Secure and Efficient Skyline Queries on Encrypted Data

Jinfei Liu, *Member, IEEE*, Juncheng Yang, *Member, IEEE*,
Li Xiong, *Member, IEEE*, and Jian Pei, *Fellow, IEEE*

Abstract—Outsourcing data and computation to cloud server provides a cost-effective way to support large scale data storage and query processing. However, due to security and privacy concerns, sensitive data (e.g., medical records) need to be protected from the cloud server and other unauthorized users. One approach is to outsource encrypted data to the cloud server and have the cloud server perform query processing on the encrypted data only. It remains a challenging task to support various queries over encrypted data in a secure and efficient way such that the cloud server does not gain any knowledge about the data, query, and query result. In this paper, we study the problem of secure skyline queries over encrypted data. The skyline query is particularly important for multi-criteria decision making but also presents significant challenges due to its complex computations. We propose a fully secure skyline query protocol on data encrypted using semantically-secure encryption. As a key subroutine, we present a new secure dominance protocol, which can be also used as a building block for other queries. Furthermore, we demonstrate two optimizations, data partitioning and lazy merging, to further reduce the computation load. Finally, we provide both serial and parallelized implementations and empirically study the protocols in terms of efficiency and scalability under different parameter settings, verifying the feasibility of our proposed solutions.

Index Terms—Skyline, secure, efficient, parallel, semi-honest

1 Introduction

As an emerging computing paradigm, cloud computing attracts increasing attention from both research and industry communities. Outsourcing data and computation to cloud server provides a cost-effective way to support large scale data storage and query processing. However, due to security and privacy concerns, sensitive data need to be protected from the cloud server as well as other unauthorized users.

A common approach to protect the confidentiality of outsourced data is to encrypt the data (e.g., [15], [33]). To protect the confidentiality of the query from cloud server, authorized clients also send encrypted queries to the cloud server. Fig. 1 illustrates our problem scenario of secure query processing over encrypted data in the cloud. The data owner outsources encrypted data to the cloud server. The cloud server processes encrypted queries from the client on the encrypted data and returns the query result to the client. During the query processing, the cloud server should not gain any knowledge about the data, data patterns, query, and query result.

Fully homomorphic encryption schemes [15] ensure strong security while enabling arbitrary computations on the encrypted data. However, the computation cost is prohibitive in practice. Trusted hardware such as Intel's Software Guard Extensions (SGX) brings a promising alternative, but still has limitations in its security guarantees [10]. Many techniques (e.g., [17], [38]) have been proposed to support specific queries or computations on encrypted data with varying degrees of security guarantee and efficiency (e.g., by weaker encryptions). Focusing on similarity search, secure k-nearest neighbor (kNN) queries, which return k most similar (closest) records given a query record, have been extensively studied [12], [20], [40], [42].

In this paper, we focus on the problem of secure skyline queries on encrypted data, another type of similarity search important for multi-criteria decision making. The *skyline* or *Pareto* of a multi-dimensional dataset given a query point consists of the data points that are not *dominated* by other points. A data point dominates another if it is closer to the query point in at least one dimension and at least as close to the query point in every other dimension. The skyline query is particularly useful for selecting similar (or best) records when a single aggregated distance metric with all dimensions is hard to define. The assumption of kNN queries is that the relative weights of the attributes are known in advance, so that a single similarity metric can be computed between a pair of records aggregating the similarity between all attribute pairs. However, this assumption does not always hold in practical applications. In many scenarios, it is desirable to retrieve similar records considering all possible relative weights of the attributes (e.g., considering only one attribute, or an arbitrary combination of attributes), which is essentially the skyline or the "pareto-similar" records.

Motivating Example. Consider a hospital who wishes to outsource its electronic health records to the cloud and the data is encrypted to ensure data confidentiality. Let P denote a sample heart disease dataset with attributes ID, age, trestbps (resting blood pressure). We sampled four patient records $\mathbf{p}_1, \ldots, \mathbf{p}_4$ from the heart disease dataset of UCI machine

- *J. Liu, J. Yang, and L. Xiong are with the Department of Mathematics and Computer Science, Emory University, Atlanta, GA 30322. E-mail: {jinfei.liu, juncheng.yang, lxiong}@emory.edu.*
- *J. Pei is with the School of Computing Science, Simon Fraser University, Burnaby, BC V5A 1S6, Canada. E-mail: jpei@cs.sfu.ca.*

Manuscript received 15 Feb. 2018; revised 2 June 2018; accepted 7 July 2018. Date of publication 19 July 2018; date of current version 31 May 2019. (Corresponding author: Jinfei Liu.)
Recommended for acceptance by J. Xu.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2018.2857471

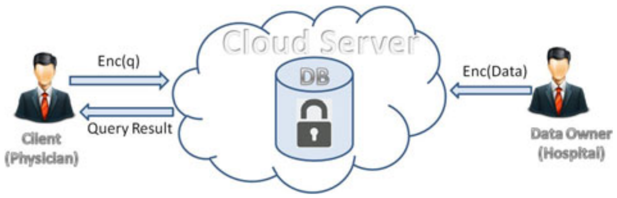


Fig. 1. Secure similarity queries.

learning repository [22] as shown in Table 1 a and Fig. 2. Consider a physician who is treating a heart disease patient $\mathbf{q} = (41, 125)$ and wishes to retrieve similar patients in order to enhance and personalize the treatment for patient $\mathbf{q}$. While it is unclear how to define the attribute weights for kNN queries ($\mathbf{p}_1$ is the nearest if only age is considered while $\mathbf{p}_2, \mathbf{p}_3$ are the nearest if only trestbps is considered), skyline provides all pareto-similar records that are not dominated by any other records. Skyline includes all possible 1NN results by considering all possible relative attribute weights, and hence can serve as a filter for users. Given the query $\mathbf{q}$, we can map the data points to a new space with $\mathbf{q}$ as the origin and the distance to $\mathbf{q}$ as the mapping function. The mapped records $\mathbf{t}_i[j] = |\mathbf{p}_i[j] - \mathbf{q}[j]| + \mathbf{q}[j]$ on each dimension j are shown in Table 1 b and also in Fig. 2. It is easy to see that $\mathbf{t}_1$ and $\mathbf{t}_2$ are skyline in the mapped space, which means $\mathbf{p}_1$ and $\mathbf{p}_2$ are skyline with respect to query $\mathbf{q}$.

Our goal is for the cloud server to compute the skyline query given $\mathbf{q}$ on the encrypted data without revealing the data, the query $\mathbf{q}$, the final result set $\{\mathbf{p}_1, \mathbf{p}_2\}$, as well as any intermediate result (e.g., $\mathbf{t}_2$ dominates $\mathbf{t}_4$) to the cloud. We note that skyline computation (with query point at the origin) is a special case of skyline queries.

Challenges. Designing a fully secure protocol for skyline queries over encrypted data presents significant challenges due to the complex comparisons and computations. Let P denotes a set of n tuples $\mathbf{p}_1, \ldots, \mathbf{p}_n$ with m attributes and $\mathbf{q}$ denotes input query tuple. In kNN queries, we only need to compute the distances between each tuple $\mathbf{p}_i$ and the query tuple $\mathbf{q}$ and then choose the k tuples corresponding to the k smallest distances. In skyline queries, for each tuple $\mathbf{p}_i$, we need to compare it with all other tuples to check dominance. For each comparison between two tuples $\mathbf{p}_a$ and $\mathbf{p}_b$, we need to compare all their m attributes and for comparison of each attribute $\mathbf{p}[j]$, there are three different outputs, i.e., $\mathbf{p}_a[j] < (=, >)\ \mathbf{p}_b[j]$. Therefore, there are 3^m different outputs for each comparison between two tuples, based on which we need to determine if one tuple dominates the other. How to determine the $2^m - 1$ cases that satisfy p_a dominates p_b efficiently while protecting intermediate results (e.g., whether two attribute values are the same) is particularly challenging.

Such complex comparisons and computations require more complex protocol design in order to carry out the computations on the encrypted data given an encryption scheme with semantic security (instead of weaker order-preserving or other property-preserving encryptions). In addition, the extensive intermediate result means more indirect information about the data can be potentially revealed (e.g., which tuple dominates which other, whether there are duplicate tuples or equivalent attribute values) even if the exact data is protected. This makes it challenging to design a fully secure and efficient skyline query protocol in which the cloud should not gain any knowledge about the data including indirect data patterns.

TABLE 1
Sample of Heart Disease Dataset

(a) Original data.

ID	age	trestbps
$\mathbf{p}_1$	40	140
$\mathbf{p}_2$	39	120
$\mathbf{p}_3$	45	130
$\mathbf{p}_4$	37	140

(b) Mapped Data.

ID	age	trestbps
$\mathbf{t}_1$	42	140
$\mathbf{t}_2$	43	130
$\mathbf{t}_3$	45	130
$\mathbf{t}_4$	45	140

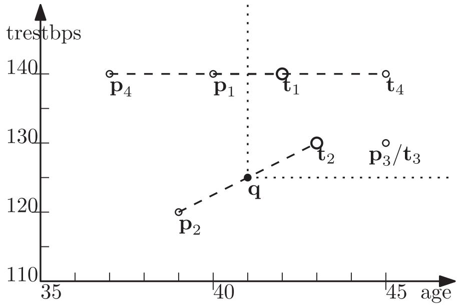


Fig. 2. Dynamic skyline query.

Contributions. We summarize our contributions as follows.

- We study the secure skyline problem on encrypted data with semantic security for the first time. We assume the data is encrypted using the Paillier cryptosystem which provides semantic security and is partially homomorphic.
- We propose a fully secure dominance protocol, which can be used as a building block for skyline queries as well as other queries, e.g., reverse skyline queries [11] and k-skyband queries [34].
- We present two secure skyline query protocols. The first one, served as a basic and efficient solution, leaks some indirect data patterns to the cloud server. The second one is fully secure and ensures that the cloud gains no knowledge about the data including indirect patterns. The proposed protocols exploit the partial (additive) homomorphism as well as novel permutation and perturbation techniques to ensure the correct result is computed while guaranteeing privacy. We provide security and complexity analysis of the proposed protocols.
- Compared with our conference version [30], we present two new optimizations, data partitioning and lazy merging, to further reduce the computation load. For the data partitioning, we theoretically analyze the optimal number of partitions given the number of points, the expected number of output skyline points, the number of decomposed bits, and the number of dimensions. In addition, we propose a lazy merging scheme that aims to reduce computation overhead due to the smaller partition sizes at the later stage of the partitioning scheme.
- We also provide a complete implementation including both serial and parallelized versions which can be deployed in practical settings. We empirically study the efficiency and scalability of the implementations under different parameter settings, verifying the feasibility of our proposed solutions.

Organization. The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 introduces background definitions as well as our problem setting.

The security subprotocols for general functions that will be used in our secure skyline protocol are introduced in Section 4. The key subroutine of secure skyline protocols, secure dominance protocol, is shown in Section 5. The complete secure skyline protocols are presented in Section 6. We illustrate two optimizations to further reduce the computation load in Section 7. We report the experimental results and findings in Section 8. Section 9 concludes the paper.

2 Related Work

Skyline. The skyline computation problem was first studied in computational geometry field [3], [25] where they focused on worst-case time complexity. [23], [29] proposed output-sensitive algorithms achieving $O(n \, log \, k)$ in worst-case where k is the number of skyline points which is far less than n in general.

Since the introduction of the skyline operator by Börzsönyi et al. [5], skyline has been extensively studied in the database field. Kossmann et al. [24] studied the progressive algorithm for skyline queries. Different variants of the skyline problem have been studied (e.g., subspace skyline [8], uncertain skyline [32] [36], group-based skyline [26], [28], [44], skyline diagram [31]).

Secure Query Processing on Encrypted Data. Fully homomorphic encryption schemes [15] enable arbitrary computations on encrypted data. Even though it is shown that [15] we can build such encryption schemes with polynomial time, they remain far from practical even with the state of the art implementations [18].

Many techniques (e.g., [17], [38]) have been proposed to support specific queries or computations on encrypted data with varying degrees of security guarantee and efficiency (e.g., by weaker encryptions). We are not aware of any formal work on secure skyline queries over encrypted data with semantic security. Bothe et al. [6] and their demo version [7] illustrated an approach about skyline queries on so-called "encrypted" data without any formal security guarantee. Another work [9] studied the verification of skyline query result returned by an untrusted cloud server.

The closely related work is secure kNN queries [12], [19], [20], [35], [37], [40], [42], [43] which we discuss in more detail here. Wong et al. [40] proposed a new encryption scheme called asymmetric scalar-product-preserving encryption. In their work, data and query are encrypted using slightly different encryption schemes and all clients know the private key. Hu et al. [20] proposed a method based on provably secure privacy homomorphism encryption scheme. However, both schemes are vulnerable to the chosen-plaintext attacks as illustrated by Yao et al. [42]. Yao et al. [42] proposed a new method based on secure Voronoi diagram. Instead of asking the cloud server to retrieve the exact kNN result, their method retrieve a relevant encrypted partition such that it is guaranteed to contain the kNN of the query point. Hashem et al. [19] identified the challenges in preserving user privacy for group nearest neighbor queries and provided a comprehensive solution to this problem. Yi et al. [43] proposed solutions for secure kNN queries based on oblivious transfer paradigm. Recently, Elmehdwi et al. [12] proposed a secure kNN query protocol on data encrypted using Paillier cryptosystem that ensures data privacy and query privacy, as well as low (or no) computation overhead on client and data owner using two non-colluding cloud servers. Our work follows this setting and addresses skyline queries.

TABLE 2
The Summary of Notations

Notation	Definition
P	dataset of n points/tuples/records
$\mathbf{p}_i[j]$	the jth attribute of $\mathbf{p}_i$
$\mathbf{q}$	query tuple of client
n	number of points in P
m	number of dimensions
k	number of skyline
l	number of bits
K	key size
pk/sk	public/private key
$[\![a]\!]$	encrypted vector of the individual bits of a
$\hat{a}$	binary bit
$(a)_B^{(i)}$	the ith bit of binary number a

Other works studied kNN queries in the secure multi-party computation (SMC) setting [37] (data is distributed between two parties who want to cooperatively compute the answers without revealing to each other their private data), or private information retrieval (PIR) setting [35] (query is private while data is public), which are different from our settings.

Secure Multi-party Computation (SMC). SMC was first proposed by Yao [41] for two-party setting and then extended by Goldreich et al. [16] to multi-party setting. SMC refers to the problem where a set of parties with private inputs wish to compute some joint function of their inputs. There are techniques such as garbled circuits [21] and secret sharing [2] that can be used for SMC. In this paper, all protocols assume a two-party setting, but different from the traditional SMC setting. Namely, we have party $\mathcal{C}_1$ with encrypted input and party $\mathcal{C}_2$ with the private key sk. The goal is for $\mathcal{C}_1$ to obtain an encrypted result of a function on the input without disclosing the original input to either $\mathcal{C}_1$ or $\mathcal{C}_2$.

3 Preliminaries and Problem Definitions

In this section, we first illustrate some background knowledge on skyline computation and dynamic skyline query, and then describe the security model we use in this paper. For references, a summary of notations is given in Table 2.

3.1 Skyline Definitions

Definition 1 (Skyline). *Given a dataset $P = \{\mathbf{p}_1, ..., \mathbf{p}_n\}$ in m-dimensional space. Let $\mathbf{p}_a$ and $\mathbf{p}_b$ be two different points in P, we say $\mathbf{p}_a$ dominates $\mathbf{p}_b$, denoted by $\mathbf{p}_a \prec \mathbf{p}_b$, if for all j, $\mathbf{p}_a[j] \leq \mathbf{p}_b[j]$, and for at least one j, $\mathbf{p}_a[j] < \mathbf{p}_b[j]$, where $\mathbf{p}_i[j]$ is the jth dimension of $\mathbf{p}_i$ and $1 \leq j \leq m$. The skyline points are those points that are not dominated by any other point in P.*

Definition 2 (Dynamic Skyline Query) [11]. *Given a dataset $P = \{\mathbf{p}_1, ..., \mathbf{p}_n\}$ and a query point $\mathbf{q}$ in m-dimensional space. Let $\mathbf{p}_a$ and $\mathbf{p}_b$ be two different points in P, we say $\mathbf{p}_a$ dynamically dominates $\mathbf{p}_b$ with regard to the query point $\mathbf{q}$, denoted by $\mathbf{p}_a \prec \mathbf{p}_b$, if for all j, $|\mathbf{p}_a[j] - \mathbf{q}[j]| \leq |\mathbf{p}_b[j] - \mathbf{q}[j]|$, and for at least one j, $|\mathbf{p}_a[j] - \mathbf{q}[j]| < |\mathbf{p}_b[j] - \mathbf{q}[j]|$, where $\mathbf{p}_i[j]$ is the jth dimension of $\mathbf{p}_i$ and $1 \leq j \leq m$. The skyline points are those points that are not dynamically dominated by any other point in P.*

The traditional skyline definition is a special case of dynamic skyline query in which the query point is the origin. On the other hand, dynamic skyline query is equivalent to traditional skyline computation if we map the points to a new space with the query point $\mathbf{q}$ as the origin and the absolute distances to $\mathbf{q}$ as mapping functions. So the protocols we will present in the paper also work for traditional skyline computation (without an explicit query point).

Example 1. Consider Table 1 and Fig. 2 as a running example. Given data points $\mathbf{p}_1$ to $\mathbf{p}_4$ and query point $\mathbf{q}$, the mapped data points are computed as $\mathbf{t}_i[j] = |\mathbf{p}_i[j] - \mathbf{q}[j]| + \mathbf{q}[j]$. We see that $\mathbf{t}_1, \mathbf{t}_2$ are the skyline in the mapped space, and $\mathbf{p}_1, \mathbf{p}_2$ are the skyline with respect to query $\mathbf{q}$ in the original space.

3.2 Skyline Computation

Skyline computation has been extensively studied as we discussed in Section 2. We illustrate an iterative skyline computation algorithm (Algorithm 1) which will be used as the basis of our secure skyline protocol. We note that this is not the most efficient algorithm to compute skyline for plaintext compared to the divide-and-conquer algorithm [25]. We construct our secure skyline protocol based on this algorithm for two reasons: 1) the divide-and-conquer approach is less suitable if not impossible for a secure implementation compared to the iterative approach, 2) the performance of the divide-and-conquer algorithm deteriorate with the "curse of dimensionality".

The general idea of Algorithm 1 is to first map the data points to the new space with the query point as origin (Lines 1-3). Given the new data points, it computes the sum of all attributes for each tuple $S(\mathbf{t}_i)$ (Line 6) and chooses the tuple $\mathbf{t}_{\min}$ with smallest $S(\mathbf{t}_i)$ as a skyline because no other tuples can dominate it. It then deletes those tuples dominated by $\mathbf{t}_{\min}$. The algorithm repeats this process for the remaining tuples until an empty dataset T is reached.

Algorithm 1. Skyline Computation

```
   Input: A dataset $P$ and a query $\mathbf{q}$.
   Output: Skyline of $P$.
 1 for $i = 1$ to $n$ do
 2   for $j = 1$ to $m$ do
 3     $\mathbf{t}_i[j] = |\mathbf{p}_i[j] - \mathbf{q}[j]|$;
 4 while the dataset $T$ is not empty do
 5   for $i = 1$ to size of dataset $T$ do
 6     $S(\mathbf{t}_i) = \sum_{j=1}^{m} \mathbf{t}_i[j]$;
 7     choose the tuple $\mathbf{t}_{\min}$ with smallest $S(\mathbf{t}_i)$ as a skyline;
 8     add corresponding tuple $\mathbf{p}_{\min}$ to the skyline pool;
 9     delete those tuples dominated by $\mathbf{t}_{\min}$ from $T$;
10     delete tuple $\mathbf{t}_{\min}$ from $T$;
11 return skyline pool;
```

Example 2. Given the mapped data points $\mathbf{t}_1, ..., \mathbf{t}_4$, we begin by computing the attribute sum for each tuple as $S(\mathbf{t}_1) = 16$, $S(\mathbf{t}_2) = 7$, $S(\mathbf{t}_3) = 9$, and $S(\mathbf{t}_4) = 19$. We choose the tuple with smallest $S(\mathbf{t}_i)$, i.e., $\mathbf{t}_2$, as a skyline tuple, delete $\mathbf{t}_2$ from dataset T and add $\mathbf{p}_2$ to the skyline pool. We then delete tuples $\mathbf{t}_3$ and $\mathbf{t}_4$ from T because they are dominated by $\mathbf{t}_2$. Now, there is only $\mathbf{t}_1$ in T. We add $\mathbf{p}_1$ to the skyline pool. After deleting $\mathbf{t}_1$ from T, T is empty and the algorithm terminates. $\mathbf{p}_1$ and $\mathbf{p}_2$ in the skyline pool are returned as the query result.

3.3 Problem Setting

We now describe our problem setting for secure skyline queries over encrypted data. Consider a data owner (e.g., hospital, CDC) with a dataset P. Before outsourcing the data, the data owner encrypts each attribute of each record $\mathbf{p}_i[j]$ using a semantically secure public-key cryptosystem. Fully homomorphic encryption schemes ensure strong security while enabling arbitrary computations on the encrypted data. However, the computation cost is prohibitive in practice. Partially homomorphic encryption is much more efficient but only provides partially (either additive or multiplicative) homomorphic properties. Among them, we chose Paillier [33] mainly due to its additive homomorphic properties as we employ significantly more additions than multiplications in our protocol. Furthermore, we can also utilize its homomorphic multiplication between ciphertext and plaintext. We use pk and sk to denote the public key and private key, respectively. Data owner sends $E_{pk}(\mathbf{p}_i[j])$ for $i = 1, ..., n$ and $j = 1, ..., m$ to cloud server $\mathcal{C}_1$.

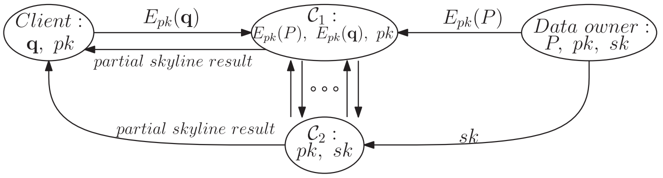


Fig. 3. Overview of protocol setting.

Consider an authorized client (e.g., physician) who wishes to query the skyline tuples corresponding to query tuple $\mathbf{q} = (\mathbf{q}[1], ..., \mathbf{q}[m])$. In order to protect the sensitive query tuple, the client uses the same public key pk to encrypt the query tuple and sends $E_{pk}(\mathbf{q}) = (E_{pk}(\mathbf{q}[1]), ..., E_{pk}(\mathbf{q}[m]))$ to cloud server $\mathcal{C}_1$.

Our goal is to enable the cloud server to compute and return the skyline to the client without learning any information about the data and the query. In addition to guaranteeing the correctness of the result and the efficiency of the computation, the computation should require no or minimal interaction from the client or the data owner for practicality. To achieve this, we assume there is an additional non-colluding cloud server, $\mathcal{C}_2$, which will hold the private key sk shared by the data owner and assist with the computation. This way, the data owner does not need to participate in any computation. The client also does not need to participate in any computation except combining the partial result from $\mathcal{C}_1$ and $\mathcal{C}_2$ for final result. An overview of the protocol setting is shown in Fig. 3.

3.4 Security Model

Adversary Model. We adopt the *semi-honest* adversary model in our study. In any multi-party computation setting, a *semi-honest* party correctly follows the protocol specification, yet attempts to learn additional information by analyzing the transcript of messages received during the execution. By semi-honest model, this work implicitly assumes that the two cloud servers do not collude.

There are two main reasons to adopt the semi-honest adversary model in our study. First, developing protocols under the semi-honest setting is an important first step towards constructing protocols with stronger security guarantees [21]. Using zero-knowledge proofs [14], these protocols can be transformed into secure protocols under the malicious model. Second, the semi-honest model is realistic in current cloud market. $\mathcal{C}_1$ and $\mathcal{C}_2$ are assumed to be two cloud servers, which are legitimate, well-known companies (e.g., Amazon, Google, and Microsoft). A collusion between them is highly unlikely. Therefore, following the work done in [12], [27], [45], we also adopt the semi-honest adversary model for this paper. Please see Security Definition in the Semi-honest Model and Paillier Cryptosystem in the Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2018.2857471.

Desired Privacy Properties. Our security goal is to protect the data and the query as well as the query result from the cloud servers. We summarize the desired privacy properties below. After the execution of the entire protocol, the following should be achieved.

- *Data Privacy.* Cloud servers $\mathcal{C}_1$ and $\mathcal{C}_2$ know nothing about the exact data except the size pattern, the client knows nothing about the dataset except the skyline query result.
- *Data Pattern Privacy.* Cloud servers $\mathcal{C}_1$ and $\mathcal{C}_2$ know nothing about the data patterns (indirect data knowledge) due to intermediate result, e.g., which tuple dominates which other tuple.
- *Query Privacy.* Data owner, cloud servers $\mathcal{C}_1$ and $\mathcal{C}_2$ know nothing about the query tuple $\mathbf{q}$.
- *Result Privacy.* Cloud servers $\mathcal{C}_1$ and $\mathcal{C}_2$ know nothing about the query result, e.g., which tuples are in the skyline result.

4 Basic Security Subprotocols

In this section, we present a set of secure subprotocols for computing basic functions on encrypted data that will be used to construct our secure skyline query protocol. All protocols assume a two-party setting, namely, $\mathcal{C}_1$ with encrypted input and $\mathcal{C}_2$ with the private key sk as shown in Fig. 3. The goal is for $\mathcal{C}_1$ to obtain an encrypted result of a function on the input without disclosing the original input to either $\mathcal{C}_1$ or $\mathcal{C}_2$. We note that this is different from the traditional two-party secure computation setting with techniques such as garbled circuits [21] where each party holds a private input and they wish to compute a function of the inputs. For each function, we describe the input and output, present our proposed protocol or provide a reference if existing solutions are available. Due to limited space, we omit the security proof which can be derived by the simulation and composition theorem in a straightforward way. Please see Secure Multiplication (SM), Secure Bit Decomposition (SBD), and Secure Boolean Operations in the appendix, available in the online supplemental material.

4.1 Secure Minimum and Secure Comparison

Secure minimum protocol and secure comparison protocol have been extensively studied in cryptography community [1], [13], [39] and database community [12], [45]. Secure comparison protocol can be easily adapted to secure minimum protocol, and vice versa. For example, if we set $E_{pk}(out)$ as the result of secure comparison $E_{pk}(Bool(a \le b))$ known by cloud server $\mathcal{C}_1$ (it will be $E_{pk}(1)$ when $a \le b$ and $E_{pk}(0)$ when $a > b$), $\mathcal{C}_1$ can get $E_{pk}(min(a, b))$ by computing $E_{pk}(a * out + b * \neg out)$.

We analyzed the existing protocols and observed that both secure minimum (SMIN) algorithms [12], [45] from database community for selecting a minimum have a security weakness, i.e., $\mathcal{C}_2$ can determine whether the two numbers are equal to each other. We point out the security weakness in the appendix, available in the online supplemental material.

Therefore, we adapted the secure minimum/comparison protocols [39] from cryptography community in this paper. The basic idea of those protocols is that for any two l bit numbers a and b, the most significant bit (z_l) of $z = 2^l + a - b$ indicates the relationship between a and b, i.e., $z_l = 0 \Leftrightarrow a < b$. We list the secure minimum/comparison protocols we used in this paper below.

Secure Less Than or Equal (SLEQ). Assume a cloud server $\mathcal{C}_1$ with encrypted input $E_{pk}(a)$ and $E_{pk}(b)$, and a cloud server $\mathcal{C}_2$ with the private key sk, where a and b are two numbers not known to $\mathcal{C}_1$ and $\mathcal{C}_2$. The goal of the SLEQ protocol is to securely compute the encrypted boolean output $E_{pk}(Bool(a \le b))$, such that only $\mathcal{C}_1$ knows $E_{pk}(Bool(a \le b))$ and no information related to a and b is revealed to $\mathcal{C}_1$ or $\mathcal{C}_2$.

Secure Equal (SEQ). Assume a cloud server $\mathcal{C}_1$ with encrypted input $E_{pk}(a)$ and $E_{pk}(b)$, and a cloud server $\mathcal{C}_2$ with the private key sk, where a and b are two numbers not known to $\mathcal{C}_1$ and $\mathcal{C}_2$. The goal of the SEQ protocol is to securely compute the encrypted boolean output $E_{pk}(Bool(a == b))$, such that only $\mathcal{C}_1$ knows $E_{pk}(Bool(a == b))$ and no information related to $Bool(a == b)$ is revealed to $\mathcal{C}_1$ or $\mathcal{C}_2$.

Secure Less (SLESS). Assume a cloud server $\mathcal{C}_1$ with encrypted input $E_{pk}(a)$ and $E_{pk}(b)$, and a cloud server $\mathcal{C}_2$ with the private key sk, where a and b are two numbers not known to $\mathcal{C}_1$ and $\mathcal{C}_2$. The goal of the SLESS protocol is to securely compute the encrypted boolean output $E_{pk}(Bool(a < b))$, such that only $\mathcal{C}_1$ knows $E_{pk}(Bool(a < b))$ and no information related to $Bool(a < b)$ is revealed to $\mathcal{C}_1$ or $\mathcal{C}_2$. This can be simply implemented by conjunction from the output of SEQ and SLEQ.

Secure Minimum (SMIN). Assume a cloud server $\mathcal{C}_1$ with encrypted input $E_{pk}(a)$ and $E_{pk}(b)$, and a cloud server $\mathcal{C}_2$ with the private key sk, where a and b are two numbers not known to both parties. The goal of the SMIN protocol is to securely compute encrypted minimum value of a and b, $E_{pk}(min(a, b))$, such that only $\mathcal{C}_1$ knows $E_{pk}(min(a, b))$ and no information related to a and b is revealed to $\mathcal{C}_1$ or $\mathcal{C}_2$. Benefiting from the probabilistic property of Paillier, the ciphertext of $min(a, b)$, i.e., $E_{pk}(min(a, b))$ is different from the ciphertext of a, b, i.e., $E_{pk}(a)$, $E_{pk}(b)$. Therefore, $\mathcal{C}_1$ does not know which of a or b is $min(a, b)$. In general, assume $\mathcal{C}_1$ has n encrypted values, the goal of SMIN protocol is to securely compute encrypted minimum of the n values.

5 Secure Dominance Protocol

The key to compute skyline is to compute dominance relationship between two tuples. Assume a cloud server $\mathcal{C}_1$ with encrypted tuples $\mathbf{a} = (\mathbf{a}[1], ..., \mathbf{a}[m])$, $\mathbf{b} = (\mathbf{b}[1], ..., \mathbf{b}[m])$ and a cloud server $\mathcal{C}_2$ with the private key sk, where $\mathbf{a}$ and $\mathbf{b}$ are not known to both parties. The goal of the secure dominance (SDOM) protocol is to securely compute $E_{pk}(Bool(\mathbf{a} \prec \mathbf{b}))$ such that only $\mathcal{C}_1$ knows $E_{pk}(1)$ if $\mathbf{a} \prec \mathbf{b}$, otherwise, $E_{pk}(0)$.

Protocol Design. Given any two tuples $\mathbf{a} = (\mathbf{a}[1], ..., \mathbf{a}[m])$ and $\mathbf{b} = (\mathbf{b}[1], ..., \mathbf{b}[m])$, recall the definition of skyline, we say $\mathbf{a} \prec \mathbf{b}$ if for all j, $\mathbf{a}[j] \le \mathbf{b}[j]$ and for at least one j, $\mathbf{a}[j] < \mathbf{b}[j]$ $(1 \le j \le m)$. If for all j, $\mathbf{a}[j] \le \mathbf{b}[j]$, we have either $\mathbf{a} = \mathbf{b}$ or $\mathbf{a} \prec \mathbf{b}$. We refer to this case as $\mathbf{a} \preceq \mathbf{b}$. The basic idea of secure dominance protocol is to first determine whether $\mathbf{a} \preceq \mathbf{b}$, and then determine whether $\mathbf{a} = \mathbf{b}$.

The detailed protocol is shown in Algorithm 2. For each attribute, $\mathcal{C}_1$ and $\mathcal{C}_2$ cooperatively use the secure less than or equal (SLEQ) protocol to compute $E_{pk}(Bool(\mathbf{a}[j] \le \mathbf{b}[j]))$.

And then $\mathcal{C}_1$ and $\mathcal{C}_2$ cooperatively use SAND to compute $\Phi = \delta_1 \wedge, ..., \wedge\delta_m$. If $\Phi = E_{pk}(1)$, it means $\mathbf{a} \preceq \mathbf{b}$, otherwise, $\mathbf{a} \npreceq \mathbf{b}$. We note that, the dominance relationship information Φ is known only to $\mathcal{C}_1$ in ciphertext. Therefore, both $\mathcal{C}_1$ and $\mathcal{C}_2$ do not know any information about whether $\mathbf{a} \preceq \mathbf{b}$.

Algorithm 2. Secure Dominance Protocol

```
    **Input:** $\mathcal{C}_1$ has $E_{pk}(\mathbf{a}), E_{pk}(\mathbf{b})$ and $\mathcal{C}_2$ has $sk$.
    **Output:** $\mathcal{C}_1$ gets $E_{pk}(1)$ if $\mathbf{a} \prec \mathbf{b}$, otherwise, $\mathcal{C}_1$ gets $E_{pk}(0)$.
$\mathcal{C}_1$ and $\mathcal{C}_2$:
for $j = 1$ to $m$ do
    $\mathcal{C}_1$ gets $\delta_j = E_{pk}(Bool(\mathbf{a}[j] \leq \mathbf{b}[j]))$ by SLEQ;
use SAND to compute $\Phi = \delta_1 \wedge ..., \wedge\delta_m$;
$\mathcal{C}_1$:
compute $\alpha = E_{pk}(\mathbf{a}[1])\times, \ldots, \times E_{pk}(\mathbf{a}[m])$;
compute $\beta = E_{pk}(\mathbf{b}[1])\times, \ldots, \times E_{pk}(\mathbf{b}[m])$;
$\mathcal{C}_1$ and $\mathcal{C}_2$:
$\mathcal{C}_1$ gets $\sigma = E_{pk}(Bool(\alpha < \beta))$ by employing SLESS;
$\mathcal{C}_1$ gets $\Psi = \sigma \wedge \Phi$ as the final dominance relationship using SAND;
```

Next, we need to determine if $\mathbf{a} \neq \mathbf{b}$. Only if $\mathbf{a} \neq \mathbf{b}$, then $\mathbf{a} \prec \mathbf{b}$. One naive way is to employ SEQ protocol for each pair of attribute and then take the conjunction of the output. We propose a more efficient way which is to check whether $S(\mathbf{a}) < S(\mathbf{b})$, where $S(\mathbf{a})$ is the attribute sum of tuple $\mathbf{a}$. If $S(\mathbf{a}) < S(\mathbf{b})$, then it is impossible that $\mathbf{a} = \mathbf{b}$. As the algorithm shows, $\mathcal{C}_1$ computes the sum of all attributes $\alpha = E_{pk}(\mathbf{a}[1] + ... + \mathbf{a}[m])$ and $\beta = E_{pk}(\mathbf{b}[1] + ... + \mathbf{b}[m])$ based on the additive homomorphic property. Then $\mathcal{C}_1$ and $\mathcal{C}_2$ cooperatively use SLESS protocol to compute $\sigma = E_{pk}(Bool(\alpha < \beta))$. Finally, $\mathcal{C}_1$ and $\mathcal{C}_2$ cooperatively use SAND protocol to compute the final dominance relationship $\Psi = \sigma \wedge \Phi$ which is only known to $\mathcal{C}_1$ in ciphertext. $\Psi = E_{pk}(1)$ means $\mathbf{a} \prec \mathbf{b}$, otherwise, $\mathbf{a} \nprec \mathbf{b}$.

Security Analysis. Based on the composition theorem (Theorem 2), the security of secure dominance protocol relies on the security of SLEQ, SLESS, and SAND, which have been shown in existing works.

Complexity Analysis. To determine $\mathbf{a} \preceq \mathbf{b}$, Algorithm 2 requires $O(m)$ encryptions and decryptions. Then to determine if $\mathbf{a} = \mathbf{b}$, Algorithm 2 requires $O(1)$ encryptions and decryptions. Therefore, our secure dominance protocol requires $O(m)$ encryptions and decryptions in total.

6 Secure Skyline Protocol

In this section, we first propose a basic secure skyline protocol and show why such a simple solution is not secure. Then we propose a fully secure skyline protocol. Both protocols are constructed by using the security primitives discussed in Section 4 and the secure dominance protocol in Section 5.

As mentioned in Algorithm 1, given a skyline query $\mathbf{q}$, it is equivalent to compute the skyline in a transformed space with the query point $\mathbf{q}$ as the origin and the absolute distances to $\mathbf{q}$ as mapping functions. Hence we first show a preprocessing step in Algorithm 3 which maps the dataset to the new space. Since the skyline only depends on the order of the attribute values, we use $(\mathbf{p}_i[j] - \mathbf{q}[j])^2$ which is easier to compute than $|\mathbf{p}_i[j] - \mathbf{q}[j]|$ as the mapping function.[1] After Algorithm 3, $\mathcal{C}_1$ has the encrypted dataset $E_{pk}(P)$ and $E_{pk}(T)$, $\mathcal{C}_2$ has the private key sk. The goal is to securely compute the skyline by $\mathcal{C}_1$ and $\mathcal{C}_2$ without participation of data owner and the client.

Algorithm 3. Preprocessing

```
    **Input:** $\mathcal{C}_1$ has $E_{pk}(P)$, $\mathcal{C}_2$ has $sk$, and the client has $\mathbf{q}$.
    **Output:** $\mathcal{C}_1$ obtains the new encrypted dataset $E_{pk}(T)$.
Client:
send $(E_{pk}(-\mathbf{q}[1]), ..., E_{pk}(-\mathbf{q}[m]))$ to $\mathcal{C}_1$;
$\mathcal{C}_1$:
for $i = 1$ to $n$ do
    for $j = 1$ to $m$ do
        $E_{pk}(temp_i[j]) = E_{pk}(\mathbf{p}_i[j] - \mathbf{q}[j]) = E_{pk}(\mathbf{p}_i[j]) \times E_{pk}(-\mathbf{q}[j]) \bmod N^2$;
$\mathcal{C}_1$ and $\mathcal{C}_2$:
use SM protocol to compute $E_{pk}(T) = (E_{pk}(\mathbf{t}_1), \ldots, E_{pk}(\mathbf{t}_n))$ only known by $\mathcal{C}_1$, where $E_{pk}(\mathbf{t}_i) = (E_{pk}(\mathbf{t}_i[1]), \ldots, E_{pk}(\mathbf{t}_i[m]))$ and $E_{pk}(\mathbf{t}_i[j]) = E_{pk}(temp_i[j]) \times E_{pk}(temp_i[j])$;
```

6.1 Basic Protocol

We first illustrate a straw-man protocol which is straightforward but not fully secure (as shown in Algorithm 4). The idea is to implement each of the steps in Algorithm 1 using the primitive secure protocols. $\mathcal{C}_1$ first determines the terminal condition, if there is no tuple exists in dataset $E_{pk}(T)$, the protocol ends, otherwise, the protocol proceeds as follows.

Compute Minimum Attribute Sum. $\mathcal{C}_1$ first computes the sum of $E_{pk}(\mathbf{t}_i[j])$ for $1 \leq j \leq m$, denoted as $E_{pk}(S(\mathbf{t}_i))$, for each tuple $\mathbf{t}_i$. Then $\mathcal{C}_1$ and $\mathcal{C}_2$ uses SMIN protocol such that $\mathcal{C}_1$ obtains $E_{pk}(S(\mathbf{t}_{\min}))$.

Select the Skyline with Minimum Attribute Sum. The challenge now is we need to select the tuple $E_{pk}(\mathbf{t}_{\min})$ with the smallest $E_{pk}(S(\mathbf{t}_i))$ as a skyline tuple. In order to do this, a naive way is for $\mathcal{C}_1$ to compute $E_{pk}(S(\mathbf{t}_i) - S(\mathbf{t}_{\min}))$ for all tuples and then send them to $\mathcal{C}_2$. $\mathcal{C}_2$ can decrypt them and determine which one is equal to 0 and return the index to $\mathcal{C}_1$. $\mathcal{C}_1$ then adds the tuple $E_{pk}(\mathbf{p}_{\min})$ to skyline pool.

Eliminate Dominated Tuples. Once the skyline tuple is selected, $\mathcal{C}_1$ and $\mathcal{C}_2$ cooperatively use SDOM protocol to determine the dominance relationship between $E_{pk}(\mathbf{t}_{\min})$ and other tuples. In order to delete those tuples that are dominated by $E_{pk}(\mathbf{t}_{\min})$, a naive way is for $\mathcal{C}_1$ to send the encrypted dominance output to $\mathcal{C}_2$, who can decrypt it and send back the indexes of the tuples who are dominated to $\mathcal{C}_2$. $\mathcal{C}_1$ can delete those tuples dominated by $E_{pk}(\mathbf{t}_{\min})$ and the tuple $E_{pk}(\mathbf{t}_{\min})$ from $E_{pk}(T)$. The algorithm continues until there is no tuples left.

Return Skyline Results to Client. Once $\mathcal{C}_1$ has the encrypted skyline result, it can directly send them to the client if the client has the private key. However, in our setting, the client does not have the private key for better security. Lines 25 to 39 in Algorithm 4 illustrate how the client obliviously obtains the final skyline query result with the help of $\mathcal{C}_1$ and $\mathcal{C}_2$, at the same time, $\mathcal{C}_1$ and $\mathcal{C}_2$ know nothing about the final result. Consider the skyline tuples $E_{pk}(\mathbf{p}_1), ..., E_{pk}(\mathbf{p}_k)$ in skyline pool, where k is the number of skyline. The idea is for $\mathcal{C}_1$ to add a random noise $r_i[j]$ to each $\mathbf{p}_i[j]$ in ciphertext and then sends the encrypted randomized values $\alpha_i[j]$ to $\mathcal{C}_2$. $\mathcal{C}_1$ also sends the noise $r_i[j]$ to client. At the same time, $\mathcal{C}_2$ decrypts the randomized values $\alpha_i[j]$ and sends the result $r'_i[j]$ to client. Client

[1] We use $|\mathbf{p}_i[j] - \mathbf{q}[j]|$ in our running example for simplicity.

receives the random noise $r_i[j]$ from $\mathcal{C}_1$ and randomized values of the skyline points $\alpha_i[j]$ from $\mathcal{C}_2$, and removes the noise by computing $\mathbf{p}_i[j] = r'_i[j] - r_i[j]$ for $i = 1, ..., k$ and $j = 1, ..., m$ as the final result.

Algorithm 4. Basic Secure Skyline Protocol

```
Input: $\mathcal{C}_1$ has $E_{pk}(P), E_{pk}(T)$ and $\mathcal{C}_2$ has $sk$.
Output: client knows the skyline query result.
1  Compute minimum attribute sum;
2  $\mathcal{C}_1$:
3  if there is no tuple in $E_{pk}(T)$ then
4      break;
5  for $i = 1$ to $n$ do
6      $E_{pk}(S(\mathbf{t}_i)) = E_{pk}(\mathbf{t}_i[1]) \times ... \times E_{pk}(\mathbf{t}_i[m]) \bmod N^2$;
7  $\mathcal{C}_1$ and $\mathcal{C}_2$:
8  $E_{pk}(S(\mathbf{t}_{\min})) = SMIN(E_{pk}(S(\mathbf{t}_1)), ..., E_{pk}(S(\mathbf{t}_n)))$;
9  Select the skyline with minimum attribute sum;
10 $\mathcal{C}_1$:
11 for $i = 1$ to $n$ do
12     $\alpha_i = E_{pk}(S(\mathbf{t}_{\min}))^{N-1} \times E_{pk}(S(\mathbf{t}_i)) \bmod N^2$;
13     $\alpha'_i = \alpha_i^{r_i} \bmod N^2$, where $r_i \in \mathbb{Z}_N^+$;
14 send $\alpha'$ to $\mathcal{C}_2$;
15 $\mathcal{C}_2$:
16 decrypt $\alpha'$ and tell $\mathcal{C}_1$ which one equals to 0;
17 $\mathcal{C}_1$:
18 add the corresponding $E_{pk}(\mathbf{p}_{\min})$ to the skyline pool;
19 Eliminate dominated tuples;
20 $\mathcal{C}_1$ and $\mathcal{C}_2$:
21 use SDOM protocol to determine the dominance relationship between $E_{pk}(\mathbf{t}_{\min})$ and other tuples;
22 delete those tuples dominated by $E_{pk}(\mathbf{t}_{\min})$ and $E_{pk}(\mathbf{t}_{\min})$;
23 GOTO Line 1;
24 Return skyline results to client;
25 $\mathcal{C}_1$:
26 for $i = 1$ to $k$ do
27     for $j = 1$ to $m$ do
28         $\alpha_i[j] = E_{pk}(\mathbf{p}_i[j]) \times E_{pk}(r_i[j]) \bmod N^2$, where $r_i[j] \in \mathbb{Z}_N^+$;
29 send $\alpha_i[j]$ to $\mathcal{C}_2$ and $r_i[j]$ to client for all $i = 1, \ldots, k; j = 1, \ldots, m$;
30 $\mathcal{C}_2$:
31 for $i = 1$ to $k$ do
32     for $j = 1$ to $m$ do
33         $r_i[j]' = D_{sk}(\alpha_i[j])$;
34 send $r_i[j]'$ to client;
35 Client:
36 receive $r_i[j]$ from $\mathcal{C}_1$ and $r_i[j]'$ from $\mathcal{C}_2$;
37 for $i = 1$ to $k$ do
38     for $j = 1$ to $m$ do
39         $\mathbf{p}_i[j] = r_i[j]' - r_i[j]$;
```

6.2 Fully Secure Skyline Protocol

The basic protocol clearly reveals several information to $\mathcal{C}_1$ and $\mathcal{C}_2$ as follows.

- When selecting the skyline tuple with minimum attribute sum, $\mathcal{C}_1$ and $\mathcal{C}_2$ know which tuples are skyline points, which violates our result privacy requirement.
- When eliminating dominated tuples, $\mathcal{C}_1$ and $\mathcal{C}_2$ know the dominance relationship among tuples with respect to the query tuple $\mathbf{q}$, which violates our data pattern privacy requirement.

Algorithm 5. Fully Secure Skyline Protocol

```
Input: $\mathcal{C}_1$ has $E_{pk}(P), E_{pk}(T)$ and $\mathcal{C}_2$ has $sk$.
Output: $\mathcal{C}_1$ knows the encrypted skyline $E_{pk}(\mathbf{p}_{sky})$.
1  Order preserving perturbation;
2  $\mathcal{C}_1$:
3  for $i = 1$ to $n$ do
4      $E_{pk}(S(\mathbf{t}_i)) = E_{pk}(\mathbf{t}_i[1]) \times ... \times E_{pk}(\mathbf{t}_i[m]) \bmod N^2$;
5  $\mathcal{C}_1$ and $\mathcal{C}_2$:
6  for $i = 1$ to $n$ do
7      $[\![E_{pk}(S(\mathbf{t}_i))]\!] = SBD(E_{pk}(S(\mathbf{t}_i)))$;
8  $\mathcal{C}_1$:
9  for $i = 1$ to $n$ do
10     $[\![E_{pk}(S(\mathbf{t}_i))]\!] = \langle E_{pk}((S(\mathbf{t}_i))_B^{(1)}), ..., E_{pk}((S(\mathbf{t}_i))_B^{(l)}),$
       $E_{pk}((S(\mathbf{t}_i))_B^{(l+1)}), ..., E_{pk}((S(\mathbf{t}_i))_B^{(l+\lceil \log n \rceil)}) \rangle$, where
       $(S(\mathbf{t}_i))_B^{(l+1)}, ..., (S(\mathbf{t}_i))_B^{(l+\lceil \log n \rceil)}$ is the binary representation
       of an exclusive vale of $[0, n-1]$;
11     $E_{pk}(S(\mathbf{t}_i)) = \prod_{\gamma=1}^{l} E_{pk}((S(\mathbf{t}_i))_B^{(\gamma)})^{2^{l-\gamma}} \bmod N^2$;
12 $\mathcal{C}_1$ and $\mathcal{C}_2$:
13 $E_{pk}(S(\mathbf{t}_{\min})) = SMIN(E_{pk}(S(\mathbf{t}_1)), ..., E_{pk}(S(\mathbf{t}_n)))$;
14 $\mathcal{C}_1$:
15 $\lambda = (E_{pk}(S(\mathbf{t}_{\min})) \times E_{pk}(MAX)^{-1})^r \bmod N^2$, where $r_i \in \mathbb{Z}_N^+$;
16 send $\lambda$ to $\mathcal{C}_2$;
17 $\mathcal{C}_2$:
18 if $D_{sk}(\lambda) = 0$ then
19     break;
20 Select skyline with minimum attribute sum;
21 $(E_{pk}(\mathbf{p}_{sky}), E_{pk}(\mathbf{t}_{sky}))$ =FindOneSkyline
   $(E_{pk}(P), E_{pk}(T), E_{pk}(S(\mathbf{t}_i)), E_{pk}(S(\mathbf{t}_{\min})))$ (Algorithm 6);
22 Eliminate dominated tuples;
23 $\mathcal{C}_1$ and $\mathcal{C}_2$:
24 for $i = 1$ to $n$ do
25     for $\gamma = 1$ to $l$ do
26         $E_{pk}((S(\mathbf{t}_i))_B^{(\gamma)}) = SOR(V_i, E_{pk}((S(\mathbf{t}_i))_B^{(\gamma)}))$;
27 $\mathcal{C}_1$:
28 for $i = 1$ to $n$ do
29     $E_{pk}(S(\mathbf{t}_i)) = \prod_{\gamma=1}^{l} E_{pk}((S(\mathbf{t}_i))_B^{(\gamma)})^{2^{l-\gamma}} \bmod N^2$;
30 $\mathcal{C}_1$ and $\mathcal{C}_2$:
31 for $i = 1$ to $n$ do
32     $V_i = SDOM(E_{pk}(\mathbf{t}_{sky}), E_{pk}(\mathbf{t}_i))$;
33 Lines 23-32;
34 GOTO Line 1;
```

To address these leakage, we propose a fully secure protocol in Algorithm 5. The step to compute minimum attribute sum and return the results to the client are the same as the basic protocol. We focus on the following steps that are designed to address the disclosures of the basic protocol.

Select Skyline with Minimum Attribute Sum. Once $\mathcal{C}_1$ obtains the encrypted minimum attribute sum $E_{pk}(S(\mathbf{t}_{\min}))$, the challenge is how to select the tuple $E_{pk}(\mathbf{t}_{\min})$ with the minimum sum $E_{pk}(S(\mathbf{t}_{\min}))$ as a skyline tuple such that $\mathcal{C}_1$ and $\mathcal{C}_2$ know nothing about which tuple is selected. We present a protocol as shown in Algorithm 6.

We first need to determine which $S(\mathbf{t}_i)$ is equal to $S(\mathbf{t}_{\min})$. Note that this can not be achieved by the SMIN protocol which only selects the minimum value. Here we propose an efficient way, exploiting the fact that it is okay for $\mathcal{C}_2$ to know there is one equal case (since we are selecting one skyline tuple) as long as it does not know which one. $\mathcal{C}_1$ first computes $\alpha'_i = E_{pk}((S(\mathbf{t}_i) - S(\mathbf{t}_{\min})) \times r_i)$, and then sends a permuted list $\beta = \pi(\alpha')$ to $\mathcal{C}_2$ based on a random permutation sequence

π. The permutation hides which sum is equal to the minimum from $\mathcal{C}_2$ while the uniformly random noise r_i masks the difference between each sum and the minimum sum. Note that α'_i is uniformly random in $\mathbb{Z}_N^+$ except when $S(\mathbf{t}_i) - S(\mathbf{t}_{\min}) = 0$, in which case $\alpha'_i = 0$. $\mathcal{C}_1$ decrypts β_i, if it is 0, it means tuple i has smallest $E_{pk}(S(\mathbf{t}_i))$. Therefore, $\mathcal{C}_2$ sends $E_{pk}(1)$ to $\mathcal{C}_1$, otherwise, sends $E_{pk}(0)$.

Algorithm 6. Find One Skyline

```
Input: $\mathcal{C}_1$ has encrypted dataset $E_{pk}(P)$, $E_{pk}(T)$, $E_{pk}(S(\mathbf{t}_i))$,
       and $E_{pk}(S(\mathbf{t}_{\min}))$, $\mathcal{C}_2$ has private key $sk$.
Output: $\mathcal{C}_1$ knows one encrypted skyline $E_{pk}(\mathbf{p}_{sky})$ and
        $E_{pk}(\mathbf{t}_{sky})$.
$\mathcal{C}_1$:
for $i = 1$ to $n$ do
    $\alpha_i = E_{pk}(S(\mathbf{t}_{\min}))^{N-1} \times E_{pk}(S(\mathbf{t}_i)) \bmod N^2$;
    $\alpha'_i = \alpha_i^{r_i} \bmod N^2$, where $r_i \in \mathbb{Z}_N^+$;
send $\beta = \pi(\alpha')$ to $\mathcal{C}_2$;
$\mathcal{C}_2$:
receive $\beta$ from $\mathcal{C}_1$;
for $i = 1$ to $n$ do
    $\beta'_i = D_{sk}(\beta_i)$;
    if $\beta'_i = 0$ then
        $U_i = E_{pk}(1)$;
    else
        $U_i = E_{pk}(0)$;
send $U$ to $\mathcal{C}_1$;
$\mathcal{C}_1$:
receive $U$ from $\mathcal{C}_2$;
$V = \pi^{-1}(U)$;
for $i = 1$ to $n$ do
    for $j = 1$ to $m$ do
        $E_{pk}(\mathbf{t}_i[j]') = SM(V_i, E_{pk}(\mathbf{t}_i[j]))$;
        $E_{pk}(\mathbf{p}_i[j]') = SM(V_i, E_{pk}(\mathbf{p}_i[j]))$;
for $j = 1$ to $m$ do
    $E_{pk}(\mathbf{t}[j]') = \prod_{i=1}^{n} E_{pk}(\mathbf{t}_i[j]') \bmod N^2$;
    $E_{pk}(\mathbf{p}[j]') = \prod_{i=1}^{n} E_{pk}(\mathbf{p}_i[j]') \bmod N^2$;
add $E_{pk}(\mathbf{p}_{sky}) = \langle E_{pk}(\mathbf{p}[1]'), ..., E_{pk}(\mathbf{p}[m]')\rangle$ to skyline pool;
use $E_{pk}(\mathbf{t}_{sky}) = \langle E_{pk}(\mathbf{t}[1]'), ..., E_{pk}(\mathbf{t}[m]')\rangle$ to compare with
    other $E_{pk}(\mathbf{t}_i)$;
```

After receiving the encrypted permuted bit vector U as the equality result, $\mathcal{C}_1$ applies a reverse permutation, and obtains an encrypted bit vector V, where one tuple has bit 1 suggesting it has the minimum sum. In order to obtain the attribute values of this tuple, $\mathcal{C}_1$ and $\mathcal{C}_2$ employ SM protocol to compute encrypted product of the bit vector and the attribute values, $E_{pk}(\mathbf{t}_i[j]')$ and $E_{pk}(\mathbf{p}_i[j]')$. Since all other tuples except the one with the minimum sum will be 0, we can sum all $E_{pk}(\mathbf{t}_i[j]')$ and $E_{pk}(\mathbf{p}_i[j]')$ on each attribute and $\mathcal{C}_1$ can obtain the attribute values corresponding to the skyline tuple.

Order Preserving Perturbation. We can show that Algorithm 6 is secure and correctly selects the skyline tuple if there is only one minimum. A potential issue is that multiple tuples may have the same minimum sum. If this happens, not only is this information revealed to $\mathcal{C}_2$, but also the skyline tuple cannot be selected (computed) correctly, since the bit vector contains more than one 1 bit. To address this, we employ order-preserving perturbation which adds a set of mutually different *bit sequence* to *a set of values* such that: 1) if the original values are equal to each other, the perturbed values are guaranteed not equal to each other, and 2) if the original values are not equal to each other, their order is preserved. The perturbed values are then used as the input for Algorithm 6.

Concretely, given n numbers in their binary representations, we add a $\lceil log\ n \rceil$-bit sequence to the end of each $E_{pk}(S(\mathbf{t}_i))$, each represents a unique bit sequence in the range of $[0, n-1]$. This way, the perturbed values are guaranteed to be different from each other while their order is preserved since the added bits are the least significant bits. Line 10 of Algorithm 5 shows this step. We note that we can multiply each sum $E_{pk}(S(\mathbf{t}_i))$ by n and uniquely add a value from $[0, n-1]$ to each $E_{pk}(S(\mathbf{t}_i))$, hence guarantee they are not equal to each other. This will be more efficient than adding a bit sequence, however, since we will need to perform the bit decomposition later in the protocol to allow bit operators, we run decomposition by the SBD protocol for *l* bits in the beginning of the protocol rather than $l + \lceil \log n \rceil$ bits later.

'Eliminate Dominated Tuples. Once the skyline tuple is selected, it can be added to the skyline pool and then used to eliminate dominated tuples. In order to do this, $\mathcal{C}_1$ and $\mathcal{C}_2$ cooperatively use SDOM protocol to determine the dominance relationship between $E_{pk}(\mathbf{t}_{\min})$ and other tuples. The challenge is then how to eliminate the dominated tuples without $\mathcal{C}_1$ and $\mathcal{C}_2$ knowing which tuples are being dominated and eliminated. Our idea is that instead of eliminating the dominated tuples, we "flag" them by securely setting their attribute values to the maximum domain value. This way, they will not be selected as skyline tuples in the remaining iterations. Concretely, we can set the binary representation of their attribute sum to all 1s so that it represents the domain maximum. Since we added $\lceil \log n \rceil$ bits to $[\![E_{pk}(S(\mathbf{t}_i))]\!]$, the new $[\![E_{pk}(S(\mathbf{t}_i))]\!]$ has $l + \lceil \log n \rceil$ bits. Therefore, the maximum value $MAX = 2^{l+\lceil \log n \rceil} - 1$. To obliviously set the attributes of only dominated tuples to MAX, based on the encrypted dominance output V_i of the dominance protocol, $\mathcal{C}_1$ and $\mathcal{C}_2$ cooperatively employ SOR of the dominance boolean output and the bits of the $S(\mathbf{t}_i)$. This way, if the tuple is dominated, it will be set to MAX. Otherwise, it will remain the same. If $E_{pk}(S(\mathbf{t}_{\min})) = E_{pk}(MAX)$, it means all the tuples are processed, i.e., flagged either as a skyline or a dominated tuple, the protocol ends.

Example 3. We illustrate the entire protocol through the running example shown in Table 3. Please note that all column values are in encrypted form except columns π and β'. Given the mapped data points $\mathbf{t}_i$, $\mathcal{C}_1$ first computes the attribute sum $E_{pk}(S(\mathbf{t}_i))$ shown in the third column. We set $l = 5$, $\mathcal{C}_1$ gets the binary representation of the attribute sum $[\![E_{pk}(S(\mathbf{t}_i))]\!]$. Because $n = 4$, $\mathcal{C}_1$ obliviously adds the order-preserving perturbation $\lceil \log 4 \rceil = 2$ bits to the end of $[\![E_{pk}(S(\mathbf{t}_i))]\!]$ respectively to get the new $E_{pk}(S(\mathbf{t}_i))$ (shown in the sixth column). Then $\mathcal{C}_1$ gets $E_{pk}(S(\mathbf{t}_{\min})) = E_{pk}(30)$ by employing SMIN.

The protocol then turns to the subroutine Algorithm 6 to select the first skyline based on the minimum attribute sum. $\mathcal{C}_1$ computes $\alpha_i = E_{pk}(S(\mathbf{t}_i) - S(\mathbf{t}_{\min}))$. Assume the random noise vector $r = \langle 3, 9, 31, 2\rangle$ and the permutation sequence $\pi = \langle 2, 1, 4, 3\rangle$, $\mathcal{C}_1$ sends the encrypted permuted and randomized difference vector β to $\mathcal{C}_2$. After decrypting β, $\mathcal{C}_2$ gets β' and then sends U to $\mathcal{C}_1$. $\mathcal{C}_1$ computes V by applying a reverse permutation. By employing SM with V, $\mathcal{C}_1$ computes $(E_{pk}(\mathbf{t}_i[1]'), E_{pk}(\mathbf{t}_i[2]'))$ and $(E_{pk}(\mathbf{p}_i[1]'), E_{pk}(\mathbf{p}_i[2]'))$.

TABLE 3
Example of Algorithm 5

	$\mathcal{C}_1$:								$\mathcal{C}_2$:		$\mathcal{C}_1$:					
$\mathbf{t}_i$	$(\mathbf{t}_i[1], \mathbf{t}_i[2])$	$S(\mathbf{t}_i)$	$\llbracket S(\mathbf{t}_i)\rrbracket$	pert.	$S(\mathbf{t}_i)$	$S(\mathbf{t}_i) - S(\mathbf{t}_{\min})$	r	π	β'	U	V	$(\mathbf{t}_i[1]', \mathbf{t}_i[2]')$	$(\mathbf{p}_i[1]', \mathbf{p}_i[2]')$	$S(\mathbf{t}_i)$	V	$S(\mathbf{t}_i)$
$\mathbf{t}_1$	$(1, 15)$	16	1,0,0,0,0	1,1	67	$67 - 30$	3	2	0	1	0	$(0, 0)$	$(0, 0)$	67	0	67
$\mathbf{t}_2$	$(2, 5)$	7	0,0,1,1,1	1,0	30	$30 - 30$	9	1	111	0	1	$(2, 5)$	$(39, 120)$	127	0	127
$\mathbf{t}_3$	$(4, 5)$	9	0,1,0,0,1	0,1	37	$37 - 30$	31	4	92	0	0	$(0, 0)$	$(0, 0)$	37	1	127
$\mathbf{t}_4$	$(4, 15)$	19	1,0,0,1,1	0,0	76	$76 - 30$	2	3	217	0	0	$(0, 0)$	$(0, 0)$	76	1	127

After summing all column values, $\mathcal{C}_1$ adds $E_{pk}(\mathbf{p}_{sky}) = (E_{pk}(39), E_{pk}(120))$ to skyline pool and uses $E_{pk}(\mathbf{t}_{sky}) = (E_{pk}(2), E_{pk}(5))$ to eliminate dominated tuples.

The protocol now turns back to the main routine in Algorithm 5 to eliminate dominated tuples. $\mathcal{C}_1$ and $\mathcal{C}_2$ use SOR with V to make $E_{pk}(S(\mathbf{t}_{\min})) = E_{pk}(127)$ and $E_{pk}(S(\mathbf{t}_i)) = E_{pk}(S(\mathbf{t}_i))$ for $i \neq min$. Now, only $E_{pk}(S(\mathbf{t}_{\min})) = E_{pk}(S(\mathbf{t}_2))$ has changed to $E_{pk}(127)$ which is "flagged" as MAX. We emphasize that $\mathcal{C}_1$ does not know this value has changed because the ciphertext of all tuples has changed. Next, $\mathcal{C}_1$ and $\mathcal{C}_2$ find the dominance relationship between $E_{pk}(\mathbf{t}_{sky})$ and $E_{pk}(\mathbf{t}_i)$ by SDOM protocol. $\mathcal{C}_1$ obtains the dominance vector V. Using same method, $\mathcal{C}_1$ flags $E_{pk}(S(\mathbf{t}_3))$ and $E_{pk}(S(\mathbf{t}_4))$ to $E_{pk}(127)$. The protocol continues until all are set to MAX.

Security Analysis. Based on Theorem 1, the protocol is secure if the subprotocols are secure and the intermediate results are random or pseudo-random. We focus on the intermediate result here. From $\mathcal{C}_1$'s view, the intermediate result includes U. Because U is ciphertext and $\mathcal{C}_1$ does not have the secret key, $\mathcal{C}_1$ can simulate U based on its input and output. From $\mathcal{C}_2$'s view, the intermediate result includes β. β contains one $E_{pk}(0)$ and $m-1$ ciphertext of any positive value. After the permutation π of $\mathcal{C}_1$, $\mathcal{C}_2$ cannot determine where is the $E_{pk}(0)$. Therefore, $\mathcal{C}_2$ can simulate β based on its input and output. Hence the protocol is secure.

Computational Complexity Analysis. The subroutine Algorithm 6 requires $O(n)$ decryptions in Line 9, $O(nm)$ encryptions and decryptions in Lines 20 and 21. Thus, Algorithm 6 requires $O(nm)$ encryptions and decryptions in all. In Algorithm 5, Line 7 requires $O(nl)$ encryptions and decryptions. Line 10 requires $O(n\lceil\log n\rceil)$ encryptions. Line 12 requires $O((l + \lceil\log n\rceil)n)$ encryptions and decryptions. Line 26 requires $O(l + \lceil\log n\rceil)$ encryptions and decryptions. Line 32 requires $O(nm)$ encryptions and decryptions. Thus, this part requires $O((l + \lceil\log n\rceil)n + nm)$ encryptions and decryptions. Because this part runs k times, the fully secure skyline protocol requires $O(k(l + \lceil\log n\rceil)n + knm)$ encryptions and decryptions in total.

7 Performance Analysis and Optimizations

In this section, we illustrate two optimizations to further reduce the computation load. We first show a data partitioning optimization in Section 7.1, and then show a lazy merging optimization in Section 7.2.

7.1 Optimization of Data Partitioning

As shown in the previous section, the overall run time complexity depends on the number of points (n), the number of skyline points (k), the number of decomposed bits (l) which is determined by the domain of the attribute values, and the number of dimensions (m). A straightforward way to enhance the performance is to partition the input dataset into subdatasets and then we can use a divide-and-conquer approach to avoid unnecessary computations. Furthermore, the partitioning also allows effective parallelism.

Algorithm 7. Parallel Implementation via Data Partitioning

```
Input: A dataset $P$ of $n$ points in $m$ dimensions.
Output: Skyline of $P$.
1  divide $n$ points into $s$ partitions and compute the skyline
   points in each partition;
2  set the state of all partitions as uncomputed;
3  $n_p \leftarrow$ number of uncomputed partitions;
4  $n_t \leftarrow$ number of threads;
5  $n_{it} \leftarrow$ number of idle threads;
6  $n_{um} \leftarrow$ number of computed and unmerged results;
7  while $n_p > 0$ or $n_{it} > 0$ do
8    if $n_p > 0$ and $n_{it} > 0$ then
9      assign one uncomputed partition to each idle thread;
10   else
11     if $n_p == 0$ and $n_{it} == n_t - 1$ and $n_{um} == 0$ then
12       break;
13     wait until at least one thread finishes;
14     set the state of computed partition as unmerged;
15     if $n_{um} > 1$ then
16       merge each two into one new partition;
17       set new partition state as uncomputed;
```

The basic idea of data partitioning is to divide the dataset into a set of initial partitions, compute the skyline in each partition, and then continuously merge the skyline result of the partitions into new partitions and compute their skyline, until all partitions are merged into the final result. This can be implemented with either a single thread (sequentially) or multiple threads (in parallel). We describe our data partitioning scheme in Algorithm 7. Given an input dataset, the number of partitions s is specified as one parameter. We will show how to calculate the optimal number of partitions in Subsection 7.1.1. We first divide the input data into s partitions and compute the skyline in each partition in Line 1, and then set the state of all partitions as uncomputed in Line 2. In Line 7, the algorithm continues with uncomputed partitions or idle threads. In Line 8, if there are some uncomputed partitions and there are some idle threads, we assign one uncomputed partition to each idle thread in Line 9. In Line 11, if there is no uncomputed partition ($n_p == 0$), all computed partitions are merged ($n_{um} == 0$), and there is only one working thread ($n_{it} == n_t - 1$), that means all partitions are computed and merged, the algorithm finishes. Otherwise, we wait until at least one thread finishes and set the state of computed partition which now only contains skylines in that partition as unmerged in Lines 13-14. In Line 15, if there are some computed and unmerged partitions, we merge each two into one new partition and set the state as uncomputed in Lines 16-17.

7.1.1 Discovery of Optimal Number of Partitions

In this subsection, we show how to calculate the optimal number of partitions for minimizing the total computation load given an independent and identically distributed random dataset. We first show the theorem of the expected number of skyline points as follows.

Theorem 1 (Number of Skyline Points) [4]. *Given an independent and identically distributed random dataset of n points in m dimensional space, the expected number of skyline points is* $O(\ln^{m-1} n)$.

In the computational complexity analysis of fully secure skyline protocol, the time complexity is $O(kn(l+m+\lceil \log n \rceil))$. According to Theorem 1, the expected output size of input data with size $\frac{n}{s}$ in m dimensional space is $\ln^{m-1}(\frac{n}{s})$. Therefore, in this step, the computation load required for each partition is $\ln^{m-1}(\frac{n}{s}) \times \frac{n}{s} \times (log(\frac{n}{s}) + m + l)$. Since we have s partitions, the total computation load required is $s \times ln^{m-1}(\frac{n}{s}) \times \frac{n}{s} \times (\log(\frac{n}{s}) + m + l) = n \times \ln^{m-1}(\frac{n}{s}) \times (\log(\frac{n}{s}) + m + l)$. This is the initial layer of the computation, which we refer to $layer_0$. We use 0 because the following layers have a slightly different formula.

Before we proceed, we denote the number of layers excluding $layer_0$ as n_{layer}. For each layer i, we denote the number of partitions that needs to be computed as $n_{p,i}$, the size of a single input partition as $size_{in,i}$, the output size of a single partition as $size_{out,i}$, and the amount of computation load as W_{layer_i}. A visual graph about the layer structure is shown in Fig. 4. In the ideal case, we have $s = 2^h$ partitions, where h is an integer. For each layer, we reduce the number of partitions by merging two partitions to form a new partition which contains skyline points of those two merged partitions. After h layers' merging, we obtain only one partition which is the final skyline result.

Number of Partitions and Layers. To simplify the analysis, we assume the merging of two partitions happens at the same layer (although mergings from different layers may happen at the same time). As shown in Fig. 4, the datasets for $layer_i$ $(i > 1)$ comes from the merging of two computed partitions from $layer_{i-1}$. Therefore, in $layer_i$, the number of partitions $(n_{p,i})$ is $\frac{s}{2^i}$ given the number of partitions in $layer_1$ is $\frac{s}{2}$. Meanwhile, $layer_0$ has s partitions, $layer_1$ has $\frac{s}{2}$ partitions, and the last layer has one partition, so the number of layers excluding $layer_0$ (n_{layer}) is $\log s$.

Output Size. A partition in $layer_i$ is merged from 2^i partitions in $layer_0$. Therefore, the expected output size of one partition at $layer_i$ corresponds to the expected output size of 2^i partitions in $layer_0$. That is, in $layer_i$, the expected output size of a single partition $(size_{out,i})$ is $ln^{m-1}(\frac{2^i n}{s})$.

Input Size. In $layer_i$, the size of each input partition $(size_{in,i})$ is twice of the single partition output size from the last layer because it is the merging of two outputs from the last layer. In other words, $size_{in,i} = 2 \times size_{out,i-1} = 2 \times ln^{m-1}(\frac{2^{i-1}n}{s})$. For example, the expected single partition output size of $layer_0$ is $\ln^{m-1}(\frac{n}{s})$ and the expected size of each input partition in $layer_1$ is $2 \times \ln^{m-1}(\frac{n}{s})$.

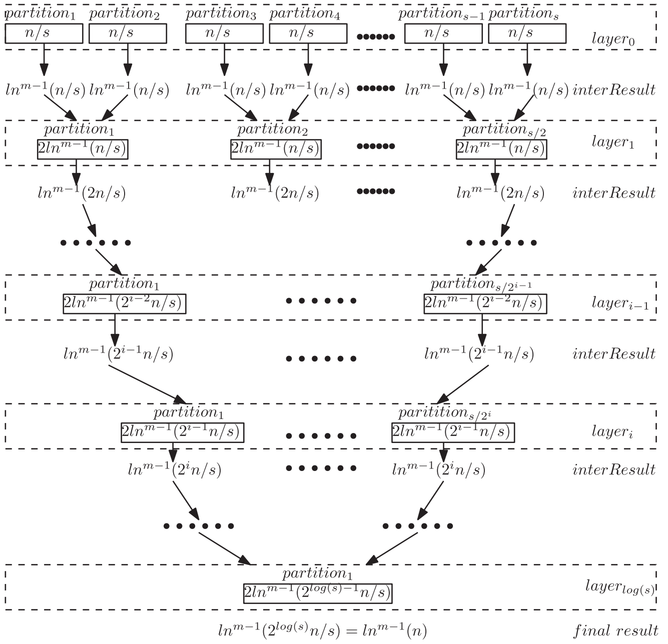


Fig. 4. Layer structure (interResult is short for intermediate result).

Computation Load. With $n_{p,i}$, $size_{in,i}$, and $size_{out,i}$, we can obtain the general formula for computation load of $layer_i$ $(i \neq 0)$ as $W_{layer_i} = n_{p,i} \times size_{out,i} \times size_{in,i} \times (m + \log(size_{in,i}))$ according to the time complexity of our fully secure skyline protocol. And since we have n_{layer} layers, the overall computation load is calculated as follows.

$$\begin{aligned} W_{all} &= W_{layer_0} + \sum_{1}^{n_{layer}} W_{layer_i} \\ &= W_{layer_0} + \sum_{1}^{n_{layer}} n_{p,i} \times size_{out,i} \times size_{in,i} \times (m + \log(size_{in,i})) \\ &= n \times \ln^{m-1}\left(\frac{n}{s}\right) \times \left(\log\frac{n}{s} + m + l\right) + \sum_{i=1}^{\log s} \frac{s}{2^i} \times \ln^{m-1}\left(\frac{2^i n}{s}\right) \\ &\quad \times 2ln^{m-1}\left(\frac{2^{i-1}n}{s}\right) \times \left(\log\left(2ln^{m-1}\left(\frac{2^{i-1}n}{s}\right)\right) + m + l\right) \end{aligned}$$

Optimal Number of Partitions. Without loss of generality, from now on, we assume $n = 2^u$ and $s = 2^v$, where $u, v \in \mathbb{Z}^+$ and $1 \leq v < u$. To find out the optimal number of partitions, our goal is to minimize W_{all} against s or v. Because $n = 2^u$ and $s = 2^v$, we have the computation load $W(v)$ corresponding to the number of partition $s = 2^v$ as follows.

$$\begin{aligned} W(v) =& 2^u \times (u-v)^{m-1} \times \ln^{m-1} 2 \times (u - v + m + l) \\ &+ \sum_{i=1}^{v} 2^{v-i+1} \times (i + u - v)^{m-1} \times (i - 1 + u - v)^{m-1} \\ &\times \ln^{2m-2} 2 \times (\log(2 \times (i - 1 + u - v)^{m-1} \ln^{m-1} 2) + m + l) \end{aligned}$$

We denote the part after $\sum$ as $WI_{v,i}$. Notice that $WI_{v,i} = WI_{v+1,i+1}$, we have

$$\begin{aligned} W(v+1) - W(v) &= W_{layer_0,v+1} - W_{layer_0,v} + \sum_{i=1}^{v+1} WI_{v+1,i} - \sum_{i=1}^{v} WI_{v,i} \\ &= W_{layer_0,v+1} - W_{layer_0,v} + WI_{v+1,1} \end{aligned}$$

Notice that the minimal value of W lies at the position where $W(v+1) - W(v)$ changes from negative to positive. Observe that in our setting, all variables can only be positive integer, which means we need to find out the integer v such that $f(v) = W(v+1) - W(v)$ changes from negative to positive. By letting $x = u - v$, we have

$$\begin{aligned} f(x) &= WI_{v+1,1} + W_{layer_0,v+1} - W_{layer_0,v} \\ &= 2^{v+1} \times (x)^{m-1} \times (x-1)^{m-1} \times \ln^{2m-2} 2 \\ &\quad \times (\log(2 \times (x-1)^{m-1} \ln^{m-1} 2) + m + l) \\ &\quad + 2^u \times (x-1)^{m-1} \times \ln^{m-1} 2 \times (x - 1 + m + l) \\ &\quad - 2^u \times x^{m-1} \times \ln^{m-1} 2 \times (x + m + l) \\ &= 2^u \ln^{m-1} 2 \times (2^{1-x} \times x^{m-1} \times (x-1)^{m-1} \times \ln^{m-1} 2 \\ &\quad \times (\log(2 \times (x-1)^{m-1} \ln^{m-1} 2) + m + l) \\ &\quad + ((x-1)^{m-1} \times (x - 1 + m + l) - x^{m-1} \times (x + m + l))) \end{aligned}$$

To obtain the minimal value of $f(x)$, we can ignore the preceding $2^u \ln^{m-1} 2$ which is always positive. Then we can easily solve the problem to find out x where $f(x)$ changes from positive to negative given m and l.

For example, we set $l = 20$ in our experiments, if $m = 2$, then the minimal value of $W(v)$ is obtained at $x = 1$, i.e., $u - v = 1$. This actually corresponds to the case where each initial partition has two data points. If $m = 3$, we have $x = 6$, i.e., $u - v = 6$. That is, for three dimensional datasets, the optimal number of partitions is 2^{u-6} and each partition has 2^6 points.

7.2 Optimization of Lazy Merging

In this subsection, we show another optimization with lazy merging.

Lazy Merging. In the hierarchical divide-and-conquer approach proposed in the last subsection, results from any two computed partitions are merged immediately as a new partition for computing skyline points. However, immediate merging might not be optimal in the later stage of the program because it requires 1) more merging overhead and 2) more unnecessary computations. In the later stage of the program, there are only a few points in each partition. At this time, merging overhead is high compared to the computation time. Therefore, we can employ lazy merging which incurs less merging overhead. Furthermore, in the later stage of the program, those remaining points are likely to follow an anti-correlated distribution as they are skyline points of a partition at a previous layer. For anti-correlated dataset, data partitioning will incur more unnecessary computations. Consider an extreme example, if all the remaining points are the final skyline points, all the computations in each partition are unnecessary. Therefore, we can employ lazy merging to avoid those unnecessary computations and delay the merging operation to a later time when more computed results are ready.

Merging Timing. With lazy merging, we can reduce running time if and only if the timing for lazy merging is perfect. Merging too early (immediate merging) or merging too late does not provide enough benefit or even jeopardizes the performance. As shown in the last subsection, for a given dataset, we can calculate the optimal number of partitions, which is related to the dataset size. For example, given $l = 20$ and $m = 3$, we have the number of optimal partitions as $\frac{n}{2^6}$, which effectively states that the optimal size of each partition should be $2^6 = 64$ in the initial layer. Therefore, in our algorithm, we heuristically wait until the size of merged partitions reach 64 before sending it for computation in the previous example. That is, there are at least 64 points in each partition (excluding the final partition which contains the final skyline points) to compute the skyline points.

Security Analysis. The cloud servers can tell if the subsets are skew or uniformly distributed in the extreme case when the distribution of entire dataset is different from the distribution of subsets based on the different number of returned skyline points from each partition. However, the probability is very low because we randomly partition the dataset, and the distribution of subsets should be very similar to the distribution of entire dataset. Moreover, this attack can be easily fixed by returning all the tuples in each iteration. That is, cloud servers $\mathcal{C}_1$ and $\mathcal{C}_2$ return all skyline tuples with true values and non-skyline tuples with MAX values. In this way, the cloud servers cannot know the skyline distribution of subsets, thus, the cloud servers cannot get any new information from the partitions.

8 Experiments

In this section, we describe our experimental setup and optimized parallel system design. For comparison purposes, we have implemented both protocols: the Basic Secure Skyline Protocol (*BSSP*) in Section 6.1, and the Fully Secure Skyline Protocol (*FSSP*) in Section 6.2. Since there is no existing solution for secure skyline computation, we use the basic approach as a baseline which is efficient but leaks some indirect data patterns to the cloud server. We have also designed a parallel framework for effective reducing computation time together with the two optimizations, data partitioning and lazy merging.

8.1 Experiment Setup

We implemented all algorithms in C with all multithreading using POSIX threads and all communication using sockets. We ran single-machine-experiments on a machine with Intel Core i7-6700K 4.0 GHz running Ubuntu 16.04. The distributed version was tested on a cluster of 64 machines with Intel Core i7-2600 3.40 GHz running CentOS 6, which we will provide more details in the next section. In our experiment setup, both $\mathcal{C}_1$ and $\mathcal{C}_2$ were running on the same machine. The reported computation time is the total computation time of the $\mathcal{C}_1$ and $\mathcal{C}_2$.

Datasets. We used both synthetic datasets and a real NBA dataset in our experiments. To study the scalability of our methods, we generated independent (INDE), correlated (CORR), and anti-correlated (ANTI) datasets following the seminal work [5]. We also built a dataset that contains 2384 NBA players who are league leaders of playoffs.[2] Each player has five attributes that measure the player's performance: Points (PTS), Rebounds (REB), Assists (AST), Steals (STL), and Blocks (BLK).

Data Partitioning. This procedure can be done either using single thread or multiple threads. We conducted single thread experiment for verifying the optimal number of partitions. And we refer to multithreading implementation as local parallelism. The algorithm is shown in Algorithm 7.

[2] The data was extracted from http://stats.nba.com/leaders /alltime/?ls=iref:nba:gnav on 04/15/2015

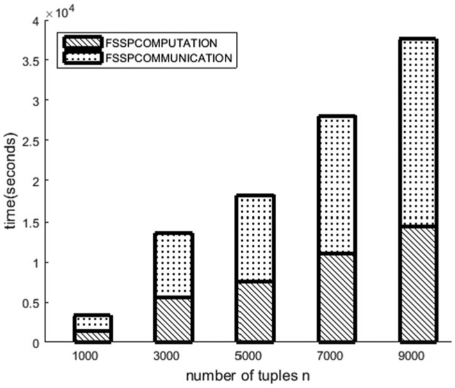


Fig. 5. Computation and communication time cost of different $n(m = 2, K = 512)$.

To further demonstrate the scalability of our algorithm, we also implemented a distributed version, which employs a manager-worker model. The manager distributes partitions to workers, the workers compute the skyline points in any given dataset and return the results to the manager, which works similarly as the local parallelism. The only difference is that the manager could implement sophisticated load balancing algorithm to fully utilize the computation resources. The overall data partitioning scheme is very similar to the existing MapReduce approach. However, we didn't employ existing MapReduce framework because existing crypto library in Java does not satisfy our requirements.

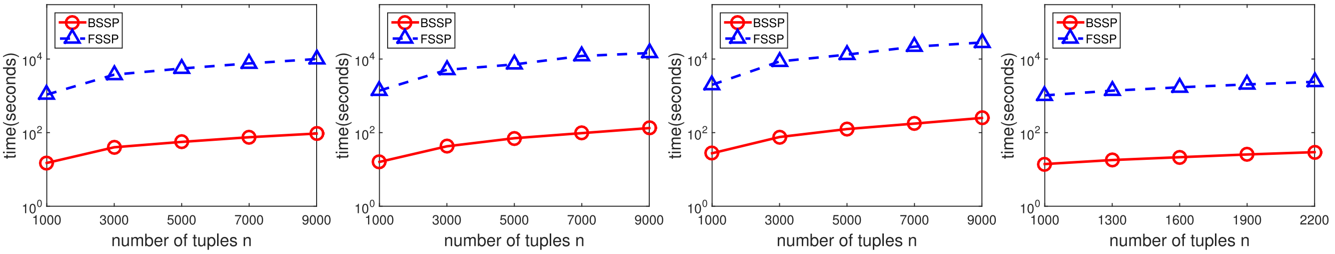
Lazy Merging. The lazy merging delays the merging operation until there are enough results to form a partition with optimal size, which is detailed shown in Section 7.1.1. All experiments using optimizations are conducted using 10 different independent and identically distributed random datasets of size 512 and dimension 3 with three repeated runs for each dataset.

8.2 Impact of Parameters

In this subsection, we evaluate our protocols by varying the number of tuples (n), the number of dimensions (m), and the key size (K) on datasets of various distributions.

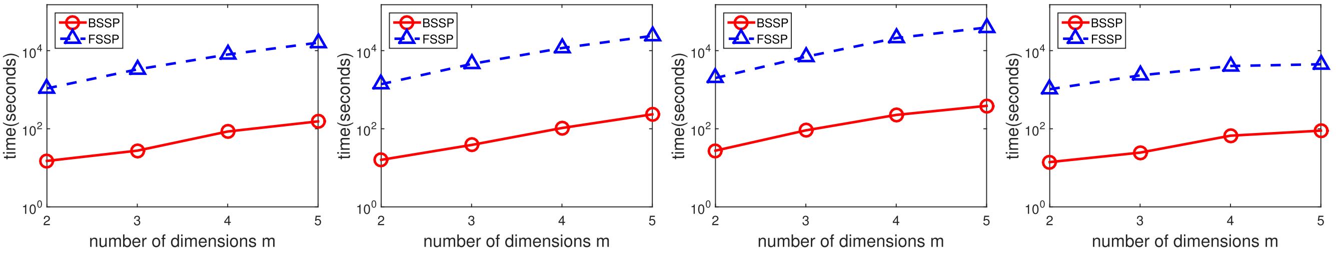
Impact of Number of Tuples n. Fig. 6a, 6b, 6c, 6d show the time cost of different n on CORR, INDE, ANTI, and NBA datasets, respectively. We observe that for all datasets, the time cost increases approximately linearly with the number of tuples n, which is consistent with our complexity analysis. While BSSP is very efficient, FSSP does incur more computational overhead for full security. Comparing different datasets, the time cost is in slightly increasing order for CORR, INDE, and ANTI, due to the increasing number of skyline points of the datasets. The time for NBA dataset is low due to its small number of tuples.

Impact of Number of Dimensions m. Fig. 7a, 7b, 7c, 7d show the time cost of different m on CORR, INDE, ANTI, and NBA



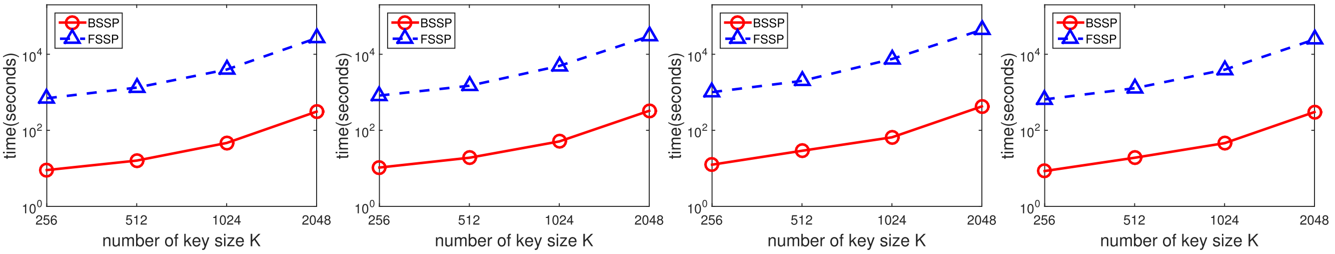
(a) time cost of CORR (b) time cost of INDE (c) time cost of ANTI (d) time cost of NBA

Fig. 6. The impact of $n(m = 2, K = 512)$.



(a) time cost of CORR (b) time cost of INDE (c) time cost of ANTI (d) time cost of NBA

Fig. 7. The impact of $m(n = 1000, K = 512)$.



(a) time cost of CORR (b) time cost of INDE (c) time cost of ANTI (d) time cost of NBA

Fig. 8. The impact of $K(n = 1000, m = 2)$.

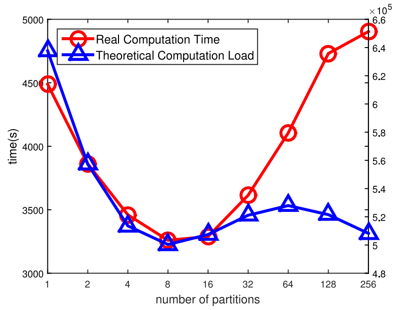


Fig. 9. Theoretical and experimental results.

datasets, respectively. For all datasets, the time cost increases approximately linearly with the number of dimensions m. FSSP also shows more computational overhead than BSSP. The different datasets show a similar comparison as in Fig. 6. The time for NBA dataset is lower than the CORR dataset which suggests that the NBA data is strongly correlated.

Impact of Encryption Key Size K. Fig. 8a, 8b, 8c, 8d show the time cost with different key size used in the Paillier cryptosystem on CORR, INDE, ANTI, and NBA datasets, respectively. A stronger security indeed comes at the price of computation overhead, i.e., the time cost increases significantly, almost exponential, when K grows.

Communication Overhead. We also measured the overall time which includes computation time reported earlier and the communication time between the two server processes. Fig. 5 shows the computation and communication time of different n on INDE dataset of FSSP. We observe that computation time only takes about one third of the total time in this setting.

8.3 Effect of Optimizations

In this subsection, we evaluate the efficiency of our proposed two optimizations, data partitioning and lazy merging.

Data Partitioning. Fig. 9 shows the relationship between theoretical computation load and real computation time. The theoretical computation load has an optimal value at the partition $2^{9-6} = 8$, which indicates dividing the original dataset into 8 partitions will give the smallest amount of computation load. Using ten datasets and three repeated runs for each dataset, we obtained the real computation time, which perfectly matches the theoretical computation load at the region with small number of partitions. With large number of partitions, the experimental results deviate from theoretical derivations. The reason for the deviation is that when the number of points in each partition is too small for large number of partitions, the number of skyline points in each partition violates our assumption of data distribution. For example, it is hard to say a dataset with only five points is an independent and identically distributed random dataset. Therefore, computation time for each partition does not follow our derivation. Furthermore, the large number of partitions will incur more merging overhead.

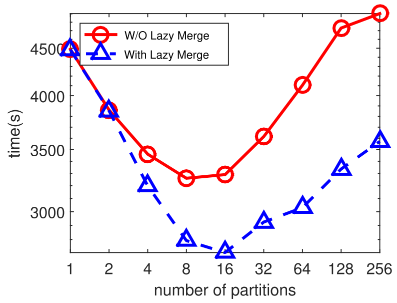
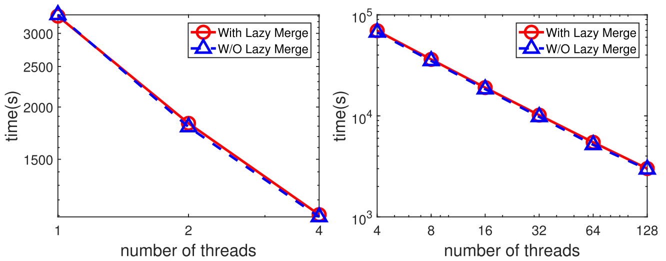


Fig. 10. Computation time with and without lazy merging.

Lazy Merging. As yet another optimization, lazy merging plays an important role especially when the number of partitions is large. In Fig. 10, we show the computation time with and without lazy merging, respectively. It can be seen that overall with lazy merging, the run time can be effectively reduced. The larger number of partitions, the larger number of time difference, which is reasonable because the larger number of partitions, the larger number of merging operations and more rounds of computation. We can also see that for one partition (no partition) and two partitions, there is no time reduction, the reasons are that there is no merging operation need for one partition and there is no lazy merging operation for two partitions.

To summarize, both data partitioning and lazy merging have been proven effective and can significantly reduce the computation time even using single thread.



(a) Local parallelism.

(b) Global parallelism.

Fig. 11. Local parallelism and global parallelism.

8.4 Effect of Parallelism

In this subsection, we demonstrate the speedup of our protocol by using multithreading (local parallelism) on independent and identically distributed random datasets with 512 points and distributed computing with 64 commercial desktops (global parallelism) on independent and identically distributed random datasets with 65536 points.

As shown in Fig. 11, if we use one machine with up to 4 threads, the protocol almost shows a linear speedup. As the number of threads doubles, the computation time reduces to half. However, as we further increase the number of threads, we only see sub-linear speedup. We believe this is due to the small size of the dataset. In distributed computation experiments, we employed 4, 8, 16, 32, 64, and 128 threads, respectively. It is clear that at the beginning the protocol shows a linear speedup. While the number of threads reaches 64, it switches to sub-linear speedup again due to the small size of dataset. In both local and global parallelism, we observe that the difference between with lazy merging and without lazy merging is too small to be observed. In other words, when we have enough computation power, lazy merging provides limited improvement, which is opposite to what we see in single-thread experiment.

9 Conclusions

In this paper, we proposed a fully secure skyline protocol on encrypted data using two non-colluding cloud servers under the semi-honest model. It ensures semantic security in that the cloud servers knows nothing about the data including indirect data patterns, query, as well as the query result. In addition, the client and data owner do not need to participate in the computation. We also presented a secure dominance protocol which can be used by skyline queries as well as other queries. Furthermore, we demonstrated two optimizations, data partitioning and lazy merging, to further reduce the computation load. Finally, we presented our implementation of the protocol and demonstrated the feasibility and efficiency of the solution. As for future work, we plan to optimize the communication time complexity to further improve the performance of the protocol.

ACKNOWLEDGMENTS

This research is supported in part by the Patient-Centered Outcomes Research Institute (PCORI) under award ME-1310-07058, the National Institute of Health (NIH) under award R01GM114612, and an NSERC Discovery grant.

REFERENCES

[1] F. Baldimtsi and O. Ohrimenko, "Sorting and searching behind the curtain," in *Proc. Int. Conf. Financial Cryptography Data Secur.*, 2015, pp. 127–146.

[2] A. Beimel, "Secret-sharing schemes: A survey," in *Proc. Int. Conf. Coding Cryptology*, 2011, pp. 11–46.

[3] J. L. Bentley, "Multidimensional divide-and-conquer," *Commun. ACM*, vol. 23, no. 4, pp. 214–229, 1980.

[4] J. L. Bentley, H. T. Kung, M. Schkolnick, and C. D. Thompson, "On the average number of maxima in a set of vectors and applications," *J. ACM*, vol. 25, no. 4, pp. 536–543, 1978.

[5] S. Börzsönyi, D. Kossmann, and K. Stocker, "The skyline operator," in *Proc. 17th Int. Conf. Data Eng.*, , 2001, pp. 421–430.

[6] S. Bothe, A. Cuzzocrea, P. Karras, and A. Vlachou, "Skyline query processing over encrypted data: An attribute-order-preserving-free approach," in *Proc. 1st Int. Workshop Privacy Secuirty Big Data*, 2014, pp. 37–43.

[7] S. Bothe, P. Karras, and A. Vlachou, "eskyline: Processing skyline queries over encrypted data," *Proc. VLDB Endowment*, vol. 6, no. 12, pp. 1338–1341, 2013.

[8] C. Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang, "Finding k-dominant skylines in high dimensional space," in *Proc. ACM SIGMOD Conf.*, 2006, pp. 503–514.

[9] 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.

[10] V. Costan and S. Devadas, "Intel sgx explained," Technical report, Cryptology ePrint Archive, Report 2016/086, 2016. [Online]. Available: http://eprint.iacr.org

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

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

[13] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft, "Privacy-preserving face recognition," in *Proc. Int. Symp. Privacy Enhancing Technol. Symp.*, 2009, pp. 235–253.

[14] U. Feige, A. Fiat, and A. Shamir, "Zero-knowledge proofs of identity," *J. Cryptology*, vol. 1, no. 2, pp. 77–94, 1988.

[15] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annu. ACM Symp. Theory Comput.*, 2009, pp. 169–178.

[16] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game or A completeness theorem for protocols with honest majority," in *Proc. ACM Symp. Theory Comput.*, 1989, pp. 218–229.

[17] H. Hacigümüs, B. R. Iyer, C. Li, and S. Mehrotra, "Executing SQL over encrypted data in the database-service-provider model," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2002, pp. 216–227.

[18] S. Halevi and V. Shoup, "Bootstrapping for helib," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Tech.*, 2015, pp. 641–670.

[19] T. Hashem, L. Kulik, and R. Zhang, "Privacy preserving group nearest neighbor queries," in *Proc. 13th Int. Conf. Extending Database Technol.*, 2010, 489–500.

[20] H. Hu, J. Xu, C. Ren, and B. Choi, "Processing private queries over untrusted data cloud through privacy homomorphism," in *Proc. IEEE 27th Int. Conf. Data Eng.*, 2011, 601–612.

[21] Y. Huang, D. Evans, J. Katz, and L. Malka, "Faster secure two-party computation using garbled circuits," in *Proc. 20th USENIX Conf. Secur.*, 2011, pp. 35–35.

[22] A. Janosi, W. Steinbrunn, M. Pfisterer, and R. Detrano, "Heart disease dataset, https://archive.ics.uci.edu/ml/datasets/heart+disease," in The UCI Archive, 1998.

[23] D. G. Kirkpatrick and R. Seidel, "Output-size sensitive algorithms for finding maximal vectors," in *Proc. Symp. Comput. Geometry*, 1985, pp. 89–96.

[24] D. Kossmann, F. Ramsak, and S. Rost, "Shooting stars in the sky: An online algorithm for skyline queries," in *Proc. 28th Int. Conf. Very Large Data Bases*, 2002, pp. 275–286.

[25] H. T. Kung, F. Luccio, and F. P. Preparata, "On finding the maxima of a set of vectors," *J. ACM*, vol. 22, no. 4, pp. 469–476, Oct. 1975.

[26] C. Li, N. Zhang, N. Hassan, S. Rajasekaran, and G. Das, "On skyline groups," in *Proc. Conf. Inf. Knowl. Manage.*, 2012, pp. 2119–2123.

[27] A. Liu, K. Zheng, L. Li, G. Liu, L. Zhao, and X. Zhou, "Efficient secure similarity computation on encrypted trajectory data," in *Proc. IEEE 31st Int. Conf. Data Eng.*, 2015, pp. 66–77.

[28] J. Liu, L. Xiong, J. Pei, J. Luo, and H. Zhang, "Finding pareto optimal groups: Group-based skyline," *Proc. VLDB Endowment*, vol. 8, no. 13, pp. 2086–2097, 2015.

[29] J. Liu, L. Xiong, and X. Xu, "Faster output-sensitive skyline computation algorithm," *Inf. Process. Lett.*, vol. 114, pp. 710–713, 2014.

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

[31] J. Liu, J. Yang, L. Xiong, J. Pei, and J. Luo, "Skyline diagram: Finding the voronoi counterpart for skyline queries," in *Proc. Int. Conf. Data Eng.*, 2018.

[32] J. Liu, H. Zhang, L. Xiong, H. Li, and J. Luo, "Finding probabilistic k-skyline sets on uncertain data," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, 2015, pp. 1511–1520.

[33] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Adv. Cryptology*, 1999, pp. 223–238.

[34] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "Progressive skyline computation in database systems," *ACM Trans. Database Syst.*, vol. 30, no. 1, pp. 41–82, 2005.

[35] S. Papadopoulos, S. Bakiras, and D. Papadias, "Nearest neighbor search with strong location privacy," *Proc. VLDB Endowment*, vol. 3, pp. 619–629, 2010.

[36] J. Pei, B. Jiang, X. Lin, and Y. Yuan, "Probabilistic skylines on uncertain data," in *Proc. 33rd Int. Conf. Very Large Data Bases*, 2007, pp. 15–26.

[37] Y. Qi and M. J. Atallah, "Efficient privacy-preserving k-nearest neighbor search," in *Proc. 28th Int. Conf. Distrib. Comput. Syst.*, 2008, pp. 311–319.

[38] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy*, 2000, pp. 44–55.

[39] T. Veugen, F. Blom, S. J. A. de Hoogh, and Z. Erkin, "Secure comparison protocols in the semi-honest model," *J. Sel. Topics Signal Process.*, vol. 9, no. 7, pp. 1217–1228, 2015.

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

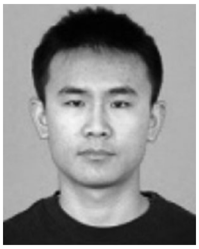
[41] A. C. Yao, "Protocols for secure computations (extended abstract)," in *Proc. 23rd Annu. Symp. Found. Comput. Sci.*, 1982, pp. 160–164.

[42] B. Yao, F. Li, and X. Xiao, "Secure nearest neighbor revisited," in *Proc. IEEE Int. Conf. Data Eng.*, 2013, pp. 733–744.

[43] X. Yi, R. Paulet, E. Bertino, and V. Varadharajan, "Practical k nearest neighbor queries with location privacy," in *Proc. IEEE 30th Int. Conf. Data Eng.*, 2014, 640–651.

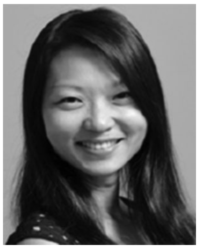
[44] W. Yu, Z. Qin, J. Liu, L. Xiong, X. Chen, and H. Zhang, "Fast algorithms for pareto optimal group-based skyline," in *Proc. ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 417–426.

[45] H. Zhu, X. Meng, and G. Kollios, "Privacy preserving similarity evaluation of time series data," in *Proc. Int. Conf. Extending Database Technol.*, 2014, pp. 499–510.



Jinfei Liu is a joint postdoctoral research fellow with Emory University and the Georgia Institute of Technology. His research interests include skyline queries, data privacy and security, and machine learning. He has published more than 20 papers in premier journals and conferences including VLDB, ICDE, CIKM, and IPL. He is a member of the IEEE.

Juncheng Yang is a working toward the master's degree at Emory University. His research interests include computer security, database, smart cache in storage, and distributed system. He has published more than 10 papers in premier conferences including ICDE and SoCC. He is a member of the IEEE.



Li Xiong is a professor of computer science and biomedical informatics with Emory University. She conducts research that addresses both fundamental and applied questions at the interface of data privacy and security, spatiotemporal data management, and health informatics. She has published more than 100 papers in premier journals and conferences including the *IEEE Transactions on Knowledge and Data Engineering*, the *Journal of the American Medical Informatics Association*, VLDB, ICDE, CCS, and WWW. She currently serves as associate editor for the *IEEE Transactions on Knowledge and Data Engineering (TKDE)* and on numerous program committees for data management and data security conferences. She is a member of the IEEE.

Jian Pei is currently a Canada research chair (Tier 1) in big data science and a professor with the School of Computing Science, Simon Fraser University, Canada. He is one of the most cited authors in data mining, database systems, and information retrieval. Since 2000, he has published one textbook, two monographs, and more than 200 research papers in refereed journals and conferences, which have been cited by more than 77,000 in literature. He was the editor-in-chief of the *IEEE Transactions of Knowledge and Data Engineering (TKDE)* in 2013-2016, and is currently a director of the Special Interest Group on Knowledge Discovery in Data (SIGKDD) of the Association for Computing Machinery (ACM). He is a fellow of the ACM and of the IEEE.