p'}} \sum_{\mu=1}^{d} \zeta(\mathtt{x_t^\mu})(\xi(\mathtt{x_s^\mu})\mathrm{Q_{s'}^\mu})^T > 0$. Since $\mathtt{r_{e'}} > \sum_{p=1}^{w} \mathtt{r_x^p}$ and $\mathtt{r_{p'}} > \sum_{j=2}^{w+1} \mathtt{r_q^j}$, we have $||\mathtt{x_t}||_{\mathrm{L}} \circ (||\mathtt{x_s}||_{\mathrm{R}} \cdot \langle \mathtt{q_s} \rangle) > 0 \Rightarrow \mathtt{x_t} \prec_{\mathcal{B}} \mathtt{x_s}$. Thus, $||\mathtt{x_t}||_{\mathrm{L}} \circ (||\mathtt{x_s}||_{\mathrm{R}} \cdot \langle \mathtt{q_s} \rangle) > 0 \Leftrightarrow \mathtt{x_t} \prec_{\mathcal{B}} \mathtt{x_s}$. □

4.4 The Description of Our Proposed Scheme

In this section, based on the above predicate encryption primitives, we present our privacy-preserving user-defined skyline query scheme, which mainly includes five phases: 1) System Initialization; 2) Local Data Outsourcing; 3) Query Token Generation; 4) User-defined Skyline Search; 5) Original Data Recovery.

4.4.1 *System Initialization*

In our scheme, the data owner $\mathcal{O}$ is responsible for initializing the whole system. Assume that the data owner holds a n-record dataset $\mathcal{X} = \{\mathtt{x}_i = (\mathtt{x}_i^1, \mathtt{x}_i^2, \cdots, \mathtt{x}_i^d) \mid 1 \leq i \leq n\}$, and each data record is a d-dimensional point. First, the data owner chooses a secure hash function $\mathtt{H}()$, e.g., SHA-256, and a secure symmetric key encryption $\mathtt{SE}()$, i.e., AES-128, and publishes them $\{\mathtt{H}(), \mathtt{SE}()\}$ with w (the number of dummy dimensions) as system parameters. Second, the data owner collects $\Delta = \{\mathtt{T}_1, \mathtt{T}_2, \cdots, \mathtt{T}_d\}$ and generates a d-dimensional lattice $\mathbb{L}_\Delta$ for the dataset $\mathcal{X}$. Finally, the data owner generates a master key k_0 and a secret key sk:

• k_0 is randomly generated and sent to the cloud via a secure channel.

• $sk = \{\mathrm{M}, \mathrm{M}', \widetilde{\mathrm{M}}\}$ is used to encrypt the data records and R-tree, where $\mathrm{M}' = \mathsf{HRIPE.Setup}(\Delta, w)$ and $(\mathrm{M}, \widetilde{\mathrm{M}}) = \mathsf{MPDPE.Setup}(\Delta, w)$.

After that, if a query user u_i would like to enjoy the user-defined skyline query services, he/she should first register to the data owner with his/her identity $\mathtt{ID}_i$. If accepted, the data owner will first generate a shared key $k_i = \mathtt{H}(k_0|\mathtt{ID}_i)$ for the query user u_i. Then, the data owner $\mathcal{O}$ authorizes $\{sk, k_i, \mathbb{L}_\Delta\}$ to the query user u_i through a secure channel.

4.4.2 *Local Data Outsourcing*

To guarantee efficiency and privacy for searching skylines, before outsourcing the dataset $\mathcal{X}$ to the cloud server $\mathcal{S}$, the data owner prepares the data with the following two steps:

Step-1. Build an R-tree over $\mathcal{X}$, denoted as Γ.

Step-2. Encrypt the R-tree Γ into $\mathtt{E}(\Gamma)$ with the secret key $sk = \{\mathrm{M}, \mathrm{M}', \widetilde{\mathrm{M}}\}$. Specifically, for internal nodes, i.e., MBRs, they are encrypted into $||\mathbf{R}'|| = \mathsf{HRIPE.Enc}(\mathbf{R}', \mathrm{M}', \mathbb{L}_\Delta)$ with M'. For leaf nodes, each leaf node $\mathtt{x} \in \mathcal{X}$ is encrypted into two ciphertexts $(||\mathtt{x}||_{\mathrm{L}}, ||\mathtt{x}||_{\mathrm{R}}) = \mathsf{MPDPE.Enc}(\mathtt{x}, \mathrm{M}, \widetilde{\mathrm{M}}, \mathbb{L}_\Delta)$, where $||\mathtt{x}||_{\mathrm{L}} = \widehat{\mathtt{x}}\,\mathrm{M}$ and $||\mathtt{x}||_{\mathrm{R}} = \widetilde{\mathtt{x}}\,\widetilde{\mathrm{M}}$.

After the above two steps, the data owner outsources the encrypted R-tree $\mathtt{E}(\Gamma)$ to the cloud server $\mathcal{S}$.

In order to reduce the number of outsourced ciphertexts, we will use $||\mathtt{x}||_{\mathrm{L}}$ instead of the ciphretext generated by $\mathsf{MPRPE.Enc}(\mathtt{x}, \mathrm{M}, \mathbb{L}_\Delta)$ to check whether a point lies inside a rectangle.

4.4.3 *Token Generation*

To obtain the desired skyline points, the query user u_i should first generate query tokens as follows.

Step-1. Construct a query rectangle $\mathbf{R} = \{(\mathtt{R}_1, \mathtt{R}_2, \cdots, \mathtt{R}_d) \mid \mathtt{R}_\mu = [\mathtt{R}_{\mu l}, \mathtt{R}_{\mu u}], 1 \leq \mu \leq d\}$. If the

μ-th dimension is ignored, the corresponding range $\mathtt{R}_\mu$ is set from 1 to $\mathtt{T}_\mu$.

Step-2. Construct a query vector $\mathtt{q_s} = (\mathtt{q_s^1}, \mathtt{q_s^2}, \cdots, \mathtt{q_s^d})$ according to user's preferences.

Step-3. Based on the query rectangle $\mathbf{R}$ and query vector $\mathtt{q_s}$, the query user generates three query tokens $\{\langle \mathtt{q}\rangle, \langle \mathtt{q_h}\rangle, \langle \mathtt{q_s}\rangle\}$, where

$$\begin{aligned}\langle \mathtt{q}\rangle &= \mathsf{MPRPE.TokenGen}(\mathbf{R}, \mathrm{M}, \mathbb{L}_\triangle)\\ \langle \mathtt{q_h}\rangle &= \mathsf{HRIPE.TokenGen}(\mathbf{R}, \mathrm{M}', \mathbb{L}_\triangle)\\ \langle \mathtt{q_s}\rangle &= \mathsf{MPDPE.TokenGen}(\mathtt{q_s}, \mathrm{M}, \widetilde{\mathrm{M}}, \mathbb{L}_\triangle).\end{aligned} \tag{15}$$

Note that, here $\langle \mathtt{q}\rangle$ is a $\theta = \sum_{i=1}^{d} \mathtt{T}_i + 3d + 1 + w$-dimensional vector, as we extend $\widetilde{\mathtt{q}}$ in our $\mathsf{MPRPE.TokenGen}(\mathbf{R}, \mathrm{M}, \mathbb{L}_\triangle)$ algorithm as follows:

$$\widetilde{\mathtt{q}} = (\mathtt{r_t}\beta(\mathtt{R}_1), \mathtt{r_t}\beta(\mathtt{R}_2), \cdots, \mathtt{r_t}\beta(\mathtt{R}_d), \underbrace{0, \cdots, 0}_{3d}, d \cdot \mathtt{r_t}, \mathtt{r_q^1}, \cdots, \mathtt{r_q^w}).$$

After generating query tokens, the query user u_i will encrypt these tokens with the shared key k_i: $\mathtt{SE}(\langle \mathtt{q}\rangle | \langle \mathtt{q_h}\rangle | \langle \mathtt{q_s}\rangle | ts,\ k_i)$, in which ts is the time stamp. Finally, the query user sends the query request: Req= $\{\mathtt{ID}_i, ts, \mathtt{SE}(\langle \mathtt{q}\rangle | \langle \mathtt{q_h}\rangle | \langle \mathtt{q_s}\rangle | ts,\ k_i)\}$, to the cloud server.

Upon receiving Req, the cloud server will extract $\mathtt{ID}_i$ and calculate the shared key with the authorized master key: $k_i = \mathtt{H}(k_0 | \mathtt{ID}_i)$. Next, by decrypting $\mathtt{SE}(\langle \mathtt{q}\rangle | \langle \mathtt{q_h}\rangle | \langle \mathtt{q_s}\rangle | ts,\ k_i)$ with k_i, the cloud server recovers the query tokens $\{\langle \mathtt{q}\rangle, \langle \mathtt{q_h}\rangle, \langle \mathtt{q_s}\rangle\}$ and time stamp ts. With the decrypted ts, the cloud server checks whether it is the same as the time stamp extracted from Req. If yes, the query request Req will be further processed, and rejected otherwise.

4.4.4 User-defined Skyline Search

Once accepting the query request from query user u_i, with these query tokens $\{\langle \mathtt{q}\rangle, \langle \mathtt{q_h}\rangle, \langle \mathtt{q_s}\rangle\}$, the cloud server can run the user-defined skyline search algorithm over the encrypted R-tree $\mathtt{E}(\Gamma)$. The encrypted R-tree contains the encrypted MBR $||\mathbf{R}'||$ for each internal node and $(||\mathtt{x}||_\mathrm{L}, ||\mathtt{x}||_\mathrm{R})$ for each leaf node. Here, we employ the BNL algorithm (introduced in Section 3.2) to retrieve skyline points. Typically, the cloud server can conduct the following steps to obtain the desired skyline points (see details in Algorithm 1).

Step-1. Perform range query on $\mathtt{E}(\Gamma)$ to obtain the points inside the user-defined query rectangle. First, the cloud server traverses $\mathtt{E}(\Gamma)$ to determine whether the query rectangle intersects with the internal nodes. It can be achieved by executing $\mathsf{HRIPE.Query}$($||\mathbf{R}'||$, $\langle \mathtt{q_h}\rangle$). When reaching the leaf nodes, the cloud server runs $\mathsf{MPRPE.Query}$($||\mathtt{x}||_\mathrm{L}$, $\langle \mathtt{q}\rangle$) to check whether the point lies inside the query rectangle. If yes, the point is added into a set, denoted as $\mathtt{S_p}$. This step is depicted from lines 12-21 in Algorithm 1.

Step-2. Find skyline points in $\mathtt{S_p}$. In order to obtain the skyline points, the cloud server needs to find the points in $\mathtt{S_p}$ that are not dominated by any other point in the set. Using our MPDPE scheme, the cloud server can check whether $\mathtt{x_t} \prec_\mathcal{B} \mathtt{x_s}$ by executing $\mathsf{MPDPE.Query}$($||\mathtt{x_t}||_\mathrm{L}$, $||\mathtt{x_s}||_\mathrm{R}$, $\langle \mathtt{q_s}\rangle$). After checking the dominating relations for all points in $\mathtt{S_p}$, the cloud server can find skyline points and add them into a set $\mathtt{S}$. See details from lines 4-10 in Algorithm 1.

Before returning the retrieved skyline points to the query user u_i, the cloud server encrypts them by the shared key k_i: $\mathtt{SE}(\mathtt{S} | ts,\ k_i)$, where ts is a new time stamp. Finally, the cloud server sends $\{ts, \mathtt{SE}(\mathtt{S} | ts,\ k_i)\}$ to the query user u_i.

Algorithm 1 User-defined Skyline Search over Ciphertexts

```
Input: An encrypted R-tree, $\mathtt{E}(\Gamma)$; three query tokens, $\{\langle q\rangle, \langle q_h\rangle, \langle q_s\rangle\}$.
Output: A set containing encrypted skyline points, $\mathtt{S}$;
 1: root ← $\mathtt{E}(\Gamma)$.getRoot();
 2: Create an empty set $\mathtt{S_p} \leftarrow \varnothing$;
 3: rangeQuery(root, $\langle q\rangle$, $\langle q_h\rangle$, $\mathtt{S_p}$)
 4: for each node $n_p$ of $\mathtt{S_p}$ do
 5:     for each node $n_s$ of $\mathtt{S}$ do
 6:         if $||x_{n_s}||_L \circ (||x_{n_p}||_R \cdot \langle q_s\rangle) > 0$ then
 7:             Break;
 8:         if $||x_{n_p}||_L \circ (||x_{n_s}||_R \cdot \langle q_s\rangle) > 0$ then
 9:             $\mathtt{S}$.remove($n_s$);
10:     $\mathtt{S}$.add($n_p$);
11:
12: function rangeQuery(node, $\langle q\rangle$, $\langle q_h\rangle$, $\mathtt{S_p}$)
13:     if node is leaf then
14:         $||x||_L$ ← node's left ciphertext;
15:         if $||x||_L \circ \langle q\rangle > 0$ then
16:             $\mathtt{S_p}$.add(node);
17:     else
18:         $||\mathbf{R}'||$ ← node.MBR();
19:         if $||\mathbf{R}'|| \circ \langle q_h\rangle > 0$ then
20:             for each childNode of node do
21:                 rangeQuery(childNode, $\langle q\rangle$, $\langle q_h\rangle$, $\mathtt{S_p}$);
```

4.4.5 Original Data Recovery

After receiving the query response, the query user u_i uses the authorized shared key k_i to obtain the encrypted time stamp ts and the set $\mathtt{S}$ in which the skyline points are encrypted by our predicate encryption. First, u_i verifies whether the external ts is consistent with the encrypted ts. If yes, the query response will be accepted, and dropped otherwise. Then, with the secret key $sk = \{\mathrm{M}, \mathrm{M}', \widetilde{\mathrm{M}}\}$, the query user can recover each point $\mathtt{x} \in \mathtt{S}$: i) with M, the query user can obtain $\widehat{\mathtt{x}}$ by calculating $||\mathtt{x}||_\mathrm{L}\mathrm{M}^{-1} = \widehat{\mathtt{x}}\ \mathrm{M}\ \mathrm{M}^{-1} = \widehat{\mathtt{x}}$; ii) it is simple to remove $\mathtt{r_e}$ from $\widehat{\mathtt{x}}$ and recover the point $\mathtt{x}$ by Eq. (1) since the elements of $\alpha(\mathtt{x}^i)$ are 0 or 1.

5 Security Analysis

In this section, we will analyze the security of the proposed user-defined skyline query scheme. Since our scheme is built on three predicate encryption primitives, namely, MPRPE, HRIPE, and MPDPE, we first prove the security of these predicate encryption primitives, and then analyze the security of our user-defined skyline query scheme. Note that, since HRIPE can be treated as applying MPRPE in the $2d$-dimensional point, the security proof of HRIPE is the same as that of MPRPE. Therefore, for space saving, we only provide the detailed security analysis of MPRPE and MPDPE in this section.

5.1 Security of MPRPE

Our MPRPE scheme is designed to support the multi-dimensional range query over ciphertexts. As the same as the security analysis of other searchable encryption schemes [24], [25], we prove that our MPRPE scheme is selectively secure in the real/ideal world security model. Specifically, the real world is our proposal, while the ideal world is an ideal function with some leaked information, which is related to the public information in our proposal and defined as a leakage function. In addition, in the ideal world, the adversary can select queries according to the previous query tokens and query results. We will prove that the real world of our proposal is indistinguishable from the ideal world, which means that our scheme can guarantee security when the queries are not independent of the previous query materials. Before delving into the detailed proof, we first define the trivial leakage function, denoted as $\mathcal{L}$, of our MPRPE scheme. Assume that $||\mathtt{x}||$ is the ciphertext of a d-dimensional point $\mathtt{x}$, and $\langle \mathtt{q} \rangle$ is the ciphertext of a d-dimensional query rectangle $\mathbf{R}$. The leakage function of our MPRPE is the inner product result between $||\mathtt{x}||$ and $\langle \mathtt{q} \rangle$, i.e., $\mathcal{L} = \mathtt{dot}(||\mathtt{x}||, \langle \mathtt{q} \rangle)$. Based on the leakage $\mathcal{L}$, we run the ideal experiment as follows.

Ideal experiment. The ideal experiment involves two participants: a probabilistic polynomial time adversary $\mathcal{A}$ and a simulator with the leakage $\mathcal{L}$. They interact as follows.

• Setup. $\mathcal{A}$ chooses p_1 d-dimensional database records $\{\mathtt{x}_i\}_{i=1}^{p_1}$ and transfers them to the simulator. On receiving them, the simulator randomly generates p_1 θ-dimensional vectors $\{||\mathtt{x}_i||'\}_{i=1}^{p_1}$ as the ciphertexts of $\{\mathtt{x}_i\}_{i=1}^{p_1}$.

• Token generation phase 1. In this phase, $\mathcal{A}$ randomly chooses p_2 d-dimensional query rectangles $\{\mathbf{R}_i\}_{i=1}^{p_2}$, and sends them to the simulator. After receiving $\{\mathbf{R}_i\}_{i=1}^{p_2}$, the simulator uses the leakage $\mathcal{L}$ and $\{||\mathtt{x}_i||'\}_{i=1}^{p_1}$ (generated in the setup phase of ideal experiment) to construct the ciphertexts of $\{\mathbf{R}_i\}_{i=1}^{p_2}$ and returns them to $\mathcal{A}$. We denote these ciphertexts as $\{\langle \mathtt{q}_i \rangle'\}_{i=1}^{p_2}$. For each $\mathbf{R}_i$, the simulator generates the corresponding ciphertext $\langle \mathtt{q}_i \rangle'$ as follows.

Step 1. The simulator generates a vector $\mathtt{V}_i$ with p_1 length, in which each element r_j $(1 \leq j \leq p_1)$ is a random real number and satisfies the leakage $\mathcal{L}$. That is:

$$\begin{cases} r_j > 0, & \text{if } ||\mathtt{x}_j|| \circ \langle \mathtt{q}_j \rangle > 0 \\ r_j < 0, & \text{if } ||\mathtt{x}_j|| \circ \langle \mathtt{q}_j \rangle < 0. \end{cases} \tag{16}$$

Step 2. The simulator randomly chooses a vector $\langle \mathtt{q}_i \rangle'$ satisfying the following equation system.

$$\begin{bmatrix} ||\mathtt{x}_1||' \\ \vdots \\ ||\mathtt{x}_{p_1}||' \end{bmatrix} (\langle \mathtt{q}_i \rangle')^T = \mathtt{V}_i^T.$$

• Challenge phase. In the challenge phase, the simulator sends $\{||\mathtt{x}_i||'\}_{i=1}^{p_1}$ to $\mathcal{A}$.

• Token generation phase 2. In the token generation phase 2, $\mathcal{A}$ continues to generate $p_2' - p_2$ d-dimensional query rectangles $\{\mathbf{R}_i\}_{i=p_2+1}^{p_2'}$, and sends them to the simulator. Once receiving $\{\mathbf{R}_i\}_{i=p_2+1}^{p_2'}$, the simulator adopts the same steps to obtain $\{\langle \mathtt{q}_i \rangle'\}_{i=p_2+1}^{p_2'}$ and sends them to $\mathcal{A}$.

As the real experiment is our proposal, the view of $\mathcal{A}$ in the real experiment is $\mathtt{View}_{\mathcal{A},\mathtt{Real}} = \{\{||\mathtt{x}_i||\}_{i=1}^{p_1}, \{\langle \mathtt{q}_i \rangle\}_{i=1}^{p_2'}\}$ that can be generated by our MPRPE scheme. While, in the ideal experiment, the view of $\mathcal{A}$ is $\mathtt{View}_{\mathcal{A},\mathtt{Ideal}} = \{\{||\mathtt{x}_i||'\}_{i=1}^{p_1}, \{\langle \mathtt{q}_i \rangle'\}_{i=1}^{p_2'}\}$ as shown in the above experiment. Based on the views of $\mathcal{A}$, we define the security of MPRPE.

Definition 6 (**Security of MPRPE**). *MPRPE is selectively secure with the leakage $\mathcal{L}$ iff for any probabilistic polynomial time adversary $\mathcal{A}$ issuing a polynomial number of database records encryption and query token generations, there exists a simulator such that the advantage that $\mathcal{A}$ can distinguish the views of real and ideal experiments is negligible. That is, $|\Pr[\mathtt{View}_{\mathcal{A},\mathtt{Real}} = 1] - \Pr[\mathtt{View}_{\mathcal{A},\mathtt{Ideal}} = 1]|$ is negligible.*

Theorem 1. *MPRPE is selectively secure with $\mathcal{L}$.*

Proof. According to Definition 6, MPRPE is selectively secure with $\mathcal{L}$ *iff* $\mathcal{A}$ cannot distinguish $\mathtt{View}_{\mathcal{A},\mathtt{Real}} = \{\{||\mathtt{x}_i||\}_{i=1}^{p_1}, \{\langle \mathtt{q}_i \rangle\}_{i=1}^{p_2'}\}$ and $\mathtt{View}_{\mathcal{A},\mathtt{Ideal}} = \{\{||\mathtt{x}_i||'\}_{i=1}^{p_1}, \{\langle \mathtt{q}_i \rangle'\}_{i=1}^{p_2'}\}$. To prove the indistinguishability, we consider two cases: 1) the ciphertexts $\{||\mathtt{x}_i||\}_{i=1}^{p_1}$ are indistinguishable from $\{||\mathtt{x}_i||'\}_{i=1}^{p_1}$; 2) the query tokens $\{\langle \mathtt{q}_i \rangle\}_{i=1}^{p_2'}$ are indistinguishable from $\{\langle \mathtt{q}_i \rangle'\}_{i=1}^{p_2'}$.

• $\{||\mathtt{x}_i||\}_{i=1}^{p_1}$ are indistinguishable from $\{||\mathtt{x}_i||'\}_{i=1}^{p_1}$. In the real experiment, $||\mathtt{x}_i|| = \widetilde{\mathtt{x}_i}\mathrm{M}$, where $\widetilde{\mathtt{x}_i} = (\mathtt{r_e}\alpha(\mathtt{x}_i^1), \mathtt{r_e}\alpha(\mathtt{x}_i^2), \cdots, \mathtt{r_e}\alpha(\mathtt{x}_i^d), -\mathtt{r_e}, \mathtt{r}_\mathtt{x}^1, \cdots, \mathtt{r}_\mathtt{x}^w)$. Since each ciphertext $||\mathtt{x}_i||$ contains at least three random numbers $\{\mathtt{r_e}, \mathtt{r}_\mathtt{x}^1, \cdots, \mathtt{r}_\mathtt{x}^w \mid w \geq 2\}$, and the random matrix M is unknown for $\mathcal{A}$, $\{||\mathtt{x}_i||\}_{i=1}^{p_1}$ are indistinguishable from random vectors. Since $\{||\mathtt{x}_i||'\}_{i=1}^{p_1}$ are randomly generated in the ideal experiment, $\{||\mathtt{x}_i||\}_{i=1}^{p_1}$ are indistinguishable from $\{||\mathtt{x}_i||'\}_{i=1}^{p_1}$.

• $\{\langle \mathtt{q}_i \rangle\}_{i=1}^{p_2'}$ are indistinguishable from $\{\langle \mathtt{q}_i \rangle'\}_{i=1}^{p_2'}$. In the real experiment, $\langle \mathtt{q}_i \rangle$ is generated by $\widetilde{\mathtt{q}}(\mathrm{M}^{-1})^T$, where $\widetilde{\mathtt{q}} = (\mathtt{r_t}\beta(\mathtt{R}_1), \mathtt{r_t}\beta(\mathtt{R}_2), \cdots, \mathtt{r_t}\beta(\mathtt{R}_d), d \cdot \mathtt{r_t}, \mathtt{r}_\mathtt{q}^1, \cdots, \mathtt{r}_\mathtt{q}^w)$. Since each query token $\langle \mathtt{q}_i \rangle$ has at least three random numbers $\{\mathtt{r_t}, \mathtt{r}_\mathtt{q}^1, \cdots, \mathtt{r}_\mathtt{q}^w \mid w \geq 2\}$, and M^{-1} is unknown for $\mathcal{A}$, $\{\langle \mathtt{q}_i \rangle\}_{i=1}^{p_2'}$ are indistinguishable from random vectors. In the ideal experiment, each query token $\langle \mathtt{q}_i \rangle'$ is restricted by the leakage $\mathcal{L}$. However, from Eq. (3), we know that the results of $||\mathtt{x}_j|| \circ \langle \mathtt{q}_j \rangle$ are random, leading to random r_j in Eq. (16). Therefore, $\{\langle \mathtt{q}_i \rangle'\}_{i=1}^{p_2'}$ are indistinguishable from random vectors under the randomly chosen ciphertexts $\{||\mathtt{x}_i||'\}_{i=1}^{p_1}$. Thus, $\{\langle \mathtt{q}_i \rangle\}_{i=1}^{p_2'}$ are indistinguishable from $\{\langle \mathtt{q}_i \rangle'\}_{i=1}^{p_2'}$.

As a result, we can deduce that $\mathtt{View}_{\mathcal{A},\mathtt{Real}}$ is indistinguishable from $\mathtt{View}_{\mathcal{A},\mathtt{Ideal}}$, and $\mathcal{A}$ cannot distinguish the views of real and ideal experiments of MPRPE scheme. Thus, MPRPE is selectively secure with the leakage $\mathcal{L}$. □

5.2 Security of MPDPE

Our MPDPE scheme is designed to check the dominating relation between two encrypted data points. Similar to the security of MPRPE, in this section, we prove that our MPDPE scheme is selectively secure in the real/ideal world security model. Assume that $||\mathtt{x_t}||_\mathrm{L}$ is $\mathtt{x_t}$'s left ciphertext, $||\mathtt{x_s}||_\mathrm{R}$ $(s \neq t)$ is $\mathtt{x_s}$'s right ciphertext, and $\langle \mathtt{q_s} \rangle$ is a query token generated by MPDPE.TokenGen. The leakage function of the MPDPE scheme is the result of $||\mathtt{x_t}||_\mathrm{L} \circ (||\mathtt{x_s}||_\mathrm{R} \cdot \langle \mathtt{q}_s \rangle)$, denoted as $\widetilde{\mathcal{L}} = \mathtt{dotMul}(||\mathtt{x_t}||_\mathrm{L}, ||\mathtt{x_s}||_\mathrm{R}, \langle \mathtt{q_s} \rangle)$. Based on the leakage $\widetilde{\mathcal{L}}$, we run the ideal experiment as follows.

Ideal experiment. The ideal experiment involves a probabilistic polynomial time adversary $\mathcal{A}$ and a simulator with the leakage $\widetilde{\mathcal{L}}$, and they interact as follows.

• Setup. $\mathcal{A}$ chooses p_1 d-dimensional database records $\{\mathtt{x}_i\}_{i=1}^{p_1}$ and transfers them to the simulator. Once receiving them, the simulator randomly generates p_1 θ-dimensional vectors $\{||\mathtt{x}_i||_\mathrm{L}'\}_{i=1}^{p_1}$ and $\{||\mathtt{x}_i||_\mathrm{R}'\}_{i=1}^{p_1}$ as the MPDPE ciphertexts of $\{\mathtt{x}_i\}_{i=1}^{p_1}$.

• Token generation phase 1. $\mathcal{A}$ randomly chooses p_2 preference vectors $\{\mathtt{q}_{\mathtt{s}_i}\}_{i=1}^{p_2}$, and sends them to the simulator. The simulator uses the leakage $\widetilde{\mathcal{L}}$, $\{||\mathtt{x}_i||_{\mathrm{L}}'\}_{i=1}^{p_1}$, and $\{||\mathtt{x}_i||_{\mathrm{R}}'\}_{i=1}^{p_1}$ to construct the ciphertexts of $\{\mathtt{q}_{\mathtt{s}_i}\}_{i=1}^{p_2}$ and returns them to $\mathcal{A}$. We denote these ciphertexts as $\{\langle\mathtt{q}_{\mathtt{s}_i}\rangle'\}_{i=1}^{p_2}$. For each $\mathtt{q}_{\mathtt{s}_i}$, the simulator generates the corresponding ciphertext $\langle\mathtt{q}_{\mathtt{s}_i}\rangle'$ by the following steps:

Step 1. The simulator generates a $p_1 \times p_1$ matrix $\mathtt{S}_i$, in which each element r_{i_i,i_2} $(1 \le i_1, i_2 \le p_1)$ is a random real number and satisfies the leakage $\widetilde{\mathcal{L}}$. That is:

$$\begin{cases} r_{i_i,i_2} > 0, & \text{if } ||\mathtt{x}_{i_1}||_{\mathrm{L}} \circ (||\mathtt{x}_{i_2}||_{\mathrm{R}} \cdot \langle\mathtt{q}_{\mathtt{s}_i}\rangle) > 0 \\ r_{i_i,i_2} < 0, & \text{if } ||\mathtt{x}_{i_1}||_{\mathrm{L}} \circ (||\mathtt{x}_{i_2}||_{\mathrm{R}} \cdot \langle\mathtt{q}_{\mathtt{s}_i}\rangle) < 0. \end{cases} \tag{17}$$

Step 2. The simulator randomly chooses a vector $\langle\mathtt{q}_{\mathtt{s}_i}\rangle'$ satisfying the following equation system.

$$\begin{bmatrix} ||\mathtt{x}_1||_{\mathrm{L}}' \\ \vdots \\ ||\mathtt{x}_{p_1}||_{\mathrm{L}}' \end{bmatrix} \circ \left(\begin{bmatrix} ||\mathtt{x}_1||_{\mathrm{R}}' \\ \vdots \\ ||\mathtt{x}_{p_1}||_{\mathrm{R}}' \end{bmatrix} \cdot \langle\mathtt{q}_{\mathtt{s}_i}\rangle' \right) = \mathtt{S}_i.$$

• Challenge phase. The simulator sends $\{||\mathtt{x}_i||_{\mathrm{L}}'\}_{i=1}^{p_1}$ and $\{||\mathtt{x}_i||_{\mathrm{R}}'\}_{i=1}^{p_1}$ to $\mathcal{A}$.

• Token generation phase 2. $\mathcal{A}$ continues to generate preference vectors $\{\mathtt{q}_{\mathtt{s}_i}\}_{i=p_2+1}^{p_2'}$ and sends them to the simulator. The simulator adopts the same steps to obtain $\{\langle\mathtt{q}_{\mathtt{s}_i}\rangle'\}_{i=p_2+1}^{p_2'}$ and sends them to $\mathcal{A}$.

As the real experiment is our proposal, the view of $\mathcal{A}$ in the real experiment is $\mathtt{View}_{\mathcal{A},\mathtt{Real}} = \{\{||\mathtt{x}_i||_{\mathrm{L}}\}_{i=1}^{p_1}, \{||\mathtt{x}_i||_{\mathrm{R}}\}_{i=1}^{p_1}, \{\langle\mathtt{q}_{\mathtt{s}_i}\rangle\}_{i=1}^{p_2'}\}$ that can be generated by our MPDPE scheme. In the ideal experiment, the view of $\mathcal{A}$ is $\mathtt{View}_{\mathcal{A},\mathtt{Ideal}} = \{\{||\mathtt{x}_i||_{\mathrm{L}}'\}_{i=1}^{p_1}, \{||\mathtt{x}_i||_{\mathrm{R}}'\}_{i=1}^{p_1}, \{\langle\mathtt{q}_i\rangle'\}_{i=1}^{p_2'}\}$ as shown in the above experiment. Based on the views of $\mathcal{A}$, we define the security of MPDPE.

Definition 7 (**Security of MPDPE**). *MPDPE is selectively secure with the leakage $\widetilde{\mathcal{L}}$ iff for any probabilistic polynomial time adversary $\mathcal{A}$ issuing a polynomial number of database records encryption and query token generations, there exists a simulator such that the advantage that $\mathcal{A}$ can distinguish the views of real and ideal experiments is negligible. That is,* $|\Pr[\mathtt{View}_{\mathcal{A},\mathtt{Real}} = 1] - \Pr[\mathtt{View}_{\mathcal{A},\mathtt{Ideal}} = 1]|$ *is negligible.*

Theorem 2. *MPDPE is selectively secure with $\widetilde{\mathcal{L}}$.*

Proof. According to Definition 7, MPDPE is selectively secure with $\widetilde{\mathcal{L}}$ *iff* $\mathcal{A}$ cannot distinguish $\mathtt{View}_{\mathcal{A},\mathtt{Real}} = \{\{||\mathtt{x}_i||_{\mathrm{L}}\}_{i=1}^{p_1}, \{||\mathtt{x}_i||_{\mathrm{R}}\}_{i=1}^{p_1}, \{\langle\mathtt{q}_{\mathtt{s}_i}\rangle\}_{i=1}^{p_2'}\}$ and $\mathtt{View}_{\mathcal{A},\mathtt{Ideal}} = \{\{||\mathtt{x}_i||_{\mathrm{L}}'\}_{i=1}^{p_1}, \{||\mathtt{x}_i||_{\mathrm{R}}'\}_{i=1}^{p_1}, \{\langle\mathtt{q}_i\rangle'\}_{i=1}^{p_2'}\}$. Since all ciphertexts and query tokens in the ideal experiment are random generated, distinguishing $\mathtt{View}_{\mathcal{A},\mathtt{Real}}$ from $\mathtt{View}_{\mathcal{A},\mathtt{Ideal}}$ is equivalent to distinguishing $\mathtt{View}_{\mathcal{A},\mathtt{Real}}$ from random ciphertexts and random query tokens. Here we consider the following three cases: 1) the ciphertexts $\{||\mathtt{x}_i||_{\mathrm{L}}\}_{i=1}^{p_1}$ and $\{||\mathtt{x}_i||_{\mathrm{R}}\}_{i=1}^{p_1}$ are indistinguishable from random ciphertexts; 2) the query tokens $\{\langle\mathtt{q}_{\mathtt{s}_i}\rangle\}_{i=1}^{p_2'}$ are indistinguishable from random query tokens; 3) the intermediate results computed from $\{||\mathtt{x}_i||_{\mathrm{R}}\}_{i=1}^{p_1}$ and $\{\langle\mathtt{q}_{\mathtt{s}_i}\rangle\}_{i=1}^{p_2'}$ are indistinguishable from random vectors.

• $\{||\mathtt{x}_i||_{\mathrm{L}}\}_{i=1}^{p_1}$ and $\{||\mathtt{x}_i||_{\mathrm{R}}\}_{i=1}^{p_1}$ are indistinguishable from random ciphertexts. In the MPDPE.Enc algorithm, $||\mathtt{x}_i||_{\mathrm{L}} = \widehat{\mathtt{x}}_i\mathrm{M}$ and $||\mathtt{x}_i||_{\mathrm{R}} = \widetilde{\mathtt{x}}_i\widetilde{\mathrm{M}}$, where

$$\begin{aligned} \widehat{\mathtt{x}_i} =& (\mathtt{r_e}\alpha(\mathtt{x}_i^1), \mathtt{r_e}\alpha(\mathtt{x}_i^2), \cdots, \mathtt{r_e}\alpha(\mathtt{x}_i^d), \mathtt{r_{e'}}\zeta(\mathtt{x}_i^1), \\ & \mathtt{r_{e'}}\zeta(\mathtt{x}_i^2), \cdots, \mathtt{r_{e'}}\zeta(\mathtt{x}_i^d), -\mathtt{r_e}, -\mathtt{r_x}^1, \cdots, -\mathtt{r_x}^w) \text{ and} \\ \widetilde{\mathtt{x}_i} =& (\mathtt{r_s}\eta(\mathtt{x}_i^1), \mathtt{r_s}\eta(\mathtt{x}_i^2), \cdots, \mathtt{r_s}\eta(\mathtt{x}_i^d), \mathtt{r_s}\xi(\mathtt{x}_i^1), \\ & \mathtt{r_s}\xi(\mathtt{x}_i^2), \cdots, \mathtt{r_s}\xi(\mathtt{x}_i^d), -\mathtt{r_z}, \mathtt{r_z}, \cdots, \mathtt{r_z}). \end{aligned}$$

Since both $||\mathtt{x}_i||_{\mathrm{L}}$ and $||\mathtt{x}_i||_{\mathrm{R}}$ contain at least two types of random numbers, for example $\mathtt{r_e}$ and $\mathtt{r_x}$ in $||\mathtt{x}_i||_{\mathrm{L}}$, and the random matrix M and $\widetilde{\mathrm{M}}$ are unknown for $\mathcal{A}$, $||\mathtt{x}_i||_{\mathrm{L}}$ and $||\mathtt{x}_i||_{\mathrm{R}}$ are indistinguishable from random vectors. Thus, $\{||\mathtt{x}_i||_{\mathrm{L}}\}_{i=1}^{p_1}$ and $\{||\mathtt{x}_i||_{\mathrm{R}}\}_{i=1}^{p_1}$ are indistinguishable from random ciphertexts.

• $\{\langle\mathtt{q}_{\mathtt{s}_i}\rangle\}_{i=1}^{p_2'}$ are indistinguishable from random query tokens. In the MPDPE.TokenGen algorithm, $\langle\mathtt{q_s}\rangle = \widetilde{\mathrm{M}}^{-1}\mathrm{Q}(\mathrm{M}^{-1})^T$. According to Eq. (8), Q contains a random matrix RM. Also, there exists random real numbers $\{\mathtt{r_p}, \mathtt{r_{p'}}, \mathtt{r_q}^j \mid 2 \le j \le w+1\}$, and both $\widetilde{\mathrm{M}}^{-1}$ and M^{-1} are unknown for $\mathcal{A}$. Therefore, each query token $\langle\mathtt{q}_{\mathtt{s}_i}\rangle$ is indistinguishable from a random matrix. Thus, $\{\langle\mathtt{q}_{\mathtt{s}_i}\rangle\}_{i=1}^{p_2'}$ are indistinguishable from random query tokens.

• The intermediate results computed from $\{||\mathtt{x}_i||_{\mathrm{R}}\}_{i=1}^{p_1}$ and $\{\langle\mathtt{q}_{\mathtt{s}_i}\rangle\}_{i=1}^{p_2'}$ are indistinguishable from random vectors. In the real experiment, $\mathcal{A}$ may compute $||\mathtt{x}_i||_{\mathrm{R}} \cdot \langle\mathtt{q}_{\mathtt{s}_i}\rangle$ to obtain intermediate results, i.e., $||\mathtt{x}_i||_{\mathrm{R}} \cdot \langle\mathtt{q}_{\mathtt{s}_i}\rangle = \widetilde{\mathtt{x}}_i\mathrm{Q}(\mathrm{M}^{-1})^T$, where $\widetilde{\mathtt{x}}_i\mathrm{Q} = \mathtt{r_s}(\mathtt{r_p} \cdot \eta(\mathtt{x}_i^1) \cdot \mathrm{Q}_{\mathtt{s}}^1, \cdots, \mathtt{r_p} \cdot \eta(\mathtt{x}_i^d) \cdot \mathrm{Q}_{\mathtt{s}}^d, \mathtt{r_{p'}} \cdot \xi(\mathtt{x}_i^1) \cdot \mathrm{Q}_{\mathtt{s'}}^1, \cdots, \mathtt{r_{p'}} \cdot \xi(\mathtt{x}_i^d) \cdot \mathrm{Q}_{\mathtt{s'}}^d, k \cdot \mathtt{r_p}, \mathtt{r_q}^2, \cdots, \mathtt{r_q}^{w+1})$. Since each intermediate result contains many random numbers $\{\mathtt{r_s}, \mathtt{r_p}, \mathtt{r_{p'}}, \mathtt{r_q}^1, \cdots, \mathtt{r_q}^w \mid w \ge 2\}$, and M^{-1} is unknown for $\mathcal{A}$. Therefore, each intermediate result is indistinguishable from a random vector. Thus, the intermediate results computed from $\{||\mathtt{x}_i||_{\mathrm{R}}\}_{i=1}^{p_1}$ and $\{\langle\mathtt{q}_{\mathtt{s}_i}\rangle\}_{i=1}^{p_2'}$ are indistinguishable from random vectors.

As a result, we can deduce that $\mathtt{View}_{\mathcal{A},\mathtt{Real}}$ is indistinguishable from $\mathtt{View}_{\mathcal{A},\mathtt{Ideal}}$, and $\mathcal{A}$ cannot distinguish the views of real and ideal experiments of MPDPE scheme. Thus, MPDPE is selectively secure with the leakage $\widetilde{\mathcal{L}}$. □

5.3 Security of User-defined Skyline Query Scheme

In this subsection, we analyze the security of our user-defined skyline query scheme. Recalling our security model, we consider that the cloud server is honest-but-curious, who may be curious about the plaintext of data records, query requests, and query results. Since we focus on privacy-preserving properties, we will show that these values are privacy-preserving in the cloud server.

• The data records are privacy-preserving. The outsourced data records should be kept secret from the cloud server. In our scheme, one data record $\mathtt{x} \in \mathcal{X}$ will be encrypted into two ciphertexts ($||\mathtt{x}||_{\mathrm{L}}$, $||\mathtt{x}||_{\mathrm{R}}$) = MPDPE.Enc(x, M, $\widetilde{\mathrm{M}}$, $\mathbb{L}_\triangle$). As proved in Section 5.2, the security of MPDPE can guarantee that the cloud server cannot obtain the plaintexts of data records from their ciphertexts. Meanwhile, the cloud server may try to deduce the plaintexts of data records when performing the user-defined skyline queries. First, the cloud server can compute $||\mathtt{x}||_{\mathrm{R}} \cdot \langle\mathtt{q_s}\rangle = \widetilde{\mathtt{x}}\mathrm{Q}(\mathrm{M}^{-1})^T$. Since the cloud server has no idea about the secret M^{-1}, it even cannot obtain $\tilde{\mathtt{x}}\mathsf{Q}$. Second, the cloud server can further run MPRPE.Query and MPDPE.Query to obtain the inner product of data records and query requests. However, each inner product result contains many random numbers (see Eq. (3) and Eq. (14)). Therefore, the cloud server cannot recover the data records from inner product results. Finally, the cloud server can know which data records are retrieved as skyline points and deduce the plaintexts of data records according to the dominating relations. However, in our scheme, the cloud server neither knows which dimensions are selected nor what are the preferences of selected dimensions. Therefore, the cloud server cannot infer the plaintexts of data records by the dominating relations. Thus, the data records are privacy-preserving in our scheme.

• The query requests and query results are privacy-preserving. The query requests and query results should be kept secret from the cloud server. In our scheme, there are three query tokens, i.e., $\{\langle \mathtt{q}\rangle, \langle \mathtt{q_h}\rangle, \langle \mathtt{q_s}\rangle\}$, for the query request $\{\mathbf{R}, \mathtt{q_s}\}$. According to Eq. (15), $\mathbf{R}$ is encrypted by MPRPE.TokenGen and HRIPE.TokenGen, while $\mathtt{q_s}$ is encrypted by MPDPE.TokenGen. Therefore, the security of MPRPE, HRIPE and MPDPE can guarantee that the cloud server cannot deduce any plaintext information of $\mathbf{R}$ and $\mathtt{q_s}$ from these query tokens. Besides, in the process of the user-defined skyline queries, the cloud server may infer the query requests with the inner product results obtained from MPRPE.Query, HRIPE.Query, and MPDPE.Query. However, each inner product result is protected by many random real numbers, and the cloud server has no idea about the data record $\mathtt{x}$. Therefore, the cloud cannot recover the plaintexts of query requests from inner product results. In our scheme, the cloud server returns the ciphertexts of data records to the query users. As discussed previously, the cloud server cannot obtain any information about the plaintexts of data records without the secret key sk. Therefore, the query results are privacy-preserving in the cloud server.

Besides, although each query user has the same secret key sk to decrypt query tokens and query results, the secure symmetric encryption $\mathtt{SE}()$ guarantees that no query user can recover query tokens and query results of other query users. It is because that these tokens and results are encrypted by the shared key k_i held only by the corresponding query user u_i. Thus, the query requests and query results are also privacy-preserving for other query users.

6 Performance Evaluation

In this section, we experimentally evaluate the performance of our proposed scheme and compare it with the alternative scheme in terms of computational costs. Specifically, we first explore the impacts of different datasets and query workloads on our scheme. Then, we compare our scheme with the existing privacy-preserving skyline query scheme that is close to our scheme. Since the performance of our scheme is related to the dataset size, the number of dimensions, and the domains, it is reasonable to synthesize a dataset varying with the above mentioned parameters to facilitate the observation of the impact of these parameters on our scheme. The synthesized dataset has 20 dimensions with 10000 tuples, in which ten of the dimensions have a domain size of 100, and the others have a domain size of 50. We implemented all schemes in Java and set the number of dummy dimensions $w = 2$. All experiments are conducted on 3.4 GHz Intel(R) CORE(TM) i7-3770 processes and Ubuntu OS with 16GB RAM.

TABLE 1
Token generation time (*ms*) of MPDPE varying with d

d	3	4	5	6	7	8	9	10
Domain T=100	45	110	216	360	572	854	1264	1792
Domain T=50	6	13	26	46	74	109	156	212

TABLE 2
Token generation time (*ms*) of MPDPE varying with k

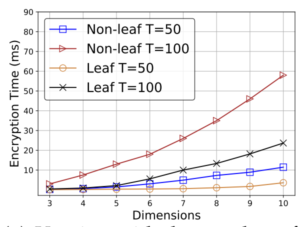
k	2	3	4	5	6	7	8	9
Domain T=100	1803	1803	1802	1803	1803	1802	1803	1803
Domain T=50	210	210	209	210	210	209	210	210

6.1 Performance of Our Proposed Scheme

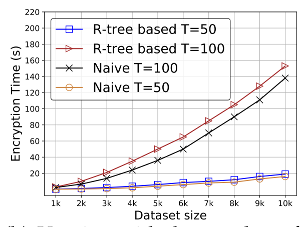
Our privacy-preserving skyline query scheme has four phases, namely, data outsourcing, token generation, skyline search, and original data recovery. For different phases, their performance will be affected by different parameters. For example, the performance of data outsourcing is related to the dataset, while the performance of token generation is related to the query workload. Therefore, in the following, we will evaluate the performance of these phases separately.

Data outsourcing. In the data outsourcing phase, our scheme builds an R-tree over plaintexts and then encrypts the R-tree, in which the leaf nodes are encrypted by MPDPE.Enc, and the non-leaf nodes are encrypted by HRIPE.Enc. First, we will evaluate the performance of encryption schemes applied on the leaf and non-leaf nodes. Then, we compare the R-tree based data outsourcing with the naive approach that only encrypts data records and outsources them to the cloud, which can clearly show the extra costs of R-tree for improving search efficiency. Fig. 3(a) depicts the computational costs of encrypting a leaf and non-leaf node varying with the number of dimensions. With the dimension growing, both the encryption time of the leaf and non-leaf node are increasing, especially, for the larger domain cases. It is straightforward since the secret key used in our scheme is a random matrix, and its size grows with the dimensions and domain increasing. Besides, we can see that encrypting non-leaf nodes is more expensive than leaf nodes. It is because that the secret key applied on non-leaf nodes has almost $4\times$ size than that of leaf nodes. Fig. 3(b) plots the computational costs of R-tree based data outsourcing and naive approach varying with the number of data records. Due to the operations of building R-tree and encrypting non-leaf nodes, the R-tree based data outsourcing inevitably incurs more computational costs than that of naive approach. However, they are necessary trade-offs for improving search efficiency.

Token generation. Our scheme has three query tokens generated by MPRPE.TokenGen, HRIPE.TokenGen, and MPDPE.TokenGen, respectively. First, we evaluate the computational costs for each token generation varying with the number of total dimensions d. Then, we explore whether the number of selected dimension k affects the performance of the token generation. Since the MPDPE.TokenGen algorithm needs to convert a query vector to the $\theta \times \theta$ query matrix Q and encrypt Q using two secret keys, this token generation requires more time than the other two tokens that

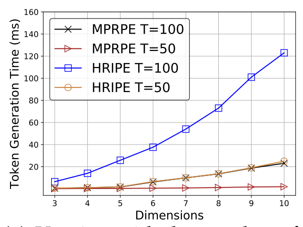


(a) Varying with the number of dimensions d.

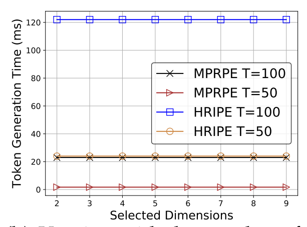


(b) Varying with the number of data records $|\mathcal{X}|$.

Fig. 3. Encryption time. T is domain size. (a) Encryption time per node varying with the number of dimensions d; (b) Total encryption time varying with the number of data records $|\mathcal{X}|$.

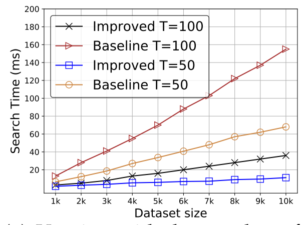


(a) Varying with the number of dimensions d.

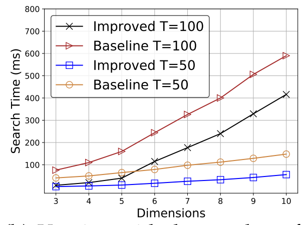


(b) Varying with the number of selected dimensions k.

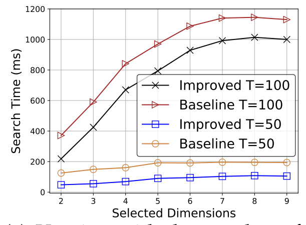
Fig. 4. Token generation time, T is domain size, and set $|\mathcal{X}|$ =10000. (a) Varying with the number of dimensions d. (b) Varying with the number of selected dimensions k.



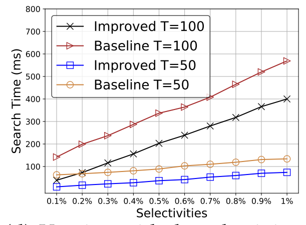
(a) Varying with the number of data records $|\mathcal{X}|$.



(b) Varying with the number of dimensions d.

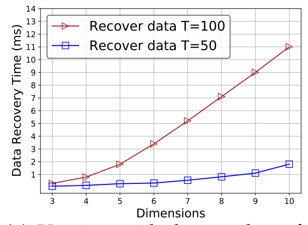


(c) Varying with the number of selected dimensions k.

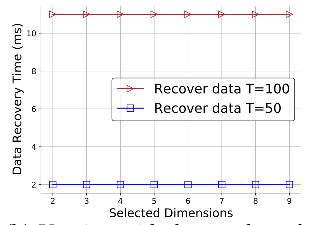


(d) Varying with the selectivity.

Fig. 5. User-defined skyline search time. T is domain size and set $|\mathcal{X}|$ =10000. (a) Varying with the number of data records $|\mathcal{X}|$. Keep d=5, k=3, and selectivity= 0.1%. (b) Varying with the number of dimensions d. Keep $|\mathcal{X}|$=10000, k=3, and selectivity= 0.1%. (c) Varying with the number of selected dimensions k. Keep $|\mathcal{X}|$=10000, d=10, and selectivity= 0.1%; (d) Varying with the selectivity. Keep $|\mathcal{X}|$=10000, d=5, and k=3.



(a) Varying with the number of dimensions d.



(b) Varying with the number of selected dimensions k.

Fig. 6. Original data recovery time per data record. T is domain size, and set $|\mathcal{X}|$ =10000. (a) Varying with the number of dimensions d. (b) Varying with the number of selected dimensions k.

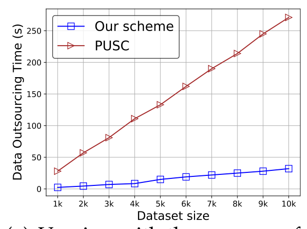
are encrypted only by one secret key. As a result, we plot the performance of MPRPE.TokenGen and HRIPE.TokenGen in Fig. 4 and list the performance of MPDPE.TokenGen in Table 1 and Table 2. From Fig. 4, we can see that HRIPE.TokenGen takes more time than MPRPE.TokenGen in token generation. It is because that the query vector in HRIPE.TokenGen is expanded to almost a $2d$-dimensional vector and then encrypted by the corresponding secret key. In addition, Fig. 4(b) shows that the number of selected dimensions has no impact on the performance of token generation. It is reasonable since we hide the information about the selected dimensions into query tokens, which protects this privacy from leakage.

User-defined skyline search. If the search efficiency is not considered, we can achieve the privacy-preserving user-defined skyline query just with MPRPE and MPDPE, which can be denoted as the *baseline scheme*. In the *baseline scheme*, the cloud server needs to check the points that fall within a hyper rectangle one by one. To improve efficiency, we propose HRIPE and employ it to support the multi-dimensional range queries on encrypted R-tree. We say the R-tree based skyline search scheme as the *improved scheme*. To clearly show the improvement, we evaluate the performance of the user-defined skyline search by comparing the *baseline scheme* and *improved scheme*, as shown in Fig. 5, in which we vary the parameters of the number of data records $|\mathcal{X}|$, the number of dimensions d, the number of selected dimensions k, and the selectivity. All of the evaluation results show that the *improved scheme* has better performance compared to the *baseline scheme*. It is intuitive since using R-tree to determine point-inside-rectangle is more efficient than checking points one by one. Both Fig. 5(a) and Fig. 5(d) demonstrate that the search time is linearly increasing with the corresponding parameters, i.e., the number of data records and the selectivity. However, Fig. 5(a) has an increasing difference, while the difference is stable for Fig. 5(d). The reason is that increasing the number of data records indicates filtering out more data records by checking the non-leaf nodes. While, for Fig. 5(d), since the data records are fixed, the differences are relatively stable with the growth of selectivities. Due to the similar reason, Fig. 5(b) also shows a stable difference between *baseline* and *improved* schemes varying with the number of dimensions. For Fig. 5(c), it shows a very different trend, i.e., the search time will keep stable when $k > 7$. It is because the more dimensions are selected, the less likely for the points to dominate each other. When the amount of data records returned stabilizes, the search time will tend to be stable.

Original data recovery. In data recovery phase, the query user needs to decrypt $||\mathrm{x}||_{\mathrm{L}}$ for recovering the original data record x. Fig. 6 shows the average data recovery time for each data record. Both Fig. 6(a) and Fig. 6(b) demonstrate that the data recovery time will increase with the domain size T growing. As similar reasons to the trend of the token generation time (in Fig. 4), the data recovery time increases with the number of dimensions growing and is not affected by the number of selected dimensions.

6.2 Comparison with Existing Scheme

In this section, we compare our scheme with PUSC [14], which is built upon the protocols proposed in [26]. To the best of our knowledge, PUSC is the only scheme that is close to our scheme since it achieves the privacy-preserving


(a) Varying with the numner of data records

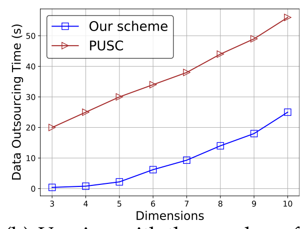
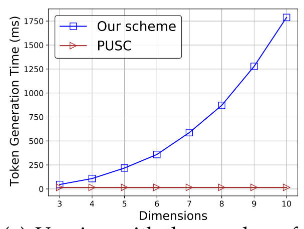

(b) Varying with the number of dimensions d

Fig. 7. Computational costs of data outsourcing. (a) Varying with the number of data records and set d=5, k=3; (b) Varying with the number of dimensions and set k=3.


(a) Varying with the number of dimensions d

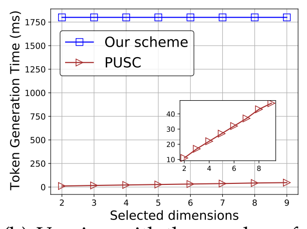
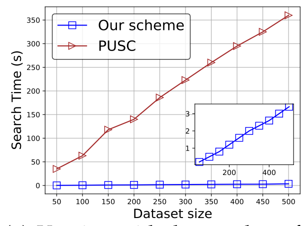
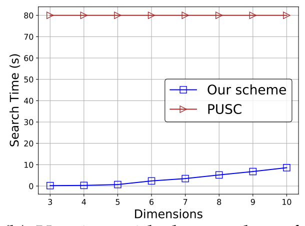

(b) Varying with the number of selected dimensions k

Fig. 8. Computational costs of token generation. (a) Varying with the number of dimensions and set k=3; (b) Varying with the number of selected dimensions and set d=10.


(a) Varying with the number of data records.


(b) Varying with the number of dimensions d

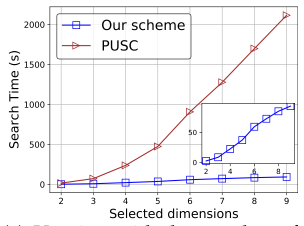
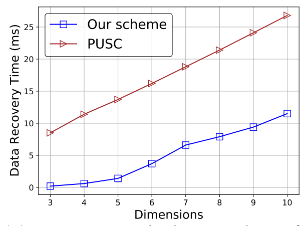

(c) Varying with the number of selected dimensions k

Fig. 9. Computational costs of searching user-defined skyline. (a) Varying with dataset size and set d=5, k=3; (b) Varying with the number of dimensions and set k=3; (c) Varying with the number of selected dimensions and set d=10.


(a) Varying with the number of dimensions d

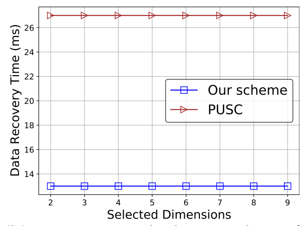

(b) Varying with the number of selected dimensions k

Fig. 10. Computational costs of original data recovery. (a) Varying with the number of dimensions and set k=3; (b) Varying with the number of selected dimensions and set d=10.

skyline queries allowing the users to select attributes and preferences. As PUSC cannot support the constrained region in our user-defined skyline queries, we exclude this feature from our scheme and adopt the aforementioned *baseline scheme* as our scheme. Both the schemes are evaluated on the dimensions with domain $\mathtt{T} = 100$ and the security parameter with 512. Since PUSC can also be divided into four phases, i.e., data outsourcing, token generation, skyline search, and original data recovery, we evaluate and compare the computational costs of our scheme and PUSC in these phases.

Data outsourcing. In this phase, the main operation of these two schemes is to encrypt data records before sending them to the cloud. PUSC adopts a double trapdoor public key encryption [27], which is more expensive than our predicate encryption. As a result, our scheme is much more efficient than PUSC in data outsourcing. Fig. 7(a) shows that the computational costs of both schemes are linearly increasing with the number of data records. However, PUSC increases sharply, and our scheme can achieve up to an order of magnitude better performance than PUSC in data outsourcing. Fig. 7(b) describes the computational costs of both schemes varying with the number of dimensions d. When d=10, our scheme is still at least 2× faster than PUSC.

Token generation. In Fig. 8, we depict the computational costs of both schemes in token generation. Fig. 8(a) shows that the computational cost of our scheme quadratically increases with the number of dimensions, while PUSC is independent of the number of dimensions. The essence reason is that our scheme generates and encrypts the query matrix $\mathrm{Q}^{\theta\times\theta}$, where $\theta = \sum_{i=1}^{d} \mathtt{T}_i + 3d + 3$, while PUSC only encrypts the selected dimensions to generate tokens. Fig. 8(b) plots a completely different trend. That is, our scheme remains stable with the number of selected dimensions, while PUSC increases. It indicates that, in the token generation phase, the performance of our scheme is related to the number of the total dimensions, while the performance of PUSC is related to the number of selected dimensions. In fact, PUSC leaks information about how many and which dimensions are selected, while our scheme hides this information by generating and encrypting the query matrix Q. Therefore, our scheme sacrifices the performance in token generation for improve security.

User-defined skyline search. Generally, the computational cost of searching skylines is the main concern for a privacy-preserving skyline scheme. Fig. 9 depicts the computational costs of our scheme and PUSC varying with the number of data records, dimensions, and selected dimensions. Fig. 9(a) shows that the costs of our scheme and PUSC linearly increase with the number of data records, and the performance of our scheme is always two orders of magnitude better than PUSC. Note that since PUSC is much slow in searching skyline points, we have to use a small number of data records from 50 to 500. As shown in Fig. 9(b), although PUSC remains stable when the number of dimensions increases, our scheme is at least 9× faster than PUSC. In Fig. 9(c), we can see that the costs in both schemes are increasing with the number of selected dimensions. Meanwhile, Fig 9(c) shows that our scheme can achieve up to an order of magnitude better performance than PUSC when $d = 10$. The reason why our scheme has such a significant performance advantage is that PUSC needs to decrypt ciphertexts to determine the order relation and apply homomorphic computations to obtain the dominating relations, while our scheme can directly determine the dominating relations by calculating the inner product of two ciphertexts.

Original data recovery. In the original data recovery phase, both our scheme and PUSC mainly involve the decryption operations. Fig. 10 shows the average computational costs of these two schemes in recovering one encrypted point. From Fig. 10(a), we can see that the computational costs of both schemes increase with the number of dimensions. However, our scheme is at least $2\times$ faster than PUSC in recovering an encrypted data. It is because that decrypting a ciphertext by matrix encryption is more efficient than the double trapdoor public key encryption. In Fig. 10(b), we plot the computational costs of these two schemes varying with the number of selected dimensions. As usual, our scheme keeps stable on this parameter. However, different from other phases, PUSC is independent of the number of selected dimensions in the data recovery phase. It is because, in this phase, PUSC needs to decrypt all dimensions' values and not just the selected dimensions. Besides, Fig. 10(b) shows that our scheme is always around $2\times$ faster than PUSH when $d = 10$.

7 Related work

Privacy-preserving multi-dimensional range queries. Achieving multi-dimensional range queries over encrypted data has been extensively studied. Boneh et al. [28] proposed a private multi-dimensional range query scheme by using the constructed hidden vector encryption, which can achieve a linear search efficiency. To further improve efficiency, Wang et al. [24] designed a sub-linear search scheme by integrating R-tree into Boneh et al.'s scheme [28]. However, both schemes cannot preserve query privacy. Shi et al. [23] presented a secure multi-dimensional range query scheme based on anonymous identity-based encryption. However, this scheme has a linear search efficiency and employs expensive public-key cryptography. In [29], Wang et al. combined the $\hat{\text{R}}$-tree and asymmetric scalar-product preserving encryption to support the multi-dimensional range query over encrypted data, while Mei et al. [30] adopted the interval tree. Although these schemes are efficient, they cannot preserve single-dimensional privacy. Based on the bucketization approach, Hore et al. [31] and Lee [32] separately proposed a secure multi-dimensional range query scheme to minimize the risk of disclosure while keeping query cost under a certain threshold value. Unfortunately, both schemes are in an approximate manner, i.e., the returned results contain false positives. Besides, some privacy-preserving multi-dimensional range query schemes [33], [34] were proposed based on order preserving encryption (OPE). However, the former leaks order relations for each dimension, while the latter cannot dynamically support range queries for any combination of dimensions and suffers from huge storage overheads. Recently, Yang et al. [17] proposed two privacy-preserving multi-dimensional range query schemes: TRQED and TRQED+, in which TRQED is over a single cloud, while TRQED+ is over a two-server setting. Although TRQED also uses matrix encryption, its prediction result is determined by every single dimension and thus is weak in terms of preserving single-dimensional privacy. Different from the above schemes, our privacy-preserving multi-dimensional range query scheme can preserve data privacy, query privacy, and single-dimension privacy while returning accurate results.

Privacy-preserving skyline queries. With the emergence of data outsourcing, privacy-preserving skyline queries have attracted considerable attention [10]–[14], [35], [36]. In [10], Bothe et al. proposed a prototype system *eSkyline* that enables the processing of skyline queries over encrypted data. This scheme uses the matrix encryption as the cryptographic primitive, and the key idea of this scheme is very different from our scheme since it does not consider the user-defined skyline queries. In [35], Liu et al. proposed a secure skyline computation scheme across multiple domains for supporting basic skyline computation. This scheme adopted a lightweight addition homomorphic encryption and designed secure protocols on a two-server model. In 2018, Liu et al. [11] designed a secure dynamic skyline query scheme over encrypted data. This work also employed the addition homomorphic encryption and the two-server model. In [12], Zheng et al. presented a privacy-preserving skyline computation scheme based on *Index* and leftist tree, which can perform skyline queries over merged encrypted data. Recently, Zhang et al. [13] adopted a variant of ElGamal encryption to encrypt data records and proposed a privacy-preserving probabilistic skyline computation to select workers. However, all of these schemes are achieved in a two-server model, which needs extra communication costs and cannot be applied to support the privacy-preserving user-defined skyline queries. Although the work in [36] designed a secure skyline computation scheme over the single cloud, it adopted the order-revealing encryption (ORE) as the cryptographic primitive, leading to the order information leakage in the single dimension. A closely related work was proposed by Liu et al. [14], which achieves a simple-version user-defined skyline query over encrypted data by employing the double trapdoors public-key cryptosystem. However, this scheme is also deployed in a two-server model and cannot be directly used in our user-defined skyline query scenario.

8 Conclusion

In this paper, we have proposed the first privacy-preserving user-defined skyline query scheme without an additional server. In particular, we first formally defined the user-defined skyline query. Then, we proposed three predicate encryption schemes, i.e., MPRPE, HRIPE, and MPDPE, to determine point-inside-rectangle relations, intersection relations, and dominating relations, respectively. Security analysis showed that these predicate encryption schemes are selectively secure, and our proposed scheme is indeed privacy-preserving under the defined security model. In addition, extensive performance experiments validated its efficiency. In our future work, we expect to hide the access pattern of our proposed scheme and further improve the performance in generating query tokens.

Acknowledgements

This research was supported in part by NSERC Discovery Grants (04009 and 03787), ZJNSF (LZ18F020003), NSFC (U1709217).

REFERENCES

[1] S. Borzsony, D. Kossmann, and K. Stocker, "The skyline operator," in *ICDE*. IEEE, 2001, pp. 421–430.

[2] K.-L. Tan, P.-K. Eng, B. C. Ooi *et al.*, "Efficient progressive skyline computation," in *VLDB*, 2001, pp. 301–310.

[3] D. Kossmann, F. Ramsak, and S. Rost, "Shooting stars in the sky: An online algorithm for skyline queries," in *VLDB*, 2002, pp. 275–286.

[4] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "Progressive skyline computation in database systems," *TODS*, pp. 41–82, 2005.

[5] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with presorting," in *ICDE*, vol. 3, 2003, pp. 717–719.

[6] P. Godfrey, R. Shipley, J. Gryz *et al.*, "Maximal vector computation in large data sets," in *VLDB*, 2005, pp. 229–240.

[7] I. Bartolini, P. Ciaccia, and M. Patella, "Salsa: Computing the skyline without scanning the whole sky," in *CIKM*, 2006, pp. 405–414.

[8] Y. Yuan, X. Lin, Q. Liu, W. Wang, J. X. Yu, and Q. Zhang, "Efficient computation of the skyline cube," in *VLDB*, 2005, pp. 241–252.

[9] R. C.-W. Wong, J. Pei, A. W.-C. Fu, and K. Wang, "Online skyline analysis with dynamic preferences on nominal attributes," *IEEE TKDE*, pp. 35–49, 2008.

[10] S. Bothe, P. Karras, and A. Vlachou, "eskyline: Processing skyline queries over encrypted data," *VLDB*, pp. 1338–1341, 2013.

[11] J. Liu, J. Yang, L. Xiong, and J. Pei, "Secure and efficient skyline queries on encrypted data," *IEEE TKDE*, pp. 1397–1411, 2018.

[12] Y. Zheng, R. Lu, B. Li, J. Shao, H. Yang, and K.-K. R. Choo, "Efficient privacy-preserving data merging and skyline computation over multi-source encrypted data," *Information Sciences*, vol. 498, pp. 91–105, 2019.

[13] X. Zhang, R. Lu, J. Shao, H. Zhu, and A. A. Ghorbani, "Secure and efficient probabilistic skyline computation for worker selection in mcs," *IEEE Internet of Things Journal*, pp. 11 524–11 535, 2020.

[14] X. Liu, K.-K. R. Choo, R. H. Deng, Y. Yang, and Y. Zhang, "Pusc: privacy-preserving user-centric skyline computation over multiple encrypted domains," in *TrustCom/BigDataSE*. IEEE, 2018, pp. 958–963.

[15] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data." in *NDSS*, 2015, p. 4325.

[16] X. Wang, J. Ma, F. Li, X. Liu, Y. Miao, and R. H. Deng, "Enabling efficient spatial keyword queries on encrypted data with strong security guarantees," *IEEE TIFS*, vol. 16, pp. 4909–4923, 2021.

[17] W. Yang, Y. Geng, L. Li, X. Xie, and L. Huang, "Achieving secure and dynamic range queries over encrypted cloud data," *IEEE TKDE*, 2020.

[18] D. Cash, J. Jaeger, S. Jarecki, C. S. Jutla, H. Krawczyk, M.-C. Rosu, and M. Steiner, "Dynamic searchable encryption in very-large databases: data structures and implementation." in *NDSS*, vol. 14. Citeseer, 2014, pp. 23–26.

[19] S. K. Kermanshahi, J. K. Liu, R. Steinfeld, S. Nepal, S. Lai, R. Loh, and C. Zuo, "Multi-client cloud-based symmetric searchable encryption," *IEEE TDSC*, 2019.

[20] J. Pei, W. Jin, M. Ester, and Y. Tao, "Catching the best views of skyline: A semantic approach based on decisive subspaces," *Space (X, Y)*, p. 1, 2005.

[21] A. Vlachou, C. Doulkeridis, Y. Kotidis, and M. Vazirgiannis, "Efficient routing of subspace skyline queries over highly distributed data," *IEEE TKDE*, pp. 1694–1708, 2009.

[22] E. Dellis, A. Vlachou, I. Vladimirskiy, B. Seeger, and Y. Theodoridis, "Constrained subspace skyline computation," in *CIKM*, 2006, pp. 415–424.

[23] E. Shi, J. Bethencourt, T. H. Chan, D. Song, and A. Perrig, "Multi-dimensional range query over encrypted data," in *IEEE Symposium on Security and Privacy*. IEEE, 2007, pp. 350–364.

[24] B. Wang, Y. Hou, M. Li, H. Wang, and H. Li, "Maple: scalable multi-dimensional range search over encrypted cloud data with tree-based index," in *ASIA CCS*, 2014, pp. 111–122.

[25] S. Lai, S. Patranabis, A. Sakzad, J. K. Liu, D. Mukhopadhyay, R. Steinfeld, S.-F. Sun, D. Liu, and C. Zuo, "Result pattern hiding searchable encryption for conjunctive queries," in *CCS*, 2018, pp. 745–762.

[26] X. Liu, R. H. Deng, K.-K. R. Choo, and J. Weng, "An efficient privacy-preserving outsourced calculation toolkit with multiple keys," *IEEE TIFS*, pp. 2401–2414, 2016.

[27] E. Bresson, D. Catalano, and D. Pointcheval, "A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications," in *ASIACRYPT*, 2003, pp. 37–54.

[28] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Theory of cryptography conference*. Springer, 2007, pp. 535–554.

[29] P. Wang and C. V. Ravishankar, "Secure and efficient range queries on outsourced databases using rp-trees," in *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. IEEE, 2013, pp. 314–325.

[30] Z. Mei, H. Zhu, Z. Cui, Z. Wu, G. Peng, B. Wu, and C. Zhang, "Executing multi-dimensional range query efficiently and flexibly over outsourced ciphertexts in the cloud," *Information Sciences*, vol. 432, pp. 79–96, 2018.

[31] B. Hore, S. Mehrotra, M. Canim, and M. Kantarcioglu, "Secure multidimensional range queries over outsourced data," *The VLDB Journal*, vol. 21, no. 3, pp. 333–358, 2012.

[32] Y. Lee, "Secure ordered bucketization," *IEEE Transactions on Dependable and Secure Computing*, vol. 11, no. 3, pp. 292–303, 2014.

[33] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order preserving encryption for numeric data," in *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, 2004, pp. 563–574.

[34] M. Kesarwani, A. Kaul, G. Singh, P. M. Deshpande, and J. R. Haritsa, "Collusion-resistant processing of sql range predicates," *Data Science and Engineering*, vol. 3, no. 4, pp. 323–340, 2018.

[35] X. Liu, R. Lu, J. Ma, L. Chen, and H. Bao, "Efficient and privacy-preserving skyline computation framework across domains," *Future Generation Computer Systems*, vol. 62, pp. 161–174, 2016.

[36] W. Wang, H. Li, Y. Peng, S. S. Bhowmick, P. Chen, X. Chen, and J. Cui, "Scale: An efficient framework for secure dynamic skyline query processing in the cloud," in *International Conference on Database Systems for Advanced Applications*. Springer, 2020, pp. 288–305.

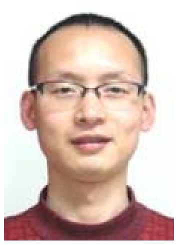
Songnian Zhang received his M.S. degree from Xidian University, China, in 2016 and he is currently pursuing his Ph.D. degree in the Faculty of Computer Science, University of New Brunswick, Canada. His research interest includes cloud computing security, big data query and query privacy.

Suprio Ray is an Associate Professor with the Faculty of Computer Science, University of New Brunswick, Fredericton, Canada. He received a Ph.D. degree from the Department of Computer Science, University of Toronto, Canada, in 2015. His research interests include big data and database management systems, run-time systems for scalable data science, provenance and privacy issues in big data and query processing on modern hardware. E-mail: sray@unb.ca

Rongxing Lu (S'09-M'11-SM'15-F'21) is a University Research Scholar, an associate professor at the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Canada. Before that, he worked as an assistant professor at the School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore from April 2013 to August 2016. Rongxing Lu worked as a Postdoctoral Fellow at the University of Waterloo from May 2012 to April 2013. He was awarded the most prestigious "Governor General's Gold Medal", when he received his PhD degree from the Department of Electrical & Computer Engineering, University of Waterloo, Canada, in 2012; and won the 8th IEEE Communications Society (ComSoc) Asia Pacific (AP) Outstanding Young Researcher Award, in 2013. Dr. Lu is an IEEE Fellow. His research interests include applied cryptography, privacy enhancing technologies, and IoT-Big Data security and privacy. He has published extensively in his areas of expertise (with H-index 76 from Google Scholar as of October 2021), and was the recipient of 9 best (student) paper awards from some reputable journals and conferences. Currently, Dr. Lu serves as the Chair of IEEE ComSoc CIS-TC (Communications and Information Security Technical Committee), and the founding Co-chair of IEEE TEMS Blockchain and Distributed Ledgers Technologies Technical Committee (BDLT-TC). Dr. Lu is the Winner of 2016-17 Excellence in Teaching Award, FCS, UNB.

Yandong Zheng received her M.S. degree from the Department of Computer Science, Beihang University, China, in 2017 and she is currently pursuing her Ph.D. degree in the Faculty of Computer Science, University of New Brunswick, Canada. Her research interest includes cloud computing security, big data privacy and applied 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.



Jun Shao received the Ph.D. degree from the Department of Computer Science 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.