Published in final edited form as:
Proc Int Conf Data Eng. 2017 April ; 2017: 633–644. doi:10.1109/ICDE.2017.117.

Secure Skyline Queries on Cloud Platform

Jinfei Liu[*], **Juncheng Yang**[*], **Li Xiong**[*], and **Jian Pei**[†]

[*]Department of Mathematics & Computer Science, Emory University

[†]School of Computing Science, Simon Fraser University

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. 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.

I. 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 needs 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., [14], [31]). To protect the confidentiality of the query from cloud server, authorized clients also send encrypted queries to the cloud server. Figure 1 illustrates our problem scenario of secure query processing over encrypted data in the cloud. The data owner outsources their 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 [14] ensure strong security while enabling arbitrary computations on the encrypted data, however, the computation cost is prohibitive in practice. Trusted hardware such as the latest Intel's Software Guard Extensions (SGX) brings a promising alternative, however still has limitations in its security guarantees [9]. Many techniques (e.g., [17], [37]) 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 (*k*NN) queries, which return *k* most similar (closest) records given a query record, have been extensively studied [11], [20], [39], [41].

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 *k*NN 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 learning repository [22] as shown in Table I(a) and Figure 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 *k*NN 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. 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 I(b) and also in Figure 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. Our protocol can be also used for skyline computation.

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.

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 skyline query protocol in which the cloud should not gain any knowledge about the data including indirect data patterns.

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 [10] and k-skyband queries [32].
- 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. 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 II presents the related work. Section III 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 IV. The key subroutine of secure skyline protocols, secure dominance protocol, is shown in Section V. The complete secure skyline protocols are presented in Section VI. We report the experimental results and findings in Section VII. Section VIII concludes the paper.

II. 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 $O(nlogk)$ 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. [4], 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 [7], uncertain skyline [34] [30], group-based skyline [28], [26]).

Secure query processing on encrypted data

Fully homomorphic encryption schemes [14] enable arbitrary computations on encrypted data. Even though it is shown that [14] 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], [37]) 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. [5] and their demo version [6] illustrated an approach about skyline queries on so-called “encrypted” data without any formal security guarantee. Another work [8] studied the verification of skyline query result returned by an untrusted cloud server.

The closely related work is secure *k*NN queries [11], [19], [20], [33], [35], [39], [41], [42] which we discuss in more detail here. Wong et al. [39] 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. [41]. Yao et al. [41] proposed a new method based on secure Voronoi diagram. Instead of asking the cloud server to retrieve the exact *k*NN result, their method retrieve a relevant encrypted partition E(R) for E(Q) such that R is guaranteed to contain the *k*NN of Q. 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. [42] proposed solutions for secure *k*NN queries based on oblivious transfer paradigm. Recently, Elmehdwi et al. [11] proposed a secure *k*NN 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.

Other works studied *k*NN queries in the secure multi-party computation (SMC) setting [35] (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 [33] (query is private while data is public), which are different from our settings.

Secure Multi-party Computation (SMC)

SMC was first proposed by Yao [40] 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 $\mathscr{C}_1$ with encrypted input and $\mathscr{C}_2$ with the private key *sk*. The goal is for $\mathscr{C}_1$ to obtain an encrypted result of a function on the input without disclosing the original input to either $\mathscr{C}_1$ or $\mathscr{C}_2$.

III. 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 II.

A. Skyline Definitions

Definition 1: (Skyline)—Given a dataset $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ 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, $\mathbf{p}_a[j] \le \mathbf{p}_b[j]$, and for at least one j, $\mathbf{p}_a[j] < \mathbf{p}_b[j]$, where $\mathbf{p}_i[j]$ is the j^{th} dimension of $\mathbf{p}_i$ and $1 \le j \le m$. The skyline points are those points that are not dominated by any other point in P.

Definition 2: (Dynamic Skyline Query) [10]—Given a dataset $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ 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, $|\mathbf{p}_a[j] - \mathbf{q}[j]| \le |\mathbf{p}_b[j] - \mathbf{q}[j]|$ 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 j^{th} dimension of $\mathbf{p}_i$ and $1 \le j \le m$. 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).

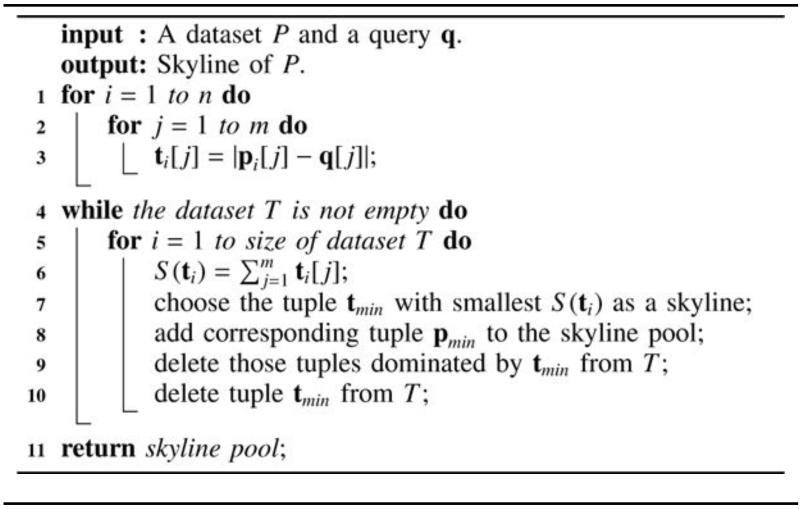
Example 1—Consider Table I and Figure 2 as a running example. Given data points $\mathbf{p}_1$ to $\mathbf{p}_4$ and query point $\mathbf{q}$, the mapped data points are computed as $\mathbf{t}_i[j] = |\mathbf{p}_i[j] - \mathbf{q}[j]| + \mathbf{q}[j]$. We see that $\mathbf{t}_1$, $\mathbf{t}_2$ are the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

B. 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”.

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;
```

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.

Example 2—Given the mapped data points $\mathbf{t}_1, \ldots, \mathbf{t}_4$, 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.

C. 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 (we employ the Pailliar cryptosystem [31] as explained later in the section). We use pk and sk to denote the public key and private key, respectively. Data owner sends $E_{pk}(\mathbf{p}_i[j])$ for $i = 1, \ldots, n$ and $j = 1, \ldots, m$ to cloud server $\mathscr{C}_1$.

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

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, $\mathscr{C}_2$, 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 $\mathscr{C}_1$ and $\mathscr{C}_2$. An overview of the protocol setting is shown in Figure 3.

D. 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 [13], these protocols can be transformed into secure protocols under the malicious model. Second, the semi-honest model is realistic in current cloud market. $\mathscr{C}_1$ and $\mathscr{C}_2$ 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 [11], [27], [43], we also adopt the semi-honest adversary model for this paper.

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 $\mathscr{C}_1$ and $\mathscr{C}_2$ 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 $\mathscr{C}_1$ and $\mathscr{C}_2$ 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 $\mathscr{C}_1$ and $\mathscr{C}_2$ know nothing about the query tuple **q**.
- **Result Privacy**. Cloud servers $\mathscr{C}_1$ and $\mathscr{C}_2$ know nothing about the query result, e.g., which tuples are in the skyline result.

Security Definition in the Semi-honest Model—Considering the privacy properties above, we adopt the formal security definition from the multi-party computation setting under the semi-honest model [15]. Intuitively, a protocol is secure if whatever can be computed by a party participating in the protocol can be computed based on its input and output only. This is formalized according to the simulation paradigm. Loosely speaking, we require that a party's view in a protocol execution to be simulative given only its input and output. This then implies that the parties learn nothing from the protocol execution. We omit the definition due to the limited space, for the detailed and strict definition, please see [15].

Theorem 1

Composition Theorem [15]: If a protocol consists of subprotocols, the protocol is secure as long as the subprotocols are secure and all the intermediate results are random or pseudo-random.

In this work, the proposed secure skyline protocols are constructed based on a sequential composition of subprotocols. To formally prove the security under the semi-honest model, according to the composition theorem given in Theorem 1, one needs to show that the simulated view of each subprotocol was computationally indistinguishable from the actual execution view and the protocol produces random or pseudo-random shares as intermediate results.

E. Paillier Cryptosystem

We use the Paillier cryptosystem [31] as the encryption scheme in this paper and briefly describe Paillier's additive homomorphic properties which will be used in our protocols.

- Homomorphic addition of plaintexts:

$$D_{sk}(E_{pk}(a) \times E_{pk}(b) mod\ N^2) = (a+b) mod\ N$$

- Homomorphic multiplication of plaintexts:

$$D_{sk}(E_{pk}(a)^{b}\ mod\ N^{2}) = a \times b\ mod\ N$$

IV. 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, $\mathscr{C}_1$ with encrypted input and $\mathscr{C}_2$ with the private key *sk* as shown in Figure 3. The goal is for $\mathscr{C}_1$ to obtain an encrypted result of a function on the input without disclosing the original input to either $\mathscr{C}_1$ or $\mathscr{C}_2$. 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.

Secure Multiplication (SM)

Assume a cloud server $\mathscr{C}_1$ with encrypted input $E_{pk}(a)$ and $E_{pk}(b)$, and a cloud server $\mathscr{C}_2$ with the private key *sk*, where *a*, *b* are two numbers not known to $\mathscr{C}_1$ and $\mathscr{C}_2$. The Secure Multiplication (SM) protocol [11] (based on the additively homomorphic property of Paillier) securely computes encrypted result of multiplication of *a*, *b*, $E_{pk}(a \times b)$, such that only $\mathscr{C}_1$ knows $E_{pk}(a \times b)$ such that only $\mathscr{C}_1$ knows $E_{pk}(a \times b)$, and no information related to *a*, *b* is revealed to $\mathscr{C}_1$ or $\mathscr{C}_2$.

Secure Bit Decomposition (SBD)

Assume a cloud server $\mathscr{C}_1$ with encrypted input $E_{pk}(a)$ and a cloud server $\mathscr{C}_2$ with the private key *sk*, where *a* is a number not known to $\mathscr{C}_1$ and $\mathscr{C}_2$. The Secure Bit Decomposition (SBD) protocol [36] securely computes encrypted individual bits of the binary representation of *a*, denoted as $[\![a]\!] = \langle E_{pk}((a))_B^{(1)}, \ldots, E_{pk}((a)_B^{(l)}))\rangle$, where *l* is the number of bits, $(a)_B^{(1)}$ and $(a)_B^{(l)}$ denote the most and least significant bits of *a*, respectively. At the end of the protocol, the output $[\![a]\!]$ is known only to $\mathscr{C}_1$ and no information related to *a* is revealed to $\mathscr{C}_1$ or $\mathscr{C}_2$.

A. Secure Boolean Operations

Secure OR (SOR)—Assume a cloud sever $\mathscr{C}_1$ with encrypted input $E_{pk}(\hat{a})$ and $E_{pk}(\hat{b})$, and a cloud server $\mathscr{C}_2$ with the private key *sk*, where $\hat{a}$ and $\hat{b}$ are two bits not known to $\mathscr{C}_1$ and $\mathscr{C}_2$. The Secure OR (SOR) protocol [11] securely computes encrypted result of the bit-wise OR of the two bits, $E_{pk}(\hat{a} \vee \hat{b})$, such that only $\mathscr{C}_1$ knows $E_{pk}(\hat{a} \vee \hat{b})$ and no information related to $\hat{a}$ and $\hat{b}$ is revealed to $\mathscr{C}_1$ or $\mathscr{C}_2$.

Secure AND (SAND)—Assume a cloud server $\mathscr{C}_1$ with encrypted input $E_{pk}(\hat{a})$ and $E_{pk}(\hat{b})$, and a cloud server $\mathscr{C}_2$ with the private key *sk*, where $\hat{a}$ and $\hat{b}$ are two bits not known to $\mathscr{C}_1$ and $\mathscr{C}_2$. The goal of the SAND protocol is to securely compute encrypted result of the bit-wise

AND of the two bits, $E_{pk}(\hat{a} \wedge \hat{b})$, such that only $\mathscr{C}_1$ knows $E_{pk}(\hat{a} \wedge \hat{b})$ and no information related to $\hat{a}$ and $\hat{b}$ is revealed to $\mathscr{C}_1$ or $\mathscr{C}_2$. We can simply use the secure multiplication (SM) protocol on the two bits.

Secure NOT (SNOT)—Assume a cloud server $\mathscr{C}_1$ with encrypted input $E_{pk}(\hat{a})$ and a cloud server $\mathscr{C}_2$ with the private key *sk*, where $\hat{a}$ is a bit not known to $\mathscr{C}_1$, $\mathscr{C}_2$. The goal of the SNOT protocol is to securely compute the encrypted complement bit of $\hat{a}$, $E_{pk}(\neg\hat{a})$, such that only $\mathscr{C}_1$ knows $E_{pk}(\neg\hat{a})$ and no information related to $\hat{a}$ is revealed to $\mathscr{C}_1$ or $\mathscr{C}_2$. Secure NOT protocol can be easily implemented by $E_{pk}(1 - \hat{a}) = E_{pk}(1)E_{pk}(\hat{a})^{N-1}$.

B. Secure Minimum and Secure Comparison

Secure minimum protocol and secure comparison protocol have been extensively studied in cryptography community [1], [12], [38] and database community [11], [43]. 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 $E_{pk}(Bool(a \leq b))$ known by cloud server $\mathscr{C}_1$ (it will be $E_{pk}(1)$ when $a \leq b$ and $E_{pk}(0)$ when $a > b$), $\mathscr{C}_1$ 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 [11], [43] from database community for selecting a minimum have a security weakness, i.e., $\mathscr{C}_2$ can determine whether the two numbers are equal to each other. We point out the security weakness as follows.

Disclosure of Binary based SMIN—Given two numbers in binary representations, the idea of the Binary representation based SMIN protocol (BSMIN)[1] [11] is for $\mathscr{C}_1$ to randomly choose a boolean functionality F (by flipping a coin), where F is either $a > b$ or $b > a$, and then securely compute F with $\mathscr{C}_2$, such that the output of F is oblivious to both $\mathscr{C}_1$ and $\mathscr{C}_2$. Based on the output and chosen F, $\mathscr{C}_1$ computes $min(a, b)$ locally using homomorphic properties. More specifically, given the binary representation of the two numbers, for each bit, $\mathscr{C}_1$ computes an encrypted boolean output W_i of the two bits based on F (e.g., if F is $a > b$, $W_i = E_{pk}(1)$, if $(a)_B^{(i)} > (b)_B^{(i)}$ and $E_{pk}(0)$ otherwise) and an encrypted randomized difference between $(a)_B^{(i)}$ and $(b)_B^{(i)}$. This way, the order and difference of the two numbers are not disclosed to $\mathscr{C}_2$. However, when $a = b$, whatever F is, we have $W_i = E_{pk}(0)$ for all bits. We can show that through the intermediate result (the encrypted randomized difference between $(a)_B^{(i)}$ and $(b)_B^{(i)}$, $\Gamma_i = E_{pk}(r_i)$ for $1 \leq i \leq l$, the bit-wise XOR of $(a)_B^{(i)}$ and $(b)_B^{(i)}$, $G_i = E_{pk}(0)$ for $1 \leq i \leq l$), $\mathscr{C}_2$ can determine a equals to b.

Disclosure of Perturbation based SMIN—The Perturbation based SMIN protocol (PSMIN) [43] assumes $\mathscr{C}_1$ has $E_{pk}(a)$ and $E_{pk}(b)$. $\mathscr{C}_1$ generates a set of v random values uniformly from a certain range $\{r_1, \ldots, r_v | r_1 < r_i, i \geq 2\}$. $\mathscr{C}_1$ then sends a set of $2+v-1$ encrypted values $\{E_{pk}(a+r_1), E_{pk}(b+r_1), E_{pk}(x_2+r_2), \ldots, E_{pk}(x_v+r_v)\}$ to $\mathscr{C}_2$, where x_i, $i \geq 2$

[1] The SMIN protocol for *n* values can be constructed by employing BSMIN for two values at a time in a hierarchical fashion as suggested in [11] or simply a linear fashion.

are randomly chosen from a, b. The idea is that the smallest number, after being perturbed by r_1 (which is smaller than r_i, $i \geq 2$), will remain the smallest. The perturbation hides the order of the numbers to $\mathscr{C}_2$. Although not mentioned by the original paper, we point out $\mathscr{C}_1$ also needs to shuffle the encrypted values before sending them to $\mathscr{C}_2$, otherwise the differences between the values will be disclosed to $\mathscr{C}_2$ after decryption. After decrypting those $2+v-1$ values, $\mathscr{C}_2$ takes the minimal min and sends $E_{pk}(min)$ to $\mathscr{C}_1$. $\mathscr{C}_1$ computes $E_{pk}(min - r_1)$ as result. The security weakness of PSMIN is due to the fact that if two numbers are equal, their perturbed values remain equal. Since $\mathscr{C}_1$ sends $\{E_{pk}(a + r_1), E_{pk}(b + r_1), E_{pk}(x_2 + r_2), \ldots, E_{pk}(x_v + r_v)\}$ to $\mathscr{C}_2$, $\mathscr{C}_2$ can learn two numbers are equal based on $a + r_1$ and $b + r_1$.

Therefore, we adapted the secure minimum/comparison protocols [38] 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 $\mathscr{C}_1$ with encrypted input $E_{pk}(a)$ and $E_{pk}(b)$, and a cloud server $\mathscr{C}_2$ with the private key sk, where a and b are two numbers not known to $\mathscr{C}_1$ and $\mathscr{C}_2$. The goal of the SLEQ protocol is to securely compute the encrypted boolean output $E_{pk}(Bool(a \leq b))$, such that only $\mathscr{C}_1$ knows $E_{pk}(Bool(a \leq b))$ and no information related to a and b is revealed to $\mathscr{C}_1$ or $\mathscr{C}_2$.

Secure Equal (SEQ)—Assume a cloud server 1 with encrypted input $E_{pk}(a)$ and $E_{pk}(b)$, and a cloud server $\mathscr{C}_2$ with the private key sk, where a and b are two numbers not known to $\mathscr{C}_1$ and $\mathscr{C}_2$. The goal of the SEQ protocol is to securely compute the encrypted boolean output $E_{pk}(Bool(a == b))$, such that only $\mathscr{C}_1$ knows $E_{pk}(Bool(a == b))$ and no information related to $Bool(a == b)$ is revealed to $\mathscr{C}_1$ or $\mathscr{C}_2$.

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

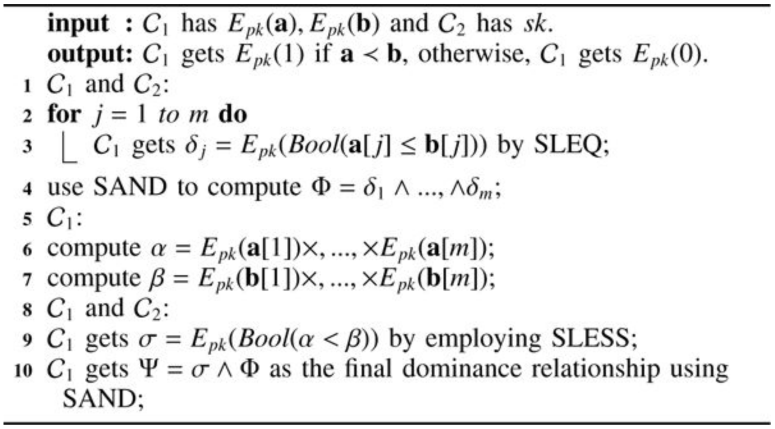
Secure Minimum (SMIN)—Assume a cloud server $\mathscr{C}_1$ with encrypted input $E_{pk}(a)$ and $E_{pk}(b)$, and a cloud server $\mathscr{C}_2$ with the private key sk, where a and b are two numbers not known to both parties. The goal of the SMIN protocol is to securely compute encrypted minimum value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

V. 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 $\mathbf{a} = (\mathbf{a}[1], \ldots, \mathbf{a}[m])$, $\mathbf{b} = (\mathbf{b}[1], \ldots, \mathbf{b}[m])$ and a cloud server $\mathcal{C}_2$ with the private key sk, where **a** and **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, $E_{pk}(0)$.

Algorithm 2

Secure Dominance Protocol.



```
input : $C_1$ has $E_{pk}(\mathbf{a}), E_{pk}(\mathbf{b})$ and $C_2$ has $sk$.
output: $C_1$ gets $E_{pk}(1)$ if $\mathbf{a} \prec \mathbf{b}$, otherwise, $C_1$ gets $E_{pk}(0)$.
$C_1$ and $C_2$:
for $j = 1$ to $m$ do
    $C_1$ gets $\delta_j = E_{pk}(Bool(\mathbf{a}[j] \le \mathbf{b}[j]))$ by SLEQ;
use SAND to compute $\Phi = \delta_1 \wedge \ldots, \wedge \delta_m$;
$C_1$:
compute $\alpha = E_{pk}(\mathbf{a}[1]) \times, \ldots, \times E_{pk}(\mathbf{a}[m])$;
compute $\beta = E_{pk}(\mathbf{b}[1]) \times, \ldots, \times E_{pk}(\mathbf{b}[m])$;
$C_1$ and $C_2$:
$C_1$ gets $\sigma = E_{pk}(Bool(\alpha < \beta))$ by employing SLESS;
$C_1$ gets $\Psi = \sigma \wedge \Phi$ as the final dominance relationship using
  SAND;
```

Protocol Design

Given any two tuples $\mathbf{a} = (\mathbf{a}[1], \ldots, \mathbf{a}[m])$ and $\mathbf{b} = (\mathbf{b}[1], \ldots, \mathbf{b}[m])$, 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 $\Phi = \delta_1 \wedge, \ldots, \wedge \delta_m$. 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 $\mathbf{a} \preceq \mathbf{b}$.

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 **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 $\alpha = E_{pk}(\mathbf{a}[1] + \ldots + \mathbf{a}[m])$ and $\beta = E_{pk}(\mathbf{b}[1] + \ldots + \mathbf{b}[m])$ 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 $\mathscr{C}_1$ 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 1), 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.

VI. Secure Skyline Protocol

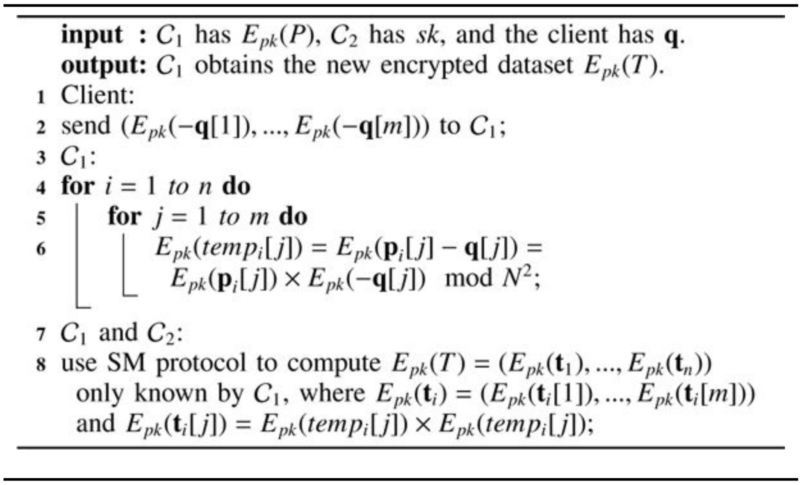
In this section, we first propose a basic secure skyline protocol and show why such a simple solution is not secure. Then we propose a fully secure skyline protocol. Both protocols are constructed by using the security primitives discussed in Section IV and the secure dominance protocol in Section V.

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[2]. After Algorithm 3, $\mathscr{C}_1$ has the encrypted dataset $E_{pk}(P)$ and $E_{pk}(T)$, $\mathscr{C}_2$ has the private key sk. The goal is to securely compute the skyline by $\mathscr{C}_1$ and $\mathscr{C}_2$ without participation of data owner and the client.

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

Algorithm 3

Preprocessing.



```
input : $C_1$ has $E_{pk}(P)$, $C_2$ has $sk$, and the client has $\mathbf{q}$.
output: $C_1$ obtains the new encrypted dataset $E_{pk}(T)$.
Client:
send $(E_{pk}(-\mathbf{q}[1]), ..., E_{pk}(-\mathbf{q}[m]))$ to $C_1$;
$C_1$:
for $i = 1$ to $n$ do
    for $j = 1$ to $m$ do
        $E_{pk}(temp_i[j]) = E_{pk}(\mathbf{p}_i[j] - \mathbf{q}[j]) =$
          $E_{pk}(\mathbf{p}_i[j]) \times E_{pk}(-\mathbf{q}[j]) \mod N^2$;
$C_1$ and $C_2$:
use SM protocol to compute $E_{pk}(T) = (E_{pk}(\mathbf{t}_1), ..., E_{pk}(\mathbf{t}_n))$
  only known by $C_1$, where $E_{pk}(\mathbf{t}_i) = (E_{pk}(\mathbf{t}_i[1]), ..., E_{pk}(\mathbf{t}_i[m]))$
  and $E_{pk}(\mathbf{t}_i[j]) = E_{pk}(temp_i[j]) \times E_{pk}(temp_i[j])$;
```

A. 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. $\mathscr{C}_1$ 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— $\mathscr{C}_1$ first computes the sum of $E_{pk}(\mathbf{t}_i[j])$ for $1 \le j \le m$, denoted as $E_{pk}(S(\mathbf{t}_i))$, for each tuple $\mathbf{t}_i$. Then $\mathscr{C}_1$ and $\mathscr{C}_2$ uses SMIN protocol such that $\mathscr{C}_1$ 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 $\mathscr{C}_1$ to compute $E_{pk}(S(\mathbf{t}_i) - S(\mathbf{t}_{min}))$ for all tuples and then send them to $\mathscr{C}_2$. $\mathscr{C}_2$ can decrypt them and determine which one is equal to 0 and return the index to $\mathscr{C}_1$. $\mathscr{C}_1$ then adds the tuple $E_{pk}(\mathbf{p}_{min})$ to skyline pool.

Eliminate dominated tuples—Once the skyline tuple is selected, $\mathscr{C}_1$ and $\mathscr{C}_2$ 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 $\mathscr{C}_1$ to send the encrypted dominance output to $\mathscr{C}_2$, who can decrypt it and send back the indexes of the tuples who are dominated to $\mathscr{C}_2$. $\mathscr{C}_1$ 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 $\mathscr{C}_1$ 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 $\mathscr{C}_1$ and $\mathscr{C}_2$ at the same time, $\mathscr{C}_1$ and $\mathscr{C}_2$ know nothing about the final result. Given 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 $\mathscr{C}_1$ to add a random noise $r_i[j]$ to each $\mathbf{p}_i[j]$ in ciphertext and then sends the encrypted randomized values $a_i[j]$ to $\mathscr{C}_2$. $\mathscr{C}_1$ also sends the noise $r_i[j]$ to client. At the same time, $\mathscr{C}_2$ decrypts the randomized values $a_i[j]$ and sends the result $r_i^{\prime}[j]$ to client. Client receives the random noise $r_i[j]$ from $\mathscr{C}_1$ and randomized values of the skyline points $a_i[j]$ from $\mathscr{C}_2$, and removes the noise by computing $\mathbf{p}_i[j]=r_i^{\prime}[j]-r_i[j]$ for $i = 1, \ldots, k$ and $j = 1, \ldots, m$ as the final result.

B. Fully Secure Skyline Protocol

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

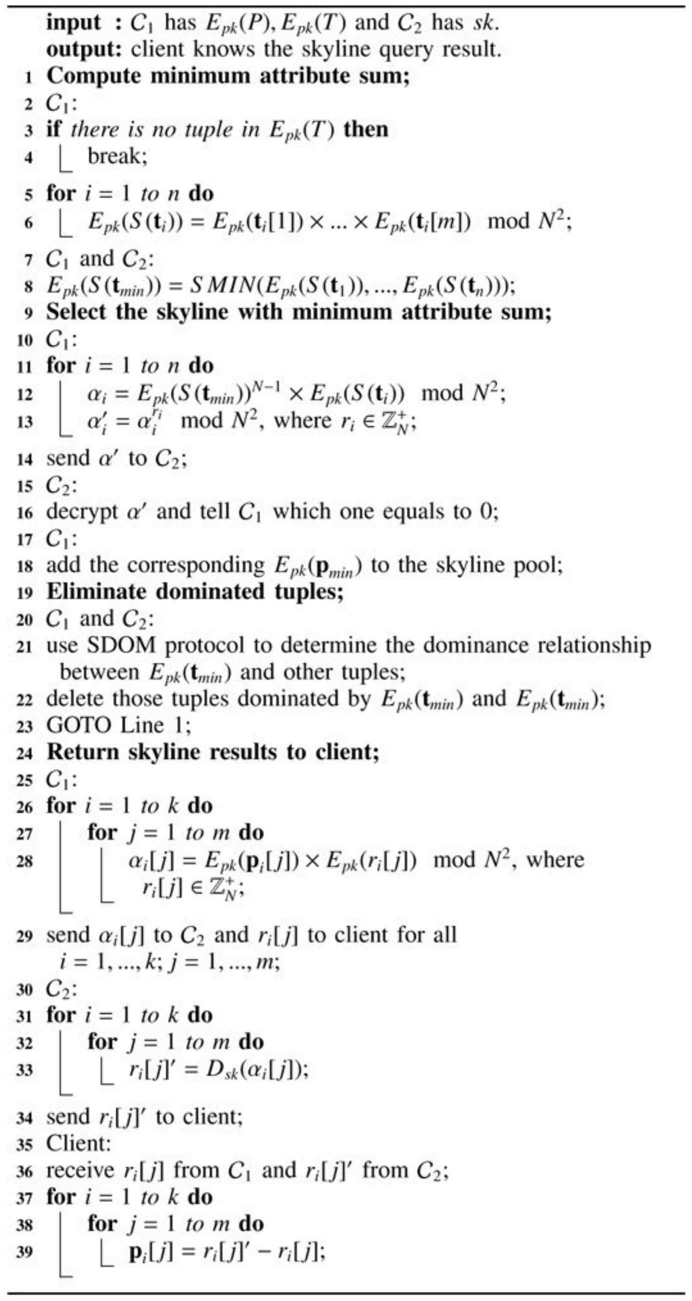
- When selecting the skyline tuple with minimum attribute sum, $\mathscr{C}_1$ and $\mathscr{C}_2$ know which tuples are skyline points, which violates result privacy requirement.
- When eliminating dominated tuples, $\mathscr{C}_1$ and $\mathscr{C}_2$ know the dominance relationship among tuples with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

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 disclosure risks of the basic protocol.

Select skyline with minimum attribute sum—Once $\mathscr{C}_1$ 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 $\mathscr{C}_1$ and $\mathscr{C}_2$ know nothing about which tuple is selected. We present a protocol as shown in Algorithm 6.

Algorithm 4

Basic Secure Skyline Protocol.



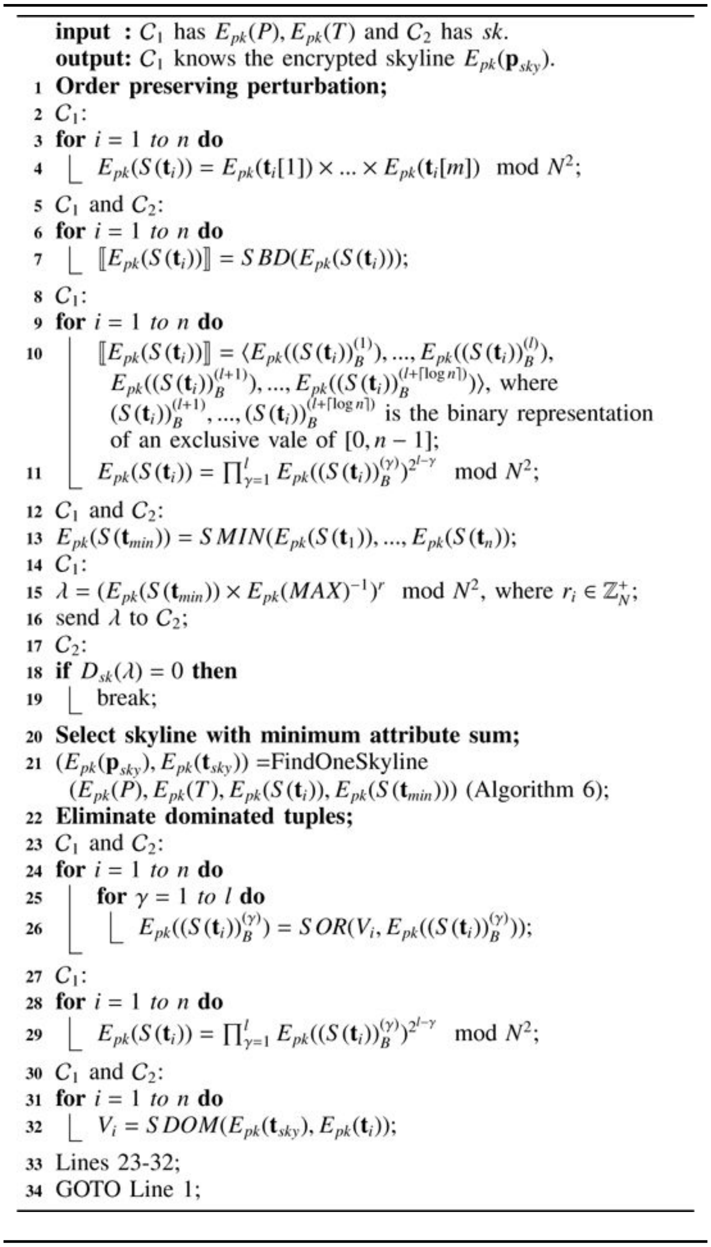
```
input  : $C_1$ has $E_{pk}(P), E_{pk}(T)$ and $C_2$ has $sk$.
output: client knows the skyline query result.
1  Compute minimum attribute sum;
2  $C_1$:
3  if there is no tuple in $E_{pk}(T)$ then
4      break;
5  for $i = 1$ to $n$ do
6      $E_{pk}(S(\mathbf{t}_i)) = E_{pk}(\mathbf{t}_i[1]) \times ... \times E_{pk}(\mathbf{t}_i[m]) \mod N^2$;
7  $C_1$ and $C_2$:
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 $C_1$:
11 for $i = 1$ to $n$ do
12     $\alpha_i = E_{pk}(S(\mathbf{t}_{min}))^{N-1} \times E_{pk}(S(\mathbf{t}_i)) \mod N^2$;
13     $\alpha'_i = \alpha_i^{r_i} \mod N^2$, where $r_i \in \mathbb{Z}_N^+$;
14 send $\alpha'$ to $C_2$;
15 $C_2$:
16 decrypt $\alpha'$ and tell $C_1$ which one equals to 0;
17 $C_1$:
18 add the corresponding $E_{pk}(\mathbf{p}_{min})$ to the skyline pool;
19 Eliminate dominated tuples;
20 $C_1$ and $C_2$:
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 $C_1$:
26 for $i = 1$ to $k$ do
27     for $j = 1$ to $m$ do
28         $\alpha_i[j] = E_{pk}(\mathbf{p}_i[j]) \times E_{pk}(r_i[j]) \mod N^2$, where
             $r_i[j] \in \mathbb{Z}_N^+$;
29 send $\alpha_i[j]$ to $C_2$ and $r_i[j]$ to client for all
     $i = 1, ..., k; j = 1, ..., m$;
30 $C_2$:
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 $C_1$ and $r_i[j]'$ from $C_2$;
37 for $i = 1$ to $k$ do
38     for $j = 1$ to $m$ do
39         $\mathbf{p}_i[j] = r_i[j]' - r_i[j]$;
```

We first need to determine which $S(\mathbf{t}_j)$ 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 $\mathscr{C}_2$ to know there is one equal case (since we are selecting one skyline tuple) as long as it does not know which one. $\mathscr{C}_1$ first computes

$\alpha_i'=E_{pk}((S(\mathbf{t}_i)-S(\mathbf{t}_{min}))\times r_i)$, and then sends a permuted list $\boldsymbol{\beta} = \boldsymbol{\pi}(\boldsymbol{\alpha}')$ to $\mathscr{C}_2$ based on a random permutation sequence $\boldsymbol{\pi}$. The permutation hides which sum is equal to the minimum from $\mathscr{C}_2$ 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 $\alpha_i'=0$. $\mathscr{C}_1$ decrypts β_i, if it is 0, it means tuple i has smallest $E_{pk}(S(\mathbf{t}_i))$. Therefore, $\mathscr{C}_2$ sends $E_{pk}(1)$ to $\mathscr{C}_1$, otherwise, sends $E_{pk}(0)$.

Algorithm 5

Fully Secure Skyline Protocol.



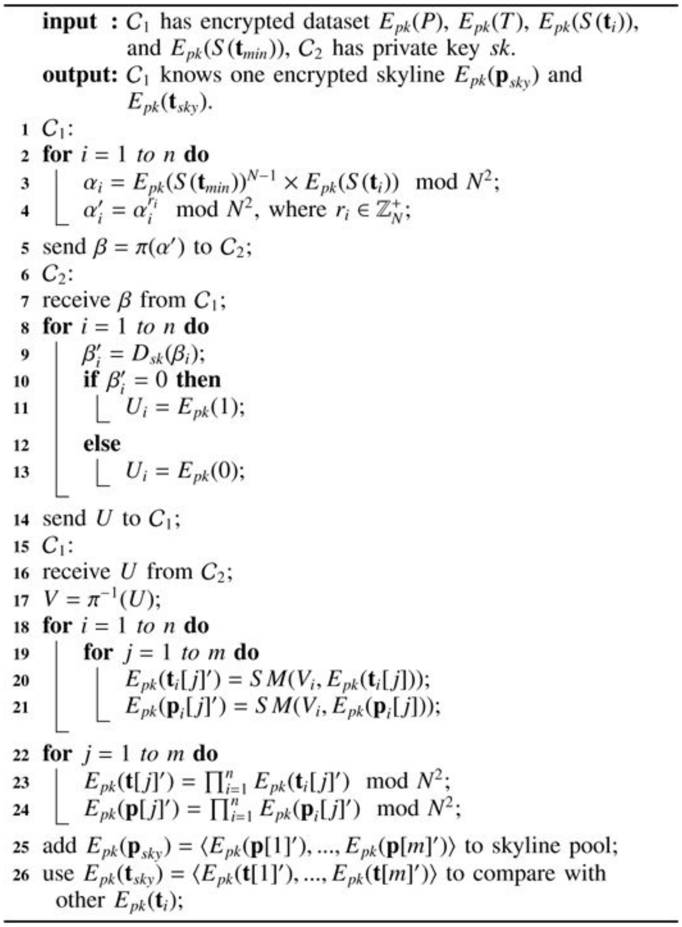
```
   input : $C_1$ has $E_{pk}(P), E_{pk}(T)$ and $C_2$ has $sk$.
   output: $C_1$ knows the encrypted skyline $E_{pk}(\mathbf{p}_{sky})$.
1  Order preserving perturbation;
2  $C_1$:
3  for $i = 1$ to $n$ do
4      $E_{pk}(S(\mathbf{t}_i)) = E_{pk}(\mathbf{t}_i[1]) \times ... \times E_{pk}(\mathbf{t}_i[m]) \mod N^2$;
5  $C_1$ and $C_2$:
6  for $i = 1$ to $n$ do
7      $[\![E_{pk}(S(\mathbf{t}_i))]\!] = SBD(E_{pk}(S(\mathbf{t}_i)))$;
8  $C_1$:
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     $E_{pk}(S(\mathbf{t}_i)) = \prod_{\gamma=1}^{l} E_{pk}((S(\mathbf{t}_i))_B^{(\gamma)})^{2^{l-\gamma}} \mod N^2$;
12 $C_1$ and $C_2$:
13 $E_{pk}(S(\mathbf{t}_{min})) = SMIN(E_{pk}(S(\mathbf{t}_1)), ..., E_{pk}(S(\mathbf{t}_n)))$;
14 $C_1$:
15 $\lambda = (E_{pk}(S(\mathbf{t}_{min})) \times E_{pk}(MAX)^{-1})^r \mod N^2$, where $r_i \in \mathbb{Z}_N^+$;
16 send $\lambda$ to $C_2$;
17 $C_2$:
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 $C_1$ and $C_2$:
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 $C_1$:
28 for $i = 1$ to $n$ do
29     $E_{pk}(S(\mathbf{t}_i)) = \prod_{\gamma=1}^{l} E_{pk}((S(\mathbf{t}_i))_B^{(\gamma)})^{2^{l-\gamma}} \mod N^2$;
30 $C_1$ and $C_2$:
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;
```

After receiving the encrypted permuted bit vector U as the equality result, $\mathscr{C}_1$ 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, $\mathscr{C}_1$ and $\mathscr{C}_2$ 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 $\mathscr{C}_1$ 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 $\mathscr{C}_2$, 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 have equal cases, 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.

Algorithm 6

Find One Skyline.



```
input : $C_1$ has encrypted dataset $E_{pk}(P)$, $E_{pk}(T)$, $E_{pk}(S(\mathbf{t}_i))$,
        and $E_{pk}(S(\mathbf{t}_{min}))$, $C_2$ has private key $sk$.
output: $C_1$ knows one encrypted skyline $E_{pk}(\mathbf{p}_{sky})$ and
        $E_{pk}(\mathbf{t}_{sky})$.
$C_1$:
for $i = 1$ to $n$ do
    $\alpha_i = E_{pk}(S(\mathbf{t}_{min}))^{N-1} \times E_{pk}(S(\mathbf{t}_i)) \mod N^2$;
    $\alpha'_i = \alpha_i^{r_i} \mod N^2$, where $r_i \in \mathbb{Z}_N^+$;
send $\beta = \pi(\alpha')$ to $C_2$;
$C_2$:
receive $\beta$ from $C_1$;
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 $C_1$;
$C_1$:
receive $U$ from $C_2$;
$V = \pi^{-1}(U)$;
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
    $E_{pk}(\mathbf{t}[j]') = \prod_{i=1}^{n} E_{pk}(\mathbf{t}_i[j]') \mod N^2$;
    $E_{pk}(\mathbf{p}[j]') = \prod_{i=1}^{n} E_{pk}(\mathbf{p}_i[j]') \mod N^2$;
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 $E_{pk}(\mathbf{t}_i)$;
```

Concretely, given n numbers in their binary representations, we add a $\lceil logn \rceil$-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, $\mathscr{C}_1$ and $\mathscr{C}_2$ 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 $\mathscr{C}_1$ and $\mathscr{C}_2$ 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 $[\![E_{pk}(S(\mathrm{t}_i))]\!]$, the new $[\![E_{pk}(S(\mathrm{t}_i))]\!]$ 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, $\mathscr{C}_1$ and $\mathscr{C}_2$ 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 III. Please note that all column values are in encrypted form except columns π and β'. Given the mapped data points $\mathbf{t}_i$, $\mathscr{C}_1$ first computes the attribute sum $E_{pk}(S(\mathbf{t}_i))$ shown in the third column. We set $l = 5$, $\mathscr{C}_1$ gets the binary representation of the attribute sum $[\![E_{pk}(S(\mathbf{t}_i))]\!]$. Because $n = 4$, $\mathscr{C}_1$ obliviously adds the order-preserving perturbation $\lceil \log 4 \rceil = 2$ bits to the end of $[\![E_{pk}(S(\mathrm{t}_i))]\!]$ respectively to get the new $E_{pk}(S(\mathbf{t}_i))$ (shown in the sixth column). Then $\mathscr{C}_1$ 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. $\mathscr{C}_1$ 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 $\pi = \langle 2, 1, 4, 3 \rangle$, $\mathscr{C}_1$ sends the encrypted permuted and randomized difference vector β to $\mathscr{C}_2$. After decrypting β, $\mathscr{C}_2$ gets β' and then sends U to $\mathscr{C}_1$. $\mathscr{C}_1$ computes V by applying a reverse permutation. By employing SM with V, $\mathscr{C}_1$ computes $(E_{pk}(\mathbf{t}_i[1]'), E_{pk}(\mathbf{t}_i[2]'))$ and $(E_{pk}(\mathbf{p}_i[1]'), E_{pk}(\mathbf{p}_i[2]'))$. After summing all column values, $\mathscr{C}_1$ 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. $\mathscr{C}_1$ and $\mathscr{C}_2$ 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 $\mathscr{C}_1$ does not know this value has changed because the ciphertext of all tuples has changed. Next, $\mathscr{C}_1$ and $\mathscr{C}_2$ find the dominance relationship between $E_{pk}(\mathbf{t}_{sky})$ and $E_{pk}(\mathbf{t}_i)$ by SDOM protocol. $\mathscr{C}_1$ obtains the dominance vector V. Using same method, $\mathscr{C}_1$ 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 $\mathscr{C}_1$'s view, the intermediate result includes U. Because U is ciphertext and $\mathscr{C}_1$ does not have the secret key, $\mathscr{C}_1$ can simulate U based on its input and output. From $\mathscr{C}_2$'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 $\mathscr{C}_1$, $\mathscr{C}_2$ cannot determine where is the $E_{pk}(0)$. Therefore, $\mathscr{C}_2$ 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 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 $O(nm)$ 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.

VII. Experiments

In this section, we evaluate the performance and scalability of our protocols under different parameter settings. For comparison purposes, we implemented and evaluated both protocols: the Basic Secure Skyline Protocol (**BSSP**) in Section VI-A, and the Fully Secure Skyline Protocol (**FSSP**) in Section VI-B.

A. Experiment Setup

We implemented all algorithms in C and ran experiments on a machine with Intel Core i7-6700K 4.0GHz running Ubuntu 16.01. We also implemented a parallel version of the protocols and tested on a cluster of machines with Intel Core i7-2600 3.40GHz running CentOS 6, which we will describe in Section VII-C.

In our experiment setup, both $\mathscr{C}_1$ and $\mathscr{C}_2$ are running on the same workstation, but since we implemented the communication using sockets, it can be easily run on two machines without modification which we have tested. Moreover, the query points used in our setup are randomly chosen. The reported computation time unless otherwise noted is the total computation time of the $\mathscr{C}_1$ and $\mathscr{C}_2$.

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 [4]. We also built a dataset that contains 2384 NBA players who are league leaders of playoffs[3]. Each player has five attributes that measure the player's performance: Points (PTS), Rebounds (REB), Assists (AST), Steals (STL), and Blocks (BLK).

B. Performance Results

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.

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

Impact of number of tuples *n*—Figures 4(a)(b)(c)(d) 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*—Figures 5(a)(b)(c)(d) show the time cost of different *m* on CORR, INDE, ANTI, and NBA 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 Figure 4. 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*—Figures 6(a)(b)(c)(d) 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. Figure 7 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.

C. Performance Improvements through Parallel Implementation

In order to reduce the skyline query processing time, we demonstrate that our algorithm can be parallelized, using a hierarchical divide-and-conquer paradigm with POSIX threads. First we divide our dataset into *v* subdatasets (*v* refers to the number of threads) and assign one to each thread. Each thread computes and returns the skyline in the subdataset when it finishes. When the main thread receives skyline result from two or more threads, it merges them into one new subdataset and sends it to an idle thread, this process iterates until finally there is only one set of skyline, which is the final result. We refer to this implementation local parallelism using multi-threading.

To further demonstrate the scalability of our algorithm, we also implemented a distributed version, which employs manager-worker model. The manager plays the role of distributing data to workers, while workers (multi-threading) compute skylines in any given working set and return them to the manager, which works similarly as the multi-threading parallelism. The only difference here is that the manager could implement sophisticated load balancing algorithm to fully utilize computation resources (not implemented in our current prototype).

In the experiment setup, we used workstations of the same configurations as described earlier. In multi-threading parallelism, we run $\mathscr{C}_1$ and $\mathscr{C}_2$ on the same machine with $\mathscr{C}_1$

running 8 threads. As for distributed version, we tested it with 2, 4, 8, 16, 32, 64 worker machines.

Figures 8 (a)(b) show the time cost of parallelized FSSP of the multi-threading and distributed version, respectively. Figure 8(a) indicates that multi-threading version (with 8 threads) is about 6 times faster than the serial version. Figure 8(b) shows the time cost with varying number of worker machines and varying number of tuples. We observe that our distributed version is very effective in reducing computation time when scaling to large datasets. And it shows a sub-linear time increase with respect to the number of worker machines.

VIII. Conclusions

In this paper, we presented 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. 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

The authors would like to thank the anonymous reviewers for their helpful comments. 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. Baldimtsi F, Ohrimenko O. Sorting and searching behind the curtain. FC 2015. 2015:127–146.
2. Beimel, A. International Conference on Coding and Cryptology. Springer; 2011. Secret-sharing schemes: a survey; p. 11-46.
3. Bentley JL. Multidimensional divide-and-conquer. Commun ACM. 1980; 23(4):214–229.
4. Börzsöonyi S, Kossmann D, Stocker K. The skyline operator. ICDE. 2001
5. Bothe S, Cuzzocrea A, Karras P, Vlachou A. Skyline query processing over encrypted data: An attribute-order-preserving-free approach. PSBD@CIKM. 2014:37–43.
6. Bothe S, Karras P, Vlachou A. eskyline: Processing skyline queries over encrypted data. PVLDB. 2013
7. Chan CY, Jagadish HV, Tan KL, Tung AKH, Zhang Z. Finding k-dominant skylines in high dimensional space. SIGMOD Conference. 2006:503–514.
8. Chen W, Liu M, Zhang R, Zhang Y, Liu S. Secure outsourced skyline query processing via untrusted cloud service providers. INFOCOM. 2016
9. Costan, V., Devadas, S. Intel sgx explained. Technical report, Cryptology ePrint Archive, Report 2016/086. 2016. http://eprint.iacr.org
10. Dellis E, Seeger B. Efficient computation of reverse skyline queries. VLDB. 2007:291–302.
11. Elmehdwi Y, Samanthula BK, Jiang W. Secure k-nearest neighbor query over encrypted data in outsourced environments. ICDE. 2014

12. Erkin Z, Franz M, Guajardo J, Katzenbeisser S, Lagendijk I, Toft T. Privacy-preserving face recognition. PETS. 2009:235–253.
13. Feige U, Fiat A, Shamir A. Zero-knowledge proofs of identity. J Cryptology. 1988; 1(2):77–94.
14. Gentry C. Fully homomorphic encryption using ideal lattices. STOC. 2009
15. Goldreich, O. The Foundations of Cryptography - Volume 2, Basic Applications. Cambridge University Press; 2004.
16. Goldreich O, Micali S, Wigderson A. How to play any mental game or A completeness theorem for protocols with honest majority. ACM Symposium on Theory of Computing. 1987:218–229.
17. Hacigümüs H, Iyer BR, Li C, Mehrotra S. Executing SQL over encrypted data in the database-service-provider model. SIGMOD 2002. 2002:216–227.
18. Halevi S, Shoup V. Bootstrapping for helib. EUROCRYPT 2015. 2015:641–670.
19. Hashem T, Kulik L, Zhang R. Privacy preserving group nearest neighbor queries. EDBT. 2010
20. Hu H, Xu J, Ren C, Choi B. Processing private queries over untrusted data cloud through privacy homomorphism. ICDE. 2011
21. Huang Y, Evans D, Katz J, Malka L. Faster secure two-party computation using garbled circuits. USENIX 2011. 2011
22. Janosi, A., Steinbrunn, W., Pfisterer, M., Detrano, R. Heart disease dataset. The UCI Archive. 1998. https://archive.ics.uci.edu/ml/datasets/heart+disease
23. Kirkpatrick DG, Seidel R. Output-size sensitive algorithms for finding maximal vectors. Symposium on Computational Geometry. 1985:89–96.
24. Kossmann D, Ramsak F, Rost S. Shooting stars in the sky: An online algorithm for skyline queries. VLDB 2002. 2002
25. Kung HT, Luccio F, Preparata FP. On finding the maxima of a set of vectors. JACM. 1975
26. Li C, Zhang N, Hassan N, Rajasekaran S, Das G. On skyline groups. CIKM. 2012:2119–2123.
27. Liu A, Zheng K, Li L, Liu G, Zhao L, Zhou X. Efficient secure similarity computation on encrypted trajectory data. ICDE. 2015:66–77.
28. Liu J, Xiong L, Pei J, Luo J, Zhang H. Finding pareto optimal groups: Group-based skyline. PVLDB. 2015; 8(13):2086–2097.
29. Liu J, Xiong L, Xu X. Faster output-sensitive skyline computation algorithm. Inf Process Lett. 2014
30. Liu J, Zhang H, Xiong L, Li H, Luo J. Finding probabilistic k-skyline sets on uncertain data. CIKM. 2015:1511–1520.
31. Paillier P. Public-key cryptosystems based on composite degree residuosity classes. Advances in Cryptology - EUROCRYPT ’99. 1999:223–238.
32. Papadias D, Tao Y, Fu G, Seeger B. Progressive skyline computation in database systems. ACM Trans Database Syst. 2005; 30(1):41–82.
33. Papadopoulos S, Bakiras S, Papadias D. Nearest neighbor search with strong location privacy. PVLDB. 2010
34. Pei J, Jiang B, Lin X, Yuan Y. Probabilistic skylines on uncertain data. VLDB. 2007:15–26.
35. Qi Y, Atallah MJ. Efficient privacy-preserving k-nearest neighbor search. ICDCS. 2008
36. Samanthula BK, Hu C, Jiang W. An efficient and probabilistic secure bit-decomposition. ASIA CCS. 2013:541–546.
37. Song DX, Wagner D, Perrig A. Practical techniques for searches on encrypted data. IEEE Symposium on Security and Privacy. 2000
38. Veugen T, Blom F, de Hoogh SJA, Erkin Z. Secure comparison protocols in the semi-honest model. J Sel Topics Signal Processing. 2015; 9(7):1217–1228.
39. Wong WK, Cheung DW, Kao B, Mamoulis N. Secure knn computation on encrypted databases. SIGMOD. 2009
40. Yao AC. Protocols for secure computations (extended abstract). FOCS. 1982:160–164.
41. Yao B, Li F, Xiao X. Secure nearest neighbor revisited. ICDE. 2013
42. Yi X, Paulet R, Bertino E, Varadharajan V. Practical k nearest neighbor queries with location privacy. ICDE. 2014

43. Zhu H, Meng X, Kollios G. Privacy preserving similarity evaluation of time series data. EDBT. 2014:499–510.

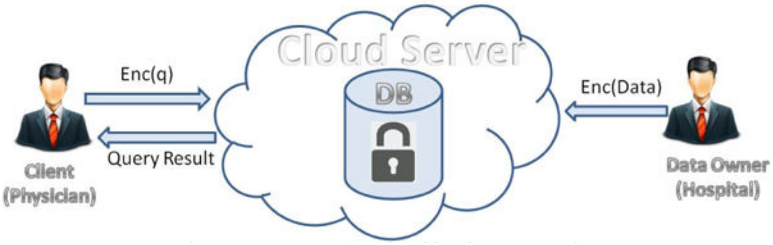


Fig. 1.
Secure similarity queries.

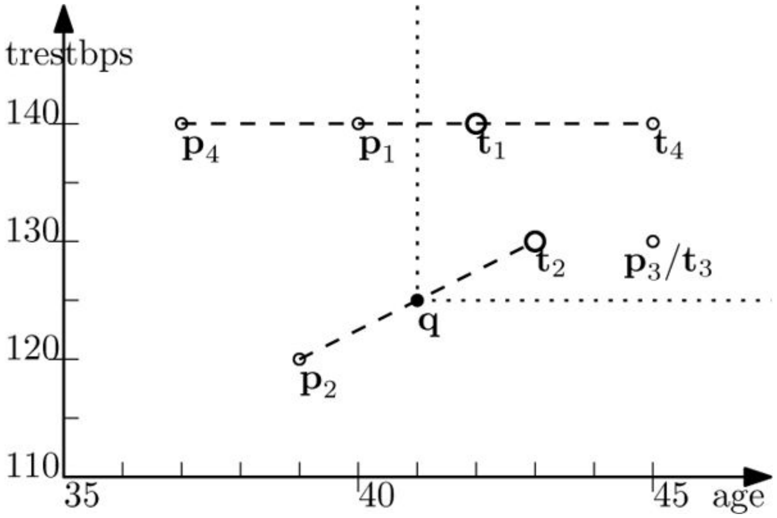


Fig. 2.

Dynamic skyline query.

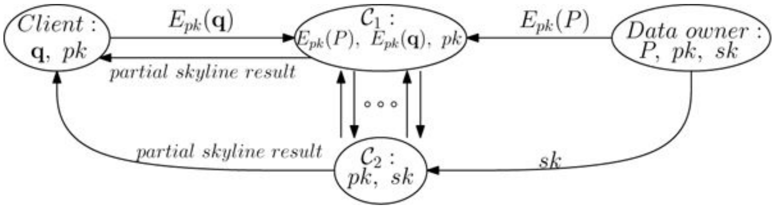
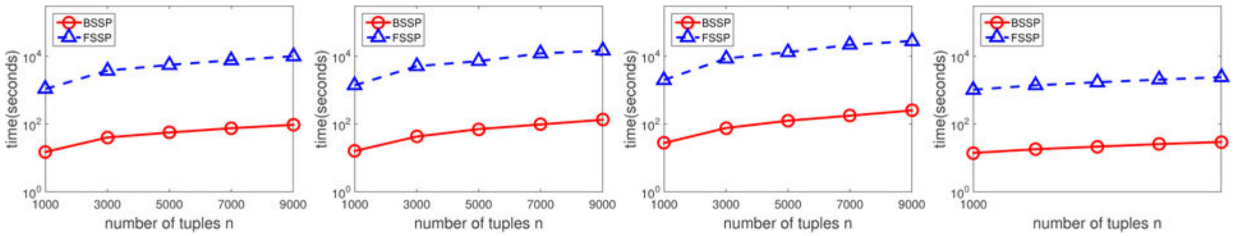


Fig. 3.

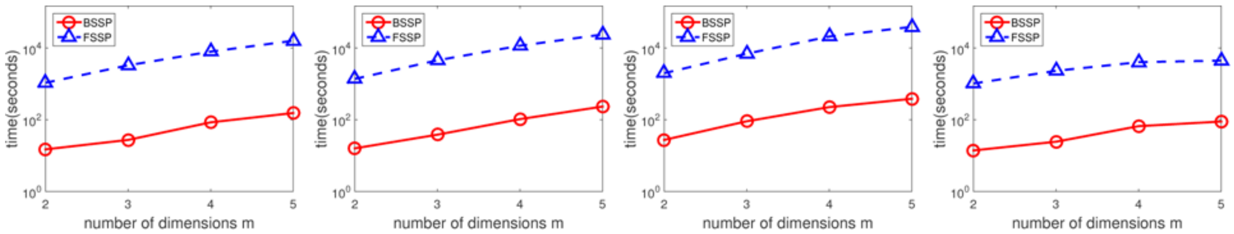
Overview of protocol setting.



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

Fig. 4.

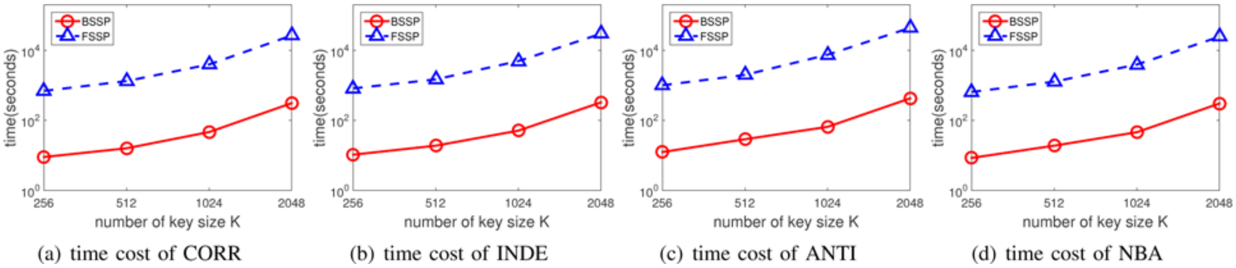
The impact of n (m=2, K=512).



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

Fig. 5.

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. 6.

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

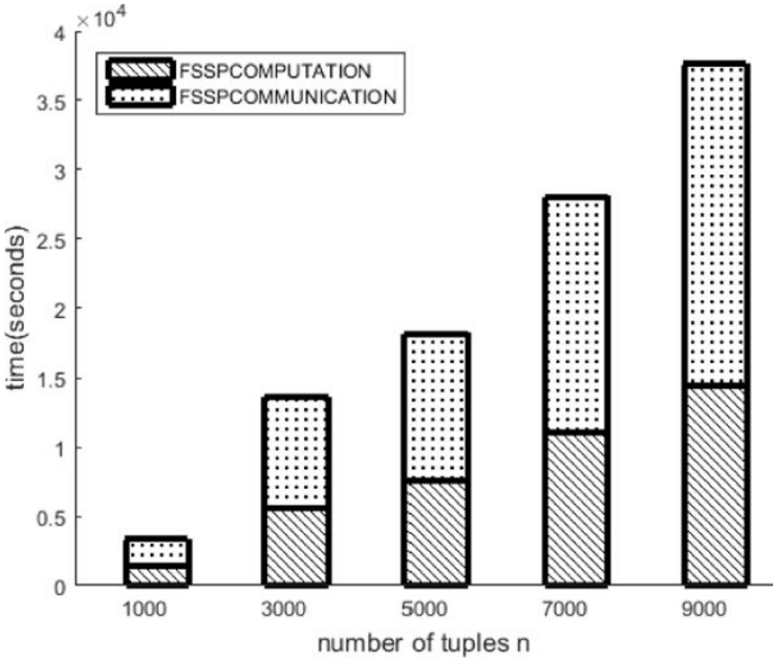
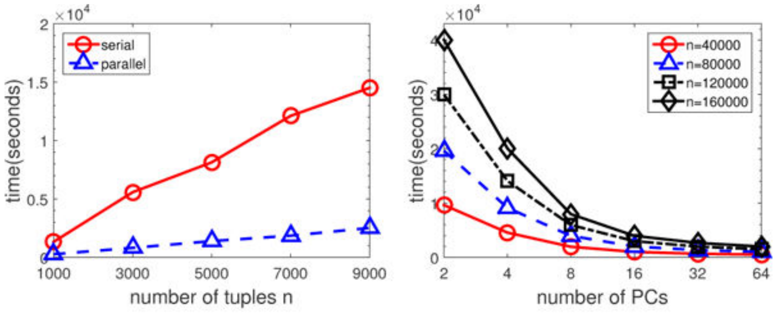


Fig. 7.

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



(a) serial vs. multi-threading.

(b) distributed parallel implementation.

Fig. 8.

Parallel implementations (m=2, K=512).

TABLE I

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

TABLE II

The summary of notations.

Notation	Definition
P	dataset of n points/tuples/records
$\mathbf{p}_i[j]$	the j^{th} 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 i^{th} bit of binary number a

TABLE III

Example of Algorithm 8.

$\mathscr{C}_1$:									$\mathscr{C}_2$:		$\mathscr{C}_1$:					
t$_i$	**(t**$_i$**[1], t**$_i$**[2])**	S**(t**$_i$**)**	**⟦**S**(t**$_i$**)⟧**	**pert.**	S**(t**$_i$**)**	S**(t**$_i$**)** − S**(t**$_{min}$**)**	r	π	β'	U	V	**(t**$_i$**[1]′, t**$_i$**[2]′)**	**(p**$_i$**[1]′, p**$_i$**[2]′)**	S**(t**$_i$**)**	V	S**(t**$_i$**)**
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
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
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
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