information

MDPI

Article

# Privacy-Aware MapReduce Based Multi-Party Secure Skyline Computation

**Saleh Ahmed [1,2,*], Mahboob Qaosar [1,3], Asif Zaman [3], Md. Anisuzzaman Siddique [3], Chen Li [1], Kazi Md. Rokibul Alam [4] and Yasuhiko Morimoto [1]**

[1] Graduate School of Engineering, Hiroshima University, Higashi-Hiroshima 739-8527, Japan; D172517@hiroshima-u.ac.jp or qaosar@ru.ac.bd (M.Q.); D165000@hiroshima-u.ac.jp (C.L.); morimo@hiroshima-u.ac.jp (Y.M.)

[2] Department of Computer Science, Bangabandhu Sheikh Mujibur Rahman Science and Technology University, Gopalganj 8100, Bangladesh

[3] Department of Computer Science and Engineering, University of Rajshahi, Rajshahi 6205, Bangladesh; asif@ru.ac.bd (A.Z.); anisuzzaman@ru.ac.bd (M.A.S.)

[4] Department of Computer Science and Engineering, Khulna University of Engineering and Technology, Khulna 9203, Bangladesh; rokib@cse.kuet.ac.bd

* Correspondence: D162694@hiroshima-u.ac.jp or saleh@bsmrstu.edu.bd

Received: 22 April 2019; Accepted: 5 June 2019; Published: 8 June 2019

**Abstract:** Selecting representative objects from a large-scale dataset is an important task for understanding the dataset. Skyline is a popular technique for selecting representative objects from a large dataset. It is obvious that the skyline computation from the collective databases of multiple organizations is more effective than the skyline computed from a database of a single organization. However, due to privacy-awareness, every organization is also concerned about the security and privacy of their data. In this regards, we propose an efficient multi-party secure skyline computation method that computes the skyline on encrypted data and preserves the confidentiality of each party's database objects. Although several distributed skyline computing methods have been proposed, very few of them consider the data privacy and security issues. However, privacy-preserving multi-party skyline computing techniques are not efficient enough. In our proposed method, we present a secure computation model that is more efficient in comparison with existing privacy-preserving multi-party skyline computation models in terms of computation and communication complexity. In our computation model, we also introduce MapReduce as a distributive, scalable, open-source, cost-effective, and reliable framework to handle multi-party data efficiently.

**Keywords:** skyline; MapReduce; distributed system; information security; order-preserving encryption; homomorphic encryption; big data; data privacy; semi-honest model; multi-party computation

## 1. Introduction

In the present era of information technology, organizations with a similar type of service collect various information from their clients. For reliable and effective analysis, they want to perform a study on their collective databases. This kind of analysis is called a multi-party computation; examples of multi-party computations are joint data analysis, data mining, statistical data analysis, etc. Business services may contain sensitive data, such as personal, financial, or health-related data of their clients. Disclosure of such data significantly violate clients' privacy and may cause a financial or goodwill loss to the organization. Therefore, the organizations never wants to disclose their sensitive data to others. However, during joint data mining operations, the participating parties are willing to obtain the result from their combined databases without revealing the sensitive information of the clients.

Skyline has received considerable attention in the database community during the past few decades. It is an important tool in many multi-criteria decision-making applications like business planning, hotel management, etc. Given a dominance relationship in a dataset, a skyline query returns the objects that are not dominated by any other objects within the dataset.

Let us assume a number of organizations willing to conduct surveys about commission cost and risk prediction where all the organizations have collected similar types of private data from their clients. Maintaining the privacy of the information of each client is a prime responsibility for each organization. As we know, skyline computation requires the comparison of attribute values among the objects of each party; without disclosing the attribute values, organizations are unable to compute the multi-party skyline from the union of their databases. The data of Table 1 plotted on the Figure 1 illustrates this scenario where *P*1, *P*2, *P*3, *P*4, and *P*5 are five records of Organization 1 and *Q*1, *Q*2, *Q*3, *Q*4, and *Q*5 are five records of Organization 2 with their costs (*d*1) and risks (*d*2). If both organizations want to find a reasonable recommendation list considering minimum cost and risk using skyline query, the skyline objects for the individual database of Organization 1 will be *P*1, *P*2, and *P*4, and the skyline objects for Organization 2 will be *Q*1, *Q*3, and *Q*4. However, the skyline objects of their combined databases will be *Q*1, *P*4, *Q*4, and *P*2. Although the object *P*1 and the object *Q*3 are in the skyline of Organization 1 and Organization 2, respectively, they are not present in their combined skylines. The object *P*4 of Organization 1 dominates the object *Q*3 of Organization 2 and the objects *Q*1 of Organization 2 dominates the object *P*1 of Organization 1. Therefore, cost and risk prediction using a skyline query are more reliable and meaningful if they are computed from the data of both organizations. Therefore, the organizations want to calculate skyline objects from their combined data. However, for security reason, they do not wish to disclose the attribute values in the objects with others. Therefore, we need a secured system that can compute the skyline from combined data of both parties without revealing the real attribute values during computation.

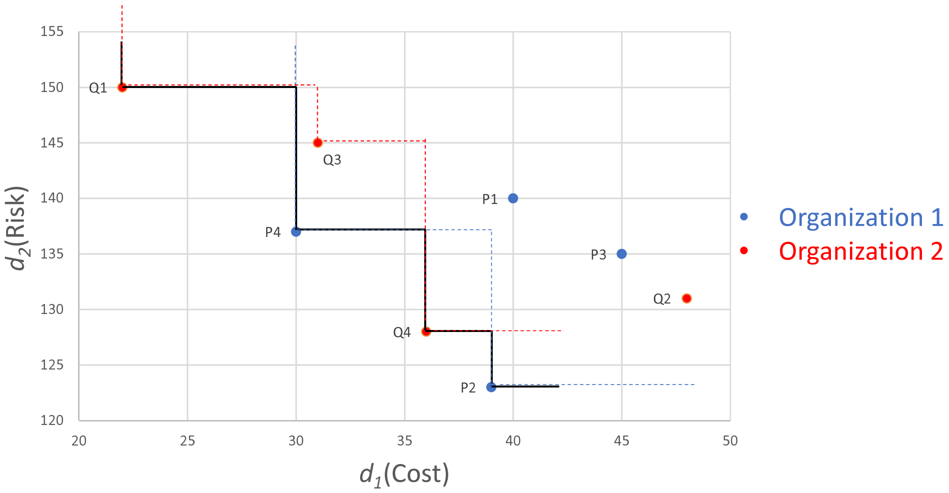


**Figure 1.** The skyline and multiparty skyline of Organizations 1 and 2.

**Table 1.** Organization database.

| Organization 1 | | | Organization 2 | | |
|---|---|---|---|---|---|
| **ID** | **Cost** | **Risk** | **ID** | **Cost** | **Risk** |
| *P*1 | 27 | 33 | *Q*1 | 22 | 30 |
| *P*2 | 39 | 3 | *Q*2 | 48 | 11 |
| *P*3 | 45 | 15 | *Q*3 | 32 | 25 |
| *P*4 | 30 | 17 | *Q*4 | 36 | 8 |
| *P*5 | 45 | 30 | *Q*5 | 42 | 37 |

Usually, skyline computation requires a massive comparison of objects' attributes to determine whether an object is in the skyline or not. It may need many object dominance checks, and each check may involve all of the objects' dimensions. Skyline computation is both IO-consuming and CPU-intensive in centralized settings [1,2]. Therefore, in the interest of overall efficiency, it is useful to compute skylines in distributed and parallel environments.

On the other hand, MapReduce is increasingly used to process massive data due to its scalability and fault tolerance. The availability of scalable and open-source MapReduce systems, such as Hadoop [3], makes it desirable for large-scale parallel skyline computation. Though the MapReduce framework usually has been composed in a local area network, which is owned by one organization, the distributed computation can be extended to computing the skyline from the union of databases that are owned by multiple organizations. In such cases, users have to send each database independently and securely to the MapReduce framework. In this way, the distributed computing of MapReduce will be used for multi-party databases. However, we also have to secure the privacy of values during the processing of multi-party data. We, therefore, think that privacy-aware computations of multi-party data in MapReduce have to be considered.

Although a number of skyline computation methods [1,2,4–6] utilize the MapReduce framework to calculate the skyline in a distributed environment, except [2], none of them consider the security issues for the multi-party skyline. The previously-proposed secure skyline computation method [2] only used MapReduce operation for sorting the attribute values in the multi-party objects and demanded multiple rounds of MapReduce operations.

Moreover, several methods have been proposed for secure skyline query [7–10]. The methods in [7,8] only considered the secure computation of the skyline for clients from single party data stored in the cloud platform. The method in [9] computed the skyline from two-party data and exchanged a significant amount of data during the secure comparison. It also needed multiple two-party skyline computation to obtain the multi-party skyline. Although our previously-proposed encrypted substitution vector-based framework [10] improved the efficiency, compared to other secure skyline computation frameworks, it also required sharing an encrypted substitution vector before secure computation of the skyline.

In our proposed method, we introduce an efficient multi-party skyline computation method that generates skyline objects from multi-party data and preserves the privacy of individual objects during multi-party skyline computation. Our method simultaneously processes multi-party data, concurrently executes operations for skyline computation in each phase, and uses only two rounds of MapReduce operations. For our proposed method, a minimum number of data exchanged is required among the parties.

This paper is organized as follows:

Section 2 discusses and reviews the related work; Section 3 explains the required preliminary knowledge; Section 4 explains the methodology of computing the secure multi-party skyline with an example; Section 5 specifies the scalability and the application of the method; Section 6 specifies the security issues; Section 7 provides the theoretical analysis of our proposed method; Section 8 discusses the experiment details and explains the effectiveness and efficiency of our method under various settings; Section 9 concludes the proposed work.

## 2. Related Work

The following sections discuss the various related work.

### 2.1. Skyline Query

Borzsonyi et al., first proposed the skyline operator over a large dataset and provided three algorithms: Block-Nested-Loops (BNL), Divide-and-Conquer (D&C), and B-tree-based schemes [11]. The BNL algorithm compares every object with every other object in the database, then non-dominant objects are collected as a result. The D&C algorithm divides the data in such a way that they can fit into memory; then, the candidate skylines are computed in each partition. After merging the candidate skyline from each partition, it gets the final skyline. The B-tree-based schemes [11] compute the skyline using an ordered index; e.g., a B-tree, an R-tree, etc. Chomicki et al., improved BNL by presorting data and proposed a variant of BNL as Sort-Filter-Skyline (SFS) [12]. The Branch-and-Bound Skyline (BBS) [13], proposed by Papadias et al., is a progressive algorithm based on the best-first nearest neighbor (BF-NN) algorithm. In the same way, Kossmann et al., improved the performance of the D&C algorithm and proposed the nearest neighbor (NN) [14] algorithm that prunes out dominated objects by iteratively dividing the data space based on the closest objects in the space. Furthermore, none of the above works considered the multi-party skyline or security issues.

### 2.2. Secure Skyline Query

Because of the privacy awareness of the present era, each organization expresses concern about the security of their information. Privacy-preserving secure data analytics is one of the major research areas in "big data" processing. In different application aspects, it is necessary to compute the multi-party skyline without revealing the vital information to others. Liu et al. [7] proposed secure skyline queries that can execute the skyline query in encrypted form on the cloud platform. To compute the secure skyline, Liu et al., used the secure comparison protocol proposed by Veugen et al. [15] and the secure bit-decomposition method proposed by Samanthula et al. [16]. Hua et al., proposed a privacy-preserving skyline computation model named CINEMA [8]. In their work, they proposed a solution for computing the secure skyline based on the user's dynamic query. In this proposed method, a user can hide the dynamic query point from the database owner, and the database owner can also protect the data from the user during computation. Although their model provided a secure computation environment concerning data privacy, their circumstances were different from ours. Moreover, their model used computationally-expensive secure comparison protocols.

Liu et al., provided another privacy-preserving skyline computation system [9] using the additivity property of the skyline [17] to reduce the number of secure comparisons. They computed the local skyline object set at first. Then, from the local skyline, they used secure dominance relationship computation and calculated the global skyline object set. However, for several participating parties, it needs a pair-wise secure skyline computation for computing the global skyline. Therefore, as the number of participating party grows, the computational complexity increases rapidly. Besides, the complexity of the zero-encoding and one-encoding schemes used in their proposed method increased with the domain length of the attribute values. From the above discussion, we can assume that the work mentioned above for multi-party skyline query is not efficient enough when the number of parties increases and also needs to exchange a significant amount information during computation among the parties.

Qaosar et al., proposed a secure multi-party skyline computation method [10]. It improved efficiency, compared to other secured skyline computation frameworks, but it required sharing an encrypted substitution vector before secure computation of the skyline.

In our proposed method, we solve the problems by keeping the exchange of information minimum and by applying concurrent and distributed computation of skyline in each phase.

### 2.3. MapReduce-Based Skyline Query

Recently, the distributed computing paradigm has become very popular for skyline computation. Kasper Mullesgaard et al. [1] proposed efficient skyline computation in the MapReduce framework. They designed a grid partitioning scheme to divide the data space into several partitions and employed a bit-string to represent the partitions. The bit-string was efficiently obtained in MapReduce, and it helped to prune partitions (and tuples) that could not have skyline tuples. Hyeong-Cheol Ryu et al., used adaptive two-level grids to process the skyline query in MapReduce [4]. Ji Zhang et al. [5] in their scheme considered data partitioning, filtering, and parallel skyline evaluation as a holistic query process. To improve the parallel local skyline calculation, they proposed two partition-aware filtering methods that kept skyline candidates in a balanced manner. Yoonjae Park et al. [6] proposed efficient parallel algorithms for processing the skyline and its variants using MapReduce. They effectively pruned out non-skyline points in advance with the help of histograms calculated from all points. Then, using the quadtree, they divided the data into partitions, where each partition contained the same number of data points. In the first MapReduce phase, it computed the candidate skyline in each partition, then in the next step, it combined the candidate skyline to generate the final skyline. Conversely, the above-discussed methods did not consider multi-party databases and privacy issues regarding multi-party skyline.

Asif Zaman et al. [2] introduced the secure objects' ordering-based skyline computation framework. In his work, all participating parties constructed their database objects' order with the help of a semi-honest third party, called the coordinator. At first, the method generated the order of the values for each dimension in the multi-party databases. This computation required one round of MapReduce operation for each digit in a dimension. For example, in the multi-party databases, if there were $D$ dimensions, and each dimension contained $M$ digits, then it needed $D * M$ number of MapReduce rounds for order generation. Afterwards, from the order of the values in each dimension of multi-party databases, it computed the skyline using another round of the MapReduce operation. In our current proposed method, we use two MapReduce rounds for calculating the privacy-preserving skyline, thus improving the performance.

## 3. Preliminaries

### 3.1. Adversary Model

In this work, we consider the semi-honest adversary model for multi-party computation. In the semi-honest adversary model, no parties are allowed to share information with any other party, other than permitted by the protocol. Each party may be honest, but curious and try to analyze the data during computation. The security threats in this model depend on how each party successfully obtains private information from the data during calculation. Berlin and Heidelberg in their book [18] provided a proper definition and the security proof of the semi-honest model.

### 3.2. Dominance and Skyline

Assume a dataset $P = \{P_1, P_2, \cdots, P_n\}$ of $n$ objects with $m$-dimensions $\{d_1, d_2, \cdots, d_m\}$. We consider $P_i.d_j$ to denote the value of the $j$-th dimension of object $P_i$. Without loss of generality, we consider the lower value in each attribute to be better while calculating the skyline.

**Dominance:** An object $P_i \in P$ is dominant over another object $P_j \in P$, denoted as $P_i \prec P_j$, if $P_i.d_r \leq P_j.d_r$ $(1 \leq r \leq d)$ for all $d$ dimensions and $P_i.d_t < P_j.d_t$ $(1 \leq t \leq d)$ for at least one attribute. We say such a $P_i$ is a dominant object and such a $P_j$ is a dominated object between $P_i$ and $P_j$. For example, in Figure 1, object $Q3(31, 145)$ is dominated by object $P4(30, 137)$; because $P4$ is better than $Q3$ in both dimensions. At the same time, objects $P4(30, 137)$ and $P2(39, 123)$ are not dominating each other because $P4$ is better than $P2$ when we consider Dimension 1 (cost) and $P2$ is better than $P4$ when we consider Dimension 2 (risk).

**Skyline:** An object $P_i \in P$ is said to be a *skyline object* of $P$, if and only if there is no such object $P_j \in P$ ($j \neq i$) that dominates $P_i$.

The skyline of $P$, denoted by $Sky(P)$, is the set of skyline objects in $P$. For the dataset shown in Figure 1, objects $\{Q1, P4, Q4, P2\}$ are not dominated by any other objects. Thus, the skyline query retrieves $Sky(P) = \{Q1, P4, Q4, P2\}$, where $P = \{P1, P2, P3, P4, Q1, Q2, Q3, Q4\}$.

**Local skyline:** In multi-party secure skyline computation, we consider the skyline of the single party as the local skyline. In Figure 1, the object sets $\{P1, P2, P4\}$ and $\{Q1, Q3, Q4\}$ are the local skyline of Organization 1 and Organization 2, respectively.

**Global skyline:** The skyline computed from all objects of the private parties in a secure multi-party skyline computation is called the global skyline. In Figure 1, the objects $\{Q1, P4, Q4, P2\}$ are the global skylines of Organization 1 and Organization 2.

**Additivity of skyline computation [17]:** Suppose a dataset $P$ (a union of datasets of $d$ number of datasets) such that $P = P_1 \cup \cdots \cup P_d$, then the additive property of skyline computation ensures that: $Sky(P) = Sky(Sky(P_1) \cup \cdots \cup Sky(P_d))$.

### 3.3. Order-Preserving Encryption

Order-preserving Encryption (OPE) [19] is a symmetric key encryption technique whose encryption function preserves the numerical order of plaintext. Consider a database $D$ containing $|D|$ number of *plaintext* and represented as $D = d_1, d_2, ..., d_{|D|}$ where $d_i < p_{i+1}$. After encrypting the *plaintext* values into *ciphertext* values, we get $\tilde{C} = \tilde{c}_1, \tilde{c}_2, ..., \tilde{c}_{|D|}$, which ensures that $\tilde{c}_i < \tilde{c}_{i+1}$ $(i = 1, ...|D| - 1)$.

OPE usually preserves the order in the encrypted values and in the *plaintext* values, but generates a different distribution for values in ciphertext than the values in plaintext.

### 3.4. Paillier Cryptosystem

Paillier cryptosystem is an asymmetric key-based homomorphic encryption mechanism [20]. In this method, both the public and secret key are used in integer form. We can consider *plaintext* as $p$ and *ciphertext* as $c$, respectively. Let the public key be $Paillier_{pk}(n, g)$ and the secret key be $Paillier_{sk}(\lambda, \mu)$.

We can define the Paillier encryption and decryption by the following functions:

$c = g^p \cdot r^n \bmod n^2$

$p = L(c^\lambda \bmod n^2) \cdot \mu \bmod n$, where $L(x) = \frac{x-1}{n}$

The scheme is an additive-homomorphic cryptosystem; one can add two numbers in the encrypted form.

Suppose, $m_1$ and $m_2$ are in plaintext and the corresponding encrypted messages are $\zeta_1$ and $\zeta_2$, where $\zeta_1 = Pk_x(m_1)$ and $\zeta_2 = Pk_x(m_2)$, where $Pk_x(y)$ is a Paillier encryption function and $Pk_x$ is a Paillier public key of $x$.

Then, we can perform homomorphic addition using:

- Homomorphic Addition

$$(\zeta_1 \times \zeta_2) \bmod n^2 = Pk_x((m_1 + m_2) \bmod n)$$

### 3.5. Hadoop MapReduce

Hadoop is an open-source implementation of the MapReduce framework, maintained by the Apache Software Foundation. This framework is designed to allow users to define a MapReduce job only by specifying the map and reduce functions. In this framework, data are represented as <key, value> pairs, and computations are distributed across a shared-nothing cluster of autonomous machines. Jobs to be performed using the MapReduce framework mainly refer to two user-defined functions, called Mapand Reduce:

- Map function $Map(k_1, v_1)$ to $(k_2, v_2)$
- Reduce function $Reduce(k_2, list(v_2))$ to $list(v_3)$

The Map function (sometimes called Mapper) processes on each <key, value> pair of input data and produces intermediate <key, value> pairs. The intermediate <key, value> pairs are then sorted and grouped, associated with the same intermediate key. The Reduce function (sometimes called Reducer) takes a key and a list of values for that key, applies the processing algorithm, and generates the final result.

MapReduce is increasingly used to process massive data due to its scalability and fault-tolerance. The availability of scalable and open-source MapReduce systems, such as Hadoop [3], makes it desirable to leverage such systems for large-scale parallel skyline computation. We tried to deploy MapReduce using Apache Hadoop because of its positive features like being easy to implement, its popularity, flexibility, and fault-tolerance. We can deploy and scale it using a general purpose computer, so it is a cost-effective solution for the distributed computing environment. The algorithm design in the Apache Hadoop platform can be easily transferable to Apache Spark. It also produces the desired accurate result and efficiency in the case of the implementation of our proposed method. Therefore, we used Hadoop MapReduce to handle the data generated from multiple parties during the computation of skyline. Moreover, there were several skyline computations found that use the MapReduce framework to calculate skyline efficiently: the works in [1,4,5,21] showed that MapReduce-based parallel skyline computation is more efficient than the centralized skyline computations and can process a large amount of data.

## 4. Proposed Model

In our proposed system, we introduce a skyline computation method that can compute the skyline with the help of coordinators using multiple parties' databases with privacy and security. The participating parties never want to disclose the real attribute values of the databases; therefore, in our proposed method, each party encrypted the values before sending them to the coordinators, and the coordinators computed the skyline on the encrypted attribute values. We considered two coordinators: Coordinator 1 and Coordinator 2. Coordinator 2 computes the skyline cell-wise, and Coordinator 1 computes the final multi-party skyline. During skyline computation, we considered four types of privacies. The privacies were:

1. The privacy of the original values of attributes.
2. The privacy of the initial distribution of the values in each attribute of the multi-party databases.
3. The privacy of the original order of attribute values in each database.
4. The privacy of information about the source of data, which means the privacy of information about which data came from which party.

In our proposed model, during computation, we ensured Privacies 1, 2, and 3 were in Coordinator 2 and Privacies 1, 2, and 4 were in Coordinator 1. We also considered each party and the coordinators as a semi-honest adversary. Therefore, they can try to guess private values during computation, but never exchange any information with other parties except those permitted by the proposed method.

For efficiency, we simultaneously computed the local skyline and encryption of the local skyline in each party, concurrently executing operations in each coordinator.

Figure 2 describes the simplified block-diagram of our proposed privacy-preserving skyline computation model. Here, Party 1, Party 2, ..., Party $N$want to calculate the global skyline from their local databases without disclosing their attribute values. Coordinator 1 and Coordinator 2 calculate the global skyline from the data of each party without knowing the actual attribute values of individual databases. Our proposed algorithm consists of five steps.

1. Initialization
2. Local skyline computation, order-preserving encryption of local skyline objects, and perturbation of the original order of attribute values
3. Cell-wise candidate skyline computation distributively and concurrently in each cell
4. Global skyline computation from the cell-wise candidate skyline
5. Decryption of the global skyline

For simplicity, we provide an explanation of each step of the proposed method by two-dimensional data, as shown in Table 2. Here, Party A and Party B are two participating parties.

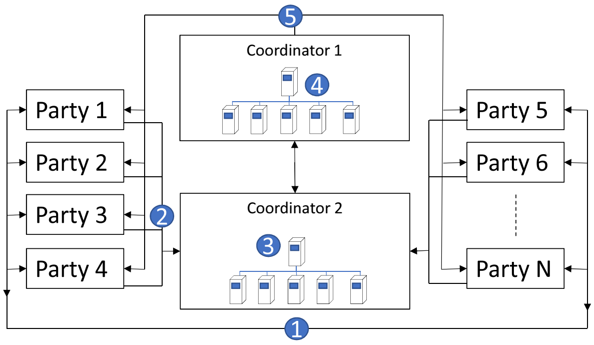


**Figure 2.** MapReduce-based multi-party secure skyline computation model.

**Table 2.** Data of Party A and Party B.

| Party A | | | Party B | | |
|---|---|---|---|---|---|
| **ID** | **Cost** | **Risk** | **ID** | **Cost** | **Risk** |
| A01 | 105 | 154 | B01 | 113 | 151 |
| A02 | 113 | 149 | B02 | 127 | 111 |
| A03 | 124 | 102 | B03 | 131 | 101 |
| A04 | 133 | 99 | B04 | 134 | 92 |
| A05 | 191 | 85 | B05 | 145 | 84 |
| A06 | 144 | 72 | B06 | 159 | 98 |
| A07 | 167 | 64 | B07 | 167 | 70 |
| A08 | 176 | 55 | B08 | 176 | 60 |
| A09 | 191 | 53 | B09 | 191 | 102 |
| A10 | 167 | 151 | B10 | 174 | 149 |
| A11 | 167 | 98 | B11 | 174 | 87 |
| A12 | 191 | 53 | B12 | 191 | 55 |

### *4.1. Initialization*

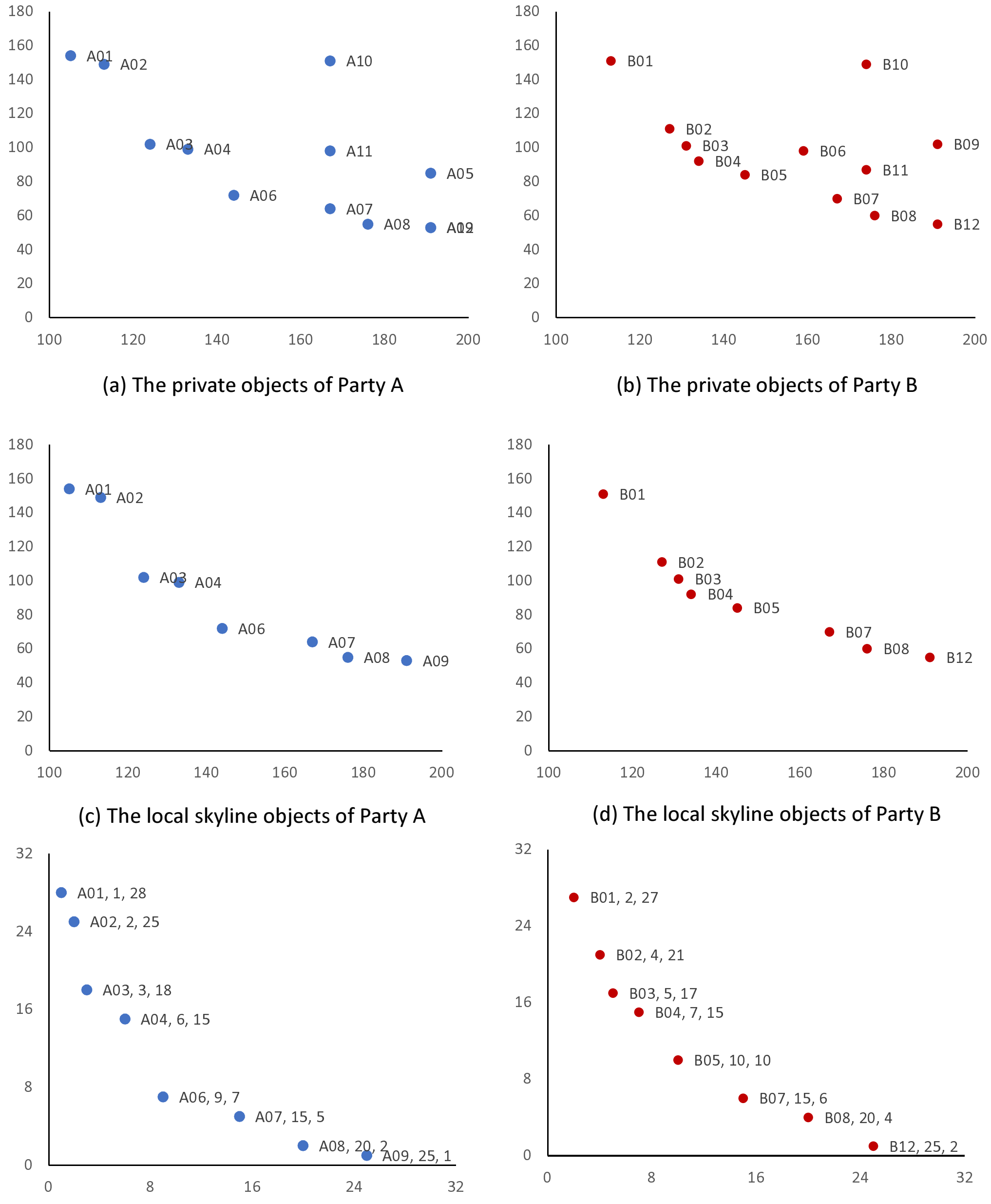
We assumed that all databases of the participating parties contained the same database schema. Therefore, all participating parties had the same number of attributes in each object, and the same amount of bits was needed in each attribute to store the values. Coordinator 1 initiates the process by sending a start signal to all parties and sends its Paillier public key ($Pk_{C1}$) to all parties with a random number $R_{C1}$. Coordinator 1 randomly selects a party. The selected party decides the OPES key for each attribute and the number of partitions they make in each attribute. Then, the selected party sends the OPE keys and partition for each attribute to all participating parties.

### *4.2. Local Skyline Computation, OPE of Original Values, and Perturbation of Original Order*

Before sending any objects to the coordinator, each party computes the local skyline, performs order-preserving encryption, and perturbs the original order. The operations are discussed in the following subsections.

### 4.2.1. Local Skyline Computation

All the parties calculate the local skyline from their private databases. As a result, all the dominated objects from the private databases of each party are filtered out. Figure 3a,b show the objects in the database of Party A and Party B. Figure 3c,d show the objects in the local skyline of Party A and Party B.



(a) The private objects of Party A

(b) The private objects of Party B

(c) The local skyline objects of Party A

(d) The local skyline objects of Party B

(e) The local skyline objects of Party A after OPE

(f) The local skyline objects of Party B after OPE

**Figure 3.** Local skyline of Party A and Party B from their private objects. OPE, Order-preserving Encryption.

### 4.2.2. OPE of the Original Attribute Values of Objects in the Local Skyline

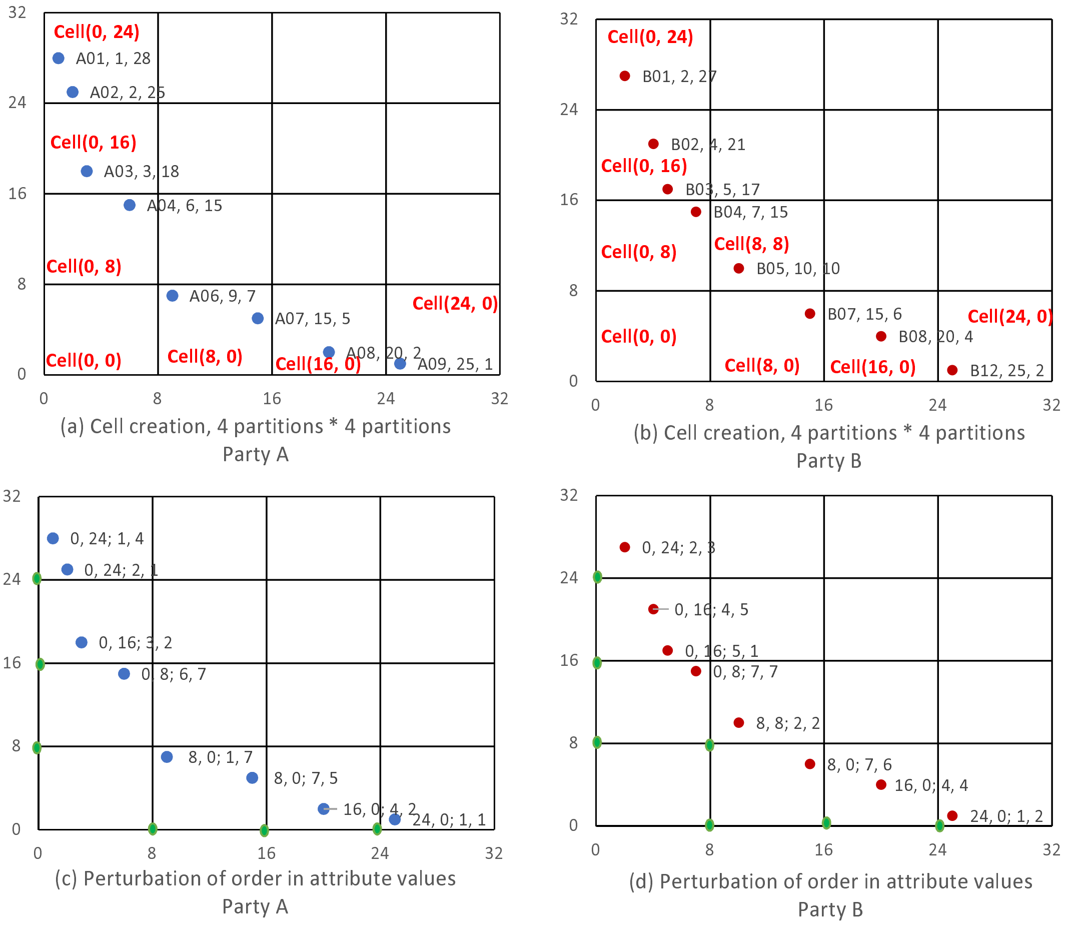
Here, every party performs order-preserving encryption for each attribute value by applying the OPE key (they get OPE keys for each attribute during initialization) for the corresponding attribute of the local skyline objects. Order-preserving encryption changes the attribute values and distribution of the values, but maintains the relative order in each attribute value. Figure 3c,d show the local skyline of objects with the original attribute values of Party A and Party B. Figure 3e,f show the order preserving encrypted attribute values of the local skyline of Party A and Party B.

### 4.2.3. Perturbation of the Original Order

OPE changes the attribute values without changing the relative order of values in each attribute. The objects in the local skyline of all parties have to be sent to Coordinator 2 for global skyline computation. If they send the values with the relative order in each attribute of objects, then the coordinator can analyze the relative position of all the objects of the parties. This is also a significant privacy and security concern.

Therefore, each party perturbs the original order of the values in each attribute before sending it to Coordinator 2. For perturbation, each party divides the object space into several cells and disguises the order of the attribute values in such a way that the order of values within a cell is maintained, but the order of the values in the different cells is not retained.

Cells are created by dividing the domain of each attribute by the corresponding partition number. Since the partition number is provided to each party during initialization, all the parties will have an equal number of cells. Figure 4 shows the cell division by applying four partitions in each attribute.



(a) Cell creation, 4 partitions * 4 partitions
Party A

(b) Cell creation, 4 partitions * 4 partitions
Party B

(c) Perturbation of order in attribute values
Party A

(d) Perturbation of order in attribute values
Party B

**Figure 4.** Cell creation and attribute value perturbation of objects in the local skyline of Party A and Party B.

Figure 4a,b exhibit that Party A and Party B have the same number of cells. We considered the minimum integer attribute values in a cell as the cell id. For example, in Figure 4a, cell(0, 8) is a cell with id (0, 8) because zero and eight are the minimum integer attribute values in this cell.

We subtracted attribute values in an object with the values in the cell id. For example, in Figure 4a, the object A01 of Party A with attribute value (1, 28) in cell(0, 24) becomes (1, 28)−(0, 24) = (1, 4). We can consider an object as <cell id><subtracted value>. For example, objects with attribute value (1, 28) in cell(0, 24) can be considered as <0, 24; 1, 4>. Figure 4c,d show the object attribute value as <cell id; value>.

The participating party encrypt the cell id of each object by the public key of Coordinator 1 and $R_{C1}$ (all parties collect $R_{C1}$ at the time of initialization). At this point, the objects become <encrypted id; values>. For example, object <0, 24; 1, 4> becomes <$Pk_{C1}$(0, 24); 1, 4>. Here, $Pk_{C1}$(0, 24) means (0, 24) is encrypted by the public key of Coordinator 1. The encrypted cell id disguises the inter-cell relative order in attribute values. All the participating parties send the value of the object with the encrypted id to Coordinator 2. Each party performs all the tasks using Algorithm 1 where Line 2 creates the divisions, Lines 5–12 generate the cell id, as well as the cell-wise attribute values of each object, and Line 13 sends the objects with an encrypted cell id to Coordinator 2.

**Algorithm 1:** Cell-wise object generation with encrypted cell id and translated attribute values.

```
Input  : $O_j (1 \leq j \leq m)$ $m$ objects with $n$ attributes each, the bit length $b_i (1 \leq i \leq n)$,
         the number of partitions $Par_i (1 \leq i \leq n)$, the Paillier public key of Coordinator 1
         $Pk_{C1}$, random number $r_1$
Output:  Cell-wise objects with encrypted cell id and translated attribute values

1  for $i \leftarrow 1$ to $n$ do
2      $div_i = (2^{b_i} + 1)/Par_i$;
3  end
4  Define P as a temporary object;
5  for $j \leftarrow 1$ to $m$ do
6      $cellid$=null
7      for $i \leftarrow 1$ to $n$ do
8          $P(i) = O_j(i) \bmod div_i$     /* here, $O_j(i)$ means the $i$-th attribute of the $j$-th
               object                                                                    */
9          $cell_i = O_j(i) - P(i)$     /* cell id generation in the $i_{th}$ attribute         */
10         $O_j(i) = P(i)$     /* value of the $i^{\text{th}}$ attribute of the $j$-th object         */
11         $cellid$ = concatenate ($cellid$, $cell_i$)
12     end
13     send value <$Pk_{C1}(cellid, R_1)$><$O_j$> to Coordinator 2     /* encryption of cell id by
           the                                Paillier public key of Coordinator 1
           using $R_1$                                                                    */
14 end
```

### 4.3. Cell-Wise Candidate Skyline Computation Distributively and Concurrently in Each Cell

Coordinator 2 receives objects as $< encrypted\_cell\_id; attributes\_values >$ from all parties. Since the ids of the objects are encrypted, it is impossible for Coordinator 2 to guess which id corresponds to which cell. As we know, relative order is sufficient for a dominance check between two objects; as a result, we can calculate the skyline from the relative order. The Coordinator 2 could compute the skyline of objects in each cell because objects' attribute values in a particular cell maintain their relative order. Coordinator 2 uses the mapper and reducer functions to execute the skyline in each cell concurrently. Figure 5a shows the objects received from all the parties. Figure 5b shows the split of the encrypted cell id of the object as the mapper-key and the cell-wise attribute values of objects as the mapper-value. Figure 5c shows cell-wise objects. Figure 5d shows the candidate skyline objects in each cell after the reducing operation.

Coordinator 2 performs $Pk_{C1}$(cell id) + $Pk_{C1}$(value) for each object in the cell-wise skyline. Since it is a homomorphic addition, $Pk_{C1}$(cell id) + $Pk_{C1}$(value) = $Pk_{C1}$(cell id + value) (Figure 6). After performing such homomorphic addition, it sends all the cell-wise skyline objects to Coordinator 1. Figure 6 shows the homomorphic addition process.

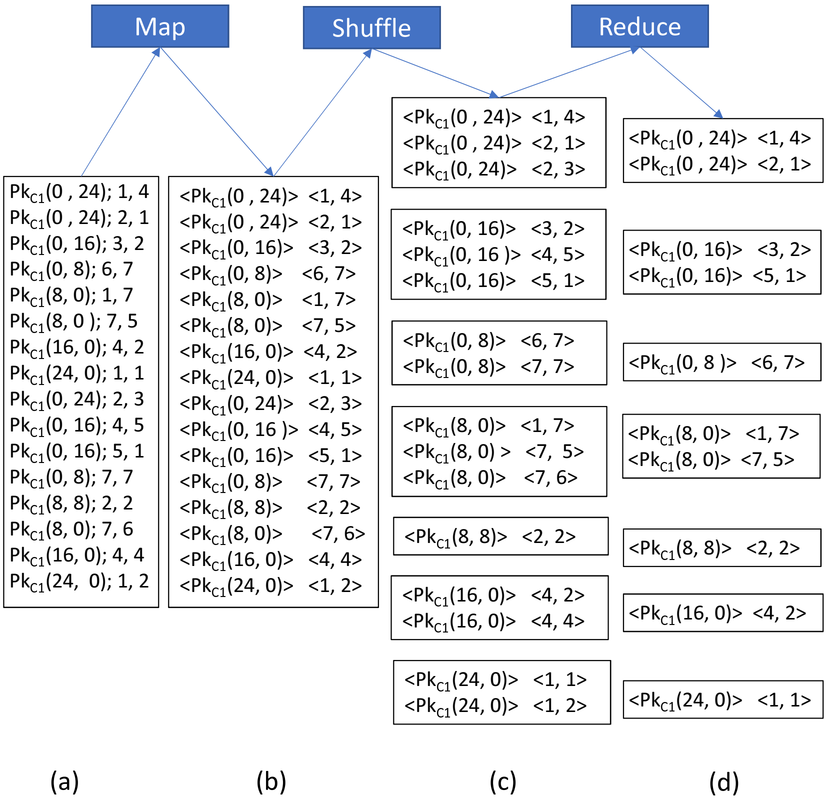


**Figure 5.** MapReduce-based cell-wise skyline computation. Here, $Pk_{C1}(x, y)$ means $(x, y)$ is encrypted by Coordinator 1's public key.

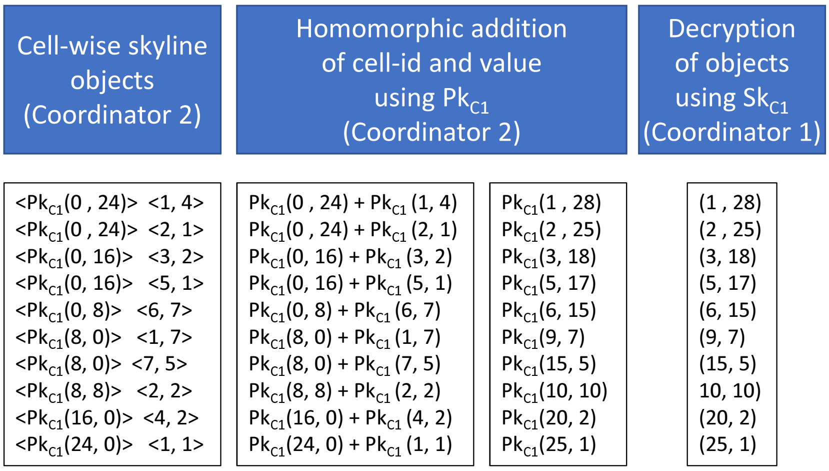

Cell-wise skyline objects (Coordinator 2)
Homomorphic addition of cell-id and value using Pk$_{C1}$ (Coordinator 2)
Decryption of objects using Sk$_{C1}$ (Coordinator 1)
<Pk$_{C1}$(0 , 24)> <1, 4>
<Pk$_{C1}$(0 , 24)> <2, 1>
<Pk$_{C1}$(0, 16)> <3, 2>
<Pk$_{C1}$(0, 16)> <5, 1>
<Pk$_{C1}$(0, 8)> <6, 7>
<Pk$_{C1}$(8, 0)> <1, 7>
<Pk$_{C1}$(8, 0)> <7, 5>
<Pk$_{C1}$(8, 8)> <2, 2>
<Pk$_{C1}$(16, 0)> <4, 2>
<Pk$_{C1}$(24, 0)> <1, 1>
Pk$_{C1}$(0 , 24) + Pk$_{C1}$ (1, 4)
Pk$_{C1}$(0 , 24) + Pk$_{C1}$ (2, 1)
Pk$_{C1}$(0, 16) + Pk$_{C1}$ (3, 2)
Pk$_{C1}$(0, 16) + Pk$_{C1}$ (5, 1)
Pk$_{C1}$(0, 8) + Pk$_{C1}$ (6, 7)
Pk$_{C1}$(8, 0) + Pk$_{C1}$ (1, 7)
Pk$_{C1}$(8, 0) + Pk$_{C1}$ (7, 5)
Pk$_{C1}$(8, 8) + Pk$_{C1}$ (2, 2)
Pk$_{C1}$(16, 0) + Pk$_{C1}$ (4, 2)
Pk$_{C1}$(24, 0) + Pk$_{C1}$ (1, 1)
Pk$_{C1}$(1 , 28)
Pk$_{C1}$(2 , 25)
Pk$_{C1}$(3, 18)
Pk$_{C1}$(5, 17)
Pk$_{C1}$(6, 15)
Pk$_{C1}$(9, 7)
Pk$_{C1}$(15, 5)
Pk$_{C1}$(10, 10)
Pk$_{C1}$(20, 2)
Pk$_{C1}$(25, 1)
(1 , 28)
(2 , 25)
(3, 18)
(5, 17)
(6, 15)
(9, 7)
(15, 5)
10, 10)
(20, 2)
(25, 1)

**Figure 6.** Homomorphic addition by Coordinator 2 and decryption of global skyline objects by Coordinator 1. Here, $Pk_{C1}(x, y)$ means $(x, y)$ is encrypted by Coordinator 1's public key.

### 4.4. Global Skyline Computation from the Cell-Wise Candidate Skyline

Coordinator 1 decrypts all the objects attribute values by its secret-key ($Sk_{C1}$) (Figure 6) and computes the global skyline using quadtree-based [6] skyline computation. The quadtree-based method divides the objects using the quadtree structure and concurrently computes the skyline in each node using the MapReduce framework. After computation of the skyline, skyline objects are sent to all the parties. Since the cell id and values are added together before sending each object to Coordinator 1, the coordinator can know the original order of all objects. On the other hand, the objects are directly coming from Coordinator 2 (not from other parties) to Coordinator 1. Therefore, Coordinator 1 has no knowledge about which objects came from which party; thus, Coordinator 1 is never able to analyze the relative order of the objects of each party.

### *4.5. Decryption of the Global Skyline*

Dominated objects are pruned out in each phase, and finally, only the non-dominated objects are retained. We know that the skyline is the set of non-dominated objects; as a result, all the parties obtain the desired global skyline. Each party receives the global skyline where each attribute value is encrypted with the corresponding OPE key. Then, they decrypt the attribute values using the OPE key for each attribute and obtain the global skyline in plaintext form. Table 3 shows the decrypted values of the global skyline.

**Table 3.** Decrypted objects of the global skyline.

| Global Skyline (OPE) | | Global Skyline (Plaintext) | |
|---|---|---|---|
| 1 | 28 | 105 | 154 |
| 2 | 25 | 113 | 149 |
| 3 | 18 | 124 | 102 |
| 5 | 17 | 131 | 101 |
| 6 | 15 | 133 | 99 |
| 9 | 7 | 144 | 72 |
| 15 | 5 | 167 | 64 |
| 20 | 2 | 176 | 55 |
| 25 | 1 | 191 | 53 |

## 5. Scalability and Application of the Proposed Method

Local skyline calculation, OPE, and perturbation are performed concurrently in each participating party, so the system is scalable as the number of participating parties grows. We used the Hadoop MapReduce distributed computing system in Coordinator 1 and Coordinator 2. Since MapReduce is a highly scalable and distributed system, our system is scalable for a higher number of participating parties or a significant volume of data.

We may extend our work where there are multi-party databases and they want to perform secure computation without disclosing the actual values in the data. The inferring fine-grained urban flows [22] can be extended to multi-party secure urban traffic flow analysis. Spatiotemporal computation using ST(Spatiotemporal)-Hadoop [23] can be extended for multi-party secure computation. K-nearest skyline query in spatiotemporal databases [24] may be deployable as secure multi-party computation in our proposed model. In the urban area, if there are multiple water quality testing services, then our method can extend the work proposed by Ye Liu et al. [25] for secure analysis of data from multiple water-quality testing service databases.

## 6. Privacy and Security

To build the proposed secured privacy-preserving skyline computation, we utilized order-preserving encryption and Paillier encryption to meet the secure computation and data privacy requirements. As per the semi-honest adversary model [18], no party is allowed to share any data with any other party violating the protocol assigned to each party.

Since no party shares their private data with other parties, so no party has any idea about data from other parties. Only data are shared with Coordinator 1 and Coordinator 2, so we have to ensure the privacy of data in Coordinator 1 and Coordinator 2.

In the case of Coordinator 2, the values are encrypted by order-preserving encryption, and orders are perturbed by encrypting each cell id. It can obtain the actual order of values if it is successful at rearranging the cells correctly. Let us consider that there are M attributes in the objects and each attribute is divided into $N$ partition; then, we can arrange $N^M$ cells in $(N^M)!$ different ways. Therefore, the probability of correctly organizing all cells will be $1/(N^M)!$. Moreover, Coordinator 2 has no mechanism to determine the correct arrangement of cells because each arrangement produces similar kinds of results, which are not distinguishable from each other. Therefore, it is quite impossible for Coordinator 2 to guess the correct order. Thus, the privacy of the values in each attribute and the privacy of the relative order in each attribute are preserved during computation in Coordinator 2.

On the other hand, in the case of Coordinator 1, it does not know the actual values of each attribute. However, it knows the relative order of the values in each attribute of limited objects, because a considerable number of objects are pruned out at the time of local skyline computation and cell-wise skyline computation. Besides, Coordinator 1 does not know which objects came from which party. Therefore, it is never able to analyze the relative order of the objects of multiple parties.

## 7. Theoretical Analysis of the Proposed Method

In this section, we discuss the theoretical comparison of our method with the existing methods. In our proposed system, the computational complexity of the secure skyline depended on the following operations:

1. The time required for the calculation of the local skyline, OPE, perturbation, and cell-wise value generation.
2. The time needed for the calculation of the cell-wise candidate skyline.
3. The time needed for the calculation of the global skyline.

All the participating parties concurrently calculate the local skyline, OPE, perturbation, and cell-wise values. Thus, if the number of parties increases, the time required for these operations does not vary. On the other hand, Coordinator 2 calculates the candidate skylines concurrently in every cell, prunes a considerable number of dominated object, and improves the efficiency. Moreover, in global skyline computation, it maps all the data into a quadtree, then calculates the final skyline simultaneously in each node of the quadtree, thus improving the efficiency. Besides, during the calculation, the coordinators do not need to exchange any information with the participating parties or other coordinators; this also enhances the overall efficiency.

The method proposed in [9] securely compared the pair-wise objects' attributes and computed the dominance of objects between two parties. In their method, they did not consider the coordinator for computing the multi-party skyline. Therefore, it cannot handle multiple parties simultaneously; it can only compute the skyline between two parties. Therefore, for n parties, it needs ${}^nC_2 = \frac{n(n-1)}{2}$ two-party skyline computations to compute the n-party skyline. Moreover, it requires secure comparison in each object attribute for the dominance check, which is also time consuming and needs several rounds of data exchange between the parties to compare each attribute value.

On the other hand, the complexity of our proposed method depends on the total local skyline objects of multiple parties, not on the number of participants. Moreover, it does not require any rounds of data exchange between any pair of participating parties and also compares objects directly on the encrypted values for the dominance check.

For the method proposed in [2], all the participating parties constructed their database objects' order with the help of a semi-honest third party, called the *coordinator*. To generate the join order in each dimension, every digit in the dimension needed one round of MapReduce operation in the coordinator. For example, if there were $D$ dimensions and each dimension contained $M$ digits, $D * M$ MapReduce rounds were required for order generation and one extra round for producing the skyline. In our recent work, wee only needed two MapReduce rounds.

Our previous work [10] improved the efficiency, compared to other secure skyline computation frameworks, but it required sharing an encrypted substitution vector before secure computation of the skyline. For 32-bit integer values and two equal 16-bit partitions, it needed $2 \times 2^{16}$ 32-bit integer values as a substitution vector and should be shared among the parties before skyline computation.

## 8. Experimental Analysis of the Proposed Method

### *8.1. Experimental Setup and Datasets*

Here, we discuss the performance and efficiency of our proposed method. For participating parties, we used computers with a fourth-generation Intel® Core™i7, 3.4-GHz CPU, and 8 GB main memory, running on the 64-bit Microsoft Windows 10 Enterprise edition operating system. For Coordinator 1 and Coordinator 2, we configured a cluster of two commodity PCs in a high-speed Gigabit Ethernet network, each of which had an Intel Core 2 Duo E8500 3.16-GHz CPU and 8 GB memory. We compiled the source codes under Java V8. We used Hadoop Version 2.5.2 and 64-bit Cent-OS 7. We set the replication parameter of the Hadoop cluster to two.

We evaluated our proposed privacy-preserving secure skyline algorithm in a multi-party distributed environment on synthetic datasets. As benchmark databases, we used the skyline benchmark data generator proposed by Borzsonyi et al. [11], in which we could generate three types of synthetic data distributions: correlated, anticorrelated, and independent.

### *8.2. Analysis of Our Proposed Method for Different Data Distributions*

We know that the standard way to analyze the skyline computation is how the complexity of the calculation varies with correlated, anticorrelated, and independent data distributions. Most of the related work used these three distributions to analyze the complexity of skyline computation. Usually, correlated data generate less non-dominated objects in skyline computation, thus requiring less time. On the other hand, anticorrelated data generate most non-dominated objects in skyline computation; therefore, they require the longest time. However, independent data produce non-dominated objects in between the number of non-dominated objects generated by correlated and anticorrelated data. We also want to verify how the complexity of our proposed algorithm varied with different data distributions. For this experiment, we varied each participating parties' object numbers from 10–50 k, each object containing two attributes, and values were in a 32-bit integer. We also considered 30 partitions per attribute.

According to Figure 7, We found that the skyline computation was more efficient for the correlated dataset and less efficient for the anticorrelated dataset. However, the performance for the independent dataset lied in between the performance for the anticorrelated and correlated datasets. For correlated data, a paramount number of objects are pruned out during local skyline computation. In the case of independent data, the average number of objects is pruned out. For anticorrelated data, less pruning occurred. We also found that the time needed for skyline increased when the number of objects per party increased because it needed a dominance check for each object of one party with the objects of the other parties.

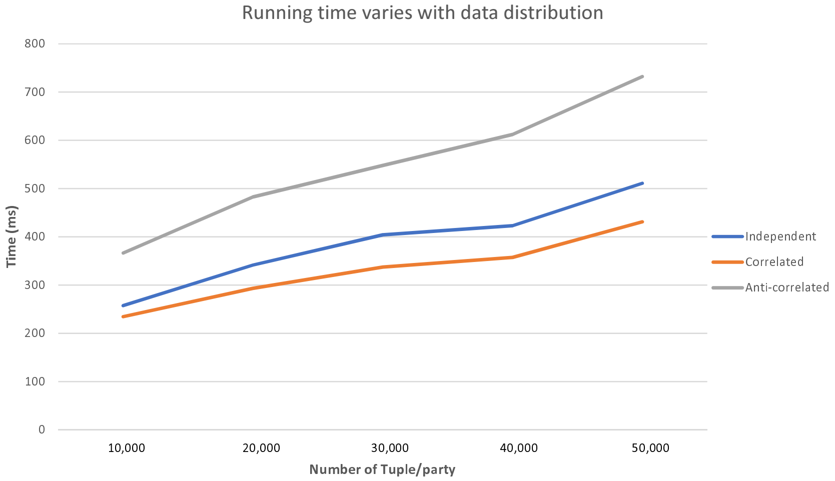


**Figure 7.** Running time varies with data distribution (attribute: 2, partitions: 30/attribute, value: 32-bit).

### *8.3. Analysis of Our Proposed Method with Variation in Object Dimensions*

Another way to analyze the complexity of skyline computation is how the computation time varies with the variation in object dimensions. We know skyline computation requires comparison in each dimension to compute non-dominating objects from the dataset. Therefore, the complexity of skyline computation increases with the increase in the object dimensions. Here, we discuss how our proposed skyline computation time varied with the variation in the objects' dimensions.

Figure 8 shows how time varied with the variation of the data dimension for computing skyline. For this experiment, we varies data dimensions from 2–6 and tried to find out how the computation time varied with the variation in the data dimensions.

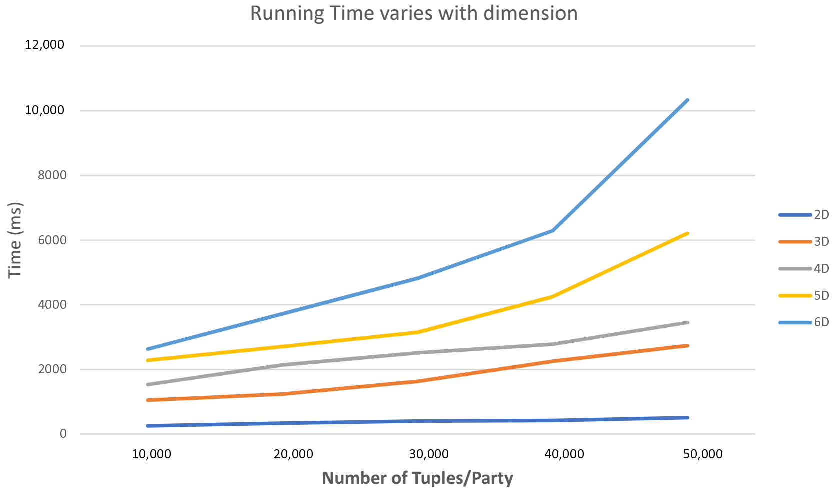


**Figure 8.** Running time varies with objects attributes (distribution: independent, partitions: 30/attribute).

Since the number of required attribute partitions along with the number of comparisons and the amount of qualified local skyline objects increases with the object dimension, the process execution time additionally increases. Therefore, in Figure 8, we find that the running time increased when object dimensions grew. We also found that the time needed for skyline increased when the number of objects per party increased because it needed a dominance check for each object of one party with the objects of the other parties.

### *8.4. Comparison of the Proposed Method with the Encrypted Substitution Vector-Based Method*

Recently, an encrypted substitution vector (ESV)-based [10] method has been proposed for multi-party distributed skyline computation, and the method is efficient compared to other contemporary multi-party secure skyline computations. In this section, we compare our proposed method with the ESV method. For this experiment, we considered that each party had 10–50 k objects, each object containing three attributes, and the values in each attribute contained a 32-bit value. We also considered 30 partitions per attribute. For the ESV method, we considered an 11-bit bit-slice length for creating the encrypted substitution vector.

The ESV-based method requires sharing an encrypted substitution vector among the parties before secure computation of the skyline. Besides, it does not consider the concurrent computation of the skyline in the coordinator. In our proposed method, the exchange of information among the parties was only OPE keys for each attribute and the number of partitions in each attribute. We concurrently executed the operation in each phase to compute the global skyline. Thus, the comparison results show that our proposed method needed less time than the ESV-based method. Figure 9a–c show that our proposed method outperformed the ESV-based method for the independent, correlated, and anticorrelated datasets.

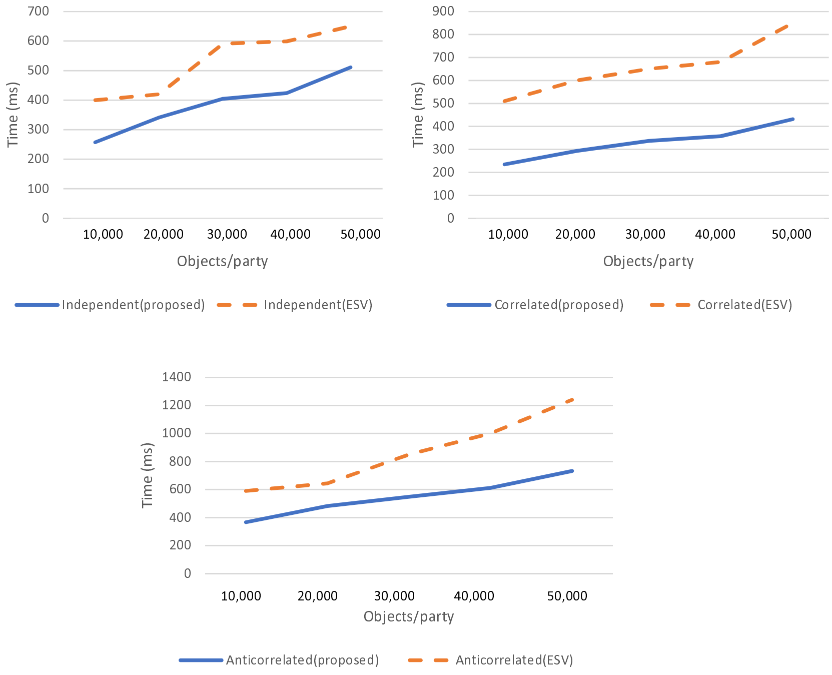


**Figure 9.** Running time comparison with ESVand the proposed method in different data distributions (attribute: 2, partitions: 30/attribute, value: 32-bit, bit-slice length: 11-bit, slices/attribute:3). ESV, encrypted substitution vector.

### 8.5. Comparison of the Proposed Method with Variation in the Number of Participating Parties

We discussed that our method was efficient even for the increases in the number of parties. Here, we varied the number of participating parties and determined the time required for multi-party secure skyline computation. For this experiment, we considered each party to have 50 k objects, each object containing two attributes, and each attribute value was a 32-bit unsigned integer. We also considered 30 partitions per attribute.

Figure 10 shows the time required for skyline computation with variation in the number of participating parties. The number of parties varied from 2–8. Since our proposed method did not share data among the parties during computation and did not need pair-wise computation, thus the computation time grew linearly with the growth in the number of participating parties. Therefore, the time required for our proposed method in Figure 10 showed a steady increase in time with an increase in participating parties' databases.

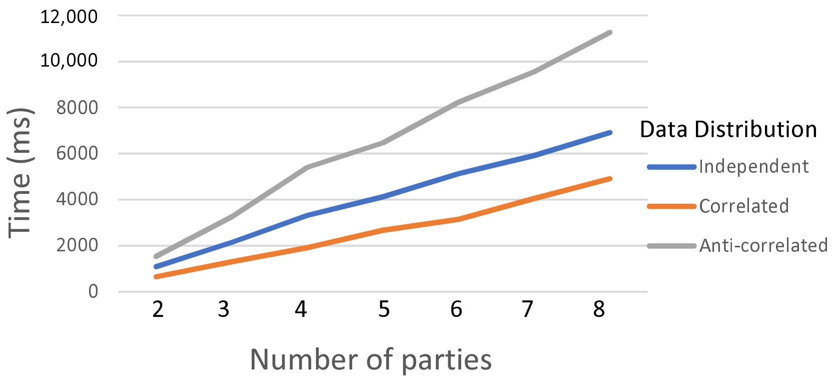


**Figure 10.** Running time varies with participating parties (attribute: 2, partition: 30/attribute).

## 9. Conclusions

In our proposed method, we efficiently handled multi-party data without disclosing the original values in the attributes during the computation of the secure skyline. We simultaneously performed local skyline computation and encryption of local skyline objects in each party. Moreover, Coordinator 1 and Coordinator 2 parallelly executed operations for computing the global skyline. We also kept a minimum exchange of information among other parties during the computation of the skyline. Thus, our proposed method showed better performance. Both of the coordinators used the MapReduce framework; therefore, our approach can handle big data from multi-party reliably and cost-effectively.

**Author Contributions:** Conceptualization, S.A. and Y.M.; formal analysis, S.A., A.Z., M.A.S., and Y.M.; investigation, S.A.; methodology, S.A.; supervision, Y.M.; validation, S.A., M.Q., A.Z., and M.A.S.; writing, original draft, S.A; writing, review and editing, S.A., M.Q., C.L., K.M.R.A., and Y.M.

**Funding:** This work is supported by KAKENHI (16K00155, 17H01823), Japan. Saleh Ahmed and Mahboob Qaosar are supported by the Japanese Government MEXT Scholarship. Asif Zaman and Md. Anisuzzaman Siddique were supported by the Japanese Government MEXT Scholarship.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Mullesgaard, K.; Laurits Pederseny, J.; Lu, H.; Zhou, Y. Efficient Skyline Computation in MapReduce. In Proceedings of the 17th International Conference on Extending Database Technology (EDBT), Athens, Greece, 24–28 March 2014; pp. 37–48.
2. Zaman, A.; Siddique, M.A.; Annisa; Morimoto, Y. Secure Computation of Skyline Query in MapReduce. In *Advanced Data Mining and Applications*; Li, J., Li, X., Wang, S., Li, J., Sheng, Q.Z., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 345–360.
3. Apache. Welcome to ApacheTMHadoop. Available online: http://hadoop.Apache.org (accessed on 1 May 2019).
4. Ryu, H.C.; Jung, S. MapReduce-based Skyline Query Processing Scheme Using Adaptive Two-level Grids. *Clust. Comput.* **2017**, *20*, 3605–3616, doi:10.1007/s10586-017-1203-y.
5. Zhang, J.; Jiang, X.; Ku, W.; Qin, X. Efficient Parallel Skyline Evaluation Using MapReduce. *IEEE Trans. Parallel Distrib. Syst.* **2016**, *27*, 1996–2009, doi:10.1109/TPDS.2015.2472016.
6. Park, Y.; Min, J.K.; Shim, K. Parallel Computation of Skyline and Reverse Skyline Queries Using Mapreduce. *Proc. VLDB Endow.* **2013**, *6*, 2002–2013, doi:10.14778/2556549.2556580.
7. Liu, J.; Yang, J.; Xiong, L.; Pei, J. Secure Skyline Queries on Cloud Platform. In Proceedings of the 2017 IEEE 33rd International Conference on Data Engineering (ICDE), San Diego, CA, USA, 19–22 April 2017; pp. 633–644, doi:10.1109/ICDE.2017.117.
8. Hua, J.; Zhu, H.; Wang, F.; Liu, X.; Lu, R.; Li, H.; Zhang, Y. CINEMA: Efficient and Privacy-Preserving Online Medical Primary Diagnosis with Skyline Query. *IEEE Internet Things J.* **2018**, *6*, 1450–1461, doi:10.1109/JIOT.2018.2834156.
9. Liu, X.; Lu, R.; Ma, J.; Chen, L.; Bao, H. Efficient and privacy-preserving skyline computation framework across domains. *Future Gener. Comput. Syst.* **2016**, *62*, 161–174, doi:10.1016/j.future.2015.10.005.
10. Qaosar, M.; Zaman, A.; Siddique, M.A.; Annisa.; Morimoto, Y. Privacy-Preserving Secure Computation of Skyline Query in Distributed Multi-Party Databases. *Information* **2019**, *10*, 119, doi:10.3390/info10030119.
11. Borzsonyi, S.; Kossmann, D.; Stocker, K. The skyline operator. In Proceedings of the IEEE International Conference on Data Engineering (ICDE), Heidelberg, Germany, 2–6 April 2001; pp. 421–430.
12. Chomicki, J.; Godfrey, P.; Gryz, J.; Liang, D. Skyline with Presorting. In Proceedings of the IEEE International Conference on Data Engineering (ICDE), Bangalore, India, 5–8 March 2003; pp. 717–719.
13. Papadias, D.; Tao, Y.; Fu, G.; Seeger, B. Progressive skyline computation in database systems. *ACM Trans. Database Syst.* **2005**, *30*, 41–82.
14. Kossmann, D.; Ramsak, F.; Rost, S. Shooting stars in the sky: An online algorithm for skyline queries. In Proceedings of the International Conference on Very Large Data Bases (VLDB), Hong Kong, China, 20–23 August 2002; pp. 275–286.

15. Veugen, T.; Blom, F.; de Hoogh, S.J.A.; Erkin, Z. Secure Comparison Protocols in the Semi-Honest Model. *IEEE J. Sel. Top. Signal Process.* **2015**, *9*, 1217–1228, doi:10.1109/JSTSP.2015.2429117.
16. Samanthula, B.K.K.; Chun, H.; Jiang, W. An Efficient and Probabilistic Secure Bit-decomposition. In Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security, Hangzhou, China, 8–10 May 2013; ACM: New York, NY, USA, 2013; pp. 541–546, doi:10.1145/2484313.2484386.
17. Hose, K.; Vlachou, A. A survey of skyline processing in highly distributed environments. *VLDB J.* **2012**, *21*, 359–384, doi:10.1007/s00778-011-0246-6.
18. Hazay, C.; Lindell, Y. Definitions. In *Efficient Secure Two-Party Protocols: Techniques and Constructions*; Springer Berlin Heidelberg: Berlin/Heidelberg, Germany, 2010; pp. 19–49, doi:10.1007/978-3-642-14303-8_2.
19. Agrawal, R.; Kiernan, J.; Srikant, R.; Xu, Y. Order Preserving Encryption for Numeric Data. In Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, 13–18 June 2004; pp. 563–574.
20. Paillier, P. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In Proceedings of the Advances in Cryptology–Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)'99, Prague, Czech Republic, 2–6 May 1999; Stern, J., Ed.; Springer Berlin Heidelberg: Berlin, Heidelberg, 1999; pp. 223–238.
21. Siddique, M.A.; Tian, H.; Morimoto, Y. Distributed Skyline Computation of Vertically Splitted Databases by Using MapReduce. In Proceedings of the International Conference on Database Systems for Advanced Applications, Bali, Indonesia, 21–24 April 2014; pp. 33–45.
22. Liang, Y.; Ouyang, K.; Jing, L.; Ruan, S.; Liu, Y.; Zhang, J.; Rosenblum, D.S.; Zheng, Y. UrbanFM: Inferring Fine-Grained Urban Flows. *arXiv* **2019**, arXiv:1902.05377.
23. Alarabi, L.; Mokbel, M.F.; Musleh, M. ST-Hadoop: A MapReduce framework for spatio-temporal data. *GeoInformatica* **2018**, *22*, 785–813, doi:10.1007/s10707-018-0325-6.
24. Huang, Y.K.; He, Z.H. Processing continuous K-nearest skyline query with uncertainty in spatio-temporal databases. *J. Intell. Inf. Syst.* **2015**, *45*, 165–186, doi:10.1007/s10844-014-0344-1.
25. Liu, Y.; Zheng, Y.; Liang, Y.; Liu, S.; Rosenblum, D.S. Urban Water Quality Prediction Based on Multi-task Multi-view Learning. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, New York, NY, USA, 9–15 July 2016; AAAI Press: Menlo Park, CA, USA, 2016; pp. 2576–2582.