Achieving Efficient and Privacy-Preserving Reverse Skyline Query Over Single Cloud

Yubo Peng, Xiong Li, *Senior Member, IEEE*, Ke Gu, *Member, IEEE*, Jinjun Chen, *Fellow, IEEE*, Sajal K. Das, *Fellow, IEEE*, and Xiaosong Zhang

***Abstract*—Reverse skyline query (RSQ) has been widely used in practice since it can pick out the data of interest to the query vector. To save storage resources and facilitate service provision, data owners usually outsource data to the cloud for RSQ services, which poses huge challenges to data security and privacy protection. Existing privacy-preserving RSQ schemes are either based on a two-cloud model or cannot fully protect privacy. To this end, we propose an efficient privacy-preserving reverse skyline query scheme over a single cloud (ePRSQ). Specifically, we first design a privacy-preserving inner product's sign determination scheme (PIPSD), which can determine whether the inner product of two vectors satisfies a specific relation with 0 without leaking the vectors' information. Next, we propose a privacy-preserving reverse dominance checking scheme (PRDC) based on symmetric homomorphic encryption. Finally, we achieve ePRSQ based on PIPSD and PRDC. Security analysis shows that PIPSD and PRDC are both secure in the real/ideal world model, and ePRSQ can protect the security of the dataset, the privacy of query requests and query results. Extensive experiments show that ePRSQ is efficient. Specifically, for a 3-dimensional dataset of size 1000, the computational and communication overheads of ePRSQ for a query are 79.47 s and 0.0021 MB, respectively. The efficiency is improved by $3.78\times$ (300.58 s) and $928.57\times$ (1.95 MB) respectively compared with PPARS, and by $61.31\times$ (4872.55 s) and $407309\times$ (855.35 MB) respectively compared with OPPRS.**

***Index Terms*—Privacy-preserving, reverse dominance, reverse skyline, single cloud, symmetric homomorphic encryption.**

Received 5 April 2024; revised 1 September 2024; accepted 17 October 2024. Date of publication 29 October 2024; date of current version 26 November 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62332018 and Grant 62072078, and in part by the Open project of the Key Laboratory of Data Protection and Intelligent Management, Ministry of Education, Sichuan University, under Grant SCUSAKFKT202303Z. The work of Sajal K. Das was supported by the U.S. National Science Foundation under Grant ECCS-2319995, Grant OAC-2104078, Grant CNS-2030624, and Grant 2150210. Recommended for acceptance by S. Salihoglu. *(Corresponding author: Xiong Li.)*

Yubo Peng and Xiaosong Zhang are with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: kyrixleven@gmail.com; johnsonzxs@uestc.edu.cn).

Xiong Li is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China, also with the Key Laboratory of Data Protection and Intelligent Management, Ministry of Education, Sichuan University, Chengdu 610065, China, and also with the Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China, Shenzhen 518000, China (e-mail: lixiongzhq@163.com).

Ke Gu is with the School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 410114, China (e-mail: gk4572@163.com).

Jinjun Chen is with the Swinburne University of Technology, Melbourne, VIC 3122, Australia (e-mail: jinjun.chen@gmail.com).

Sajal K. Das is with the Department of Computer Science, Missouri University of Science and Technology, Rolla, MO 65409 USA (e-mail: sdas@mst.edu).

Digital Object Identifier 10.1109/TKDE.2024.3487646

I. Introduction

SKYLINE query is a data filtering technique that helps users pick out points of interest from a multi-dimensional dataset. Therefore, it is widely used in data mining [1], multidimensional decision-making [2], and other related fields [3], [4], [5], [6]. As a typical skyline query, the reverse skyline query (RSQ) can pick out the data vectors interested in the query request from the dataset. So, it has been applied to real-world applications such as environmental monitoring [7] and advertising push services [8]. For example, when a company is preparing to launch a new product, it can request an RSQ service from the server that stores historical user consumption data, and select user groups that are likely to purchase the new product for advertising. In this scenario, the reverse skyline query only needs to obtain the user IDs without other additional information.

To provide skyline query services, the data owner generally needs to maintain the dataset and stay online, which leads to high costs for building and maintaining the corresponding facilities. Since cloud computing can provide users with powerful computing and storage services, more and more data owners prefer to outsource their dataset and corresponding services (e.g., RSQ service) to the cloud to reduce costs. However, data outsourcing inevitably raises security and privacy concerns. When data is outsourced to the cloud server, the data owner loses physical control of the data, and the data is exposed to threats such as malicious tampering, deletion, and privacy leakage. At the same time, the user's query request and the query results returned by the server may reveal the user's behavior patterns and personal information, thereby exposing sensitive information. Therefore, the security of outsourced data and the privacy protection of query requests and query results are crucial. Although techniques such as fully homomorphic encryption show the potential for security and privacy protection, the increased computational and communication overheads make them unsuitable for practical applications. Balancing privacy protection and efficiency in skyline queries is also a core challenge.

A. Related Work

To address the above issues, the research community has introduced several privacy-preserving skyline query schemes. We review the related work from static skyline query, dynamic skyline query, and reverse skyline query, respectively. Besides, Table I lists the comparisons of security and privacy issues among some mainstream solutions.

TABLE I
COMPARISONS OF SOME MAINSTREAM SOLUTIONS FOR SKYLINE QUERY

Properties	Ours	Scheme [9]	Scheme [10]	Scheme [11]	Scheme [12]	Scheme [13]
Type of skyline query	Reverse	Reverse	Reverse	Dynamic	Dynamic	Static
Security of data	√	×	√	√	√	√
Privacy of query requests	√	√	√	√	√	√
Privacy of query results	√	√	√	×	√	√
Number of server	1	1	2	2	1	1
Query efficiency	High	Middle	Low	Low	Low	Low

Privacy-preserving Static Skyline Query: This is a basic type of skyline query that finds the data vectors of interest to the origin. In 2016, Chen et al. [14] proposed three secure skyline query schemes for the location databases. However, their main purpose is verification rather than privacy protection, and the schemes are suitable only for 1- or 2-dimensional data. In 2019, Zheng et al. [15] proposed a skyline computation protocol based on determining the dominance relationship of public key encrypted data, but it will leak single-dimensional privacy. Liu et al. [16] proposed a secure user-defined skyline computation scheme. Due to the high cost of encryption, this scheme incurs significant overhead when applied to high-dimensional data. Zhang et al. [13] designed a secure user-defined skyline query scheme based on matrix encryption. It uses a bitmap-like way to represent a number, making it difficult to support queries of data with large values. In [17], a privacy-preserving basic skyline query scheme was proposed using symmetric homomorphic encryption (SHE) to protect data security over the two-cloud model. Compared to the previous static skyline query scheme over the two-cloud model, this scheme reduces communication overhead. Since static skyline queries can only statically pick out points of interest to the origin from the dataset, their applications are limited. Dynamic skyline query, with adjustable query vectors, can pick out data according to user requirements to better meet application needs.

Privacy-preserving Dynamic Skyline Query: In 2017, Liu et al. [18] proposed a fully secure dynamic skyline query scheme, which guarantees data security, and query and result privacy. However, its performance is poor due to the inefficiency of its basic protocol. Soon after, this scheme was improved in [19] based on parallelization technology. In 2020, based on order-revealing encryption (ORE), a secure dynamic skyline query scheme was proposed [20]. Even though the adversary may not know the plaintext of the data, the result of ORE can reveal the one-dimensional privacy of the data. In the same year, Wang et al. [21] proposed a privacy-preserving dynamic skyline query scheme based on Intel Software Guard Extensions (SGX). This scheme has excellent query efficiency but requires additional hardware support and leaks the access pattern of a query. In 2021, Zeighami et al. [12] proposed a secure dynamic skyline query by using result materialization and ORE. This approach takes less time but will cause huge storage overhead when the data has more than two dimensions. Zhang et al. [11] designed a privacy-preserving dynamic skyline query scheme using SHE over a two-cloud model. In [22], a secure skyline query protocol was constructed over a two-cloud model by combining three privacy protection technologies.

Privacy-preserving Reverse Skyline Query: Both static and dynamic skyline queries are designed from the users' perspective so that users can find products that match their interests. To help the producers find out which users are interested in a particular product, the reverse skyline query is proposed. The definition and query methods of the reverse skyline are similar to those of the dynamic skyline query, although their purposes are different. In recent years, two privacy-preserving reverse skyline query schemes have been proposed. Zhang et al. [10] designed two privacy-preserving reverse skyline search algorithms over the two-cloud model based on the Paillier cryptosystem. To weaken the demand for the two-cloud model, they further proposed an aggregate reverse skyline query over a single-cloud model by using a bloom filter and fully homomorphic encryption [9]. It can protect the private information of query requests and results. However, it not only needs to store plaintexts on the server but also brings heavy overheads.

B. Motivation

Although existing schemes provide various solutions, they suffer from the following limitations:

1) *Reliance on an impractical two-cloud model:* Most schemes [10], [17], [18] utilize homomorphic encryption to achieve secure skyline queries. Since it is difficult to directly compare ciphertexts and other operations in a single cloud, they usually rely on a non-colluding two-cloud model. However, the requirement of non-colluding two cloud servers is difficult to achieve in reality.
2) *Incomplete data security and privacy protection:* Some schemes directly store the plaintext data on the cloud server, which cannot guarantee data security [9], while some other schemes cannot protect privacy, e.g., the ORE-based schemes may leak the single-dimensional privacy [20].
3) *Inefficiency:* How to efficiently realize privacy-preserving skyline queries is a challenging problem. Some schemes require more communication overhead [12], and others have high computational costs [13].

C. Main Idea and Contributions

The above limitations motivated us to explore privacy-preserving reverse skyline query (PRSQ) on a single cloud server. Generally, *compute-then-compare* is essential in PRSQ, and it involves two steps: (1) *Compute*. Based on the encrypted data and the received token, the cloud server computes two ciphertexts used to determine the skyline dominance relation;

(2) *Compare*. The cloud server outputs the reverse skyline query result by comparing the relationship of the plaintexts corresponding to the two ciphertexts through a secure comparison protocol. However, if both of the above steps are performed by a single server, the specific value of the ciphertext used to determine the skyline dominance relationship can be inferred by running the secure comparison protocol multiple times. Therefore, most solutions achieve privacy protection by using an impractical two-cloud model to separate computation and comparison. To address this challenge, our idea is that the data owner and query user locally obfuscate their data separately. Next, the cloud server calculates an intermediate result using the obfuscated data of the two parties without leaking the privacy information of the two parties. After that, the cloud server obtains the encrypted final result based on the intermediate result and the homomorphic ciphertext generated by the query user. Finally, we realize an efficient PRSQ on a single cloud server. Based on the above ideas, we make the following contributions:

1) We propose a *privacy-preserving inner product sign determination scheme* (PIPSD) to efficiently and confidentially determine if the inner product of two vectors satisfies a specific relationship with 0.
2) Based on PIPSD, we design a *privacy-preserving reverse dominance check scheme* (PRDC) to determine the dominance relationship without revealing the data plaintext and query vector.
3) Based on PIPSD and PRDC, we realize an *efficient PRSQ scheme on the single-cloud model, named ePRSQ*. It is proven to protect the dataset security, the privacy of query requests and corresponding query results.
4) Detailed experimental evaluations and comparisons show that ePRSQ is efficient in computation and communication. Taking a 3-dimensional dataset with size 1000 as an example, the query efficiency of ePRSQ outperforms related schemes [9], [10] by $3.78\times$ and $61.31\times$ in computational cost, and by $928.57\times$ and $407309\times$ in communication overhead, respectively.

The rest of this paper is organized as follows. Section II introduces some preliminaries used in our scheme. Section III describes the system model, security model, and design goals. Two basic protocols PIPSD, PRDC, and our ePRSQ are presented in Section IV. Security analysis and performance evaluation of ePRSQ are provided in Sections V and VI, respectively. Conclusions are offered in Section VII.

II. Preliminaries

In the following, we introduce the formal definition of RSQ and the tool of SHE that is used in our work.

A. Reverse Skyline Query

The reverse skyline query (RSQ) [10], [23], [24], [25] is used to pick out those vectors whose dynamic skylines contain a given query vector. Its core operation is to check whether there is a reverse dominance relationship between two data points.

Definition 1 (Reverse Dominance): Given two d-dimensional data vectors x_u, x_v and a d-dimensional query vector q, we say x_u dominates q with regard to x_v in reverse skyline query, denoted as $x_u \prec_{x_v} q$, if:

$$\forall i \in [1, d] : |x_{ui} - x_{vi}| \leq |q_i - x_{vi}|, \tag{1}$$

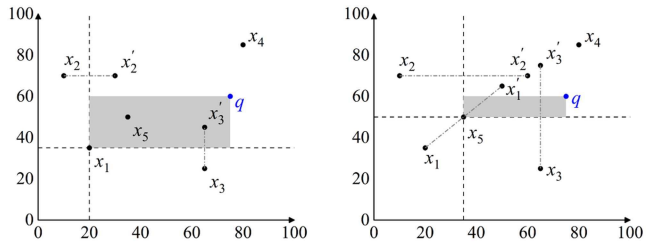
and

$$\exists i \in [1, d] : |x_{ui} - x_{vi}| < |q_i - x_{vi}|, \tag{2}$$

where x_{ui}, x_{vi} and q_i denote the i-th element in data vectors x_u, x_v and q, respectively.

Definition 2 (Reverse Skyline Query (RSQ)): Given a dataset χ, which contains multiple d-dimensional vectors, and a query vector q, a vector $x_u \in \chi$ is a reverse skyline vector of q iff there is no $x_v \in \chi$ such that $x_v \prec_{x_u} q$. RSQ returns a set S_q containing all reverse skyline vectors regarding q in χ, i.e., $S_q = \{x_i \in \chi \mid \nexists x_j \in \chi, x_j \prec_{x_i} q\}$.

Example 1: Fig. 1 illustrates an example of an RSQ process, where all the data in the dataset $\chi = \{x_i \mid 1 \leq i \leq 5\}$ and the query vector q are two-dimensional. Therefore, we can represent them as points on a plane. Fig. 1(a) and (b) depict the reverse dominance regarding x_1 and x_5, respectively. According to the above definition, x_1 is not a reverse skyline vector of the query vector q while x_5 is. To visually compare absolute values, all points in χ are mapped to a region with x_i as the origin, and a rectangular area constructed by x_i and q is marked in Fig. 1. If there exists a mapped point x_j that falls in the rectangle, it means x_j dominates q, i.e., $x_j \prec_{x_i} q$. Otherwise, x_i is a reverse skyline point. As shown in Fig. 1(a), x_5 and x_3' fall in the rectangle, implying x_5 and x_3' dominate q, and x_1 is not a reverse skyline point. Since no point falls in the rectangle in Fig. 1(b), x_5 is a reverse skyline point. By checking all points in χ, we obtain the set of reverse skyline points as $S_q = \{x_2, x_3, x_4, x_5\}$.



(a) x_1 is not a reverse skyline point regarding q

(b) x_5 is a reverse skyline point regarding q

Fig. 1. An example of RSQ.

B. Symmetric Homomorphic Encryption (SHE)

SHE [26] is a symmetric fully homomorphic encryption scheme and is proved to be semantically secure under Chosen Plaintext Attack (IND-CPA) [27]. The SHE scheme mainly consists of the following three algorithms:

- KeyGeneration(k_0, k_1, k_2): Given three security parameters k_0, k_1, k_2 satisfying $k_1 \ll k_2 < (k_0/2)$, the algorithm selects two large random prime numbers p, $\hat{p}$ and a random number $\mathcal{L}$ satisfying $\|p\| = \|\hat{p}\| = k_0$ and $\|\mathcal{L}\| = k_2$, respectively. Here, k_1 denotes the space of message m, that is $\|m\| = k_1$. Finally, the secret key sk is $(p, \mathcal{L})$ and the public parameter pk is $N = p\hat{p}$.

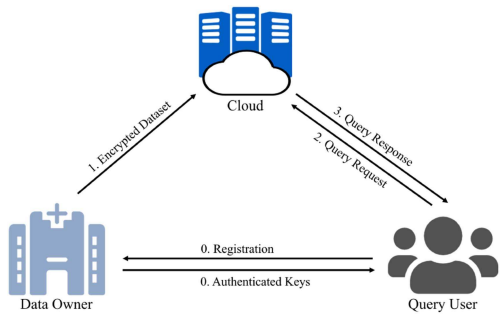


Fig. 2. System Model.

- Encryption(sk, m): The encryption algorithm takes the secret key sk and a message m as input, outputs the ciphertext $E(m) = (r\mathcal{L} + m)(1 + r'p) \bmod N$, where r and r' are two random numbers satisfying $\|r\| = k_2$ and $\|r'\| = k_0$.
- Decryption$(sk, E(m))$: Given a SHE ciphertext $E(m)$, this algorithm first generates a message $m' = (E(m) \bmod p) \bmod \mathcal{L}$ according to the secret key sk. Then, the algorithm recovers the message m as follows. If $m' < \frac{\mathcal{L}}{2}$, $m = m'$. Otherwise, $m = m' - \mathcal{L}$.

The following homomorphic properties hold for SHE:

- *Ciphertext-Ciphertext Homomorphic Addition:* $E(m_1) + E(m_2) \bmod N \Leftrightarrow E(m_1 + m_2)$;
- *Ciphertext-Plaintext Homomorphic Addition:* $E(m_1) + m_2 \bmod N \Leftrightarrow E(m_1 + m_2)$;
- *Ciphertext-Ciphertext Homomorphic Multiplication:* $E(m_1) \cdot E(m_2) \bmod N \Leftrightarrow E(m_1 m_2)$;
- *Ciphertext-Plaintext Homomorphic Multiplication:* $E(m_1) \cdot m_2 \bmod N \Leftrightarrow E(m_1 m_2)$ when $m_2 > 0$.

III. Models and Design Goals

In this section, we describe the system model, the security model, and the design goals of ePRSQ.

A. System Model

Fig. 2 depicts the system model of reverse skyline query over a single cloud for multi-dimensional data, which contains three entities: a data owner ($\mathcal{DO}$), a cloud server ($\mathcal{CS}$), and multiple query users ($\mathcal{QU}$).

- $\mathcal{DO}$: It holds a d-dimensional dataset of n data vectors, $\mathcal{D} = \{x_i = (x_{i1}, x_{i2}, \ldots, x_{id})\}_{i=1}^{n}$. All values in the dataset are assumed to be integers, which is easily satisfied. $\mathcal{DO}$ aims to provide a reverse skyline query service to take full advantage of the dataset. To save costs, $\mathcal{DO}$ outsources the dataset $\mathcal{D}$ and corresponding RSQ service to a $\mathcal{CS}$. To protect the security of dataset $\mathcal{D}$, $\mathcal{DO}$ encrypts the dataset before outsourcing it to $\mathcal{CS}$.
- $\mathcal{CS}$: The cloud server, which has powerful computing capability and sufficient storage resources, is responsible for storing $\mathcal{DO}$'s encrypted dataset and providing reverse skyline query service to $\mathcal{QU}$. When receiving a query request Req from a $\mathcal{QU}$, $\mathcal{CS}$ searches the corresponding reverse skyline regarding Req in the dataset $\mathcal{D}$ and replies the query result ciphertext S_{sky} to $\mathcal{QU}$.
- $\mathcal{QU}$: The model includes a set of query users. To enjoy the RSQ service, $\mathcal{QU}$ needs to register with $\mathcal{DO}$ first. Then, $\mathcal{QU}$ gets an authorized key and can access the reverse skyline query service with it.

B. Security Model

In the security model, we assume that $\mathcal{DO}$ is trustworthy, which means that he/she will honestly outsource an encrypted dataset and provide a valid private key to an authorized $\mathcal{QU}$. Meanwhile, we assume that $\mathcal{CS}$ is *honest-but-curious* [28], [29], i.e., it executes the protocol honestly but is curious about certain private information such as the plaintext of the dataset, query requests, and query results. We also assume that $\mathcal{QU}$ is *honest-but-curious*, since $\mathcal{QU}$ wants to get the correct query results so that he will not disobey the protocol. However, $\mathcal{QU}$ may eavesdrop on other users' query requests and results. We assume that $\mathcal{CS}$ and $\mathcal{QU}$ do not collude. First, it is reasonable for $\mathcal{CS}$ not to collude with $\mathcal{QU}$ to maintain its reputation as a cloud service provider. In addition, the cost for the query user to corrupt the cloud server is high, as the legal risk and huge economic cost. Furthermore, such a non-collusion model is widely used in various schemes [30], [31], [32].

Besides the above security assumption, we review the real/ideal world model [33], [34] with a static semi-honest adversary [29] used for security proof.

Real world model: In this model, the scheme Π will be executed between $\mathcal{CS}$ and an adversary $\mathcal{A}$. Assume that x is an input of Π and y is an auxiliary input. The definition of execution of Π with inputs x and y under $\mathcal{A}$ in this model is denoted as follows:

$$\text{REAL}_{\Pi,\mathcal{A},y}(x) \stackrel{def}{=} \{\text{Output}^{\Pi}(x), \text{View}^{\Pi}(x), y\},$$

where $\text{Output}^{\Pi}(x)$ is the output of Π executed on input x, and $\text{View}^{\Pi}(x)$ is the view of $\mathcal{CS}$ during the execution of Π with input x.

Ideal world model: In this model, $\mathcal{CS}$ only interacts with $\mathcal{F}$, which is defined as an ideal functionality for a leakage function $\mathcal{L}$. The definition of an execution $\mathcal{F}$ with inputs x and y under a simulator Sim is as follows:

$$\text{IDEAL}_{\mathcal{F},\mathcal{A},y}(x) \stackrel{def}{=} \{\mathcal{F}(x), \text{Sim}(x, \mathcal{L}(x)), y\}.$$

Definition 3 (Security against Semi-honest Adversary): Assume that $\mathcal{F}$ is a deterministic functionality and Π is a scheme of $\mathcal{CS}$, Π securely achieves $\mathcal{F}$ if there exists a $\text{Sim}(\mathcal{A})$ that satisfies probabilistic polynomial time (PPT) transformations such that:

$$\text{REAL}_{\Pi,\mathcal{A},y}(x) \stackrel{c}{\equiv} \text{IDEAL}_{\mathcal{F},\mathcal{A},y}(x),$$

where $\mathcal{A}$ is a semi-honest PPT adversary and $\stackrel{c}{\equiv}$ represents the computational indistinguishability.

C. Design Goals

In this work, we present an efficient PRSQ scheme over a single cloud. Specifically, our work is dedicated to the following goals.

TABLE II
NOTATIONS

Notation	Description
$\mathcal{D}, n, d$	The dataset, its size and dimensions
x_{ui}	The i-th dimensional data in u-th data x_u
q_i	The i-th dimensional data in query q
q_t	The query token
$A \prec_B q$	A dominates q with regard to B
T	Upper bound of the absolute value
$\langle \cdot, \cdot \rangle, \delta$	The inner product of two vectors
$[\![\cdot]\!]_i$	Obfuscated vector generated by Obf_i
∇	Condition (e.g., $\leq$)
$\| \cdot \|$	Length of data
K_i	Specific length of data
Δ	Intermediate result of inner product
$\mathcal{L}, p$	Big prime numbers
$E(\cdot)$	SHE ciphertext
pk, sk	Public key and secret key of SHE
$E(\tau)$	Sum of $E(\delta_i)$
ψ, γ, η	Intermediate query vector
$AE(\cdot)$	A secure asymmetric encryption algorithm
pk_{AE}, sk_{AE}	Public key and secret key of $AE(\cdot)$
PP	Public parameters generated by PIPSD
S_{key}	An encrypted query result
S_{ID}	A set that contains reverse skyline's IDs

- *Data security:* The security of the data can be guaranteed, i.e., any adversary cannot reveal the plaintext of the data.
- *Privacy preservation:* The privacy of query requests and query results can be protected, and any adversary cannot infer $\mathcal{QU}$s' private information.
- *Efficiency:* Generally, PRSQ schemes are inefficient due to high computational costs and communication overhead. Therefore, our work aims to improve efficiency while ensuring security and privacy.

IV. The Proposed Schemes

In the following, we first design a privacy-preserving inner-product sign determination scheme (PIPSD). Then, we propose a privacy-preserving reverse dominance checking scheme (PRDC) based on PIPSD. Finally, we develop our efficient PRSQ scheme (ePRSQ). The notations used throughout this paper are listed in Table II.

A. The PIPSD Scheme

This scheme is designed to securely determine whether the inner product of two vectors satisfies a specific relationship with 0. In particular, given two d-dimensional vectors $x = (x_1, x_2, \ldots, x_d)$, $y = (y_1, y_2, \ldots, y_d)$, and a condition ∇ (e.g., $\leq$), where x_i and y_i are integers for $1 \leq i \leq d$, PIPSD can determine whether $\langle x, y \rangle \nabla 0$ holds without revealing x, y, and $\langle x, y \rangle$. It contains four algorithms as described below.

1) *Setup*(d, T)*:* Given the dimension of the vector d and the upper limit T of the absolute value of the vector's elements, this algorithm outputs public parameters $PP = \{K_0, K_1, K_2\}$, where K_0 is the bit length of T and $K_1 > K_2 \gg \frac{K_1 + \|d\| + 1}{2} + K_0$. Besides, a secret random invertible matrix $M \in \mathbb{R}^{(d+1)\times(d+1)}$ is generated.

Algorithm 1: Obfuscation Algorithm 1.

Input: PP, x and M.
Output: A $(d+1)$-dimensional obfuscated vector $[\![x]\!]_1$.
$\alpha = RandomOfBitLength(K_1)$;
$\beta = RandomOfBitLength(K_2)$;
for $i \in [1, d]$ **do**
  $r_i = RandomOfBitLength(K_0)$;
end for
$[\![x]\!]_1 = (\alpha x_1 + r_1, \alpha x_2 + r_2, \ldots, \alpha x_d + r_d, \beta) \cdot M$
return $[\![x]\!]_1$;

Algorithm 2: Obfuscation Algorithm 2.

Input: PP, ∇, y and M^{-1}.
Output: A $(d+1)$-dimensional obfuscated vector $[\![y]\!]_2$.
$s = \begin{cases} 1, & \nabla \text{ is } > \text{ or } \geq; \\ -1, & \nabla \text{ is } < \text{ or } \leq. \end{cases}$
$s' = \begin{cases} 1, & \nabla \text{ is } \leq \text{ or } \geq; \\ -1, & \nabla \text{ is } < \text{ or } >. \end{cases}$
$\alpha' = RandomOfBitLength(K_1)$
$\beta' = RandomOfBitLength(K_2)$;
for i in $[1, d]$ **do**
  $r'_i = RandomOfBitLength(K_0)$;
end for
$[\![y]\!]_2 = M^{-1} \cdot (s\alpha' y_1 + r'_1, s\alpha' y_2 + r'_2, \ldots, s\alpha' y_d + r'_d, s'\beta')^T$
return $[\![y]\!]_2$;

2) *Obf*$_1(PP, x, M)$*:* Given the public parameter PP, a secret matrix M and a d-dimensional vector $x = (x_1, x_2, \ldots, x_d)$, this algorithm obfuscates x to $[\![x]\!]_1$ as described in Algorithm 1, where $RandomOfBitLength(\cdot)$ is to generate a random number of a specified bit length.

3) *Obf*$_2(PP, \nabla, y, M^{-1})$*:* Given the public parameter PP, a condition ∇, the inverse matrix of M and a d-dimensional vector $y = (y_1, y_2, \ldots, y_d)$, this algorithm generates an obfuscated query vector $[\![y]\!]_2$ as described in Algorithm 2.

4) *Check*$(PP, [\![x]\!]_1, [\![y]\!]_2)$*:* Given an obfuscated vector $[\![x]\!]_1$ and an obfuscated query vector $[\![y]\!]_2$, the checking algorithm determines whether $\langle x, y \rangle \nabla 0$. The algorithm calculates $\delta = \langle [\![x]\!]_1, [\![y]\!]_2 \rangle$. $\delta > 0$ represents $\langle x, y \rangle \nabla 0$ holds, while $\delta < 0$ means $\langle x, y \rangle \nabla 0$ does not hold.

Theorem 1 (Correctness of PIPSD): On inputs $[\![x]\!]_1$ and $[\![y]\!]_2$, the PIPSD can correctly determine whether $\langle x, y \rangle \nabla 0$.

Proof: According to the definition of δ, we have

$$\begin{aligned} \delta &= \langle [\![x]\!]_1, [\![y]\!]_2 \rangle \\ &= s\alpha\alpha' \sum_{i=1}^{d} x_i y_i + s'\beta\beta' + \alpha \sum_{i=1}^{d} r'_i x_i + s\alpha' \sum_{i=1}^{d} r_i y_i \end{aligned}$$

TABLE III
RELATIONSHIP BETWEEN ∇, $\langle X, y\rangle$ AND δ

∇	s	s'	$\langle x, y\rangle$	δ
$>$	1	-1	$\mathbf{>0}$	$\mathbf{>0}$
			$=0$	<0
			<0	<0
$\geq$	1	1	$\mathbf{>0}$	$\mathbf{>0}$
			$\mathbf{=0}$	$\mathbf{>0}$
			<0	<0
$<$	-1	-1	>0	<0
			$=0$	<0
			$\mathbf{<0}$	$\mathbf{>0}$
$\leq$	-1	1	>0	<0
			$\mathbf{=0}$	$\mathbf{>0}$
			$\mathbf{<0}$	$\mathbf{>0}$

$$+\sum_{i=1}^{d} r_i r_i' = s\alpha\alpha'\langle x, y\rangle + s'\beta\beta' + \lambda, \tag{3}$$

where $\lambda = s\alpha\sum_{i=1}^{d} r_i' x_i + s\alpha'\sum_{i=1}^{d} r_i y_i + \sum_{i=1}^{d} r_i r_i'$.

Because the range of all numbers in the expansion of δ is known, we can infer that $\|\lambda\| \leq 2K_0 + K_1 + \|d\|$. Since $\|\beta\beta'\| \geq 2K_2 - 1 > 2K_0 + K_1 + \|d\|$, we can infer that $|\beta\beta'| > |\lambda|$. Thus, the sign of $\beta\beta'$ determines the sign of $\beta\beta' + \lambda$. Specifically, if $\beta\beta' < 0$ (> 0), $\beta\beta' + \lambda$ is always less than (more than) 0, regardless of the sign of λ. Similarly, since $K_1 > K_2$, i.e., $|\alpha| = |\alpha'| = K_1 > |\beta| = |\beta'| = K_2$, we have $|s\alpha\alpha'\langle x, y\rangle| > |s\alpha\alpha'| > |s'\beta\beta'| > |\lambda|$, where s and s' have the same bit length. Therefore, δ's sign is identical to $s\alpha\alpha'\langle x, y\rangle$ when $\langle x, y\rangle \neq 0$, and is same to $s'\beta\beta'$ when $\langle x, y\rangle = 0$.

According to these inferences, the relationship among ∇, $\langle x, y\rangle$ and δ can be listed in Table III. For instance, when ∇ is $\leq$, we know that $s = -1$ and $s' = 1$ as defined in Algorithm 2. If $\langle x, y\rangle = 0$, we know that $\delta > 0$ since δ has the same sign as $s'\beta\beta'$. If $\langle x, y\rangle < 0$, we have $\delta > 0$ since δ has the same sign as $s\alpha\alpha'\langle x, y\rangle$. Similarly, if $\langle x, y\rangle > 0$, $\delta < 0$. Thus, whether $\langle x, y\rangle \nabla 0$ can be determined by the sign of δ. Consequently, PIPSD's check result is correct. □

B. The PRDC Scheme

Based on PIPSD, we design a privacy-preserving reverse dominance checking scheme, denoted as PRDC. It can check whether there is a reverse dominance relationship between two data in a privacy-preserving manner. Specifically, given two d-dimensional data vectors x_u, x_v and a query vector q, PRDC can determine whether $x_u \prec_{x_v} q$ holds without revealing the original vector information. If so, it outputs a SHE ciphertext $E(\tau) = E(1)$. Otherwise, it outputs $E(\tau) = E(0)$. Next, we describe the transformation of the reverse dominance problem and then introduce the PRDC.

1) Transformation of the Reverse Dominance Problem: To guarantee privacy during the determination of reverse dominance, we transform the formulas for judging the reverse dominance ((1) and (2)) into the inner product of two vectors, which can be independently constructed by $\mathcal{DO}$ and $\mathcal{QU}$.

For (1), we have

$$\begin{aligned} & |x_{ui} - x_{vi}| \leq |q_i - x_{vi}| \\ \Leftrightarrow\ & (x_{ui} - x_{vi})^2 \leq (q_i - x_{vi})^2 \\ \Leftrightarrow\ & x_{ui}^2 - 2x_{ui}x_{vi} + 2q_i x_{vi} - q_i^2 \leq 0 \\ \Leftrightarrow\ & (x_{ui}^2 - 2x_{ui}x_{vi}, x_{vi}, 1)\cdot(1, 2q_i, -q_i^2)^T \leq 0 \end{aligned} \tag{4}$$

Therefore, (1) is equivalent to (4). The transformed (4) can be expressed as the product of two vectors $(x_{ui}^2 - 2x_{ui}x_{vi}, x_{vi}, 1)$ and $(1, 2q_i, -q_i^2)^T$, which can be constructed by $\mathcal{DO}$'s data (x_u and x_v) and $\mathcal{QU}$'s query vector q, respectively. Besides, we use $E(\delta_i)$ to indicate whether (4) holds. If (4) holds, $E(\delta_i) = E(1)$, otherwise, $E(\delta_i) = E(0)$.

If both (1) and (2) hold, according to (4), we have

$$\begin{aligned} & \sum_{i=1}^{d} |x_{ui} - x_{vi}| < \sum_{i=1}^{d} |q_i - x_{vi}| \\ \Leftrightarrow\ & \sum_{i=1}^{d} (x_{ui} - x_{vi})^2 < \sum_{i=1}^{d} (q_i - x_{vi})^2 \\ \Leftrightarrow\ & \sum_{i=1}^{d} (x_{ui}^2 - 2x_{ui}x_{vi}) + \sum_{i=1}^{d} 2q_i x_{vi} - \sum_{i=1}^{d} q_i^2 < 0 \\ \Leftrightarrow\ & \left(x_{u1}^2 - 2x_{u1}x_{v1}, \ldots, x_{ud}^2 - 2x_{ud}x_{2d}, x_{v1}, \ldots, x_{vd}, 1\right) \\ & \times \left(1, \ldots, 1, 2q_1, \ldots, 2q_d, -\sum_{j=1}^{d} q_j^2\right)^T < 0 \end{aligned} \tag{5}$$

Therefore, $\sum_{i=1}^{d} (x_{ui} - x_{vi})^2 < \sum_{i=1}^{d} (q_i - x_{vi})^2$ can be expressed as the inner product of two $(2d+1)$-dimensional vectors, i.e., $(x_{u1}^2 - 2x_{u1}x_{v1}, \ldots, x_{ud}^2 - 2x_{ud}x_{2d}, x_{v1}, \ldots, x_{vd}, 1)$ and $(1, \ldots, 1, 2q_1, \ldots, 2q_d, -\sum_{j=1}^{d} q_j^2)^T$, and they can also be constructed by $\mathcal{DO}$'s data (x_u and x_v) and $\mathcal{QU}$'s query vector q, respectively. Besides, we use $E(\delta_i)$ to indicate whether (5) holds. If (5) holds, $E(\delta_i) = E(1)$, otherwise, $E(\delta_i) = E(0)$.

Lemma 1: (1) ∧ (2) ⇔ (4) ∧ (5)

Proof: We can deduce easily that (1) ∧ (2) ⇒ (4) ∧ (5) since (1) ⇔ (4) and (1) ∧ (2) ⇒ (5). On the contrary, when (4) and (5) hold simultaneously, we can infer that (2) must hold. Besides, since (4) is equivalent to (1), we have (4) ∧ (5) ⇒ (1) ∧ (2). In summary, (4) ∧ (5) ⇔ (1) ∧ (2). □

Let $E(\tau) = \prod_{i=1}^{d+1} E(\delta_i)$. According to the above analysis, $x_u \prec_{x_v} q$ holds is equivalent to (4) ∧ (5) hold. (4) holds means that $E(\delta_i) = E(1)$ for all $1 \leq i \leq d$, while (5) holds mean that $E(\delta_{d+1}) = E(1)$. Therefore, when $E(\tau) = E(1)$, $x_u \prec_{x_v} q$ holds. Otherwise, if $E(\tau) = E(0)$, $x_u \prec_{x_v} q$ does not hold.

2) Description of PRDC Scheme: The PRDC scheme contains four algorithms, and we describe them as follows.

1) *Setup*(d, T)*:* Given the dimension d of the vector in $\mathcal{D}$ and the upper limit T of the absolute value of the vector's elements, the algorithm outputs the public parameters

Algorithm 3: PRDC Encryption Algorithm.

Input: PP, M_1, M_2, x_u and x_v.
Output: $[\![\chi(u,v)]\!]^*$.
1: $\chi(u,v) =$
$(x_{u1}^2 - 2x_{u1}x_{v1}, \ldots, x_{ud}^2 - 2x_{ud}x_{vd}, x_{v1}, \ldots, x_{vd}, 1)$
2: $[\![\chi(u,v)]\!]_1 = PIPSD.Obf_1(PP, \chi(u,v), M_1)$
3: $[\![\chi(u,v)]\!]^* = [\![\chi(u,v)]\!]_1 \cdot M_2$
4: **return** $[\![\chi(u,v)]\!]^*$;

Algorithm 4: PRDC Query Generation.

Input: PP, M_1^{-1}, M_2^{-1}, q and pk.
Output: A query token q_t.
1: **for** $i \in [1,d]$ **do**
2: $\eta(i)_{j\in[1,2d+1]} = \begin{cases} 1, & j = i; \\ 2q_i, & j = d+i; \\ -q_i^2, & j = 2d+1; \\ 0, & \text{Otherwise.} \end{cases}$
3: $[\![\eta(i)]\!]_2 = PIPSD.Obf_2(PP, \leq, \eta(i), M_1^{-1})$;
4: Select random number $s_i \in \{-1, 1\}$;
5: $E(\mu_i) = \begin{cases} E(0), & s_i = 1; \\ E(1), & s_i = -1. \end{cases}$
6: **end for**
7: $\gamma = (1, \ldots, 1, 2q_1, \ldots, 2q_d, -\sum_{i=1}^d q_i^2)$;
8: $[\![\gamma]\!]_2 = PIPSD.Obf_2(PP, <, \gamma, M_1^{-1})$;
9: $\psi = M_2^{-1} \cdot (s_1[\![\eta(1)]\!]_2^T, \ldots, s_d[\![\eta(d)]\!]_2^T, s_{d+1}[\![\gamma]\!]_2^T)$;
10: $q_t = \{\psi, \{E(\mu_i)\}_{i=1}^{d+1}\}$
11: **return** q_t;

$PP = PIPSD.Setup(2d+1, T^2)$ by calling the Setup algorithm of PIPSD. Besides, two secret random invertible matrices $M_1, M_2 \in \mathbb{R}^{(2d+2)\times(2d+2)}$ are generated.

2) *Enc*(x_u, x_v, PP, M_1, M_2)*:* Given any two d-dimensional data vectors x_u and x_v, the public parameter PP and two secret matrixes M_1, M_2, the encryption algorithm outputs a $(2d+2)$-dimensional obfuscated vector $[\![\chi(u,v)]\!]^*$ for x_u and x_v. The encryption algorithm is shown in Algorithm 3.

3) *QueryGen*$(q, PP, M_1^{-1}, M_2^{-1}, pk)$*:* Given a d-dimensional query vector q, the public parameter PP, two secret matrices M_1^{-1}, M_2^{-1} and SHE's public parameter pk, the query generation algorithm, which is shown in Algorithm 4, outputs a query token q_t.

4) *Check*$([\![\chi(u,v)]\!]^*, q_t, pk)$*:* Given the public parameter pk of SHE, an obfuscated vector $[\![\chi(u,v)]\!]^*$ and a query token q_t, the checking algorithm outputs an encrypted check result $E(\tau)$, where $E(\tau) = E(1)$ if $x_u \prec_{x_v} q$ holds, otherwise, $E(\tau) = E(0)$. The algorithm is shown in Algorithm 5.

Theorem 2 (Correctness of PRDC): On inputs $[\![\chi(u,v)]\!]^*$ and q_t, the PRDC can correctly determine whether $x_u \prec_{x_j} q$.

Proof: To prove the correctness of PRDC, it is essential to demonstrate that $E(\delta_i)$ accurately reflects the relationship among x_{ui}, x_{vi} and q_i as specified by (4) for each i from 1 to d. Similarly, the correctness of $E(\delta_{d+1})$ must be verified to ensure it accurately determines whether x_u, x_v, and q conform to (5).

Algorithm 5: PRDC Checking Algorithm.

Input: $[\![\chi(u,v)]\!]^*$, q_t and pk.
Output: An encrypted check result $E(\tau)$.
1: $\Delta = [\![\chi(u,v)]\!]^* \cdot \psi$
2: **for** $i \in [1,d]$ **do**
3: $E(\delta_i) = \begin{cases} E(1) - E(\mu_i), & \Delta_i > 0; \\ E(\mu_i), & \Delta_i < 0. \end{cases}$
4: **end for**
5: $E(\tau) = \prod_{i=1}^{d+1} E(\delta_i)$;
6: **return** $E(\tau)$;

Given the data ciphertext $[\![\chi(u,v)]\!]^*$ and a query token $q_t = \{\psi, \{E(\mu_i)\}_{i=1}^{d+1}\}$, Δ can be expanded as

$$\begin{aligned} \Delta &= [\![\chi(u,v)]\!]^* \cdot \psi \\ &= [\![\chi(u,v)]\!]_1 M_2 M_2^{-1}(s_1[\![\eta(1)]\!]_2^T, \ldots, s_d[\![\eta(d)]\!]_2^T, \\ &\quad s_{d+1}[\![\gamma]\!]_2^T) \\ &= (s_1\langle[\![\chi(u,v)]\!]_1, [\![\eta(1)]\!]_2\rangle, \ldots, s_d\langle[\![\chi(u,v)]\!]_1, [\![\eta(d)]\!]_2\rangle, \\ &\quad s_{d+1}\langle[\![\chi(u,v)]\!]_1, [\![\gamma]\!]_2\rangle). \end{aligned} \tag{6}$$

For $\Delta_i = s_i\langle[\![\chi(u,v)]\!]_1, [\![\eta(i)]\!]_2\rangle$, $1 \leq i \leq d$, since $[\![\eta(i)]\!]_2 = PIPSD.Obf_2(PP, \leq, \eta(i), M_1^{-1})$, we know that Δ_i is actually s_i multiplied by a PIPSD's checking result which indicates whether $\langle\chi(u,v), \eta(i)\rangle = (x_{ui}^2 - 2x_{ui}x_{vi}) \cdot 1 + x_{vi} \cdot (2q_i) + 1 \cdot (-q_i^2) = (x_{ui} - x_{vi})^2 - (q_i - x_{vi})^2 \leq 0$ holds. Assuming $(x_{ui} - x_{vi})^2 - (q_i - x_{vi})^2 \leq 0$ holds, we have $\langle[\![\chi(u,v)]\!]_1, [\![\eta(i)]\!]_2\rangle > 0$. If $s_i = 1$, we have $E(\mu_i) = E(0)$ and $\Delta_i > 0$. According to line 3 in Algorithm 5, we can deduce that $E(\delta_i) = E(1)$. If $s_i = -1$, we have $E(\mu_i) = E(1)$ and $\Delta_i < 0$, so $E(\delta_i) = E(1)$. In contrast, when $(x_{ui} - x_{vi})^2 - (q_i - x_{vi})^2 \leq 0$ does not hold, we have $\langle[\![\chi(u,v)]\!]_1, [\![\eta(i)]\!]_2\rangle < 0$. If $s_i = 1$, we have $E(\mu_i) = E(0)$ and $\Delta_i < 0$. Then, we can deduce that $E(\delta_i) = E(0)$. If $s_i = -1$, we have $E(\mu_i) = E(1)$ and $\Delta_i > 0$, so $E(\delta_i) = E(0)$. From the above analysis, we have proved that $E(\delta_i)(1 \leq i \leq d)$ correctly indicates whether x_{ui}, x_{vi} and q_i satisfy (4).

For $\Delta_{d+1} = s_{d+1}\langle[\![\chi(u,v)]\!]_1, [\![\gamma]\!]_2\rangle$, since $[\![\gamma]\!]_2 = PIPSD.Obf_2(PP, <, \gamma, M_1^{-1})$, we know that Δ_{d+1} is actually s_{d+1} multiplied by a PIPSD's checking result which indicates whether $\langle\chi(u,v), \gamma\rangle = \sum_{i=1}^d (x_{ui}^2 - 2x_{ui}x_{vi}) \cdot 1 + \sum_{i=1}^d x_{vi} \cdot (2q_i) + \sum_{i=1}^d 1 \cdot (-q_i^2) = \sum_{i=1}^d (x_{ui} - x_{vi})^2 - \sum_{i=1}^d (q_i - x_{vi})^2 < 0$ holds. Assuming $\sum_{i=1}^d (x_{ui} - x_{vi})^2 - \sum_{i=1}^d (q_i - x_{vi})^2 < 0$ holds, we have $\langle[\![\chi(u,v)]\!]_1, [\![\gamma]\!]_2\rangle > 0$. If $s_{d+1} = 1$, we have $E(\mu_{d+1}) = E(0)$ and $\Delta_{d+1} > 0$. According to line 3 in Algorithm 5, we can deduce that $E(\delta_{d+1}) = E(1)$. If $s_{d+1} = -1$, we have $E(\mu_{d+1}) = E(1)$ and $\Delta_{d+1} < 0$ so that $E(\delta_{d+1}) = E(1)$. In contrast, when $\sum_{i=1}^d (x_{ui} - x_{vi})^2 - \sum_{i=1}^d (q_i - x_{vi})^2 < 0$ does not hold, we have $\langle[\![\chi(u,v)]\!]_1, [\![\gamma]\!]_2\rangle < 0$. If $s_{d+1} = 1$, we have $E(\mu_{d+1}) = E(0)$ and $\Delta_{d+1} < 0$. Similarly, we can deduce that $E(\delta_{d+1}) = E(0)$. If $s_{d+1} = -1$, we have $E(\mu_{d+1}) = E(1)$ and $\Delta_{d+1} > 0$ so that $E(\delta_{d+1}) = E(0)$. Now, we have proved that $E(\delta_{d+1})$ correctly indicates whether x_u, x_v and q satisfy (5).

Algorithm 6: ePRSQ Dataset Encryption.

Input: A dataset $\mathcal{D}$, public parameters PP and two matrix M_1, M_2.
Output: An encrypted dataset $[\![\mathcal{D}]\!] = \{[\![\chi(i,j)]\!]^* \mid i,j \in [1,n] \wedge i \neq j\}$.

1: $[\![\mathcal{D}]\!] = \varnothing$;
2: **for** x_i in $\mathcal{D}$ **do**
3: **for** x_j in $\mathcal{D}$ **do**
4: **if** $i \neq j$ **then**
5: $[\![\chi(i,j)]\!]^* = PRDC.Enc(x_i, x_j, PP, M_1, M_2)$;
6: $[\![\mathcal{D}]\!] \leftarrow [\![\chi(i,j)]\!]^*$;
7: **end if**
8: **end for**
9: **end for**
10: **return** $[\![\mathcal{D}]\!]$;

So far, we have proved that $E(\delta_i)(1 \leq i \leq d)$ correctly indicates whether x_{ui}, x_{vi} and q_i satisfy (4), and $E(\delta_{d+1})$ correctly indicates whether x_u, x_v and q satisfy (5). According to the definition of dominance relation, we have $E(\tau) = \prod_{i=1}^{d+1} E(\delta_i)$. So, the correctness of PRDC is proved. □

C. The ePRSQ Scheme

Our ePRSQ scheme includes five phases described below.

1) *System initialization:* Assuming $\mathcal{DO}$ has a d-dimensional dataset $\mathcal{D}$, in which the upper bound on the absolute value of the data is T. To initialize the system, $\mathcal{DO}$ first selects an asymmetric encryption algorithm $AE(\cdot)$, e.g., ElGamal, and generates a key pair $\{sk_{AE}, pk_{AE}\}$ randomly. Then, $\mathcal{DO}$ generates parameters of PRDC, i.e., $\{PP, M_1, M_2\} = PRDC.Setup(d, T)$. Finally, $\mathcal{DO}$ publishes the public parameters $\{PP, pk_{AE}\}$ and sends sk_{AE} to $\mathcal{CS}$ via a secure channel. Meanwhile, if a $\mathcal{QU}$ wants to utilize the RSQ service, he or she must first register with $\mathcal{DO}$. Upon approval from $\mathcal{DO}$, M_1^{-1} and M_2^{-1} are transmitted to $\mathcal{QU}$ via a secure communication channel.

2) *Data Outsourcing:* In this phase, $\mathcal{DO}$ first converts the dataset $\mathcal{D}$ into an encrypted dataset $[\![\mathcal{D}]\!]$ using Algorithm 6 to guarantee the security of the dataset $\mathcal{D}$. After that, $\mathcal{DO}$ outsources $[\![\mathcal{D}]\!]$ to $\mathcal{CS}$.

3) *Query Generation:* First, for a query vector q, $\mathcal{QU}$ generates a query token $q_t = PRDC.QueryGen(q, PP, M_1^{-1}, M_2^{-1}, pk)$ according to PRDC query generation algorithm. Second, $\mathcal{QU}$ encrypts the query token q_t with $\mathcal{CS}$'s public key pk_{AE}, that is $AE(pk_{AE}, q_t)$, and chooses SHE parameters $\{pk, sk\}$. Finally, $\mathcal{QU}$ sends the query request $Req = \{AE(pk_{AE}, q_t), pk\}$ to $\mathcal{CS}$ and keeps sk secretly.

4) *Reverse Skyline Search:* Once $\mathcal{CS}$ receives a query request Req from $\mathcal{QU}$, it extracts pk and decrypts $AE(pk_{AE}, q_t)$ with sk_{AE} to recover the query token q_t. Then, $\mathcal{CS}$ executes the Algorithm 7 to perform the reverse skyline query with privacy-preserving. Finally, $\mathcal{CS}$ sends the encrypted query result S_{sky} to $\mathcal{QU}$.

Algorithm 7: ePRSQ Reverse Skyline Search Algorithm.

Input: An encrypted dataset $\{[\![\chi(i,j)]\!]^* \mid i,j \in [1,n] \wedge i \neq j\}$, a query token q_t, and a SHE public parameter pk.
Output: A set containing n data indexes and SHE ciphertexts, $S_{sky} = \{i \parallel E(\theta_i) \mid 1 \leq i \leq n\}$.

1: $S_{sky} = \varnothing$;
2: $i = 1$;
3: **while** $i \leq n$ **do**
4: $j = 1$;
5: $E(\theta_i) = E(0)$;
6: **while** $j \leq n$ **do**
7: $E(\tau) = PRDC.Check([\![\chi(j,i)]\!]^*, q_t, pk)$;
8: $E(\theta_i) = E(\theta_i) + E(\tau)$;
9: $j = j + 1$;
10: **end while**
11: $S_{sky} \leftarrow (i \parallel E(\theta_i))$;
12: $i = i + 1$;
13: **end while**
14: **return** S_{sky};

Algorithm 8: ePRSQ Query Result Recover.

Input: An encrypted query result $S_{sky} = \{i \parallel E(\theta_i) \mid 1 \leq i \leq n\}$, and a SHE private parameter sk.
Output: A set S_{ID} that contains reverse skylines' IDs.

1: $S_{ID} = \varnothing$;
2: **for** $i \parallel E(\theta_i)$ in S_{sky} **do**
3: $\theta_i = SHE.Dec(sk, E(\theta_i))$;
4: **if** $\theta_i = 0$ **then**
5: $S_{ID} \leftarrow i$;
6: **end if**
7: **end for**
8: **return** S_{ID};

5) *Result Recovery:* Upon receiving the encrypted query result from $\mathcal{CS}$, $\mathcal{QU}$ performs the Algorithm 8 to recover the reverse skylines' IDs regarding q.

Note: Our ePRSQ returns the number and IDs that meet the query condition, which is reasonable in some applications where the query user only needs to know the basic information of the query results, such as in advertising push mentioned in the introduction and commercial site selection. In addition, we can extend our ePRSQ to support returning specific query data using proxy re-encryption (PRE) technology [35]. During system initialization, $\mathcal{DO}$ generates a PRE key based on each $\mathcal{QU}$'s public key and sends it to $\mathcal{CS}$. In the query phase, $\mathcal{CS}$ returns the IDs with the PRE of the corresponding ciphertext data. Finally, $\mathcal{QU}$ can recover the plaintext data by decrypting the PRE ciphertexts using the private key.

Theorem 3 (Correctness of ePRSQ): According to $\mathcal{QU}$'s query request, ePRSQ can return the correct reverse skyline query result by searching in the encrypted dataset $[\![\mathcal{D}]\!]$.

Proof: From the correctness of PRDC, $E(\tau)$ in line 7 of Algorithm 7 determines whether $x_j \prec_{x_i} q$ holds or not, so it

is easy to know that $E(\theta_i)$ represents how many data vectors reverse dominate x_i in the dataset. According to the definition of RSQ (Definition 2), the data vector x_i satisfies the condition of reverse skyline query iff $E(\theta_i) = E(0)$. Therefore, $\mathcal{QU}$ can correctly filter out reverse skylines according to Algorithm 8.□

V. Security Analysis

In this section, we first prove the security of PIPSD and PRDC, followed by the security analysis of ePRSQ.

A. Security of the PIPSD Scheme

In this section, we demonstrate that PIPSD is selective security within the framework of the real/ideal world model, as delineated in Section III-B. Let $[\![x]\!]_1$ and $[\![y]\!]_2$ be the outputs of x and y after obfuscated by $PIPSD.Obf_1(\cdot)$ and $PIPSD.Obf_2(\cdot)$, respectively, the leakage information of PIPSD is the inner product of $[\![x]\!]_1$ and $[\![y]\!]_2$, i.e., $\mathcal{L} = \langle [\![x]\!]_1, [\![y]\!]_2 \rangle$.

Ideal world: There is a PPT adversary $\mathcal{A}$ and a simulator in the ideal world interacting as follows.

- *Setup phase:* $\mathcal{A}$ selects w_1 random d-dimensional vectors $\{x_i\}_{i=1}^{w_1}$, where w_1 is a polynomial number. Next, it sends $\{x_i\}_{i=1}^{w_1}$ and the upper limit T of the absolute value of the vector's elements to the simulator. After receiving $\{x_i\}_{i=1}^{w_1}$ and T, the simulator selects w_1 random $d+1$-dimensional vectors $[\![x_i]\!]'_1$ as the obfuscated vectors of $\{x_i\}_{i=1}^{w_1}$.
- *Query phase I:* $\mathcal{A}$ sends w_2 random d-dimensional query vectors $\{y_i\}_{i=1}^{w_2}$ to the simulator, where w_2 is a polynomial number. After receiving $\{y_i\}_{i=1}^{w_2}$, the simulator constructs obfuscated query vectors $\{[\![y_i]\!]'_2\}_{i=1}^{w_2}$ according to the leakage function $\mathcal{L}$ and $\{[\![x_i]\!]'_2\}_{i=1}^{w_1}$. The simulator performs the following steps to construct each $[\![y_i]\!]'_2$.
  - It generates w_1 random real numbers $\{v_i\}_{i=1}^{w_1}$ satisfying the following equation:

$$\begin{cases} v_i > 0, & \langle [\![x_i]\!]_1, [\![y_i]\!]_2 \rangle > 0 \\ v_i < 0, & \langle [\![x_i]\!]_1, [\![y_i]\!]_2 \rangle < 0 \end{cases} \tag{7}$$

  - It randomly selects a vector $[\![y_i]\!]'_2$ satisfying $\langle [\![x_j]\!]'_1, [\![y_i]\!]'_2 \rangle = v_i$ for $j \in [1, w_1]$.

  Finally, the simulator returns $\{[\![y_i]\!]'_2\}_{i=1}^{w_2}$ to $\mathcal{A}$.
- *Challenge phase:* The simulator sends $\{[\![x_i]\!]'_1\}_{i=1}^{w_1}$ to $\mathcal{A}$.
- *Query phase II:* Adversary $\mathcal{A}$ randomly selects a set of $(w'_2 - w_2)$ d-dimensional query vectors $\{y_i\}_{i=w_2+1}^{w'_2}$, where w'_2 is a polynomial number. Subsequently, $\mathcal{A}$ forwards these vectors $\{y_i\}_{i=w_2+1}^{w'_2}$ to the simulator. Upon receipt of the query vectors, the simulator responds by returning the obfuscated query vectors $\{[\![y_i]\!]'_2\}_{i=w_2+1}^{w'_2}$ to $\mathcal{A}$, which is similar to the Query Phase I.

Since the experiment in the real world is to execute our PIPSD normally, the real world's view of $\mathcal{A}$ is $\text{View}_{\mathcal{A},\text{Real}} = \{\{[\![x_i]\!]_1\}_{i=1}^{w_1}, \{[\![q_i]\!]_2\}_{i=1}^{w'_2}\}$. Meanwhile, the view of $\mathcal{A}$ in the ideal world is $\text{View}_{\mathcal{A},\text{Ideal}} = \{\{[\![x_i]\!]'_1\}_{i=1}^{w_1}, \{[\![q_i]\!]'_2\}_{i=1}^{w'_2}\}$ as shown above.

Definition 4 (Security of the PIPSD): The PIPSD scheme achieves selective security on the leakage function $\mathcal{L}$ iff for any $\mathcal{A}$ that initiates a polynomial number of data vector obfuscations and query vector obfuscations, there exists a $\text{Sim}(\mathcal{A})$ that $\mathcal{A}$ has a negligible advantage in distinguishing the views of real or ideal worlds, i.e., $\text{View}_{\mathcal{A},\text{Real}} \overset{c}{\equiv} \text{View}_{\mathcal{A},\text{Ideal}}$.

We prove that the PIPSD scheme achieves selective security on the leakage function $\mathcal{L}$ in the following.

Theorem 4: PIPSD achieves selective security on $\mathcal{L}$.

Proof: Recalling the Definition 4, we prove that it is hard for $\mathcal{A}$ to distinguish $\text{View}_{\mathcal{A},\text{Real}}$ from $\text{View}_{\mathcal{A},\text{Ideal}}$. We must consider two cases:

1) The real-world obfuscated vectors $\{[\![x_i]\!]_1\}_{i=1}^{w_1}$ are indistinguishable from $\{[\![x_i]\!]'_1\}_{i=1}^{w_1}$.
2) The real-world obfuscated query vectors $\{[\![y_i]\!]_2\}_{i=1}^{w'_2}$ are indistinguishable from $\{[\![y_i]\!]'_2\}_{i=1}^{w'_2}$.

We discuss these two cases respectively below.

- In the real world, $[\![x_i]\!]_1 = (\alpha x_{i1} + r_1, \alpha x_{i2} + r_2, \ldots, \alpha x_{id} + r_d, \beta)M$. Since each $[\![x_i]\!]_1$ includes more than two random numbers $\{\alpha, \beta, r_1, \ldots, r_d\}$, and $\mathcal{A}$ does not know the random matrix M, $\{[\![x_i]\!]_1\}_{i=1}^{w_1}$ are indistinguishable from random vectors. Because $\{[\![x_i]\!]'_1\}_{i=1}^{w_1}$ are randomly generated in the ideal world, $\{[\![x_i]\!]_1\}_{i=1}^{w_1}$ are indistinguishable from $\{[\![x_i]\!]'_1\}_{i=1}^{w_1}$.
- In the real world, $[\![y_i]\!]_2 = M^{-1}(\alpha' y_{i1} + r'_1, \alpha' y_{i2} + r'_2, \ldots, \alpha' y_{id} + r'_d, \beta')$. Since each $[\![y_i]\!]_2$ includes more than two random numbers $\{\alpha', \beta', r'_1, \ldots, r'_d\}$, and $\mathcal{A}$ does not know the random matrix M^{-1}, it is hard to distinguish $\{[\![y_i]\!]_2\}_{i=1}^{w'_2}$ from random vectors. Each $[\![y_i]\!]'_2$ is limited by the leakage function $\mathcal{L}$ in the ideal world. According to (3), we can know that the result of $\langle [\![x_i]\!]_1, [\![y_i]\!]_2 \rangle$ is random, and v_i in (7) is also random. Therefore, $\{[\![y_i]\!]'_2\}_{i=1}^{w'_2}$ are indistinguishable from random vectors under the random generated ciphertexts $\{[\![x_i]\!]'_1\}_{i=1}^{w_1}$. Therefore, $\{[\![y_i]\!]_2\}_{i=1}^{w'_2}$ are indistinguishable from $\{[\![y_i]\!]'_2\}_{i=1}^{w'_2}$.

In summary, we can deduce that $\mathcal{A}$ cannot distinguish $\text{View}_{\mathcal{A},\text{Real}}$ or $\text{View}_{\mathcal{A},\text{Ideal}}$ of our PIPSD. Therefore, PIPSD achieves selective security on the leakage function $\mathcal{L}$. □

B. Security of the PRDC Scheme

In this section, we show that PRDC achieves selective security in the real/ideal world model. Let $[\![\chi(u,v)]\!]^*$ be the ciphertext of x_u and x_v generated by $PRDC.Enc(\cdot)$. Additionally, let ψ and $E(\mu_i)$ be the obfuscated query vector and SHE ciphertexts generated by $PRDC.QueryGen(\cdot)$, respectively. The definition of the leakage function of the PRDC scheme is $\widehat{\mathcal{L}} = \langle [\![\chi(u,v)]\!]^*, \psi \rangle$.

Ideal world: There is a PPT adversary $\mathcal{A}$ and a simulator in the ideal world interacting as follows.

- *Setup phase:* $\mathcal{A}$ sends w_1 random d-dimensional vector pairs $\{x_{u_i}, x_{v_i}\}_{i=1}^{w_1}$ to the simulator, where w_1 is a polynomial number. After receiving $\{x_{u_i}, x_{v_i}\}_{i=1}^{w_1}$, the simulator randomly generates w_1 $(2d+2)$-dimensional vectors $\{[\![\chi(u_i, v_i)]\!]^{*\prime}\}_{i=1}^{w_1}$, where $[\![\chi(u_i, v_i)]\!]^{*\prime}$ is the ciphertext of x_{u_i} and x_{v_i}.
- *Query phase I:* $\mathcal{A}$ sends w_2 random query vectors $\{q_i\}_{i=1}^{w_2}$ to the simulator, where w_2 is a polynomial number. After

receiving $\{q_i\}_{i=1}^{w_2}$, the simulator constructs the ciphertexts $\{\psi_i'\}_{i=1}^{w_2}$ and $\{E(\mu_k)_i' \mid 1 \le k \le d+1\}_{i=1}^{w_2}$ using the leakage function $\widehat{\mathcal{L}}$ and $\{[\![\chi(u_i, v_i)]\!]^{*\prime}\}_{i=1}^{w_1}$. The simulator performs the following steps to construct each ψ_i' and $\{E(\mu_k)_i' \mid 1 \le k \le d+1\}$.

- It generates $w_1 \times (d+1)$ random real numbers $\{v_{m \in w_1, n \in (d+1)}\}$ and each $v_{m,n}$ satisfies the leakage function $\widehat{\mathcal{L}}$:

$$\begin{cases} v_{m,n} < 0, & ([\![\chi(u_m, v_m)]\!]^* \cdot \psi_i)_n < 0 \\ v_{m,n} > 0, & ([\![\chi(u_m, v_m)]\!]^* \cdot \psi_i)_n > 0 \end{cases} \tag{8}$$

- It randomly selects a $(2d+2)$-dimensional vector ψ_i' satisfying $([\![\chi(u_m, v_m)]\!]^* \cdot \psi_i')_n = v_{m,n}$.
- It selects $d+1$ random integers $\{\mu_k \in \{0,1\}\}_{k=1}^{d+1}$ and then generates the ciphertexts $\{E(\mu_k)_i' \mid 1 \le k \le d+1\}$ of them using SHE encryption.

Finally, the simulator returns $\{\psi_i'\}_{i=1}^{w_2}$ and $\{E(\mu_k)_i' \mid 1 \le k \le d+1\}_{i=1}^{w_2}$ to $\mathcal{A}$.

- *Challenge phase:* The simulator sends $\{[\![\chi(u_i, v_i)]\!]^{*\prime}\}_{i=1}^{w_1}$ to $\mathcal{A}$.
- *Query phase II:* $\mathcal{A}$ sends $w_2' - w_2$ random query vectors $\{q_i\}_{i=w_2+1}^{w_2'}$ to the simulator, where w_2' is a polynomial number. After receiving them, the simulator returns the ciphertexts $\{\psi_i'\}_{i=1}^{w_2}$ and $\{E(\mu_k)_i' \mid 1 \le k \le d+1\}_{i=1}^{w_2}$ to $\mathcal{A}$, which is the same as query phase I.

Since the experiment in the real world is to execute our PRDC normally, the real world's view of $\mathcal{A}$ is $\text{View}_{\mathcal{A},\text{Real}} = \{\{[\![\chi(u_i, v_i)]\!]^*\}_{i=1}^{w_1}, \{\psi_i\}_{i=1}^{w_2'}, \{E(\mu_k)_i \mid 1 \le k \le d+1\}_{i=1}^{w_2'}\}$. Meanwhile, the ideal world's view of $\mathcal{A}$ is $\text{View}_{\mathcal{A},\text{Real}} = \{\{[\![\chi(u_i, v_i)]\!]^{*\prime}\}_{i=1}^{w_1}, \{\psi_i'\}_{i=1}^{w_2'}, \{E(\mu_k)_i' \mid 1 \le k \le d+1\}_{i=1}^{w_2'}\}$ as shown above.

Definition 5 (Security of the PRDC): The PRDC scheme achieves selective security on the leakage function $\widehat{\mathcal{L}}$ iff for any $\mathcal{A}$ that initiates a polynomial number of data vectors encryptions and query token generations, there exists a $\text{Sim}(\mathcal{A})$ such that $\mathcal{A}$ has a negligible advantage in distinguishing the views of real or ideal worlds, i.e., $\text{View}_{\mathcal{A},\text{Real}} \stackrel{c}{\equiv} \text{View}_{\mathcal{A},\text{Ideal}}$.

We prove that the PRDC scheme achieves selective security on the leakage function $\widehat{\mathcal{L}}$ in the following.

Theorem 5: PRDC achieves selective security on leakage function $\widehat{\mathcal{L}}$.

Proof: Recalling the Definition 5, we should prove that it is hard for $\mathcal{A}$ to distinguish between $\text{View}_{\mathcal{A},\text{Real}}$ and $\text{View}_{\mathcal{A},\text{Ideal}}$. To prove the indistinguishability, we must consider three cases:

1) The real-world SHE ciphertexts $\{E(\mu_k)_i \mid 1 \le k \le d+1\}_{i=1}^{w_2'}$ cannot be distinguished from $\{E(\mu_k)_i' \mid 1 \le k \le d+1\}_{i=1}^{w_2'}$.
2) The real-world data ciphertexts $\{[\![\chi(u_i, v_i)]\!]^*\}_{i=1}^{w_1}$ are indistinguishable from $\{[\![\chi(u_i, v_i)]\!]^{*\prime}\}_{i=1}^{w_1}$.
3) The real-world query ciphertexts $\{\psi_i\}_{i=1}^{w_2'}$ are indistinguishable from $\{\psi_i'\}_{i=1}^{w_2'}$.

We discuss these three cases respectively below.

- Since SHE has been proved to be IND-CPA secure [27], $\{E(\mu_k)_i \mid 1 \le k \le d+1\}_{i=1}^{w_2'}$ are indistinguishable from $\{E(\mu_k)_i' \mid 1 \le k \le d+1\}_{i=1}^{w_2'}$.
- In the real world, $[\![\chi(u_i, v_i)]\!]^* = [\![\chi(u_i, v_i)]\!]_1 \cdot M_2$. Since PIPSD is selectively secure with $\mathcal{L}$, and $\mathcal{A}$ does not know random matrix M_2, $\{[\![\chi(u_i, v_i)]\!]^*\}_{i=1}^{w_1}$ are indistinguishable from random vectors. Because $\{[\![\chi(u_i, v_i)]\!]^{*\prime}\}_{i=1}^{w_1}$ are randomly generated in the ideal world, $\{[\![\chi(u_i, v_i)]\!]^*\}_{i=1}^{w_1}$ are indistinguishable from $\{[\![\chi(u_i, v_i)]\!]^{*\prime}\}_{i=1}^{w_1}$.
- In the real world, $\{\psi_i\}_{i=1}^{w_2'} = M_2^{-1} \cdot (s_1[\![\eta(1)]\!]_2^T, \ldots, s_d[\![\eta(d)]\!]_2^T, s_{d+1}[\![\gamma]\!]_2^T)$. Since PIPSD is selectively secure with $\mathcal{L}$, and $\mathcal{A}$ does not know random matrix M_2^{-1}, it is hard to distinguish $\{\psi_i\}_{i=1}^{w_2'}$ from random vectors. Each ψ_i' is limited by the leakage function $\widehat{\mathcal{L}}$ in the ideal world. According to (6), we know that the result of $\langle [\![\chi(u_i, v_i)]\!]^*, \psi_i \rangle$ is random, so $v_{m,n}$ in (8) is random. Therefore, $\{\psi_i'\}_{i=1}^{w_2'}$ cannot be distinguished from random vectors under the randomly generated ciphertexts $\{[\![\chi(u, v)]\!]^{*\prime}\}_{i=1}^{w_1}$. As a result, $\{\psi_i\}_{i=1}^{w_2'}$ are indistinguishable from $\{\psi_i'\}_{i=1}^{w_2'}$.

In summary, we conclude that $\mathcal{A}$ cannot distinguish $\text{View}_{\mathcal{A},\text{Real}}$ or $\text{View}_{\mathcal{A},\text{Ideal}}$. Therefore, PRDC achieves selective security on the leakage function $\widehat{\mathcal{L}}$. □

C. Security of the ePRSQ Scheme

In our security model, $\mathcal{CS}$ is honest-but-curious, which means it is curious about the plaintext of the dataset, query requests, and query results. Meanwhile, $\mathcal{QU}$ may eavesdrop on the query requests and query results of other $\mathcal{QU}$s, but cannot collude with $\mathcal{CS}$. Next, we show our ePRSQ is security and privacy-preserving.

Theorem 6: Our ePRSQ is security and privacy-preserving under the proposed security model.

Proof: The security and privacy protection of the ePRSQ is mainly reflected in the security of the dataset, and the privacy protection of query requests and query results. We prove that ePRSQ meets the requirements from these two aspects respectively.

The dataset is secure: In ePRSQ, $\mathcal{DO}$ obfuscates the dataset $\mathcal{D}$ using the PRDC encryption algorithm and then outsources the obfuscated dataset $[\![\mathcal{D}]\!]$ to $\mathcal{CS}$. As shown in Section V-B, PRDC guarantees that the plaintext data cannot be recovered from the ciphertext if M_1 and M_2 are kept secret. Since these two matrices are only known by $\mathcal{DO}$ and $\mathcal{QU}$s, and $\mathcal{QU}$s do not collude with $\mathcal{CS}$, $\mathcal{CS}$ cannot recover plaintexts of $\mathcal{D}$ from $[\![\mathcal{D}]\!]$. Besides, $\mathcal{CS}$ may infer the private information of $\mathcal{D}$ during the reverse skyline search phase, such as single-dimensional privacy. Recalling Section IV-C, $\mathcal{CS}$ needs to compute $\Delta = [\![\chi(u, v)]\!]^* \cdot \psi$. According to (6), Δ_i is PIPSD's check result multiplied by a random number. So, it is impossible for $\mathcal{CS}$ to recover $\chi(u, v)$, $\eta(i)$ or γ from Δ. Moreover, since the check result is multiplied by a random number, which is 1 or -1, $\mathcal{CS}$ needs to combine the SHE ciphertexts sent by $\mathcal{QU}$ and use homomorphic operations to get the encrypted real check result. As the SHE is IND-CPA,

$\mathcal{CS}$ cannot infer any private information about $\mathcal{D}$. Therefore, the dataset is secure in ePRSQ.

The query requests and query results are privacy-preserving: In ePRSQ, the query token q_t consists of ψ and $\{E(\mu_i)\}_{i=1}^{d+1}$, where ψ is generated by the PRDC query generation algorithm. As shown in Section V-B, the security of PRDC can guarantee that q cannot be recovered from ψ and intermediate result Δ if M_1^{-1} and M_2^{-1} are kept secret. Since these two matrices are only known by $\mathcal{DO}$ and $\mathcal{QU}$s, and $\mathcal{QU}$s do not collude with $\mathcal{CS}$, $\mathcal{CS}$ cannot recover q from ψ. After finishing the reverse skyline search, $\mathcal{CS}$ gets the results of SHE ciphertext. Since it does not know the sk corresponding to SHE, $\mathcal{CS}$ cannot retrieve any private information from the encrypted query results. Therefore, the query requests and query results are privacy-preserving from $\mathcal{CS}$.

In addition, although each $\mathcal{QU}$ has the same secret keys M_1^{-1} and M_2^{-1}, $\mathcal{QU}$ encrypts his/her query request using $AE(\cdot)$ and $\mathcal{CS}$'s public key pk_{AE}. So, other $\mathcal{QU}$s cannot retrieve the plaintext of q from the query request. Meanwhile, since each $\mathcal{QU}$ holds different SHE parameters pk and sk when performing $PRDC.QueryGen(\cdot)$, other $\mathcal{QU}$ cannot decrypt $\mathcal{QU}$'s query result. Therefore, the query requests and query results are privacy-preserving from other $\mathcal{QU}$s.

In conclusion, ePRSQ is secure and privacy-preserving.

VI. Performance Evaluation

In this section, we evaluate the performance of ePRSQ and compare it with related work, i.e., PPARS [9] and OPPRS [10], from computational costs and communication overhead aspects. ePRSQ contains five phases. Since there is no calculation in the system initialization phase and only data decryption operation is required in the result recovery phase, we mainly evaluate the performance of the data outsourcing, query generation, and reverse skyline search phases.

A. Experimental Setting

Three datasets from different fields, i.e., the Eye state dataset [36], the India house prices dataset [37], and the US census dataset [38], are adopted in our experiments to evaluate the performance of the schemes and their stability in diverse datasets. Both our ePRSQ and PPARS [9] adopt SHE as the cryptography primitive. Since SHE is a leveled homomorphic encryption, the maximum depth of its homomorphic multiplication depends on k_0. Generally, there are two ways to support more homomorphic multiplication operations. One is to utilize the bootstrapping protocol [27], and the other is to enlarge k_0. As ePRSQ is designed under a single cloud model, the bootstrapping protocol cannot be used in ePRSQ, and $k_0 = 4096$ meets the requirement of ePRSQ. Therefore, we set the security parameters of SHE as $k_0 = 4096$, $k_1 = 80$, and $k_2 = 160$. OPPRS [10] uses Paillier homomorphic encryption as its cryptography primitive and its security parameter is set to 512.

We implemented the three schemes using C++ programming language on a computer with 192GB of memory, Intel (R) Xeon (R) Gold 5218R CPU @ 2.10GHz and Ubuntu 20.04 OS, and the source codes of the experiments are available on Github[1]. Besides, to facilitate the implementation of the schemes, we scale the decimals in the dataset to integers by the same multiple. Furthermore, all computational cost results in the experiments are taken from the average of 10 experimental runs.

B. Computational Costs

1) Performance of Data Outsourcing Phase: As PPARS [9] stores the dataset in plaintext on the cloud, there is no computational consumption in the data outsourcing. Therefore, we only compare the computational costs of ePRSQ and OPPRS [10] in this phase. As shown in Section IV-C, in our ePRSQ, $\mathcal{DO}$ sends the obfuscated dataset $[\![\mathcal{D}]\!] = \{[\![\chi(i,j)]\!]^* \mid i,j \in [1,n] \wedge i \neq j\}$ to $\mathcal{CS}$. Consequently, the computational costs in the data outsourcing phase are associated with the size n of the dataset and the dimension d of the data.

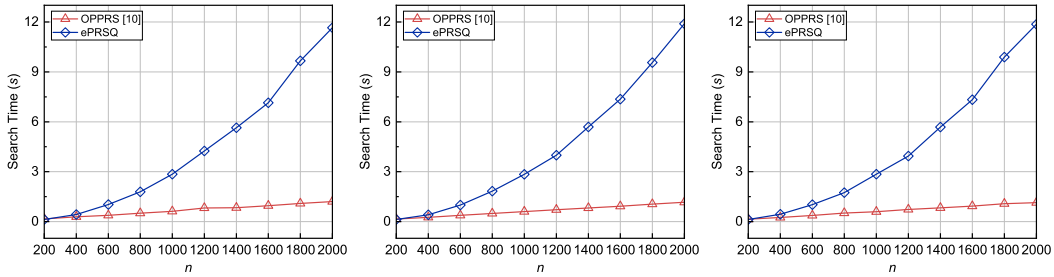
Figs. 3 and 4 show the computational costs of ePRSQ and OPPRS [10] in the data outsourcing phase. Specifically, Fig. 3(a), (b) and (c) depict the computational costs of encrypting a dataset that varies with n and $d = 3$ on three different datasets. The dataset encryption time of both ePRSQ and OPPRS [10] increases with the dataset size. Specifically, for a 3-dimensional dataset with size 2000, in ePRSQ, $\mathcal{DO}$ needs 11.65s to generate the obfuscated dataset while OPPRS [10] needs 1.20s to generate the encrypted dataset. Besides, when n is fixed, the time costs of the data outsourcing in ePRSQ and OPPRS [10] both linearly increase with the number of dimensions d. Fig. 4(a), (b) and (c) depict the computational costs of encrypting three datasets that vary with d and $n = 1000$, from which we note that for a 10-dimensional dataset with 1000 vectors, ePRSQ and OPPRS [10] need 4.12 s and 1.53 s to generate the encrypted dataset, respectively.

In summary, PPARS [9] directly outsources plaintext datasets without computational overhead, but it cannot guarantee data security from $\mathcal{CS}$. The computational efficiency of ePRSQ is somewhat inferior to that of OPPRS [10] in the outsourcing phase. However, it didn't have much impact. First, data outsourcing only needs to be performed once and can be done offline. At the same time, the computational cost of ePRSQ in the experimental dataset is at second level. Therefore, the computational cost of ePRSQ in the outsourcing phase is acceptable.

2) Performance of Query Generation Phase: According to Section IV-C, $\mathcal{QU}$ generates and sends the query request $Req = \{AE(pk_{AE}, q_t), pk\}$ to the cloud, where $q_t = \{\psi, E(\mu_i), i \in [1, d+1]\}$. Consequently, the computational costs of the query generation phase are related to the number of dimensions d.

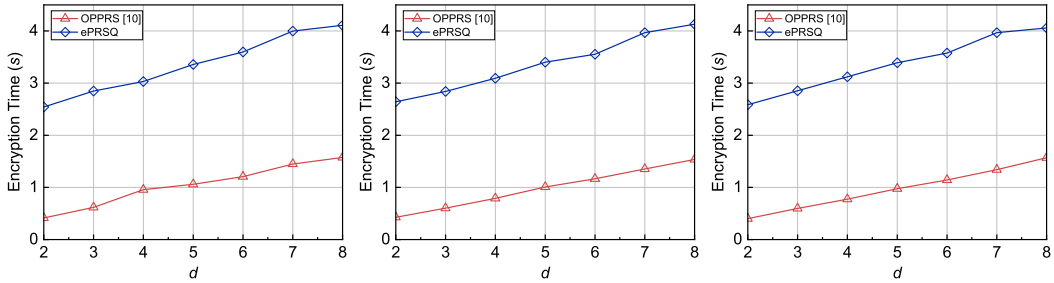
Fig. 5 illustrates the computational costs comparison among three schemes on three different datasets to generate a query request, where d changes from 2 to 8. As can be seen from the figure, the computational costs of ePRSQ and OPPRS [10] are relatively low and their growth rates are significantly slower. In contrast, the computational cost of generating query vectors in PPARS [9] is relatively large, and its growth trend is significantly higher than that of ePRSQ and OPPRS [10]. Specifically, when

[1]https://github.com/NigulasiLiu/SkylineExperiment



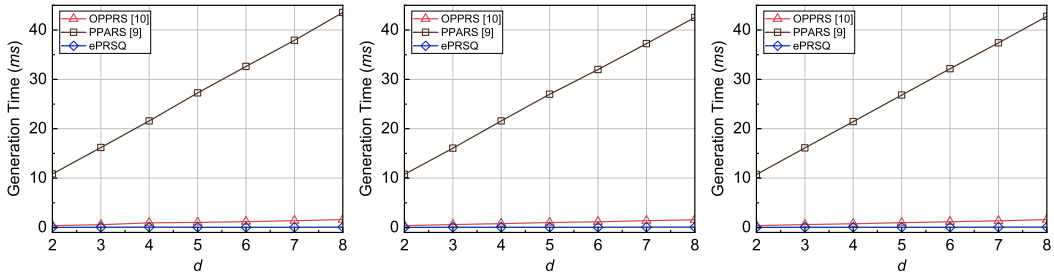
(a) EEG Eye State dataset [36] (b) India House dataset [37] (c) US Census dataset [38]

Fig. 3. Computational costs for offline data encryption on different datasets that varies with n and $d = 3$.



(a) EEG Eye State dataset [36] (b) India-house dataset [37] (c) US-census dataset [38]

Fig. 4. Computational costs for offline data encryption on different datasets that varies with d and $n = 1000$.



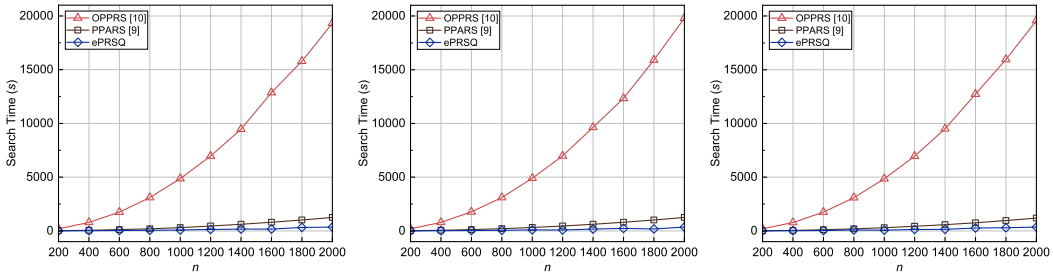
(a) EEG Eye State dataset [36] (b) India House set [37] (c) US Census dataset [38]

Fig. 5. Computational costs of query generation on different datasets that varies with d and $n = 1000$.

$n = 1000$ and $d = 8$, the computational cost of ePRSQ is only 0.076ms, while OPPRS [10] and PPARS [9] consume 1.57ms and 42.94ms, respectively. In other words, the efficiency of ePRSQ in query generation is $20.65\times$ and $565\times$ of that of OPPRS [10] and PPARS [9], respectively. The reason is that the main calculation of ePRSQ in this phase is the product of a $(2d + 2) \times (2d + 2)$ matrix and a $(2d + 2) \times (d + 1)$ matrix, which can be performed quickly. In contrast, PPARS [9] needs to generate multiple encrypted bloom filters, which is more time-consuming than matrix multiplication. OPPRS [10] needs to perform d times Paillier encryption, which is more time-consuming than SHE encryption. So, the computational costs of PPARS [9] and OPPRS [10] to generate a query request are much larger than that of ePRSQ.

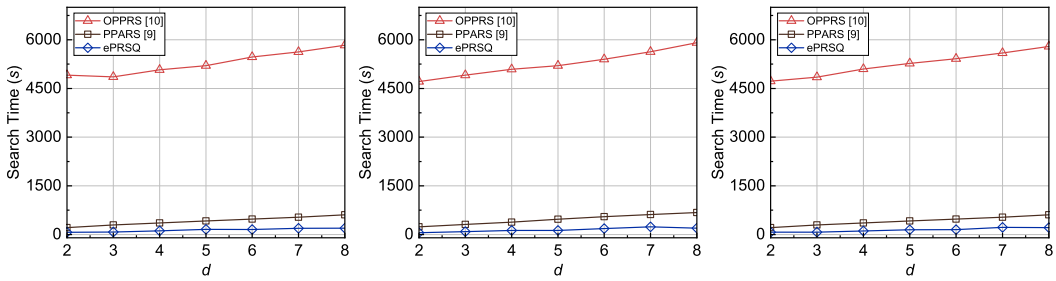
3) Performance of Reverse Skyline Search Phase: The reverse skyline search needs to be executed multiple times online and consumes the most computational cost to protect security and privacy. Therefore, the computational cost evaluation of this phase is the most important because it best reflects the computational efficiency of the protocol. Obviously, its computational cost is related to both n and d.

Figs. 6 and 7 show the computational cost of the reverse skyline search phase of ePRSQ, PPARS [9] and OPPRS [10] on three different datasets. Specifically, Fig. 6(a), (b) and (c) depict the computational costs of reverse skyline query of different schemes on different datasets varying with the size of dataset n. Each scheme needs to perform n^2 operations to ascertain the reverse skyline results. However, the figures reveal that ePRSQ is significantly more computationally efficient than that of PPARS [9] and OPPRS [10]. When d is fixed to 3 and n varies from 200 to 2000, the computational cost of ePRSQ is only 30% of that of PPARS [9] and nearly equivalent to 1.4% to 3.6% of that of OPPRS [10]. For example, for a 3-dimensional dataset of size 2000, i.e., $n = 2000$ and $d = 3$, our ePRSQ takes 352.38s, 358.31s and 356.58s to complete the reverse skyline search procedure on the three datasets, respectively, while



(a) EEG Eye State dataset [36] (b) India House dataset [37] (c) US Census dataset [38]

Fig. 6. Computational costs of searching on different datasets that varies with n and $d = 3$.



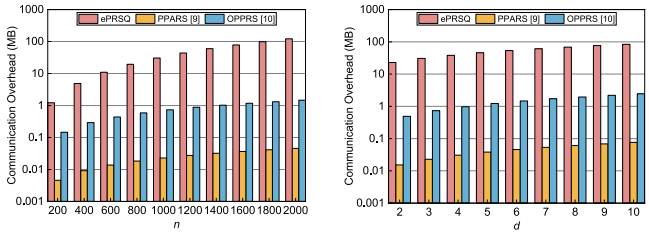
(a) EEG Eye State dataset [36] (b) India House dataset [37] (c) US Census dataset [38]

Fig. 7. Computational costs of searching on different datasets that varies with d and $n = 1000$.

PPARS [9] needs 1245.43s, 1245.42s, and 1185.41s and OPPRS [10] needs 19316.7s, 19792.76s, and 19568.40s to achieve the same objective. This is because ePRSQ requires only one matrix multiplication and $2d + 1$ SHE homomorphic operations to get the dominance relationship. In contrast, PPARS [9] needs to check multiple encrypted bloom filters, followed by SHE operations. OPPRS [10] requires executing multiple homomorphic comparison protocols, which involve a lot of time-consuming homomorphic operations.

Similarly, Fig. 7 depicts the computational costs of reverse skyline search of different schemes on three datasets varying with d and $n = 1000$. In ePRSQ, the size of the matrix determining the dominance relationship increases with the growth of d, thus the computational cost increases accordingly. For every increase of 1in the data dimension d, the PPARS [9] requires 2 additional Bloom filters to determine the dominance relationship. Similarly, OPPRS [10] requires the execution of 13 additional homomorphic operations coupled with 2 decryption processes to achieve the same goal. In addition, as the number of dimensions d increases, OPPRS [10] needs to execute a more secure window query protocol containing several homomorphic operations.

Although all three schemes show a linear growth trend, the computational costs of ePRSQ and PPARS [9] are significantly more efficient than OPPRS [10] when n is fixed. Specifically, when $n = 1000$ and d varies from 2 to 8, the computational costs of ePRSQ are about 30% of that of PPARS and 1.18% to 4.19% of that of OPPRS. Take $n = 1000$ and $d = 8$ as an example, ePRSQ's computational costs on three datasets in this phase are 196.9 s, 196.4 s and 210.6 s, respectively. But PPARS [9] consumes 609.3 s, 676.0 s and 609.5 s, respectively, and OPPRS [10] consumes 5830.2 s, 5911.8 s and 5795.6 s, respectively.



(a) Data outsourcing phase where $d = 3$ and n changes (b) Data outsourcing phase where $n = 1000$ and d changes

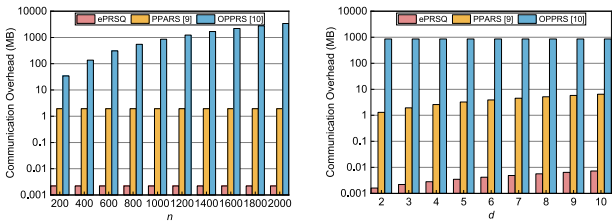
Fig. 8. Communication overhead of data outsourcing.

Based on the above analysis, it is obvious that ePRSQ significantly improves the computational efficiency of reverse skyline search compared to PPARS [9] and OPPRS [10].

C. Communication Overhead

We analyze the communication overhead of ePRSQ and compare it with PPARS [9] and OPPRS [10]. We use $TotalSize = N_{cipher} * S_{cipher}$ to represent the communication overhead between different participants, where N_{cipher} and S_{cipher} are the number of ciphertexts and the ciphertext size, respectively. Since we set SHE's parameter $k_0 = 4096$, the size of a SHE ciphertext is $2 \cdot k_0$bits $= 1024$B. The security parameter of Paillier homomorphic encryption is 512, so the size of a Paillier ciphertext is $512 * 2$bits $= 128$B. In addition, ePRSQ needs to transmit matrices between different participants, and the size of an element in matrices is set as 8B.

1) Communication Overhead of Data Outsourcing Phase: Fig. 8 shows the communication overhead of data outsourcing of three schemes. From Section IV-C, we know that $\mathcal{DO}$



(a) Query phase where $d = 3$ and n changes

(b) Query phase where $n = 1000$ and d changes

Fig. 9. Communication overhead of querying reverse skyline.

needs to send an encrypted dataset that contains $\frac{n^2-n}{2}$ $(2d+2)$-dimensional vectors to $\mathcal{CS}$ for data outsourcing. Therefore, for a d-dimensional dataset of size n, the communication overhead of ePRSQ is $4(2d+2)(n^2-n)$B. In OPPRS [10], the encrypted dataset contains nd Paillier ciphertexts, so its communication overhead is $128nd$B. Fig. 8(a) and (b) depict the communication overhead of the data outsourcing phase that varies with n and d, respectively. We can observe from them that the communication overhead of ePRSQ increases quadratically with n and increases linearly with d in the data outsourcing phase. This is because ePRSQ needs to encrypt every two different data vectors once during the data outsourcing. Since PPARS [9] stores the data in plaintext on the cloud, its communication overhead is equivalent to its dataset size. However, PPARS [9] cannot protect the security of users' data from $\mathcal{CS}$, while ePRSQ solves the security problem of users' data. The communication overhead of OPPRS [10] in this phase is small because it only needs to perform one Paillier encryption on each record in the dataset.

2) Communication Overhead of Query Generation and Search Phase: From Section IV-C, we know that $\mathcal{QU}$ needs to send a query request with a $(2d+2)\times(d+1)$-dimensional matrix and $d+1$ SHE ciphertexts to $\mathcal{CS}$. Therefore, the communication overhead of ePRSQ in this phase is $8(2d+2)(d+1)+1024(d+1)=16d^2+1056d+1040$ B. According to the experimental setting of [9], [10], we know that the communication overhead of PPARS [9] and OPPRS [10] at this phase are $1024*2*\lceil\frac{10*46}{\ln 2}\rceil\cdot d=165888\cdot d$ B and $128*d$ B, respectively. We can see from these formulas that the coefficients of d in the communication overhead of ePRSQ and PPARS [9] are $16d+1056$ and 165888 respectively. Only when d is greater than 10301, the communication overhead of ePRSQ is greater than that of PPARS [9]. But in reality, the dimension of data is often small, so the communication overhead of ePRSQ is much smaller than that of PPARS [9]. Besides, OPPRS [10] requires two clouds to cooperate during the search phase, thus incurring a communication overhead of $128\big(\frac{7}{2}n^2+nd+2kn-\frac{3}{2}n-k\big)$B, where k is a variable related to the query vector and the dataset. For convenience, we let $k=1$.

Fig. 9 shows the communication overhead of all three schemes during the reverse query. Fig. 9(a) depicts the communication overhead of the query phase varies with n and set $d=3$. We can intuitively see that OPPRS's communication overhead is very huge and far greater than that of ePRSQ and PPARS [9]. For example, when $n=1000$, OPPRS's communication overhead has reached 3419.84 MB, PPARS's is 1.95 MB, and ePRSQ's is just 0.002 MB. Similarly, Fig. 9(b) depicts the communication overhead of the query phase varies with d and set $n=1000$. It demonstrates the results of the theoretical analysis, i.e., the communication overhead of OPPRS [10] is huge and that of ePRSQ is much less than PPARS [9]. For example, when $d=10$, the communication overhead of OPPRS [10] is 857.06 MB, PPARS [9] is 6.48 MB, and ePRSQ is just 0.007 MB.

To sum up, ePRSQ has a relatively large communication overhead in the data outsourcing phase, but this communication overhead is one-time, and ePRSQ protects privacy against $\mathcal{CS}$. In addition, the ePRSQ shows a notably high communication efficiency during the reverse skyline query process. For $d\le 100$, the communication cost of ePRSQ is less than 0.3% of PPARS [9], and even more significantly, it is less than 0.03% of OPPRS [10].

D. Summary of Performance Evaluation

Based on the above extensive experimental evaluations and analysis, we summarize the performance analysis and draw the following conclusions: (1) The experimental performance trends of ePRSQ on diverse datasets in different scenarios are relatively consistent, so the performance of ePRSQ on diverse datasets is stable; (2) The computational cost and communication overhead of ePRSQ in outsourcing phase are higher than PPARS [9] and OPPRS [10]. However, PPARS [9] directly outsources plaintext and cannot guarantee data security. Furthermore, data outsourcing can be done offline and only needs to be performed once, so the computational and communication overhead of the ePRSQ in the outsourcing phase is acceptable; (3) The computational and communication efficiency of ePRSQ in reverse skyline query is far superior to PPARS [9] and OPPRS [10]. ePRSQ has improved the computational efficiency by at least $3\times$ compared with PPARS [9] that stores plaintext, and by more than $60\times$ compared with OPPRS [10] that ensures data security, while the communication overhead of ePRSQ is negligible. In summary, ePRSQ is secure and efficient.

VII. Conclusion and Future Work

In this paper, we first design PIPSD to determine whether the inner product of two vectors satisfies a specific relation with 0 without revealing the information of the vectors. Then, we design PRDC based on PIPSD and SHE to check the reverse dominance relationship in a privacy-preserving manner. Finally, we realize a privacy-preserving reverse skyline query scheme (ePRSQ) on the single-cloud model based on PIPSD and PRDC. Security analysis shows that all these three schemes achieve selective security in the real/ideal world model, and our ePRSQ guarantees the security of dataset plaintext, and the privacy of query request and query result. Moreover, ePRSQ is validated to be efficient by extensive experiments on diverse datasets. In the future, we will extend the technologies in ePRSQ to build privacy-preserving static and dynamic skyline queries. Taking the dynamic skyline query as an example, we can express the dynamic dominance relationship as two equations similar to (1) and (2) in Section II-A. Based on this, like (4) and (5) in Section IV-B1, we can transform the determination of the dynamic dominance relationship between vectors into the relationship between the inner product of the vectors and 0. Finally, similar to

ePRSQ, after receiving the encrypted query token from $\mathcal{QU}$, $\mathcal{CS}$ calculates the inner products of the query token and the vectors in the DO's outsourced dataset, and outputs the encrypted dynamic skyline query results by determining the relationship between the inner products and $E(0)$.

REFERENCES

[1] A.-T. Kuo, H. Chen, L. Tang, W.-S. Ku, and X. Qin, "ProbSky: Efficient computation of probabilistic skyline queries over distributed data," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 5, pp. 5173–5186, May 2023.

[2] X. Zhang, R. Lu, J. Shao, H. Zhu, and A. A. Ghorbani, "Continuous probabilistic skyline query for secure worker selection in mobile crowdsensing," *IEEE Internet Things J.*, vol. 8, no. 14, pp. 11758–11772, Jul. 2021.

[3] Z. Wang, X. Ding, H. Jin, and P. Zhou, "Efficient secure and verifiable location-based skyline queries over encrypted data," in *Proc. VLDB Endowment*, vol. 15, no. 9, pp. 1822–1834, 2022.

[4] Z. Wang, L. Zhang, X. Ding, K.-K. R. Choo, and H. Jin, "A dynamic-efficient structure for secure and verifiable location-based skyline queries," *IEEE Trans. Inf. Forensics Secur.*, vol. 18, pp. 920–935, 2022.

[5] Y. Zeng, Z. Yang, W. Zhang, and C. Li, "Application of processing technology based on skyline query in computer network," *Neural Comput. Appl.*, vol. 34, no. 4, pp. 2637–2647, 2022.

[6] Y. Zhang et al., "Efficient and secure skyline queries over vertical data federation," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 9, pp. 9269–9280, Sep. 2023.

[7] G. Wang, J. Xin, L. Chen, and Y. Liu, "Energy-efficient reverse skyline query processing over wireless sensor networks," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 7, pp. 1259–1275, Jul. 2012.

[8] B. Yin, K. Gu, X. Wei, S. Zhou, and Y. Liu, "A cost-efficient framework for finding prospective customers based on reverse skyline queries," *Knowl.-Based Syst.*, vol. 152, pp. 117–135, 2018.

[9] S. Zhang, S. Ray, R. Lu, Y. Guan, Y. Zheng, and J. Shao, "Toward privacy-preserving aggregate reverse skyline query with strong security," *IEEE Trans. Inf. Forensics Secur.*, vol. 17, pp. 2538–2552, 2022.

[10] S. Zhang, R. Lu, and Y. Zheng, "Towards privacy-preserving online medical monitoring with reverse skyline query," in *Proc. IEEE Glob. Commun. Conf.*, 2021, pp. 01–06.

[11] S. Zhang, S. Ray, R. Lu, Y. Zheng, Y. Guan, and J. Shao, "Achieving efficient and privacy-preserving dynamic skyline query in online medical diagnosis," *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9973–9986, Jun. 2021.

[12] S. Zeighami, G. Ghinita, and C. Shahabi, "Secure dynamic skyline queries using result materialization," in *Proc. IEEE 37th Int. Conf. Data Eng.*, 2021, pp. 157–168.

[13] S. Zhang, S. Ray, R. Lu, Y. Zheng, Y. Guan, and J. Shao, "Towards efficient and privacy-preserving user-defined skyline query over single cloud," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 2, pp. 1319–1334, Mar./Apr. 2023.

[14] W. Chen, M. Liu, R. Zhang, Y. Zhang, and S. Liu, "Secure outsourced skyline query processing via untrusted cloud service providers," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.

[15] 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," *Inf. Sci.*, vol. 498, pp. 91–105, 2019.

[16] 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 *Proc. 17th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun./12th IEEE Int. Conf. Big Data Sci. Eng.*, 2018, pp. 958–963.

[17] S. Zhang, S. Ray, R. Lu, Y. Zheng, Y. Guan, and J. Shao, "Towards efficient and privacy-preserving interval skyline queries over time series data," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 2, pp. 1348–1363, Mar./Apr. 2023.

[18] J. Liu, J. Yang, L. Xiong, and J. Pei, "Secure skyline queries on cloud platform," in *Proc. IEEE 33rd Int. Conf. Data Eng.*, 2017, pp. 633–644.

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

[20] W. Wang et al., "Scale: An efficient framework for secure dynamic skyline query processing in the cloud," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, 2020, pp. 288–305.

[21] J. Wang, M. Du, and S. S. Chow, "Stargazing in the dark: Secure skyline queries with SGX," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, Springer, 2020, pp. 322–338.

[22] X. Ding, Z. Wang, P. Zhou, K.-K. R. Choo, and H. Jin, "Efficient and privacy-preserving multi-party skyline queries over encrypted data," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 4589–4604, 2021.

[23] E. Dellis and B. Seeger, "Efficient computation of reverse skyline queries," in *Proc. 33rd Int. Conf. Very Large Data Bases*, 2007, pp. 291–302.

[24] C. Kalyvas and M. Maragoudakis, "Skyline and reverse skyline query processing in spatialhadoop," *Data Knowl. Eng.*, vol. 122, pp. 55–80, 2019.

[25] X. Lian and L. Chen, "Reverse skyline search in uncertain databases," *ACM Trans. Database Syst.*, vol. 35, no. 1, pp. 1–49, 2008.

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

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

[28] K. Bonawitz et al., "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 1175–1191.

[29] G. Oded, *Foundations of Cryptography: Basic Applications*. vol. 2, 1st ed., New York, NY, USA: Cambridge Univ. Press, 2009.

[30] G. Sheng, T. Wen, Q. Guo, and Y. Yin, "Privacy preserving inner product of vectors in cloud computing," *Int. J. Distrib. Sensor Netw.*, vol. 10, no. 5, 2014, Art. no. 537252.

[31] Y. Zheng, R. Lu, and J. Shao, "Achieving efficient and privacy-preserving k-NN query for outsourced ehealthcare data," *J. Med. Syst.*, vol. 43, pp. 1–13, 2019.

[32] Y. Zheng et al., "Efficient and privacy-preserving weighted range set sampling in cloud," *IEEE Trans. Dependable Secure Comput.*, early access, Jun. 04, 2024, doi: 10.1109/TDSC.2024.3408816.

[33] P. Ananth, Z. Brakerski, G. Segev, and V. Vaikuntanathan, "From selective to adaptive security in functional encryption," in *Proc. 35th Ann. Cryptol. Conf. Adv. Cryptol.–CRYPTO*, Santa Barbara, CA, USA, Springer, 2015, pp. 657–677.

[34] S. Lai et al., "Result pattern hiding searchable encryption for conjunctive queries," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 745–762.

[35] R. Canetti and S. Hohenberger, "Chosen-ciphertext secure proxy re-encryption," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 185–194.

[36] O. Roesler, "Eeg eye state," 2013. [Online]. Available: https://archive.ics.uci.edu/dataset/264/eeg+eye+state

[37] S. S. Brar, "House prices India," 2024. [Online]. Available: https://www.kaggle.com/datasets/sukhmandeepsinghbrar/house-prices-india/data

[38] C. Meek, B. Thiesson, and D. Heckerman, "US census data," UCI Machine Learning Repository, 1990, [Online]. Available: https://archive.ics.uci.edu/dataset/116/us+census+data+1990

Yubo Peng received the BS degree in cyberspace security from the University of Electronic Science and Technology of China, in 2021. He is currently working toward the MS degree with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China. His research interest includes privacy computing.

Xiong Li (Senior Member, IEEE) received the PhD degree in computer science and technology from the Beijing University of Posts and Telecommunications, Beijing, China, in 2012. He is a professor with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China. His research interests include information security and privacy computing.

Ke Gu (Member, IEEE) received the PhD degree from the School of Information Science and Engineering, Central South University, Changsha, China, in 2012. He is an associate professor with the School of Computer and Communication Engineering, Changsha University of Science and Technology. His research interests include network and information security.

Jinjun Chen (Fellow, IEEE) received the PhD degree in computer science from the Swinburne University of Technology, Melbourne, VIC, Australia. He is a professor with the Swinburne University of Technology. His research interests include blockchain, cloud computing, data privacy, and data systems.

Sajal K. Das (Fellow, IEEE) is a Curators' distinguished professor of computer science and Daniel St. Clair Endowed chair with the Missouri University of Science and Technology, USA. His research interests include security and privacy, cyber-physical systems, IoT, wireless sensor networks, mobile and pervasive computing, and cloud computing. He is the editor-in-chief of Elsevier's Pervasive and Mobile Computing journal, and associate editor of *IEEE Transactions on Sustainable Computing*, *IEEE Transactions on Dependable and Secure Computing*, *IEEE/ACM Tranasctions on Networking*, and *ACM Transactions on Sensor Networks*.

Xiaosong Zhang received the MS and PhD degrees in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 1999 and 2011, respectively. He is currently a professor with the University of Electronic Science and Technology of China. He is also the Cheung Kong Scholar distinguished professor. His current research interests include blockchain, Big Data security, and AI security.