Study on Privacy-preserving Data Manipulation and Secure Computation of Skyline Objects on MapReduce

(プライバシー保護データ操作およびスカイラインオブジェクトの安全な MapReduce 計算法に関する研究)

by

Saleh Ahmed

A dissertation submitted

Graduate School of Engineering, Hiroshima University

in partial fulfillment of the requirements for the degree of

Doctor of Engineering

in

Information Engineering

under supervision of

Yasuhiko Morimoto

Department of Information Engineering

Graduate School of Engineering

Hiroshima University, Japan

September, 2019

[This page was intentionally left blank]

Dissertation Summary

Database systems commonly control the access in the database to regulate authorization of confidential data. This process preserves the privacy of sensitive information, and the provided data is obtained by utilizing the required database operation interfaces. However, access control for vital and private data is often insufficient. Attacks against computer systems have confirmed that the security of data can be compromised, if an unlawful user can get access to the data files generated by the database management system, avoiding the access constraint mechanism of a database completely. For example, the Toronto Star issued an article. The article reports an occurrence where certain bank sold old disk in eBay without deleting the potential private data of the hundreds of clients. The worldwide privacy law does not support a disk holding the records of several hundred clients was being sold on eBay. So, the privacy of client data is a vital issue. As a result, organizations must encrypt the data before storing it in the drive.

People are concern about the privacy of sensitive data, such as salary, merit positions, tender evaluation, and so on, stored in a database. Several privacy preserving methods have been proposed. On the contrast, such encryption degrades the performance of the database operations. This degradation occurs due to decryption of values in the first place, then execution of the operations. Order-preserving encryption system (OPES) may solve these issues and improve the performance degradation issues. However, the order of numeric values itself is sensitive information. In such a case, it is necessary to hide the order information as well as solve the performance degradation issues.

The author proposed three methods which run on the top of OPES able to hide the order information in the values as well as solve the performance degradation issues.

On the contrary, every day, different computing organizations generate a huge amount

of information. Such a massive amount of data is accountable for information overwhelm issues. Various related works to retrieve valuable information from large data have been studied as a solution to the matter. The essential fundamental operation of information selection is to gather a little number of objects that represent the whole data from a big database. Skyline Query in one of the popular tools to select representative objects from large scale databases.

Skyline query that is also appreciated as a successful information retrieval method has been successfully utilizing to filter out dominated objects. Skyline query usually utilized to retrieve objects that are good for all organizations whose evaluation mechanism are not alike. However, it may generate a large number or a very few numbers of objects. Moreover, whenever the organizations want to get the desired result of the skyline query, it is essential to reveal the attribute values of the objects in the datasets. In various situation, to calculate the skyline query, it needs to reveal valuable, sensitive information. The author introduced a skyline object picking tool in this dissertation that accumulates the favorite skyline objects for all the organizations; meantime, it also guarantees the privacy of sensitive attribute value throughout the process of skyline calculation.

Recently, people frequently have to retrieve important objects using mobile devices like a smartphone or phablets or tablets. In such a situation, it is difficult to explain complex query requirements like top-k query evaluation function. Clients are willing to get desired objects by defining only keywords. The proposed system must be serviceable for such circumstances. To achieve the query as mentioned above, the author used the skyline query function. To handle potential big data", The author proposed a distributed algorithm in MapReduce framework to calculate the skyline query. For big data processing, MapReduce is a very popular, distributed open-source computing framework. The proposed method utilizes the MapReduce framework to manage the large-scale database.

For calculating a skyline query in conventional distributed algorithms, the attribute

values of the individual object of a local database must be revealed to other organizations. Nowadays, people are concern about the privacy of their data; as a result, such revelations of private data in traditional distributed schemes are intolerable. In the proposed scheme, the security and privacy of the distributed algorithm are improved by the author in such a way that the confidentiality of the data during the processing of the skyline query remained intact. A novel and efficient strategy is introduced by the author to calculate the skyline from data of multiple organization in distributed computing condition without revealing the local private attribute values of objects to other organizations.

The author investigates the background of the problem and provides the introduction of the problem in **Chapter 1**. Then, the literature reviews on related work of the dissertation are provided in **Chapter 2**. After that, the author divides this dissertation into different parts. In the following section of this dissertation, the author discusses the semi-order preserving encryption. Conventional encryption techniques encrypt the database values, but query execution on encrypted values is a severe performance degradation issues. Order-preserving encryption is introduced to solve this problem. But in several cases, the order in the data is also a security concern. Therefore semi-order preserving encryption effectively hides the order information and also solve the performance degradation issues. The author discusses semi-order preserving encryption in detail in **Chapter 3**. In the **Chapter 4** of this dissertation, the author discusses a novel scheme to compute the skyline query in a secured manner in a distributed way on MapReduce. The author has considered the circumstances where the owner of the dataset is multi-party rather than a private entity. They desire to compute the skyline query result but never willing to reveal the attribute values during computation. The individuals do not desire to disclose attribute value as the values may be considered as sensitive information. The author introduced an efficient solution to settle such circumstances and compute the skyline query without disclosing any attribute values of the organizations. The suggested algorithm has used MapReduce

programming infrastructure to guarantee its capacity to handle big data in a distributed manner.

Finally, a concluding study with a future guideline for enhancing the work has been given in **Chapter 5**.

Contents

Page

List of Figures

List of Tables

Life is really simple, but we insist on making it complicated.

Confucius

Acknowledgments

At first, I would like to declare my faithful praise to Allah for providing me with the chance to accomplish my research work. I desire to declare my sincere honour to my supervisor prof. YASUHIKO MORIMOTO, Graduate School of Engineering, Hiroshima University, Japan for allowing me to explore new concepts in the field of knowledge exploration and data mining. Without his generous spirit and unique guidelines, it was not viable for me to accomplish this work. His superior wisdom, motivation and above all diligence expertise in this field give me many possibilities to acquire new views to build my carrier as a researcher.

I am grateful to my Sub-advisor prof. SATOSHI FUJITA for his wisdom and the proper suggestion that help me throughout the period doctoral program and also want to show my heartiest gratitude to prof. KOJI EGUCHI for his guidelines during the last period of my doctoral program.

I am thankful to the Japanese Administration for giving the MEXT scholarship to continue my study. I am grateful to Bangabandhu Sheikh Mujibur Rahman Science and Technology University, Goplaganj, Bangladesh that has allowed me a study leave to follow the Doctoral program at Hiroshima University.

Specific gratitude to all my co-workers and lab members for their assistance and contributions during my research. Especially I'm thankful for DR. MD. ANISUZZAMAN SIDDIQUE AND DR. ASIF ZAMAN for their generous assistance and cooperation since the opening of my research work. I am also grateful to all of my countrymates for their cooperation, care, and emotional bonds throughout my stay in Japan.

Last but not least, I am profoundly thankful to my family members for their appreciation, love, care, and support during the research work.

[This page was intentionally left blank]

Chapter 1

Introduction

Daily, different sources generate enormous amounts of data. Such vast quantities of data can be utilized to evaluate entities and predict their actions more diligently. With the growing market for big-data processing and cloud computing, numerous organizations and database designers currently highlight the protection and confidentiality of sensitive information such as worker wages, age, and expense. Such sensitive data are often aggregated in a database. Some encryption systems have been introduced for protecting the privacy of sensitive data [43, 10]. Still, such methods degrade the performance of query execution completed on the encrypted database; a notable amount of time is spent for decrypting each record before computing conditional operations given as a query parameter. To address this issue, several order-preserving encryption scheme (OPES) and some of its variants have been proposed in few articles [3, 8, 9, 10, 31, 16, 29].

The enhancement of the performance of the execution of database queries can be done by maintaining the index knowledge of the original numeric values. However, in several database applications, the ordered record of numerical data itself is regarded as sensitive data. For illustration, some schools may think the merit status of a student as confidential information that should not be exposed to the public. In this concern, the author introduced

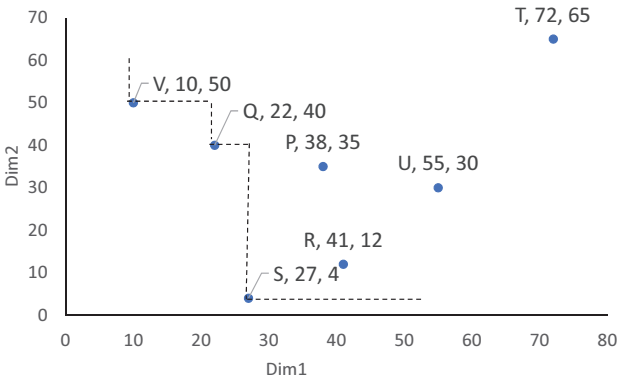
semi-order preserving techniques, which perturb the initial order index of sensitive data to enhance data privacy without degrading the performance of comparison operations over encrypted data.

The current digital world, people produce a huge amount of data every day. These huge amounts of data need to be processed and analyzed to understand the opinion of mass people. This type of analysis can be used to understand the opinion of mass people and predict their behavioural pattern. However, such a massive amount of data becomes useless if anyone failed to filter out the unpromising part efficiently. The abundance of data can be considered as both: a blessing and a curse, as it has become significantly challenging to process data in order to isolate useful and relevant information. Researchers of the database community have been keen to find efficient tools and design models for filtering out non relevant and useless data objects from a large dataset. Skyline query and its variants are considered such query tool that finds interesting data objects from a very large dataset. Such queries help users to make intelligent decisions over the complex dataset.

Skyline query [11, 24] and some of its variants [14, 49, 42] are reconnized as such foremost query. The conventional SQL queries return the query result as a total result set from the particular datasets, but the skyline queries returns the set of non-dominated objects from a supplied dataset. If an object is not inferior to any other objects in any of the attributes and is superior in at least one of the attributes then the object is stated to be a non-dominated object. Skyline queries are applied to fulfil the gap between rank-aware database retrieval [15] and set-based SQL queries. Suppose the case of Figure 1.1. In the provided example, It can be observed that Figure 1.1(a) illustrates dataset with two dimensional values: $Dim1$ and $Dim2$ – as well as data IDs. If it matches the data object T with data object Q, it can be observed that Q is better than T in every dimension (considering the less value is better). In such a case, it can be said that Q is dominating

ID	Dim1	Dim2
P	43	40
Q	27	45
R	46	17
S	32	9
T	77	70
U	60	35
V	15	55

(a) Dataset



(b) Skyline

Figure 1.1: Understanding Skyline query

T or T is dominated by Q.

While matching Q with S, none can be declared to be better in each dimension. Hence, they are not dominating each other. In Chapter 2 the author explained the mathematical formulation of dominance relation and skyline query.

The skyline query returns the set of non dominated objects from a dataset, the Figure 1.1(b) shows that $\{P, R, U, T\}$ never be the objects of the skyline query result set – because they are somehow dominated by the other objects in the dataset. However, the objects $\{Q, S, V\}$ are not dominated by any other objects in the dataset. Hence, the objects $\{Q, S, V\}$ are the outcome of the skyline query set.

The IT world produces a massive amount of data at every moment. These datasets are vast and complicated and consider as big data. Big data is such a vast collection of extensive or complicated data, to process it, a sophisticated data processing application software is needed. The traditional data processing software is not competent enough to deal with such big data. To treat big data, traditional centralized and sequential data processing schemes are not suitable enough. Parallel and distributed schemes are needed to treat such big data. While composing distributed and parallel algorithms, programmer and designers must study new issues like fault-tolerance, robustness, usability, availability etc., – as well as computational performance requirements. To assist designers and programmers,

various programming paradigms have been proposed in the last few decades. MapReduce, introduced by Google Inc., is one of the before-mentioned parallel programming paradigms. This programming model was produced for processing a huge number of data (sometimes called big data) using robust, parallel and distributed processing mechanism on a cluster. While adopting such a robust distributed framework, the user never has to think the factors such as redundancy, fault tolerance etc. The master node controls the distributed computing of other nodes in the MapReduce framework and manages the various jobs in a distributed and parallel way and controls all the interaction and transfer of the data among the other nodes. The system also provides the facilities of data redundancy and fault tolerance. We can get such implementation of MapReduce by using a software called Hadoop, which is an open-source framework for Googles' MapReduce.

1.1 Motivation

Secrecy of data of the organization has become a severe concern in the database society for several decades. Information secrecy or data secrecy is one of the vital issues, so people are concern about the acquisition and distribution of their private data. The organization may have to face a legal issue if they are not able to provide security of clients' data. The privacy requirement of mass people has to be considered during the processing of their data. The privacy issues are vital and essential whenever an organization want to gather, process and save privately identifiable and sensitive data in the digital mode. In several situations, security of individual private values needs to be protected, and in many cases, it is not permissible to disclose aggregate values from the personal data. Assume that several organizations ready to conduct studies regarding cost and risk forecast. Wherever all the organizations have accumulated a similar type of private data from their customers. Managing the privacy of the data of every client is a principal obligation for each organization. It is known that skyline reckoning needs matching of attribute values among the objects

of each party; without exposing the attribute values, organizations are incapable of computing join skyline from the combined databases. Therefore, in a traditional multi-party skyline calculation, it is impossible to get the skyline objects without showing the objects attribute values to others. Figure 4.1 demonstrates this situation wherever P1, P2, P3, P4, P5 are five objects of party-1 and Q1, Q2, Q3, Q4, Q5 are five objects of party-2 with their costs (dimention1) and risks(dimention2). If both parties require to obtain a logical recommendation list regarding least cost and risk using skyline query, the skyline objects for the own database of party-1 will be P1, P2, P4, and the skyline objects for party-2 will be Q1, Q3, Q4. However, the skyline objects of their consolidated databases will be Q1, P4, Q4, P2. Although the object P1 and the object Q3 are in the skyline of party-1 and party-2 respectively, they do not exist in their merged skyline. Because the objects P4 of party-1 dominates the object Q3 of party-2 and the objects Q1 of party-2 dominates the object P1 of party-1.

Consequently, cost and risk forecast utilizing a skyline query is more authentic and significant if it is computed from the data of both parties. So, the parties want to determine skyline objects from their combined data. But for security reason, they do not prefer to reveal the attribute values in the objects among others. As a result, it is required a secured system that can calculate skyline from consolidated data of both parties without exposing the real attribute values during the calculation.

A well-designed algorithms, like BBS(*Branch-and-Bound Skyline*) [33] or *Sort-Filter-Skyline*(SFS) [13], also need to know the attributes values before the computation skyline. Let us assume, the circumstances when the control of data belong to different bodies or organizations, they are not want to reveal the attribute values but willing to get the skyline objects. Traditional skyline query computation methods are inadequate to solve such circumstances. In this research work, the author introduced an innovative procedure to address the secure skyline computation in a distributed setting. The author presented

Organization-1		
ID	Cost	Risk
P1	27	33
P2	39	3
P3	45	15
P4	30	17
P5	45	30

Organization-2		
ID	Cost	Risk
Q1	22	30
Q2	48	11
Q3	32	25
Q4	36	8
Q5	42	37

Table 1.1: Organization Database

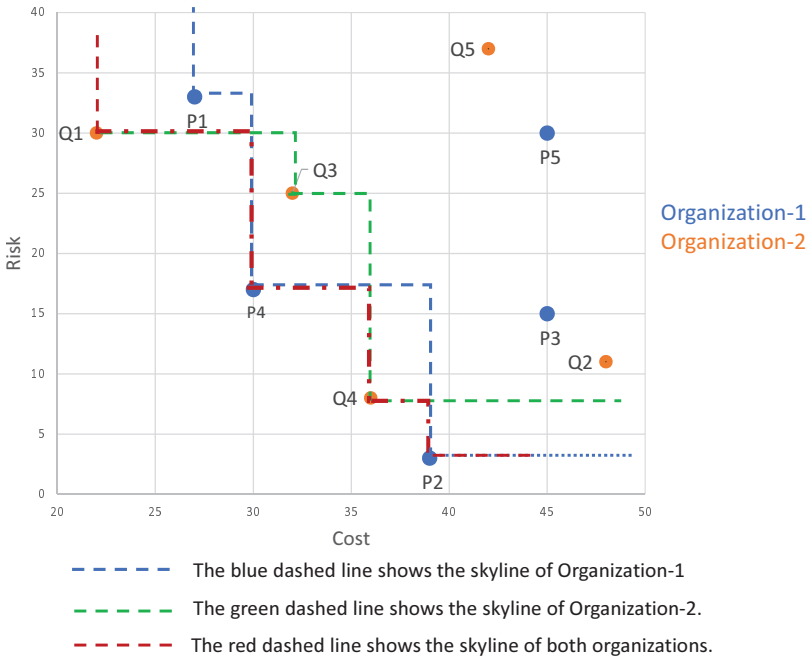


Figure 1.2: Skyline and Multiparty skyline of organization-1&2

a model which is capable to compute skyline objects set without disclosing the attribute values on the MapReduce framework.

1.2 Thesis Organization

The author organized the rest of this thesis as follows: At first, the author highlights basic background knowledge and related works regarding the proposed methods in **Chapter 2**.

In **Chapter 3** the author evaluated the model of Semi-order preserving encryption(SOPE). **Chapter 4** explain details about the secure computation of skyline query from multiple datasets without exposing attribute values. The author proposed the model that uses MapReduce framework as a parallel and distributed computing environment. Finally, **Chapter 5** provides the conclusion of the works and presents some future directions to go ahead.

Chapter 2

Background Knowledge

This chapter contains the theoretical knowledge and related works for the proposed methods in details.

2.1 Preliminaries on Semi-Order Preserving Encryption

2.1.1 Types of Attack Considered

- *Ciphertext-only attack*

  In cryptography, a known-*ciphertext* attack or ciphertext-only attack (COA) is an invasion model for cryptanalysis, here the adversary only can able to access *ciphertexts*.

- *Chosen-ciphertext attack*

  A chosen-ciphertext attack (CCA), is an invasion model for cryptanalysis, where the adversary can decrypt some portion of chosen ciphertext. From this kind of knowledge, a cryptanalyst can try to gain the private key utilized for decryption.

- *Known-plaintext attack*

  Known-plaintext attack (KPA) is an invasion model for cryptanalysis, where the

adversary can get values in plaintext form and ciphertext form. This information can be utilized for further investigation to reveal the private keys and the mapping tables.

2.1.2 Threat Model

The threat models are chosen as provided underneath.

- *Storage scheme managed by the database software is vulnerable to compromise.*

  Contemporary database schemes usually perform their storage management. But, the storage policy is performed by the operating system in which the DBMS software is installed. An adversary can get access the storage system and get the database files by utilizing a path that is different than through the database management system, or in the severe case getting the files directly from storage media. Attackers may gain access to the database file. Since the proposed system encrypts values in the data, such access may initiate a *COA*.

- *The database software is entrusted.*

  It is assumed that the database software is trusted to implement encryption of query constants and decrypt query returns. The author also believes that some values in the database softwares' memory may be accessible to opponents. An adversary can launch a KPA from those values.

- *All disk-resident data is encrypted.*

  It is considered that the database software encrypts schema information such as table and field names; metadata such as column statistics; recovery logs; and data values. Therefore, an opponent is unable to infer the data distribution.

- *The attacker may get several plaintext values of some chosen ciphertext.*

  With the cooperation from the database operator, an attacker can achieve the plain-

text values of some chosen ciphertexts, which are not a part of the initial values saved in the database. From such knowledge, the attacker can launch a CCA.

2.1.3 Order Preserving Encryption

Encryption is the most practical process to accomplish data security. Unencrypted data is named as *plaintext*, and encrypted data as *ciphertext*. In cryptography, a *key* describe the specific transformation of *plaintext* into *ciphertext*, or conversely for the time of decryption.

Assume *plaintext* and *ciphertext* be p and c, respectively. Let k_1 and k_2 be the encryption and decryption *keys*. the exhibit the encryption and the decryption can be shown by the following functions:

$c = encryption(p, k_1)$

$p = decryption(c, k_2)$

If $k_1 = k_2$, then the system is recognized as *symmetric-key cryptography* else *asymmetric-key cryptography*.

Order-preserving encryption is, to some extent, unlike ordinary encryption. Suppose a database P consists of $|P|$ *plaintext* numeric values, which are presented as $P = p_1, p_2, ..., p_{|P|}$, where $p_i < p_{i+1}$, then, when it encrypt the *plaintext* values into *ciphertext* values, which are presented as $\tilde{C} = \tilde{c}_1, \tilde{c}_2, ..., \tilde{c}_{|P|}$, it need to be confirmed that $\tilde{c}_i < \tilde{c}_{i+1}$ $(i = 1, ...|P| - 1)$. Therefore, the order of encrypted values needs to be the same as the *plaintext* values. In various cases, the order itself is private and sensitive information. Therefore, it can be considered that data privacy is not confirmed enough in the existing OPES schemes.

2.1.4 Semi-Order Preserving Encryption

SOPE is an improvement of the OPES. It is not a standalone system; it runs on top of the OPES. To improve the privacy of the order preserved encrypted values in $\tilde{C}$, these

values transform to semi-order preserving values. On transformation, $\tilde{C} = \tilde{c}_1, \tilde{c}_2, ..., \tilde{c}_{|P|}$ becomes $C = c_1, c_2, ..., c_{|P|}$. In sequence C, its original ordered sequence is perturbed to a new *ciphertext*, in which the order is no longer maintained.

2.2 Related Work on Semi-Order Preserving Encryption

Various OPES have been introduced in the literature [3, 8, 9, 10, 12, 31, 16, 29].

Agrawal *et al.* [3] came up with their OPES as follows: first, they represent the input and target distributions as linear splines. Next, they level the *plaintext* database values into a uniformly distributed database values. Moreover, they convert the uniformly distributed database values into cypher database values.

The proposed method by Bebek [8] produces a sequence of random numbers. Besides, the random numbers jth value is added to the jth integer to preserve the original order. The disadvantage of this approach is its' inefficiency in encrypting values. Moreover, the method does not consider the inclusion of new data into the database.

Boldyreva *et al.* [9] offered an order-preserving symmetric encryption. In their system, they managed a natural association between a random order-preserving function and the hypergeometric probability distribution.

Distinctive from the above systems, Boldyreva *et al.* [10] proposed cryptography-based OPES. At first, it represents the ideal OPES whose encryption function is chosen consistently at random from a set of all strictly increasing functions. Though the idea of this OPES is not feasible; it can be utilized as a security consideration for a practical OPES. They proposed a method to transform a *plaintext* x to its *ciphertext* using a binary-search process in the *ciphertext* space and then transform back the searched points using hypergeometric distribution to the space.

In the approach offered by Ozsoyoglu *et al.* [31], a series of strictly increasing polynomial function is used to build the OPES. Encrypted value of an integer x is obtained from

iterative processes of encryption functions on x. The OPES security algorithm is hard to analyze because it is not formed using basic conventional cryptographic algorithms.

Hacigümüş *et al.* [16] suggested a system that spilits the domain of *plaintext* into many partitions and then allocates an identification, which can be order preserved, for each partition. They applied a transformation function between the *plaintext* and encrypted value. A limitation of this encryption algorithm is that it unable to compare all the *plaintexts* (e.g. the *plaintexts* in the same partition).

Zheli Liu *et al.* [29] considered a system that utilizes information space expansion and nonlinear space divisions to protect data distribution and frequency.

Xiao *et al.* [46] demonstrated protocols called "DOPE" and "OE-DOPE", which can accomplish OPES in multi-user modes. To assure that no entity in the system identifies the OPES encryption key, they included a collection of key agents into the system and introduced the DOPE protocol.

Ce Yang et al. [47] presented a SOPE, though with the loss of accuracy. On the other hand, in the authors' proposed system, the security of database values is managed correctly without loss of precisions during encryption or decryption process.

All of the above mentioned algorithms need to store the order value to the disk, which is, in reality, a risky way. Consequently, this method offers a new strategy to improve security, which runs on top of the OPES algorithm [3, 9, 10, 29].

In addition to OPES, some of the secure multi-party calculations use similar procedures, which are presented in Sections 3.2 and 3.2.1. Among them, Hamada *et al.* introduced a secure multi-party computation of radix sort in [17]. The approach applied in their secure computation is the same as in Section 3.2.1, where the union of other parties' data used as an encrypted database. Besides, the method considers general database queries on the encrypted database.

2.3 Skyline Query

Skyline query is one of the most familiar and popular queries as recognized by the database researchers society. It is an efficient and effective mechanism for acquiring preferable objects from the datasets. A skyline query objects are those objects that are not dominated by other objects in the dataset. Consider the case of a two-dimensional database, presented in Figure 2.1. It is well known, the skyline is collection all non dominated objects, from the figure $\{V, Q, S\}$ are not dominated by other objects, so they are the skyline objects. Nevertheless, the objects $\{P, R, T, U\}$ are dominated by other objects in the dataset. As a result, they are filtered out from the final skyline set.

Due to understand and define dominance relationship of objects in the dataset and skyline query arithmetically, let us consider that there is an n dimension in the dataset DS, $\{d_1, d_2, \ldots d_n\}$ be the n attributes of DS and, m objects $\{Ob_1, Ob_2, \cdots, Ob_m\}$. Here, $Ob_i.d_j$ to denote the j-th dimension value of object Ob_i. Without losing of generality, it is considered that the less value in each attribute is better.

Dominance: An object $Ob_i \in DS$ is consider to dominate another object $Ob_j \in DS$, written as $Ob_i \prec Ob_j$, if $Ob_i.d_r \leq Ob_j.d_r$ $(1 \leq r \leq d)$ for all the d attribute and $Ob_i.d_t < Ob_j.d_t$ $(1 \leq t \leq d)$ for at least one dimension. One can consider such Ob_i as *dominant object* and such Ob_j as *dominated object* between Ob_i and Ob_j. As an example in Figure 2.1, R is better than U in each dimension, hence $R \prec U$ but in between R & P, $R \not\prec P$ and $P \not\prec R$.

Skyline: An object $Ob_i \in DS$ is consider to be a *skyline object* of DS, if and only if there is no other object $Ob_j \in DS$ $(j \neq i)$ that dominates Ob_i. The skyline of DS, presented as $Sky(DS)$, is the set of skyline objects in DS. As an example in Figure 2.1, $Sky(DS) = \{V, Q, S\}$.

Cells wise skyline:

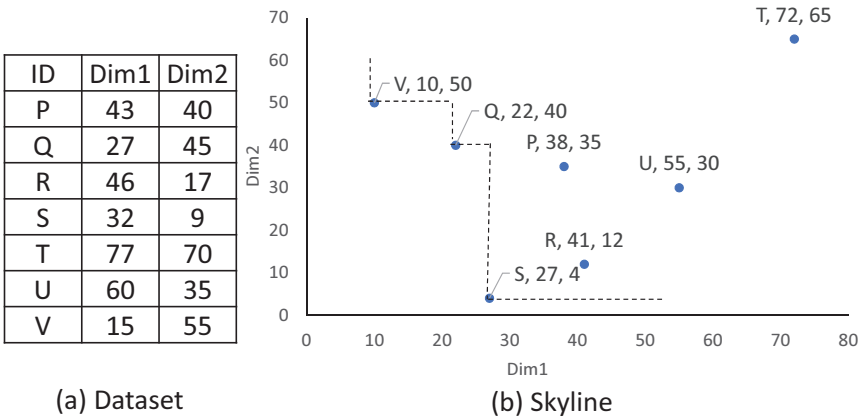


Figure 2.1: Skyline query

Cells are constructed by dividing the domain of each attribute by the corresponding partition number. Figure 4.4(a)(b) explains the cell division by applying four partitions in each attribute. While the method computes the skyline from the objects in the similar cells from all the parties, then it is called cells wise skyline. For instance in figure 4.4(c)(d) if it consider objects (1,4) ;(2,1) from cell (0,24) of party-A and object (2,3) from cell (0,24) of party-B, then the cells wise skyline objects in cell (0,24) of party-A and party-B will be (1,4) and (2,1). Because (2,3) is dominated by (2,1) in the cell (0,24) of party-A and party-B.

2.4 Secure Skyline Query

The concept of "Secure skyline query" originates from the requirement of the data privacy issue. Nowadays, the confidentiality of data, as well as complexity to compute the private data, is a pivotal concern of the database researcher community. From the discussion of the skyline, it is recognized that: to obtain the result of the skyline query; the dominance relationship has to be determined. It is not possible to determine the dominance relationship without disclosing the attribute values in each object. *Branch-and-Bound Skyline*(BBS) [33] or *Sort-Filter-Skyline*(SFS) [13] are efficient algorithms to compute

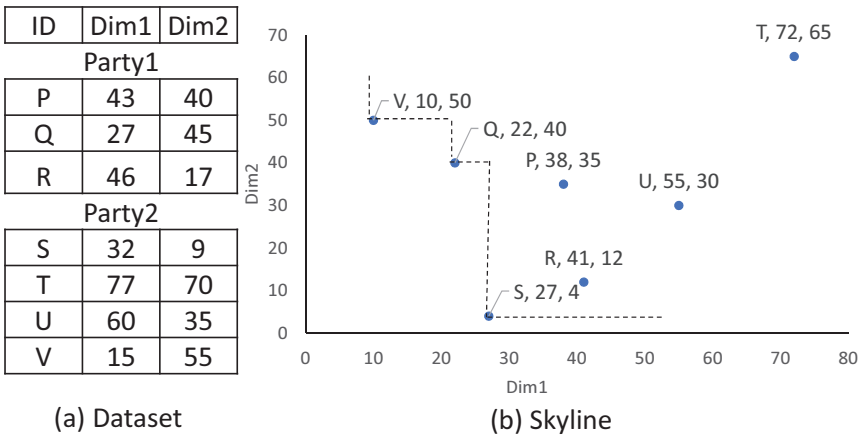


Figure 2.2: The Secured Skyline Query

skyline query, even these algorithms are not able to perform skyline query without disclosing the attribute values. In a situation where the dataset is related to multiple organization, they want to compute skyline query from their combined datasets but not willing to disclose their private data to others. To effectively understand the query problem, let us assume the example given by Figure 2.2. As illustrate in the figure, the objects $\{P, Q, R\}$ belongs to $Party1$ and $\{S, T, U, V\}$ belongs to $Party2$. In the case where $Party1$ & $Party2$ willing to determine the skyline query result from their join dataset without disclosing the attribute values in the objects, in traditional way it is not possible to compute the desired skyline. In traditional way dominance check between the objects of $Party1 : \{P, Q, R\}$ and objects of $Party2 : \{S, T, U, V\}$ are not possible without knowing the attribute values. In the proposed model, as explained in Chapter 4, addressed the problem and provided the solutions to this problem. The proposed solution uses the Googles' *MapReduce* programming paradigms on *hadoop* framework, which is designed and implemented by the Apache software foundation.

2.4.1 Paillier Cryptosystem

As an asymmetric key based homomorphic encryption mechanism Paillier cryptosystem is a popular method [32]. In this system, both public and secret key are in integer form.

We can denote *plaintext* as p and *ciphertext* as c, respectively. Consider, the public key is $Paillier_{pk}(n, g)$ and the secret key is $Paillier_{sk}(\lambda, \mu)$.

Paillier encryption and decryption process can be defined by the functions as follows:

$c = g^p \cdot r^n \mathrm{mod} n^2$

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

The method is an additive-homomorphic cryptosystem; it is possible to add two values in the encrypted form.

Let us consider, m_1 and m_2 are in plain text and the corresponding encrypted values are ζ_1 and ζ_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 accomplish homomorphic addition applying

- Homomorphic Addition

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

2.4.2 MapReduce on Hadoop framework

MapReduce is a popular distributed programming paradigm and software framework for processing big data introduced by Google Inc. In this programming paradigm, a huge amount of data can be processed in a parallel and distributed manner using a cluster of computers. It needs to design a parallel algorithm that can run on MapReduce framework. The programmer does not need to manage the factors like the redundancy of data, fault tolerance etc. during programming in this framework. The *MapReduce* framework performs the processing of distributed data by managing the various computing nodes, distributing the Map Reduce jobs in parallel, maintaining the data transfer among the different nodes with redundancy and fault tolerance.

Programmers define a map function and reduce function; the map function takes input

as a key/value pair and produce another set of key/value pairs as an intermediate result. The reduce function process all intermediate values generated by map function connected with a certain intermediate key. Many real-world problems can be implemented in this way.

The Apache Foundation maintains the *Hadoop* [5] as a popular open-source implementation of the *MapReduce* framework. A programmer only needs to assign a Map job, and a Reduce job by defining the *map*, and *reduce* functions only. In this system, data are presented as $\langle key, value \rangle$ pairs and calculations are distributed over a shared-nothing cluster of independent computers. Jobs are accomplish in this framework only utilizing two user-defined functions, named *Map* and *Reduce*:

$$Map(k_1, v_1) \rightarrow list(k_2, v_2)$$

$$Reduce(k_2, list(v_2)) \rightarrow list(v_3)$$

For each part of input data a $\langle key, value \rangle$ pairs are processed by *Map* function (usually referred to as Mapper) and as a result it produces intermediate $\langle key, value \rangle$ pairs. After that, the intermediate $\langle key, value \rangle$ pairs are sorted and arranged associated with the identical intermediate *key*. The *Reduce* function (usually referred to as Reducer) receive a a list of *key* and *values* pairs. After receiving *key* and *values* pairs it applies the final processing algorithm, and compute the final result. MapReduce is frequently utilized to process a massive amount of data due to its scalability and fault-tolerance. The readily available of scalable and open-source MapReduce architecture, such as Hadoop [1], make it a desirable system for large-scale parallel skyline calculation. The author decided to use MapReduce using Apache Hadoop because of its positive characteristics like being easy to implement, its popularity, compliance, and fault-tolerance. The user can use and scale it managing the general-purpose computers, so it is a cost-effective solution for the distributed computing system. The algorithm design for the Apache Hadoop system can be easily portable to

Apache Spark. It also provides the wanted accurate result and performance in the case of the implementation of the proposed method. Therefore, Hadoop MapReduce is adopted to process the data produced from various parties throughout the computation of skyline. Furthermore, there were many skyline computations in practice that apply the MapReduce framework to compute skyline efficiently: the works in [30, 38, 50, 41] confirmed that MapReduce-based parallel skyline calculation is more efficient than the centralized skyline calculations and can process a large volume of data.

2.4.3 Related Works for Secure Skyline Query

Here, we discussed the various related works that need to understand the skyline query and its different variations.

Skyline Query

B *Block-Nested-Loops*(BNL), *Divide-and-Conquer* ($D\&C$), and B-tree-based schemes [11] are introduced by Borzsonyi et al. They first introduce skyline operator for big datasets. In BNL schemes each object in the dataset compares with the other objects in the dataset finally the non dominated objects are returned as a skyline set. In $D\&C$ schemes the dataset is split into several partitions that can fit into the main memory. Then skyline computation is performed in each partition after merging the skyline in each partition the final skyline is computed. Kossmann et al. suggested the $D\&C$ algorithm using the nearest neighbour (NN) algorithm the method and split the data space iteratively based on the neighbouring objects in the space and efficiently filter out dominated objects [24]. Chomicki et al. suggested *Sort-Filter-Skyline*(SFS) as a slight variant of BNL [13], a presorting scheme is used to enhance the BNL algorithm. The efficient among the all skyline algorithm *Branch-and-Bound Skyline*(BBS) is suggested by Papadias et al., in this method, they use a progressive scheme utilizing *best-first nearest neighbor (BF-NN)* algorithm [33].

Nowadays, parallel computing model becomes very successful for skyline computation. W.T. Balke et al. have suggested skyline queries in distributed settings in [7]. Their work supports vertically partitioned web information. Various works had been done in the field of distributed skyline queries. Wang et al. and Chen et al. [45] both use structured P2P networks, called BATON networks and investigated skyline queries in the networks, each peer have computed a section of the skyline from the data space and finally merge the result to compute final skyline. A grid-based (AGiDS) scheme is proposed by Rocha-Junior et al. [37]. In this method, they distributively compute skyline query using a grid-based strategy where each peer holds a grid-based data summary to represent the data distribution. Arefin et al. [6] in their work utilize agent-based privacy-preserving set-skyline for distributed database. The problem solved by their approach is different from the proposed secure skyline query.

The giant software company Google develop the *MapReduce*, which is a popular framework to process queries over extensive data and capable of handle big data with scalability and fault tolerance. In [49], Zhang et al. first suggested an efficient way to compute the skyline query in *MapReduce* architecture. They proposed three different skyline computation process using the *MapReduce* framework, called MR-BNL, MR-SFS and MR-Bitmap. In [34], Park et al. also proposed an algorithm that can efficiently compute skyline in MapReduce called MR-SKY algorithm. A quad tree-based data partition is used to compute the skyline in a distributed manner. They locally prune the non dominated objects in each node then compute the global skyline by merging the node skyline.

2.4.4 Secure Skyline Query

Due to the privacy-aware present era, each institution shows concern about the safety of their data. In various application perspectives, it is essential to compute multi-party skyline without exposing private information to others. Liu et al. [26] suggested secure

skyline query that can perform the skyline query in encrypted form on the cloud platform. To accomplish the secure skyline Liu et al. use secure comparing protocol offered by Veugen et al. [44] and the secure bit-decomposition scheme suggested by Samanthula et al. [40]. Hua et al. proposed a privacy-aware *skyline* computation scheme called CINEMA [20]. In their work, they offered a solution for computing secure-skyline query on the basis of the user's dynamic query. In their system, a user can protect the dynamic query point from the database owner and the database owner can also preserve the privacy of data from the user throughout the computation. Although their scheme gives a secure computation setting concerning data privacy, their circumstances are not similar to us. Furthermore, their model applies a computationally expensive secure comparison scheme.

Liu et al. suggested another privacy-aware skyline computation model [27] using additivity property of *skyline* [19] to enhance the performance and reduce the number of secure comparison. They calculate local skyline objects set at first. Later from the local skyline, they utilize secure dominance relation to computing the global *skyline* object set. However, for several participating parties, it requires a pairwise secure skyline calculation for global skyline computation. So as the amount of the participating parties increases the computational complexity increases quickly. Besides, the complexity of 0-encoding and 1-encoding system applied in their suggested system increase with the domain space of the attribute values. From the earlier discussion, it can be understood that the methods discussed above for multi-party skyline query are not sufficient and efficient enough when the number of parties increases and also need to exchange considerable amount data throughout the computation among the parties.

Qaosar et al. proposed a secure multi-party skyline computation method [36] . It enhances the efficiency, compared to other secured skyline computation methods, but it needs to assign and share an encrypted substitution vector among the parties before secure computation of skyline.

In the recommended proposed method, the problems are solved by keeping the exchange of data minimum and by implementing parallel and distributed computation of skyline in each stage.

Skyline Query in MapReduce

Nowadays, the distributed computing model has become very familiar with skyline computation. Kasper Mullesgaard et al. [30] offered efficient skyline computation in the MapReduce framework. They produced a grid partitioning design to splits the data space into different partitions and employed a bit-string to represent the partitions. The bit-string was efficiently obtained in MapReduce, and it served to prune partitions (and tuples) that could not have skyline tuples. Hyeong-Cheol Ryu et al., applied adaptive two-level grids to compute the skyline query in MapReduce [38]. Ji Zhang et al. [50] in their system they considered data partitioning, filtering, and parallel skyline computation as a holistic query method. To enhance the parallel local skyline calculation, they proposed two partition-aware filtering schemes that kept skyline candidates in a well-balanced way. Yoonjae Park et al. [35] suggested efficient parallel scheme for processing the skyline and its alternatives using MapReduce. They expertly pruned out non-skyline points in advance with the help of histograms computed from all points. Then, applying the quadtree, they split the data into partitions, where each partition contained the same amount of data points. In the first MapReduce phase, it computed the candidate skyline in each split; then in the next step, it merged the candidate skyline to produce the final skyline. Conversely, the above-discussed systems did not consider multi-party databases and privacy concerns about the multi-party skyline.

Asif Zaman et al. [48] proposed the secure objects' ordering-based skyline computation scheme. In his proposed model, all participating parties constructed their database objects' order with the aid of a semi-honest third party, defined as a coordinator. Initially, the

scheme produced the order of the values for each attribute in the multi-party databases. This computation needed one round of MapReduce process for each digit in a dimension. For instance, in the multi-party databases, if there were D attributes, and each attribute contained M digits, then it needed $D * M$ number of MapReduce rounds for the generation of the order of the attribute values. Later, from the order of the values in each attribute of multi-party databases, it measured the skyline objects applying another round of the MapReduce process. In the currently proposed method, three MapReduce rounds are used for determining the privacy-preserving skyline objects, thus enhancing the performance.

Chapter 3

Semi-order Preserving Encryption

3.1 Introduction

The growing interest in massive data processing and the increasing popularity of cloud computing, different groups and database designers currently highlight the confidentiality and privacy of secured data such as employee wages, age, and expenses. Such sensitive data are frequently saved in a database. Various encryption procedures have been utilized for protecting the privacy of sensitive data [43, 10]. But, such procedures usually degrade the performance of the query executed directly on the encrypted data. At first, the database process decrypts the values and then execute the conditional operation in the particular query on the decrypted data. A notable amount of time is spent for decrypting each tuple before executing conditional operations contained in a query parameter.

As a consequence, several variants of the order-preserving encryption system (OPES) and some of its' other options have been proposed in the few articles [3, 8, 9, 10, 31, 16, 29].

The methods mentioned above can increase the performance of database queries by preserving the order of the data. Nevertheless, in the application of several databases, the ordered list of numerical data itself is regarded as sensitive information. For instance, some organizations may observe the merit rank of a person as private information that should

not be revealed to other peoples. For this, the semi-order preserving encryption techniques are proposed. Methods can perturb the original order or index of data and can able to enhance the privacy of data without deteriorating the efficiency of comparison operations over perturbed data.

In essence, the offerings of this research are as follows:

- In this dissertation, the author proposed three methods to perform semi-order preserving encryption (SOPE) for numeric values, and which are run on top of OPES.
- Proposed scheme can execute the query with a comparison operator from the encrypted semi-ordered values.
- The author empirically confirmed the performance and effectiveness of these stated methods through various experiments.

3.2 SOPE Technique using OPES and Two-way Perturbation

In the first scheme, the author applies the perturbation of the order information in such a way that it can quickly sort the perturbed values and perform queries efficiently.

For ease of understanding, the author considers that the database with private and sensitive data consists of one table with one column. Assume that there have a column of *plaintext* values: $[43, 253, 629, 69, 521]$ which are encrypted as $[2089, 3458, 7501, 2923, 6303]$ by using the OPES to the plaintext values. In the proposed method, the traditional OPES process is utilized for encryption; moreover, the proposed method perturbed the values that are generated from the OPES.

To improve the privacy of encrypted order preserved values in $\tilde{C}$, it map the m-digit ciphertext values for individual digit and produce perturbed m-digit ciphertext values.

After the transformation procedure, $\tilde{C} = \tilde{c}_1, \tilde{c}_2, ..., \tilde{c}_{|P|}$, in which, $\tilde{c}_i < \tilde{c}_{i+1}$ becomes $C = c_1, c_2, ..., c_{|P|}$, after that the order is no longer retained, and it addresses the sequence as: "*perturbed ciphertext values*". The order in $\tilde{C}$ is perturbed in C. Figure 3.1 demonstrates an illustration of the perturbation step.

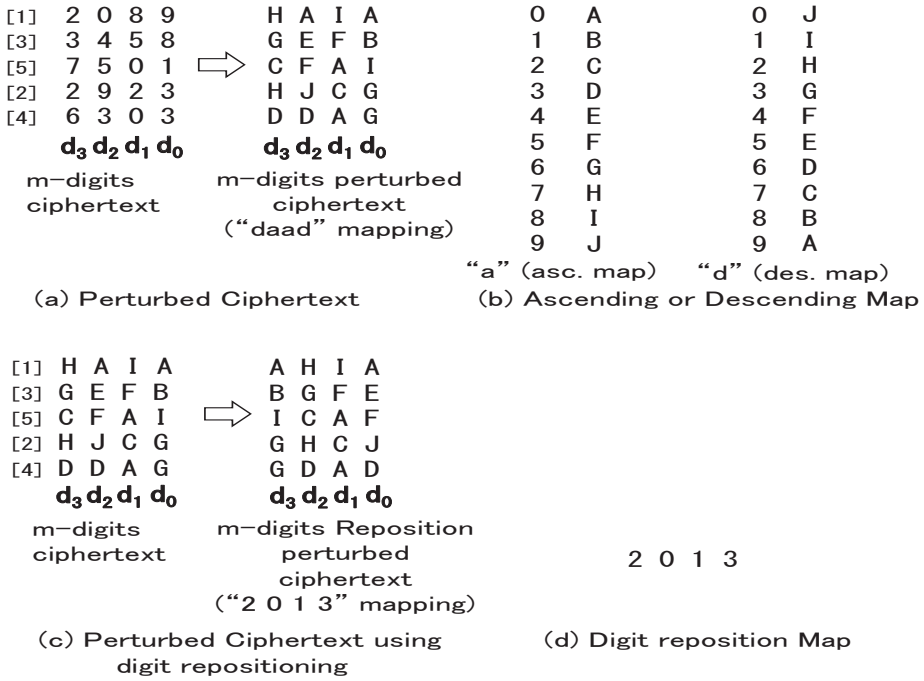


Figure 3.1: Two-way Perturbation Example

Assume $\tilde{c}_i$ be an m-digit ciphertext and $\tilde{c}_i[j]$ $(0 \leq j \leq m-1)$ be the jth digit value of $\tilde{c}_i$. it randomly select one of the two option of mappings in each digit: "an ascending order map" denoted as "a" and "a descending order map", denoted as "d". It convert $\tilde{c}_i[j]$ to $c_i[j]$ such a way that the order becomes $c_{i_1}[j] < c_{i_2}[j]$ $(c_{i_1}[j] > c_{i_2}[j])$ in the ascending (descending) order map if $\tilde{c}_{i_1}[j] < \tilde{c}_{i_2}[j]$ $(i_1 \neq i_2)$

Figure 3.1 (b) explains an illustration of the mapping. As an example, "3" is mapped to "D" in an ascending order map and to "G" in a descending order map. Remark that in each digit, the initial order in the ciphertext is maintained in the ascending order whereas, reversed in the descending order map.

Figure 3.1 (a) illustrate the perturbed ciphertext when it randomly selsect "daad"; this

signify that the digits from 3-0 are transform into “d”, “a”, “a”, and “d” in that order. For instance, “2089” is transform into the perturbed ciphertext value “HAIA”, in which the 1st value “2” is transform into the descending order value “H” and the 2nd value “0” is transform into the ascending order value “A”. Likewise, “8” and “9” are transform to “I” and “A”, respectively.

Afterwards, it randomly determines a repositioning plan and applies reposition of each digit in a number according to the plan. Figure 3.1 (c) presents the reposition of “HAIA” when it randomly select“2013” as a reposition plan. In the reposition, “H”(the 3rd digit) goes to the 2nd place, the 2nd digit “A” to the 0th place, the 1st digit “I” to the 1st place, and the least significant digit “A”(the 0th digit) to the 3rd place. Ultimately, after the repositioning, “HAIA” converts to “AHIA”.

The transform information such as “daad” and “2013” is saved in the database system are protected from the adversary. It should be noted that an adversary never infer sensitive data such as map information, the number of partitions (digits) each value is separated into, amount of bits in each partition (digit), and whether ascending or descending transform is used for each partition (digit).

In reality, ciphertext values saved in the database are neither decimal nor character values such as the ones in the earlier example; they are binary values comparable to the instance in Figure 3.2. In fact, in the figure, an OPES output “20171119” (in decimal) is split into eight digits (each digit has four bits). Each digit is mapped based upon the secret mapping “aaddddaa” into transform value with padded random noise and repositioned with the secret reposition transform “03571264.”

As a consequence, an adversary never infers the original order of the ciphertext accurately without comprehending the map data.

In contrast, the database scheme, which holds the map data, can compare the perturbed ciphertext values. For instance, the database scheme can comprehend that “AHIA” is less

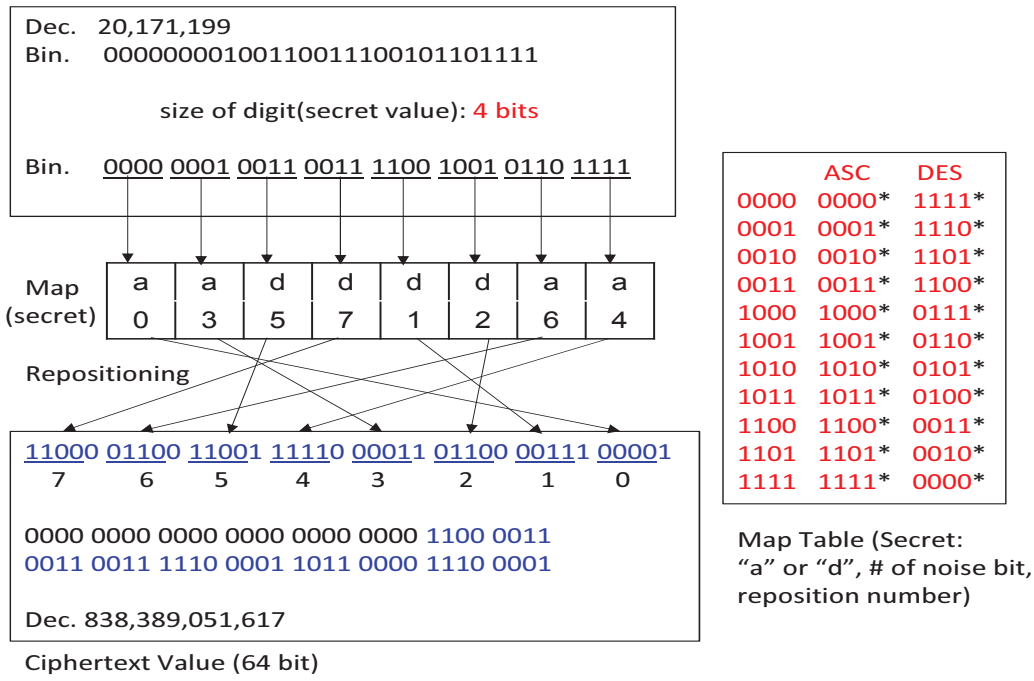


Figure 3.2: Detailed Binary Example of Two-way Perturbation

than "BGFE" because the 2nd digit (reposition of the real most significant digit or 3rd digit) in the numbers are "H or "G", where "H" is greater than "G" in descending order.

3.2.1 Sorting on Perturbed Values

Radix sort can be accomplished directly on the perturbed data, and it is well known that the complexity of radix sort is $O(mn)$ where m is the number of digits, and n is the number of values. In practice, m is very small compared with the total values of n. The sorting capability ensures that the binary search can be performed directly on perturbed values. As a consequence, it covers all the query that we can perform efficiently in the case of OPES values. As the order of each digit in perturbed values is maintained. As a result, during the query execution, the database management software can able to compare the values if it has the order and reposition map of each digit. The "radix sort" [39] scheme can be utilized to perform sorting operation in the database values. The proposed method can use the order map "daad" and reposition map "2013" to accomplish radix sort on perturbed

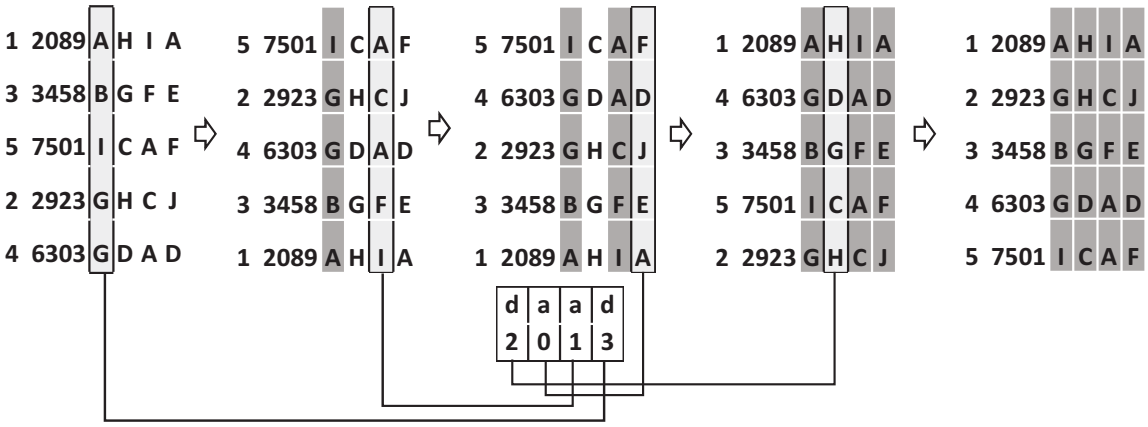


Figure 3.3: Radix Sort on Perturbed *ciphertext*

In Figure 3.3 the radix sort procedure is explained in details. Using the 3rd digit (a reposition of the original 0th digit), the database management software can sort the database values in descending order; the initial sequence {[1],[3],[5],[2],[4]} becomes {[5],[2],[4],[3],[1]}. Next, retaining the order of the 3rd digit, the database management software sort the 1st digit in ascending order, which is a reposition of the initial 1st digit. Both [5] and [4] have equal value in the 1st digit. In this case, the order of the previous result is retained and get the order as {[5],[4],[2],[3],[1]}. The method can be continued to perform the sorting process in other digits. As a result, the complete sorted order of database values can be obtained.

3.2.2 Some example of query processing in OPES scheme

Comparison Operator

SQL query uses, such as =(equal to), <> (not equal to), > (greater than), < (less than), ≥ (greater than or equal to) and ≤ (less than or equal to) operators for comparing numerical values. Figure 3.4 describes how a database management software can perform the queries applying the operators, as mentioned earlier. For example, considered that a user wants to perform the following query:

```
SELECT ... WHERE value > 70
```

Traditional OPES can recognize the possible order index of the value, i.e., 70 using the *Lookup Table* for OPES. It should be noted that the *Lookup Table* usually not cover the total order index of the data in the database but stores the order index of some sample data. The mapping function can use to recognize the order of each value. Details of the process are discussed in the proposed work found in [3]. As an example, it can be assumed that the position of 70 in the *Lookup Table* is 2.1, which is an intermediate value between 2 and 3.

Hence, the *Query Modifier* in database management software can rewrites the query as follows:

```
SELECT ... WHERE order > 2.1
```

When DBMS engine gets such a transformed query, which has the ordered information, it returns all the tuples that satisfy the particular transformed order condition in the query. By applying the order condition, the tuples with the order index lesser 2.1, i.e. {[1], [2]} are filtered out. As a result, the tuples whose plaintext values are bellow 70, i.e., {43, 69} are filtered out from final the result.

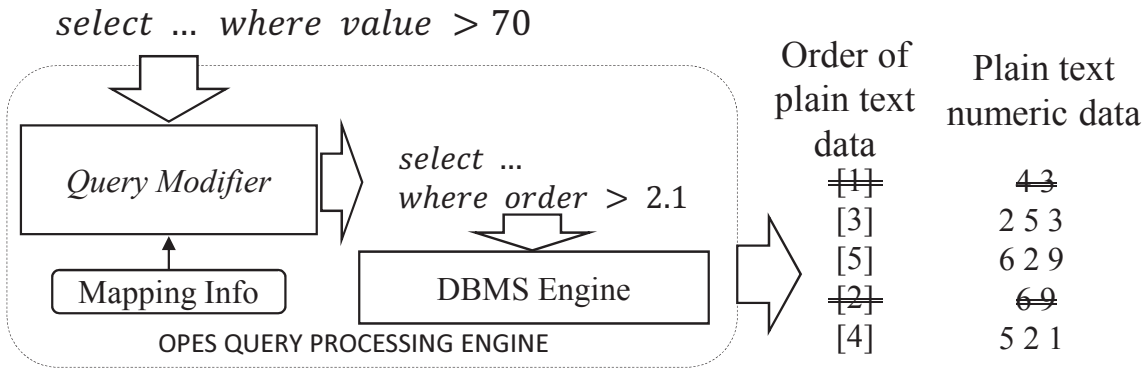


Figure 3.4: Example of Comparison Operator

Aggregation Function

Consider the figure 3.5, and assume that a client want ot perform following query:

```
SELECT MAX(value) ...
```

The query modifier in OPES database management software changes the query as:

```
SELECT MAX(order)...
```

When the transformed query is transferred to the database management software, it returns the tuple with the maximum value from the order index. Consequently, all tuples except {[5]} are filtered out. As a result, the desired value can be collected from plaintext data (i.e., the tuple with a value 629).

Nevertheless, the database management software never able to process two aggregation functions: "SUM" and "AVG". These functions cannot be processed without decrypting the SOPE value by the proposed method. It is known that all the OPES schemes are not capable of performing such operations.

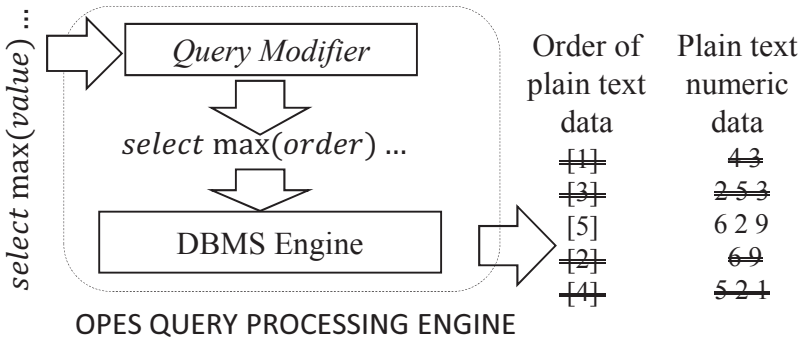


Figure 3.5: Use of Aggregate Function

Insertion and Deletion Operations

Suppose

```
INSERT ]
```

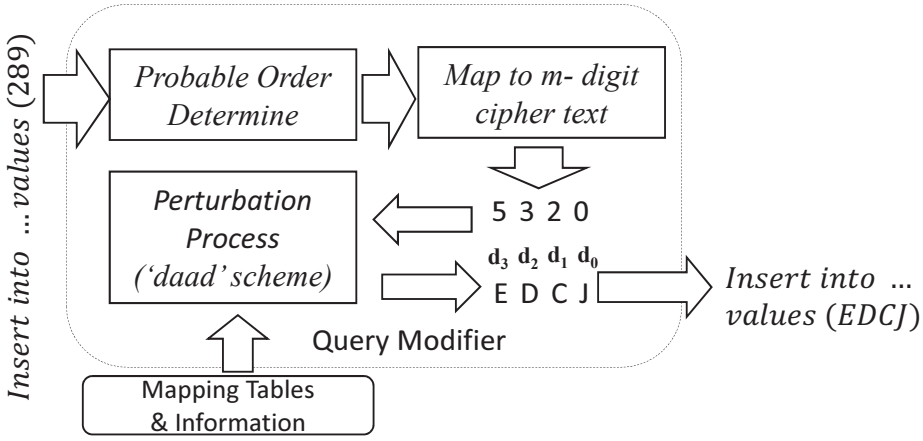


Figure 3.6: Insertion Operation

Whenever such an operator is issued to database management software, the *Query Modifier* determines the probable order index of new data value. As illustrates in the example in Figure 3.6, and according to data in the running example in Figure 3.1.

It is easy to understand that the possible order of the new data value 289 is in between the 3^{rd} and 4^{th} position. As a result, the order preserved m-digit ciphertext value would lie within 3458 and 6303. It can be considered that the order-preserving ciphertext value of 289 is 5320. The detailed process of the mapping can be found in details in the work[3].

After getting value 5320, the method applies the order preserved scheme then the DBMS can determine the encrypted value that is supplied to the database engine.

Deletion operation can be performed using comparison operators in the perturbed text. Consider that a user issues an operation:

```
DELETE ... FROM ... WHERE (VALUE > 70)
```

The method for deleting the values is almost identical, as shown in Figure 3.4. The *Query Modifier* of DBMS has to determine the probable order index of 70 and performs the query execution plan accordingly. The transformed query can be executed as the following query:

```
DELETE ... FROM ... WHERE (ORDER > 2.1)
```

In the case of no *where* condition in the query, the *delete...* process could be transferred to the database management software directly.

3.3 SOPE as a Block-wise OPES using Tree

In this suggested method, it first determines plaintext intervals applying Algorithm 1, and from this plaintext intervals it builds a B-tree in plaintext domain as illustrated in (1) in Figure 3.7. Consider N is the number of a child node in the B-tree. After that, N disjoint intervals are made in the ciphertext domain, as illustrated in (2) in the figure. Moreover, the database management software randomly assigns the intervals in ciphertext domain to each leaf node of the B-tree in plaintext domain, as represented in (3) in the figure. Depending on the B-tree, it uses the conventional OPES for each leaf, as illustrated in (4) in the figure. For instance, plaintext values reside in the $[12, 25)$ are encrypted using OPES to order preserved values interval $[100, 199]$. So that the initial order is preserved. Figure 3.8 illustrates the example how to encrypt the values using the B-tree in Figure 3.7. For example, 14 as a plaintext value arrives to leaf node of the B-tree $[12, 25)$, and by

applying OPES, the value is decrypted to a value in the ciphertext interval $[100, 199]$. In like fashion, 20 as a plaintext value can be encrypted to a ciphertext value in $[100, 199]$ using OPES. As an example, 14 and 20 are in plaintext form encrypted to 121 and 171 in ciphertext form, respectively. It can be recognized that for each node the original order of plaintext domain is retained in corresponding ciphertext domain. But, for two ciphertext values in a different node, the order information is not maintained. As a result, the privacy of order information in the ciphertext is preserved.

Interval Choosing Algorithm

The proposed method offers privacy preservation of sensitive order information in the database values. Consequently, to determine the interval in the plaintext domain is an important task. As illustrated in the Figure 3.7, the order information is maintained in the same range for plaintext and ciphertext values. As a result, an algorithm should be needed that ensures no two sensitive values are inserted in the same interval for providing this; the author designed the interval choosing algorithm. The algorithm is stated as follows-

Algorithm 1: Interval Choosing Algorithm

Result: Intervals for plaintext domain

```
Take all the plaintext values in the plaintext domain;
ListA=First value in plaintext domain;
while all plaintext values are not processed do
    if Is the current value order sensitive to values in ListA then
        Assign a new plaintext interval for ListA values and store the interval;
        empty ListA;
        ListA=Current Value;
    else
        ListA=ListA + Current Value;
    end
end
```

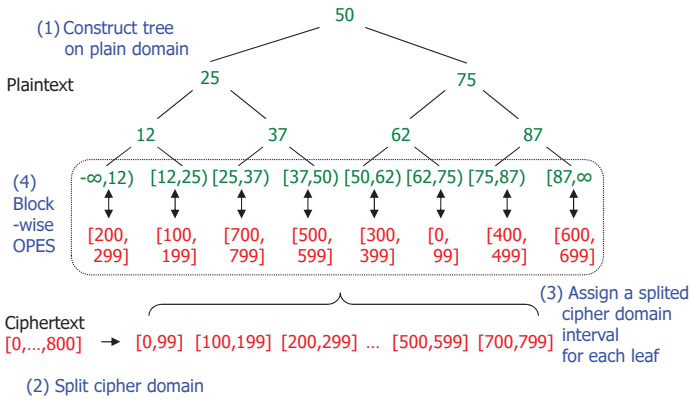


Figure 3.7: Block-wise Semi-Order Preserving Encryption using Tree

Plaintext	3	10	14	20	28	35	39	46	52	58	65	71	77	83	91	96
Ciphertext	215	245	121	171	733	772	513	562	316	361	25	65	411	433	622	651

Figure 3.8: Example of Block-wise OPES using Tree

3.3.1 SOPE Technique using Dynamic Block-wise OPES using Tree

This is considered as a variation of the procedure explained in Section 3.3. The proposed method in this section can manage two things. Firstly, it avoids two sensitive values to insert in the same intervals. Whenever someone tries to insert two sensitive information in the same interval, it creates two intervals and inserts two values at different intervals. As an illustra

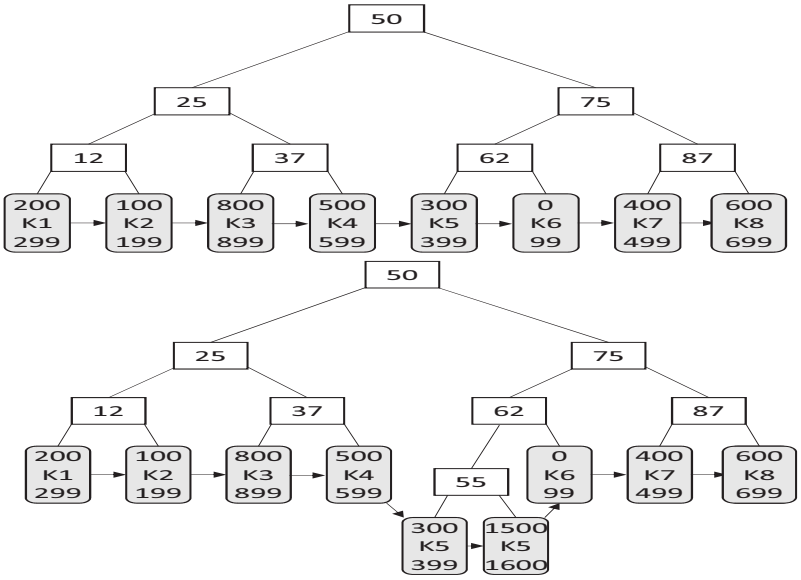


Figure 3.9: Dynamic Split of Interval

300-399 and maintain the similar order. for example a value 56 is order sensitive to any other values in the interval 50-55, the dynamic tree based method will split the interval 50-62[300-399] into 50-55[300-399] and 56-62[1500-1600] to avoid two sensitive value in the same intervals as displayed in Figure 3.9. As a result, Value 56 is transformed to ciphertext using [1500, K5, 1600] and inserted toward the database.

Secondly, in general, the database operation, service hours may have peak and off-peak hours. Throughout peak service hours, the database usually busy to perform different user queries; though, throughout the off-peak service hours, the database is comparatively less active. Therefore, during off-peak hours, the intervals can be changed in the ciphertext

domain to avoid any adversary to recognize the intervals pairs in (plaintext, ciphertext). Figure 3.

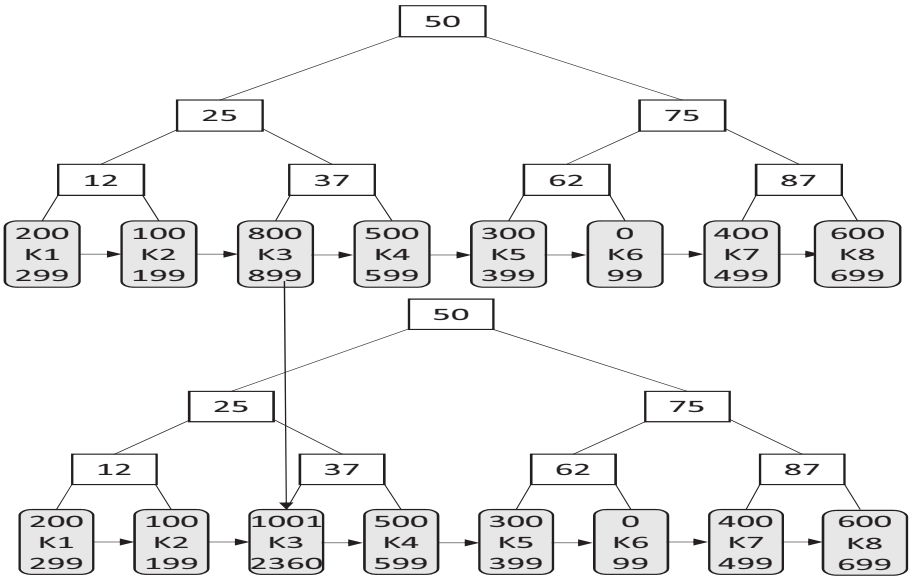


Figure 3.10: Dynamic Interval Update in "off-peak" Operation Hours

in ciphertext intervals changes to 1001-2360. Consequently, all values in the ciphertext interval 800-899 are converted to ciphertext values in the interval 1001-2360. In the case of static intervals pairs, one time obtaining the intervals pair is sufficient to get information about the values. But, in the dynamic intervals scheme, one time getting such information never ensure the adversary to obtain any sensitive information from the database.

3.3.2 Query on Block-wise OPES Values by Tree

Comparison Operator

The scheme can process a query utilizing comparison operators efficiently: $=, <>, <, >$, $>=, <=$. For example, it wants to execute a query:

```
SELECT ... FROM TABLE A WHERE VALUE > 70
```

It is considered that 70 is a plaintext value, and Table A holds encrypted values. The method intends to perform the execution of this query in encrypted data directly and wants

to get the query outcome as an encrypted value. It is apparent from the construction of B-tree when anyone passes the leaf node from left to right; the ciphertext intervals in the leaf nodes are not in an ordered fashion. However, their corresponding plaintext intervals are maintaining the order. When anyone crosses leaf nodes of the B-tree from left to right in ciphertext interval, it signifies that he is crossing the corresponding plaintext intervals in ascending order. For the execution of the query mentioned above, the database management software searched the B-tree with value 70 and arrived the leaf $[0, K5, 100]$. It utilizes the key $K5$ in the interval $[0, 99]$ to perform the encryption operation of the value 70. After encryption operation, plaintext value 70 is converted into the ciphertext value 55. As a query result, the values higher than 55 in the intervals [0,100] and all the values in the right side of the intervals [0,100] are returned.

Aggregation Function

From the discussion mentioned above, it is clear that values in the plaintext intervals, as well as the values in ciphertext intervals in the B-tree, maintain the order from left to right as ascending order. As a result, MAX values is the value in the rightmost leaf node interval in the ciphertext. In the same way, MIN value is the lowest values in the leftmost leaf node intervals in ciphertext domain.

As a result, this database management system can efficiently return the result aggregation function like `MIN` and `MAX`. For example from B-tree in figure 3.7, it is clear that the `MIN` value is the lowest value in the interval [200, 299] (which is interval in leftmost leaf node) and the `MAX` value is the maximum value in the interval [600, 699](which is the rightmost leaf node in the B-tree).

Insertion and Deletion Operations:

Assume, a user wants to perform an insertion operation:

```
INSERT INTO ... VALUES(16)
```

is issued to DBMS. For the execution of the query mentioned above, the suggested method explores the tree applying value 16 and reaches the leaf-node $[100, K2, 199]$. Moreover, by applying OPES key K2 and the ciphertext interval $[100, 199]$, 16 is transformed into 150 after that the ciphertext value 150 is inserted by the DBMS into the database.

For example, the user wants to execute a delete operation:

```
DELETE ... FROM ... WHERE (VALUE>70)
```

is issued to the DBMS. From the discussion as mentioned earlier, the intervals in the B-tree are in ascending order in plaintext domain. As well as in ascending order in the corresponding ciphertext domain. As a result, deleting values higher than 70 can be performed by deleting values greater than values in the corresponding ciphertext intervals and all the values in the right side of intervals in ciphertext domain.

For example, in the proposed method for execution of the above query, it reaches the leaf for value 70 in the B-tree, which is $[0, K6, 99]$ in ciphertext domain. Consequently, 70 is encrypted by K6 in the interval $[0, 99]$ transformed into 50 as a ciphertext value. The above query eliminates all encrypted values higher than 50 in the interval $[0, 99]$ as well as all the values in the intervals $[400, 499]$ and $[600, 699]$, which reside right side of $[0, 99]$ in the B-tree.

3.4 Evaluating SOPE Against Threat Model

The method is developed on top of the order-preserving encryption. It apply the results of OPES [3, 9, 10, 29] for the provable security. Therefore, the method has all the security features of [3, 9, 10, 29] and additionally, it can hide the order.

SOPE using OPES and Two-way Perturbation

As the adversary can not able to access the OPES key, perturbed mapping table, repositioning mapping table, and position of noise bits, therefore, if an adversary desires to identify the order information by applying attack like brute force attack. He has to arrange all the partitions into right order, also has to discard the noise bits as well as has to gain access to the ascending-descending order mapping information. Furthermore, he has to perform the reposition each digit accurately. For n digit binary numbers, the partitions can be accomplished in 2^{n-1} possible ways. If anyone takes into account k number of noise bits, they can be placed in ${}^{n}P_{k}$ possible ways in the number, and permutation of the digits can be accomplished in $n!$ ways. As a result there are possible $2^{n-1} * {}^{n}P_{k} * n!$ ways to decode a number. Therefore, by applying a brute force attack, the possibility of guessing the correct order is $1/(2^{n-1} * {}^{n}P_{k} * n!)$.

SOPE Technique using Block-wise OPES using Tree

Whenever an adversary wants to determine the exact order, he has to split the values into correct intervals as well as determine the OPES key for each interval. For example, N is the number of possible values in the database domain; the number of possible intervals can be produced from N data is 2^{N-1}. Therefore, the probability of guessing the correct interval is $1/(2^{N-1})$ whenever an adversary wants to apply a brute force attack. Consequently, it is tough to determine the exact intervals. Moreover, all intervals are encrypted by distinct OPES key. As a result, an adversary has to manage OPES key for each interval and as well have to partition each interval into correct splits. The interval generation algorithm provides extra security and avoids two sensitive value to insert into the same intervals.

SOPE by Applying Dynamic Block-wise OPES using Tree

To recognize the splits of the intervals is very hard, according to the previous discussion. The adversary may manage the splits, but in this dynamic tree based OPES process, the intervals are dynamically changed all the time. Therefore one time managing of such information does not compromise the sensitive information in the database.

3.5 Analysis of SOPE in Different Types of Attack

COA attack

Usually COA attack, the adversary only able to manage some ciphertext values but no other information like database statistics of the values. In the case of OPES([3, 9, 10, 29]) methods can withstand this kind of attack because the security of the SOPE methods is already proved in case of COA attack. The suggested method runs on the top of OPES. As a consequence, the proposed method is secured enough in these types of attacks.

Chosen-ciphertext attack

In this type of attacks, the adversary may have some information about the plaintext and corresponding ciphertext. The adversary can try to obtain the encryption key and other information from this attack. When the consideration is only OPES, this kind of attack may guess the OPES key. The proposed method utilized OPES as the intermediate encryption for SOPE. Consequently, in the proposed method plaintext values in encrypted by OPES then by the proposed method ($plaintext \rightarrow OPES \rightarrow OPES(cypher) \rightarrow SOPE \rightarrow ciphertext$). The proposed method provides an extra layer of protection for the OPES(cypher). Therefore, OPES security provided by the methods [3, 9, 10, 29]) undoubtedly applicable to SOPE. Moreover, in the case of SOPE values, the only attack the adversary can able to perform is a brute force attack. On the other hand, in the case of

tree-based(SOPE) scheme, the adversary needs the correct splits of the values as well as the OPES key for each split to guess the values.

From the preceding section, the possibility to determine the correct splits is $1/(2^{N-1})$, where N is the number of values in the database. For example, a database contains 1,000,000 number of values, then the probability of determining the correct splits is $1/(2^{1000000})$. In realistic situations, values in the database domain are much larger.

Known-plaintext attack

As the proposed system applies OPES operation to get SOPE, the SOPE can get applying operation in following ways ($plaintext \rightarrow OPES \rightarrow OPES(cipher) \rightarrow SOPE \rightarrow ciphertext$). In this case, the attacker can manage both plaintext and corresponding ciphertext. However, the OPES(cipher) consider as plaintext for SOPE cannot be obtained by the adversary. As a result, the SOPE methods disconnect the link between the original plaintext and SOPE(ciphertext). The security proved by [3, 9, 10, 29]) can be considered as a security for this type of attack.

3.6 Experiments

Various experiments are performed to analyze the proposed method and its effectiveness. The proposed approach was performed utilizing *Matlab R2016b*. The PC used in this experiment has the configuration of fourth-generation Intel® Core™i7 processor, 3.4 GHz CPU, and 8 GB main memory, operating on 64-bit Microsoft Windows 10 Enterprise system. The experiments were conducted five times, and the average value taken as the final results. The values 10k, 100k, 500k, 1m, and 10m are chosen as the input values for the experiments. Each value is generated as a 32-bit random integer. In the datasets 10k, 100k, 500k, 1m, and 10m means 10,000, 100,000, 500,000, 1,000,000, and 10,000,000 number 32 bit integer.

Order information can be hide by applying SOPE on OPES values. AES [22] is an example of a well-known encryption system. Order in the OPES values can be protected by applying AES operation on OPES values. For these experiments, AES is considered as a baseline method to evaluate the proposed schemes.

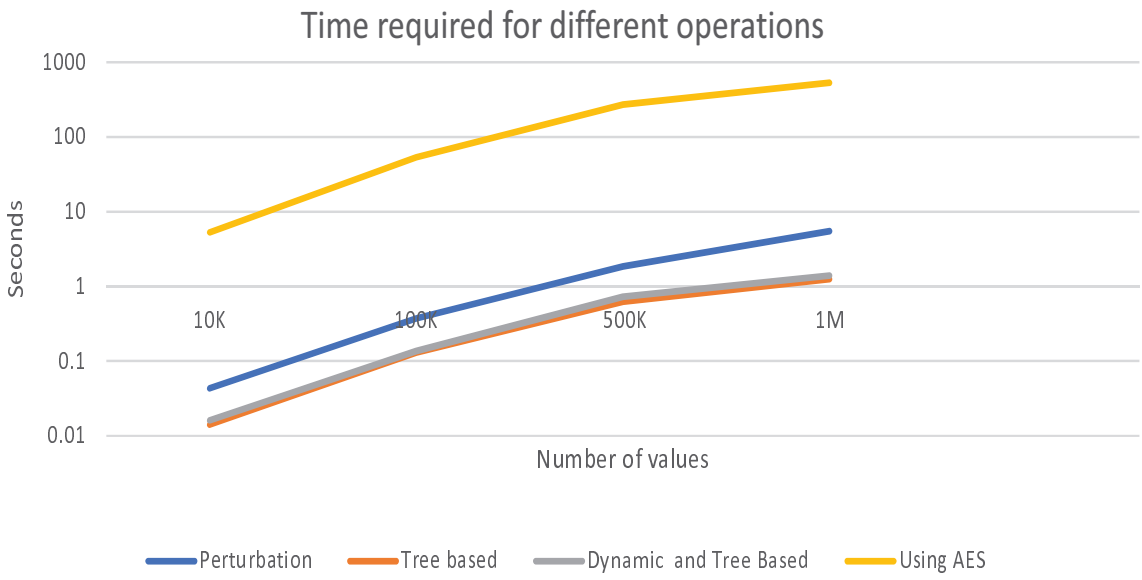


Figure 3.11: Time required for different operations

3.6.2 Order Hiding

Figure 3.12 illustrates the time required for AES based method and the three proposed method. The number of data considers form 10,000 to 1,000,000, and the experiment result shows that the proposed methods can efficiently hide the order information. Suppose n is the number of values in the database table then $2O(n)$ will be the complexity for order

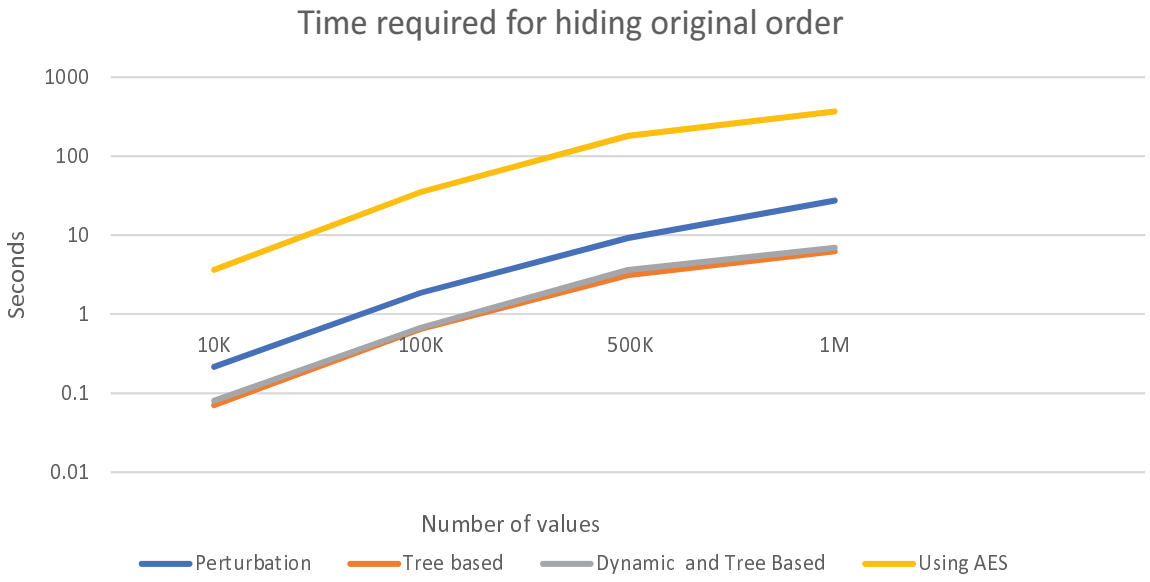


Figure 3.12: Total Time for Order Hiding

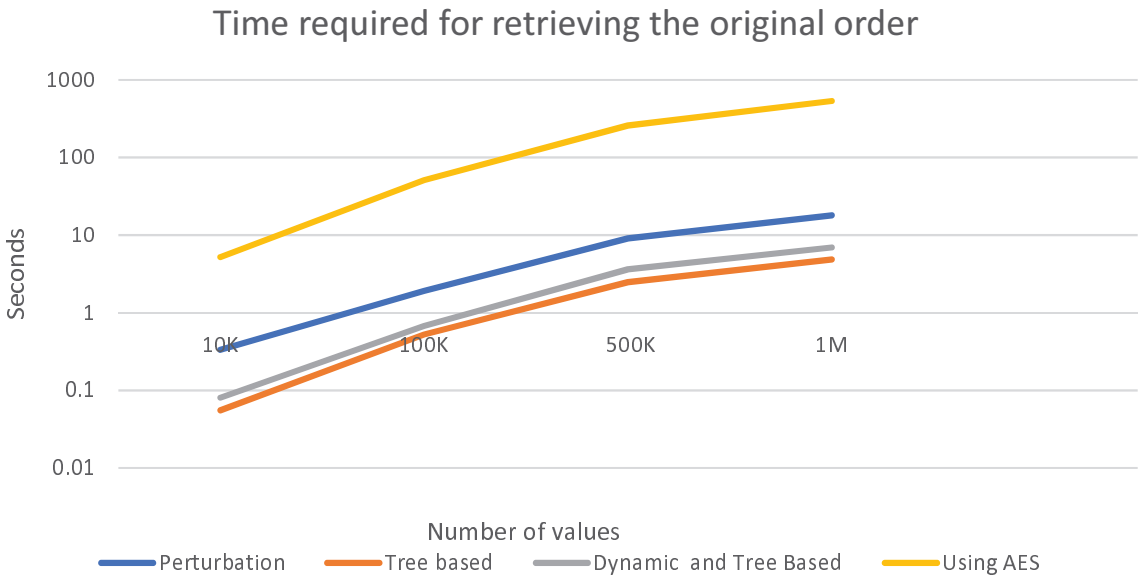


Figure 3.13: Time required for retrieving the original order.

sort is required in the case of perturbation based method to sort the values. The complexity of radix sort is $O(mn)$, where n and m is the number of data and digits in a number correspondingly. In the tree-based method, a binary search and linked list traversal are needed to sort the values; as a result; the complexity of retrieving the original order is $O(n) + log(n)$. When it is considered to order the unordered data, AES based method shows a poor efficiency. From the figure, it can be concluded that the proposed three methods are efficient enough compare to AES based method.

3.7 Conclusion

The introduction section focuses that there may be many circumstances when the user of the database never want to disclose the order information. In this kind of situations, the SOPE technique can successfully hide the secretive order information. SOPE provide an extra layer of protection over OPES based values. The experiments show that the execution times for the proposed SOPE methods are quite acceptable. The queries with the comparison operator can be directly performed from SOPE values without decrypting the SOPE value. As a result, SOPE techniques are applicable in the practical database management system.

So far, the performance of the SOPE is satisfactory. As the future direction, the author will apply SOPE operation in the distributed environment like MapReduce to handle big databases.

Chapter 4

Secure Skyline Query

The skyline query retrieves the representing objects from large datasets. The skyline query returns all non dominated objects. Currently, peoples are conscious of the privacy of their object in a database. So far, many schemes are proposed for computing the skyline query a few of them can execute the skyline query in a distributed manner and able to process "big data" [2, 23, 34]. However, except [48] none of them considers the secure computation of skyline objects without disclosing the domain values. In this chapter, the author proposed a new way to compute the skyline query in a distributed manner on MapReduce securely.

suppose a group of the organization want to compute skyline from their join datasets without disclosing the private information in the datasets. Since no organization wants to disclose sensitive information. Thus, they are not able to compute the skyline from their join datasets only able to compute skyline from their local datasets. Without any doubt, the skyline from the combined datasets is more effective and valuable than the individual local skyline.

Assume, two individual groups have accomplished some market study, and they have collected datasets regarding commission cost and risk prediction. As this information is sensitive, and both the organization never want to reveal the values of the original data.

However, those two organizations are willing to compute the skyline query from their combined survey data. In this regards, a method is proposed to compute skyline from data of both organizations without disclosing the domain values. Which is not possible in typical skyline computation method without exposing the values to other organizations or other third parties. The suggested approach can solve this problem as well as can perform skyline operation in a distributed manner on the *MapReduce* framework.

4.1 Secure Skyline Problem

In the modern era of information technology, companies with the same kind of service collect multiple information from their customers. For a reliable and practical study, they want to survey their combined databases. This kind of analysis is considered as a multi-party computation; examples of multi-party computations are joint data analysis, data mining, statistical data analysis, etc. Company services may have sensitive information, such as private, commercial, or health-related data of their customers. The revelation of such information significantly break clients' privacy and may produce a financial or goodwill loss to the company. Therefore, the companies never desire to reveal their sensitive information to others. However, throughout mutual data mining operations, the participating companies are willing to get the result from their united databases without disclosing the confidential information of the clients.

Skyline has gained significant importance in the database community throughout the past few decades. It is a vital tool in various multi-criteria decision-making applications like the business plan, hotel management, etc. Provide a dominance relationship in a dataset; a skyline query delivers the objects that are not dominated by any other objects inside the dataset.

The data of table 4.1 plotted on the figure 4.1 shows the multi-party secure skyline query where $P1$, $P2$, $P3$, $P4$, and $P5$ are five objects of Organization 1 and $Q1$, $Q2$,

$Q3$, $Q4$, and $Q5$ are five objects of Organization 2 with their costs ($d1$) and risks ($d2$). If both organizations want to obtain a feasible recommendation list regarding minimum cost and risk applying skyline query. The skyline objects for the private database of Organization 1 will be $P1$, $P2$, and $P4$, and the skyline objects for Organization 2 will be $Q1$, $Q3$, and $Q4$. However, the skyline objects of their joined databases will be $Q1$, $P4$, $Q4$, and $P2$. Though the object $P1$ and the object $Q3$ are in the skyline of Organization 1 and Organization 2, respectively, they do not exist in their joined skylines. The object $P4$ of Organization 1 dominates the object $Q3$ of Organization 2 and the objects $Q1$ of Organization 2 dominates the object $P1$ of Organization 1. Consequently, cost and risk estimation applying a skyline query is more authentic and meaningful if they are calculated from the data of both organizations. Hence, the parties want to determine skyline objects from their joined data. However, for security purpose, they do not want to reveal the attribute values in the objects with others. As a result, there requires a secured way that can calculate the skyline from joined data of both parties without exposing the real attribute values throughout the computation.

Table 4.1: Organization database.

Organization 1			**Organization 2**		
ID	**Cost**	**Risk**	**ID**	**Cost**	**Risk**
$P1$	27	33	$Q1$	22	30
$P2$	39	3	$Q2$	48	11
$P3$	45	15	$Q3$	32	25
$P4$	30	17	$Q4$	36	8
$P5$	45	30	$Q5$	42	37

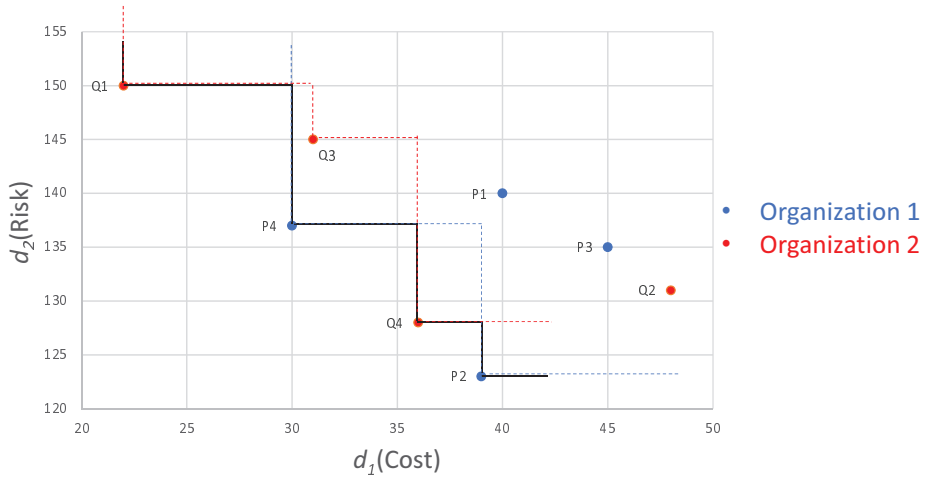


Figure 4.1: The skyline and multiparty skyline of Organizations 1 and 2.

Normally, skyline computation needs a large comparison of objects' attributes to settle whether an object is in the skyline or not. It may require multiple object dominance checks, and each check may include all of the objects' dimensions. Skyline calculation is both IO-consuming and CPU-intensive in centralized environments [30, 48]. Consequently, in the interest of overall performance, it is effective to compute skylines in distributed and parallel settings.

Besides, MapReduce is frequently used to process big data due to its scalability and fault tolerance. The availability of scalable and open-source MapReduce software, such as Hadoop [1], makes it serviceable for large-scale parallel skyline computation. Though the MapReduce framework normally has been designed in a local area network, which is maintained by one organization. The distributed computation can be extended to calculating the skyline from the union of databases that are maintained by multiple organizations. In such circumstances, users have to maintain each database independently and securely to the MapReduce framework. In this way, the distributed computing of MapReduce will be applied for multi-party databases. However, the security and the privacy of values throughout the processing of multi-party data should have to be preserved. As a result, the author thinks that privacy-aware computations of multi-party data in MapReduce have

to be considered.

Though a number of skyline calculation techniques [30, 38, 50, 35, 48] use the MapReduce framework to determine the skyline in a distributed environment, except [48], none of them consider the security concerns for the multi-party skyline. The previously-proposed secure skyline computation systems [48] only used MapReduce process for sorting the attribute values in the multi-party objects and required multiple rounds of MapReduce operations.

Furthermore, various approaches have been introduced for secure skyline query [26, 20, 27, 36]. The schemes in [26, 20] only considered the secure calculation of the skyline for clients from single party data saved in the cloud platform. The scheme in [27] computed the skyline from two-party data and interchange a significant amount of data throughout the secure comparison. It also required several two-party skyline computations to get the multi-party skyline. Though the previously-proposed encrypted substitution vector-based method [36] increased the efficiency, compared to other secure skyline computation methods, it needs to share an encrypted substitution vector before secure calculation of the skyline.

The proposed method introduces an efficient multi-party skyline computation system that computes skyline objects from multi-party data and protects the privacy of individual objects during multi-party skyline calculation. The method concurrently processes multi-party data, simultaneously performs operations for skyline computation in each phase, and uses only three rounds of MapReduce processes. For the proposed method, a minimum number of data exchanged is needed among the participants.

This chapter is organized as follows:

Section 4.2 explains the methodology of computing the secure multi-party skyline with an example; Section 4.3 specifies the scalability and the application of the method; Section 4.4 specifies the security issues; Section 4.5 provides the theoretical analysis of the

proposed method; Section 4.6 discusses the experiment details and explains the effectiveness and efficiency of the method under various settings; Section 4.7 concludes the proposed work.

4.2 Proposed Model

In the proposed system, it introduces a skyline computation process that can compute the skyline with the aides of coordinators utilizing multiple parties' databases with privacy assurance and security protection. The participating parties never desire to reveal the original attribute values of the databases; as a result, in the proposed approach, every party encrypted the database objects values before transferring them to the coordinators, and the coordinators figured out the skyline on the encrypted attribute values. The proposed method considered two coordinators: Coordinator 1 and Coordinator 2. Coordinator 2 compute the cell-wise skyline, and Coordinator 1 compute the final multi-party skyline. Through skyline computation, it considered four kinds of privacies. The privacies were:

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

In the proposed scheme, throughout the computation, it secured privacies 1, 2, and 3 were in Coordinator 2 and privacies 1, 2, and 4 were in Coordinator 1. The method considered all the parties and the coordinators as a semi-honest adversary. Consequently, they can attempt to infer private values throughout computation, but never interact with

each other and transfer any information with other parties except those allowed by the proposed system.

For performance, it concurrently computed the local skyline and encryption of the local skyline in each party, simultaneously performing operations in each coordinator.

Figure 4.2 represents the block diagram of the privacy-preserving skyline computation system. Here, Node 1, Node 2, ..., Node N want to compute the global skyline from their local databases without revealing their attribute values. Coordinator 1 and Coordinator 2 determine the global skyline from the data of all party without identifying the real attribute values of individual databases. The proposed algorithm has 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 explaining the process in a simplified manner, each step of the proposed method is described by two-dimensional data, as presented in Table 4.2. Here, Node A and Node B are two parties.

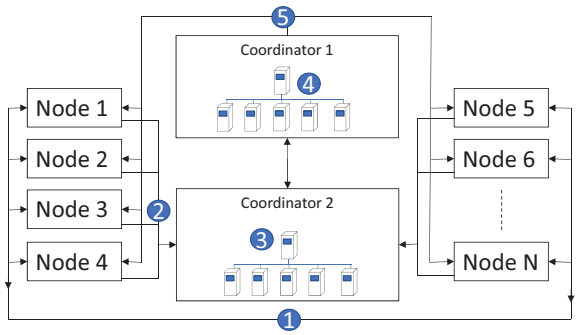


Figure 4.2: MapReduce-based multi-party secure skyline computation model.

Table 4.2: Data of Node A and Node B.

Node A			Node 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

4.2.1 Initialization

It is considered that all databases of the participating nodes held the identical database schema. Consequently, all participating nodes possessed the identical amount of attributes in each object, and the same number of bits was required in each attribute to save the values. Coordinator 1 starts the method by sending the signal to all nodes and gives its Paillier public key (Pk_{C1}) to all nodes with a random number R_{C1}. Coordinator 1 randomly chooses a node. The chosen node determines the OPES key for each attribute and the number of partitions they make in each attribute. Later, the chosen node transfers the OPE keys and partition for each attribute to all participating nodes.

4.2.2 Local Skyline, OPE, and Perturbation of Original Order

Before transferring any objects to the coordinator, each node determines the local skyline, does order-preserving encryption, and perturbs the original order. The procedures are explained in the subsequent subsections.

Local Skyline Computation

All the nodes 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 4.3a,b show the objects in the database of Node A and Node B. Figure 4.3c,d show the objects in the

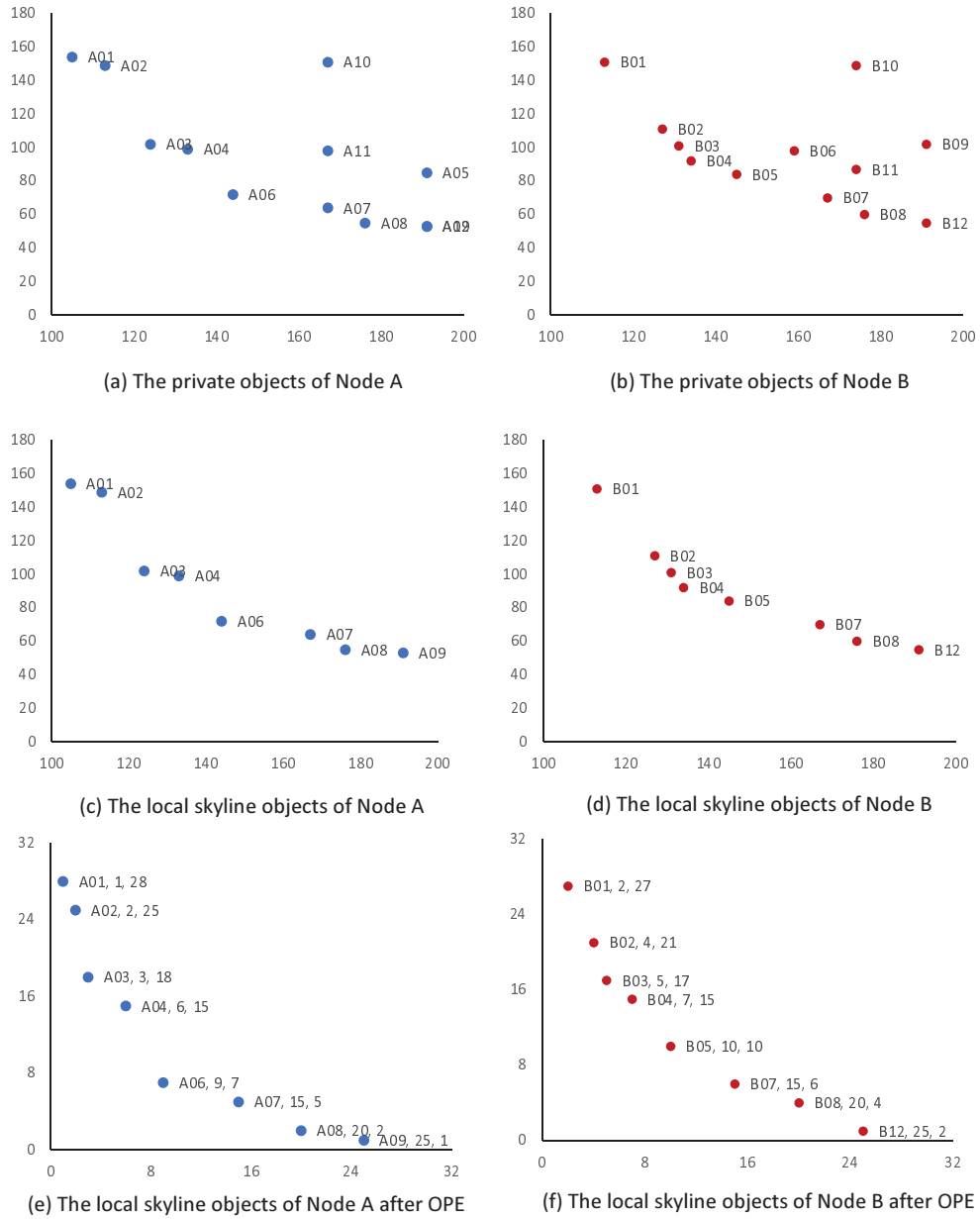


Figure 4.3: Local skyline of Node A and Node B from their private objects. OPE, Order-preserving Encryption.

OPE of Original Attribute Values in Local Skyline

Here, every node runs order-preserving encryption for each attribute value by implementing the OPE key (they get OPE keys for each attribute during initialization) for the corresponding attribute of the local skyline objects. Order-preserving encryption transforms the attribute values and distribution of the values but preserves the relative order in each attribute value. Figure 4.3c,d exhibit the local skyline of objects with the original attribute values of Node A and Node B. Figure 4.3e,f show the order-preserving encrypted attribute values of the local skyline of Node A and Node B.

Perturbation of the Original Order

OPE transforms the attribute values preserving the relative order of values in each attribute. The objects in the local skyline of all nodes have to be sent to Coordinator 2 for global skyline computation. If they give the values with the relative order in each attribute of objects, then the coordinator can examine the relative position of all the objects of the parties. This is also a vital privacy and security concern.

Consequently, each party perturbs the initial order of the values in each attribute prior to sending it to Coordinator 2. For perturbation, in each node, the object space is divided into several cells and changes the order of the attribute values in such a way that the order of values inside a cell is retained, but the order of the values in the outside of the cell is not maintained. It means, the order of values in the attributes with objects in other cells is not maintained.

Cells are constructed by dividing the domain of each attribute by the corresponding partition number. Since the partition number is given to each party in the time of initialization, all the parties will have an equal number of cells. Figure 4.4 explains the cell division by utilizing four partitions in each attribute.

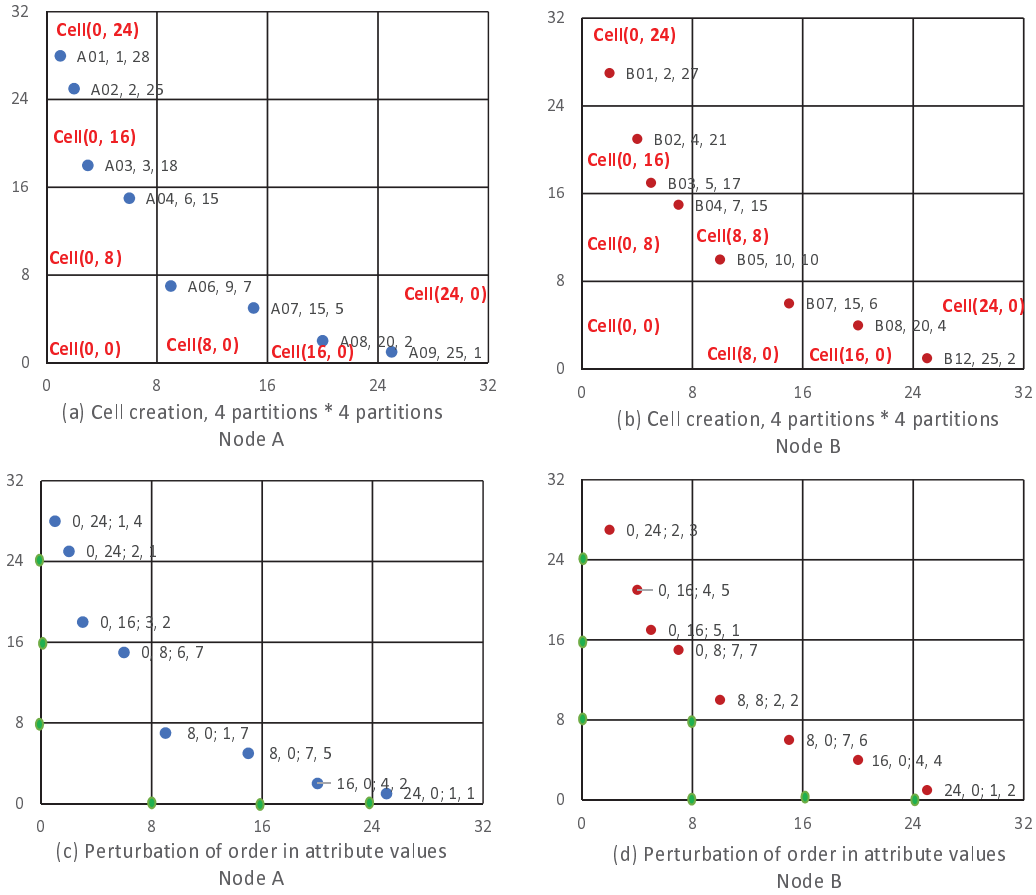


Figure 4.4: Cell creation and attribute value perturbation of objects in the local skyline of Node A and Node B.

Figure 4.4a,b show that Node A and Node B have the same number of cells. It considered the minimum integer attribute values in a cell as the cell id. For illustration, in Figure 4.4a, cell(0, 8) is a cell with id (0, 8) because zero and eight are the smallest integer attribute values in this cell.

The attribute values are subtracted in an object with the values in the cell id. As an illustration, in Figure 4.4a, the object A01 of Node A with attribute value (1, 28) in cell(0, 24) becomes (1, 28)−(0, 24) = (1, 4). Now an object can be considered as $< cellid >< subtractedvalue >$. For instance, objects with attribute value (1, 28) in cell(0, 24) can be considered as $< 0, 24; 1, 4 >$. Figure 4.4c,d show the object attribute value as $< cellid;\ value >$.

In each node, the cell id is encrypted for each object by the public key of Coordinator 1 and R_{C1} (all parties collect R_{C1} at the time of initialization). At the same time, the objects become $< encryptedid; values >$. For instance, 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 changes the inter-cell relative order in attribute values. All the nodes send the value of the object with the encrypted id to Coordinator 2. Each node completes all the tasks utilizing Algorithm 2 where Line 2 performs the divisions, Lines 5–12 forms the cell id, as well as the cell-wise attribute values of each object, and Line 13 transfers the objects with an encrypted cell id to Coordinator 2.

Algorithm 2: 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

for $i \leftarrow 1$ **to** n **do**
    div_i=$(2^{b_i} + 1)/Par_i$;
end
Define P as a temporary object;
for $j \leftarrow 1$ **to** m **do**
    $cellid$=null
    for $i \leftarrow 1$ **to** n **do**
        $P(i) = O_j(i)$ mod div_i `/* here,` $O_j(i)$ `means the` i`-th attribute of the` j`-th object */`
        $cell_i = O_j(i) - P(i)$ `/* cell id generation in the` i_{th} `attribute */`
        $O_j(i) = P(i)$ `/* value of the` i^{th} `attribute of the` j`-th object */`
        $cellid$ = concatenate ($cellid$, $cell_i$)
    end
    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 `*/`
end

4.2.3 Distributive Computation of Cell-Wise Candidate Skyline

Coordinator 2 accepts objects as $< encrypted\_cell\_id; attributes\_values >$ from all nodes. Since the ids of the objects are encrypted, it is impossible for Coordinator 2 to infer which id matches to which cell. As it is well known, relative order is adequate for a dominance check between two objects. Therefore, it is possible to compute the skyline from the relative order in each attribute values. Coordinator 2 could calculate the skyline of objects in every cell because the objects' attribute values in each cell preserve their relative order. Coordinator 2 utilizes the MapReduce to execute the skyline in every cell simultaneously. Figure 4.5a exhibits the objects received from all the nodes. Figure 4.5b explains the division 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 4.5c exhibits cell-wise objects. Figure 4.5d exhibits the candidate skyline objects in each cell following the reducing operation.

Coordinator 2 accomplish Pk_{C1}(cell id) + Pk_{C1}(value) for every objects in the cell-wise skyline. Because it is a homomorphic addition, that is why Pk_{C1}(cell id) + Pk_{C1}(value) = Pk_{C1}(cell id + value) (Figure 4.6). After doing the homomorphic addition, Coordinator 1. receives all the cell-wise skyline objects from Coordinator 2. Figure 4.6 illustrates the homomorphic addition process of cell id and values.

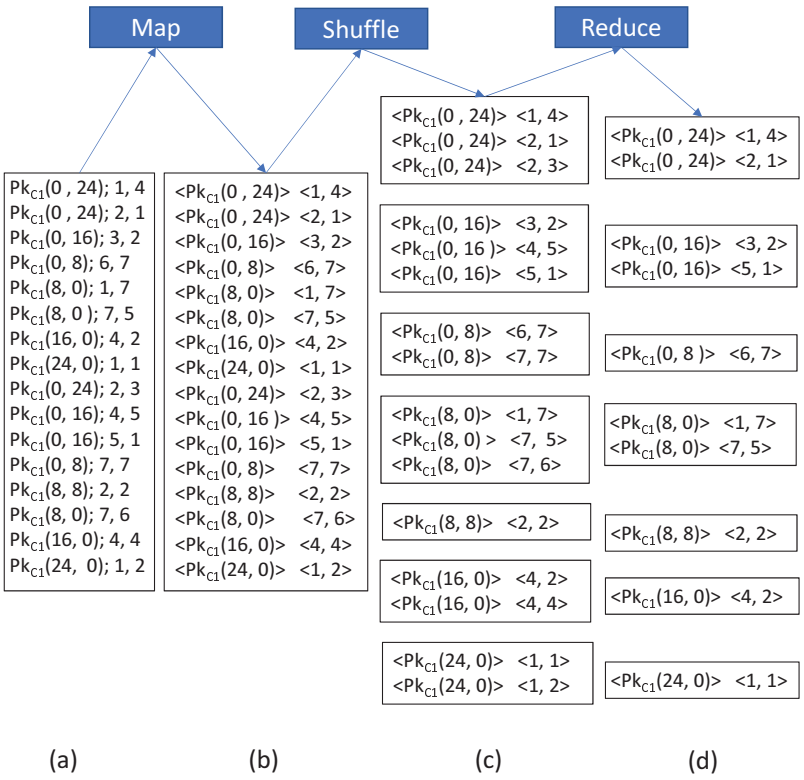


Figure 4.5: M ıs
(x, y) is encry

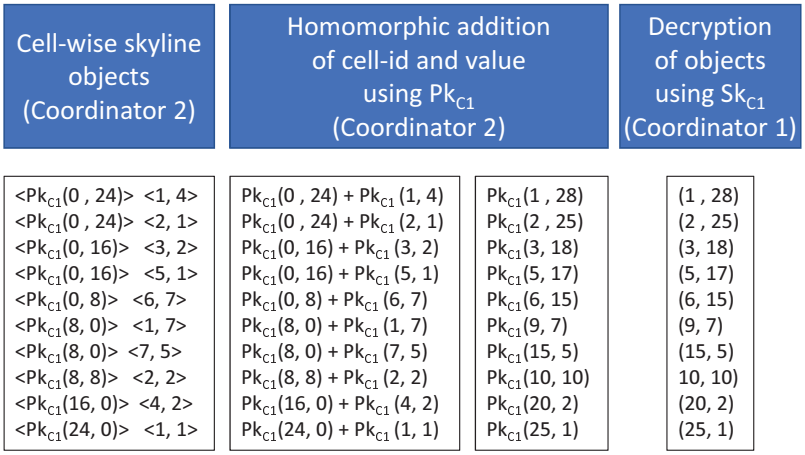


Figure 4.6: Homomorphic addition by Coordinator 2 and decryption of global skyline objects by Coordinator 1. Here, $Pk_{C1}(x, y)$ means (x, y) is encrypted by Coordinator 1's public key.

4.2.4 Global Skyline Computation from the Cell-Wise Candidate Skyline

Coordinator 1 utilize its secret-key (Sk_{C1}) and decrypts all the objects attribute values (Figure 4.6) and computes the global skyline concurrently in each node using the MapReduce framework. After calculation of the skyline, skyline objects are sent to all the nodes. Because the cell id and values are summed together before transferring each object to Coordinator 1, the coordinator can identify the initial order of all objects. Besides, the objects are straight coming from Coordinator 2 (not from other parties) to Coordinator 1. Consequently, Coordinator 1 has no idea about which objects arrived from which party; therefore, Coordinator 1 is unable to analyze the relative order of the objects of a particular party.

4.2.5 Decryption of the Global Skyline

Dominated objects are filtered out in every stage, and at last, only the non-dominated objects remained. It is known that the skyline is the set of non-dominated objects; as a consequence, all the parties obtain the required global skyline. Each party receives the global skyline where every attribute value is encrypted with the OPE key for a particular attribute. Later, they decrypt the attribute values utilizing the OPE key for every attribute and get the global skyline in plaintext mode. Table 4.3 presents the decrypted values of the global skyline.

Table 4.3: 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

4.3 Scalability and Application of the Proposed Method

Local skyline computation, OPE, and perturbation are executed simultaneously in each node, so the scheme is scalable as the number of participating parties grows. The system utilized the Hadoop MapReduce distributed computing system. Because MapReduce is a profoundly scalable and distributed arrangement, the method is scalable for a higher number of nodes or a significant size of data.

The work may be useful wherever there are multi-party datasets, and they willing to achieve secure computation without revealing the original values in the data. It can easily extend the work "The inferring fine-grained urban flows" [25] to multi-party secure urban traffic flow analysis using the proposed method. Spatiotemporal calculation utilizing ST(Spatiotemporal) -Hadoop [4] can easily extended for multi-party secure computation utilizing ST(Spatiotemporal) -Hadoop [4]. The concept of the proposed method may be deployable for secure computation of K-nearest skyline query in spatiotemporal databases [21]. In the case of the urban area, if there exist multiple water quality testing services, then the proposed method can utilize to extend the work introduced by Ye Liu et al. [28] for the secure examination of data from multiple water-quality examination service datasets.

4.4 Privacy and Security

To develop the secured privacy-preserving skyline computation system, the author employed order-preserving encryption and Paillier encryption to satisfy the secure computation and data privacy demands. It is known that in the semi-honest adversary model [18], no party is permitted to share data with any other party breaking the protocol specified to each party.

Because no party supplies their secret data with other parties; as a result, the individual party has no idea about data from other participants. Only information is shared with Coordinator 1 and Coordinator 2, so the privacy of data in Coordinator 1 and Coordinator 2 should be assured throughout the computation.

In Coordinator 2, order-preserving encryption is used to encrypt the values, and encryption of each cell id perturbed the original orders among the cells. Coordinator 2 can discover the original order of values if it is able to rearrange the cells accurately. Assume that the objects contain M number of attributes, and every attribute is split into N partition; as a result, the permutation N^M cells can be performed in $(N^M)!$ possible ways. Consequently, the probability of accurately organizing all cells will be $1/(N^M)!$. Furthermore, Coordinator 2 has no way to discover the correct arrangement of cells because each arrangement provides similar kinds of outcome, which are not distinguishable from each other. Hence, it is impossible for Coordinator 2 to infer the right order. Therefore, the privacy of the values in each attribute and the privacy of the relative order in each attribute are maintained throughout the computation of cells-wise skyline in Coordinator 2.

On the contrary, in Coordinator 1, it does not identify the actual values of each attribute because OPES encrypts the values. Nevertheless, it can detect the relative order of the values in each attribute of a minimal object, because a substantial amount of objects are filtered out throughout the local skyline calculation and cell-wise skyline calcula-

tion. Furthermore, Coordinator 1 does not identify which objects issued from which party. Consequently, it never able to analyze and determine the relative order of the objects of participating parties.

4.5 Theoretical Analysis of the Proposed Method

In this section, the author presents the theoretical comparison of the system with the existing systems. The computational complexity of the secure skyline in the proposed method depended on the following operations:

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

2. The time required for the calculation of the cell-wise candidate skyline.

3. The time required for the calculation of the global skyline.

All the nodes concurrently determine the local skyline, OPE, perturbation, and cell-wise values. Therefore, if the number of parties raises, the time needed for this step does not change. Besides, Coordinator 2 determines the cells-wise skylines simultaneously in each cell, filter out a substantial amount of dominated object, and improves the performance. Furthermore, at the time of global skyline computation, it calculates the global skyline concurrently in each node using the MapReduce framework; thus, it increases the performance. Moreover, throughout the calculation, the coordinators do not require to exchange any data with the participating parties or other coordinators; this also improves the overall performance.

The method suggested in [27] uses comparison to matched the pair-wise objects' attributes and determined the dominance of objects of two parties. In their proposed scheme, they did not use the coordinator for calculating the multi-party skyline. Consequently, it

cannot process multiple parties data concurrently; it can be only able to calculate the skyline between two parties' objects. As a result, for n parties, it requires ${}^{n}C_2 = \frac{n(n-1)}{2}$ two-party skyline computations to measure n-party skyline. Furthermore, it needs a secure comparison of values in each object attribute for the dominance comparison, which is also very time-consuming and requires many rounds of data exchange among the parties to compare each attribute value.

On the contrary, the complexity of the proposed system depends on the number of total local skyline objects of all parties, not depends on the number of participants. Furthermore, it does not need any series of data exchange among any participating parties and also compares objects directly on the encrypted values and applies the dominance check.

In the system proposed in [48], all the participating parties organized their database objects' order with the aid of a semi-honest third party, named as the *coordinator*. To produce the combined order in each attribute, every digit in the attribute required one step of MapReduce process in the coordinator. For instance, if there were D attribute, and each attribute held M digits, $D * M$ MapReduce steps were needed for order creation and one extra step for generating the skyline. In the proposed work, it only required three MapReduce steps.

The work [36] enhanced the performance, in contrast with other skyline frameworks for secure skyline computation. But it needed to compute and distribute an encrypted substitution vector to all the parties before secure computation of the skyline. For example, if we consider 32-bit integer values and 16-bit partitions for encrypted substitution vector, it required 2×2^{16} 32-bit integer values as a substitution vector and should be computed and distributed among the participants before skyline computation.

4.6 Experimental Analysis of the Proposed Method

4.6.1 Experimental Setup and Datasets

Here, the author presents and analyze the performance and efficiency of the proposed system. For each node, the scheme used machines with a fourth-generation Intel® Core™i7, 3.4-GHz CPU, and 8 GB main memory, operating on the 64-bit Microsoft Windows 10 Enterprise edition system. For Coordinator 1 and Coordinator 2, the system is configured as a group of commodity PCs in a high-speed Gigabit network, all of which had an Intel E8500 3.16-GHz CPU and 8 GB memory. The source codes are compiled with Hadoop core and Java V8. Hadoop Version 2.5.2 and 64-bit Cent-OS 7 are used throughout the process of the skyline.

The evaluation of the suggested privacy-preserving secure skyline scheme is performed in a multi-party distributed setting on synthetic datasets. The system applied randomly produced integer values, in which it considers three kinds of data distributions: correlated, anticorrelated, and independent.

4.6.2 Analysis of Our Proposed Method for Different Data Distributions

It is known that the standard approach to examine the skyline calculation is how the complexity of the computation changes with correlated, anticorrelated, and independent input distributions. Most of the relevant literature applied these three distributions to examine the complexity of skyline calculation. Typically, correlated data produce fewer non-dominated objects in skyline calculation, thus demanding a shorter time. On the contrary, anticorrelated data create a large amount of non-dominated objects in skyline calculation; consequently, they need the longest time. Nevertheless, the amount of skyline objects the independent data provides is in between the amount of non-dominated objects produced by correlated and anticorrelated data. It also requires to test how the complexity of the proposed scheme changed with various data distributions. For this experiment, each

participating parties' object numbers are ranged from 10–50 k, each object holding two attributes, and values were in a 32-bit integer and also used 30 partitions per attribute.

As in the Figure 4.7 illustrates that the skyline reckoning was more efficient for the correlated da ficiency for the indepe :related data. For cor ut local skyline calcul ount of objects is filt . From the figure, the y raised because it req objects of the others.

Figure 4.7: Running time varies with data distribution (attribute: 2, partitions: 30/attribute, value: 32-bit).

4.6.3 Analysis of Our Proposed Method with Variation in Object Dimensions

Another approach to examining the complexity of skyline calculation is how the calculation time changes with the change in object dimensions. It is known that skyline calculation needs comparison in every dimension to measure non-dominating objects from the data.

As a result, ᵗ ɔject

dimensions. I ɪ the

change in the

Figure 4.8 alcu-

lating skyline ɪ 2–6

and explain h ions.

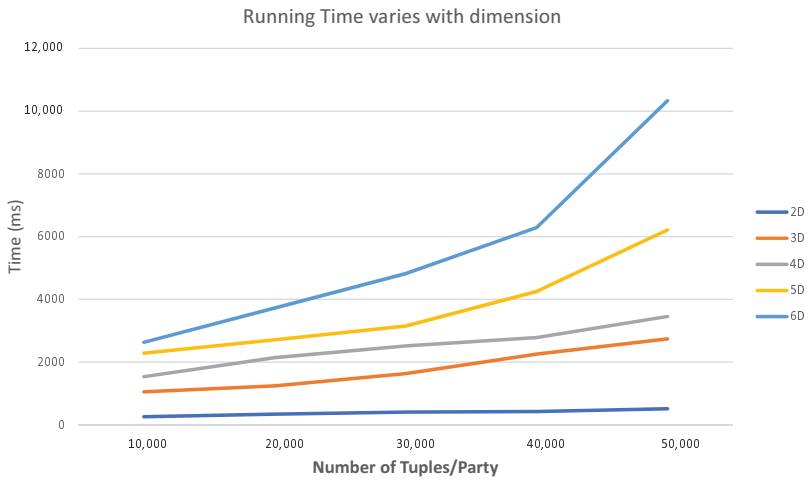


Figure 4.8: Running time varies with objects attributes (distribution: independent, partitions: 30/attribute).

Because the number of needed attribute partitions along with the number of comparisons and the amount of qualified local skyline objects grow with the object dimension, the execution time of skyline computation therefore increases. Consequently, in Figure 4.8, it is seen that the execution time grew when object dimensions increased. It is also seen that the time required for skyline computation raised when the number of objects per party grew because it required a dominance check for every object of one participant with the objects of the others.

4.6.4 Comparison with the ESV Method

An encrypted substitution vector (ESV) scheme has been introduced for multi-party skyline calculation in a distributed manner. The system is efficient in term of computation time compared with other contemporary multi-party secure skyline calculations. In this segment, the comparison of the suggested method is considered with the ESV based system. For this analysis, each party had 10k to 50 k objects, each object carrying two attributes, and the values in each attribute held a 32-bit value. It is also supposed that attribute values are divided into 30 partitions. In the case ESV based system, attribute values are partitioned into 11-bit bit-slice length for generating the encrypted substitution vector.

The ESV-based system needs determining and distributing an encrypted substitution vector among the parties before the secure calculation of the skyline. Furthermore, it does not use the simultaneous calculation of the skyline in the coordinator. In the suggested approach, the interchange of data among the participants was only OPE keys for each dimension and the number of partitions in each dimension. The operations in every stage are performed simultaneously to calculate the global skyline. Therefore, the measurement outcomes explain that the suggested system required less time than the ESV based system. Figure 4.9a–c illustrates that the suggested system outperformed the ESV based system for the independent, correlated, and anticorrelated data.

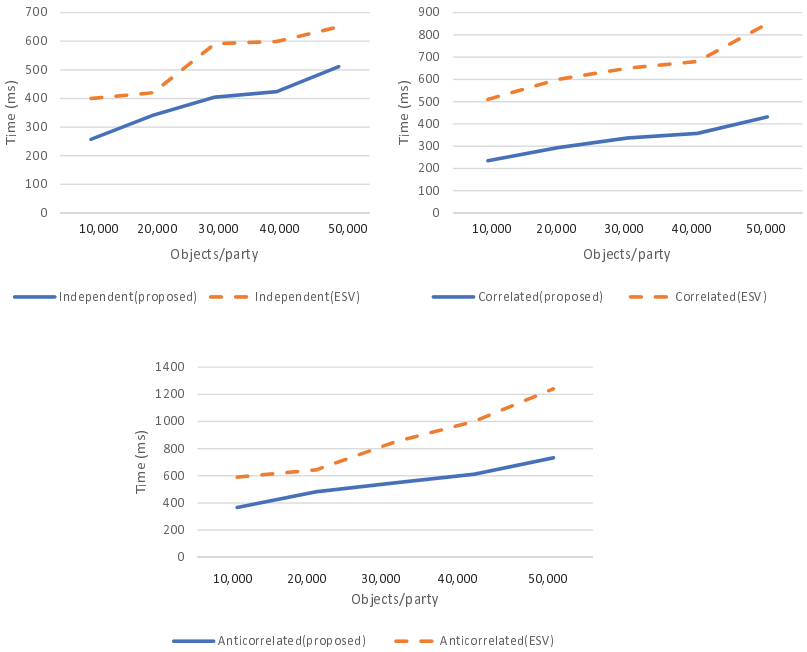


Figure 4.9: Running time comparison with ESVand the proposed method in different data distributions (attribute: 2, partitions: 30/attribute, value: 32-bit, bit-slice length: 11-bit, slices/attribute:3). ESV, encrypted substitution vector.

4.6.5 Comparison with Variation in the Number of Participating Databases

From the earlier discussion, the method is efficient even for the increases in the number of parties. Here, the variation is made for the amount of participating parties and determined the time required for multi-party secure skyline computation. For this experiment, it is considered that each party has 50 k objects, each object containing two attributes, and each attribute value was a 32-bit unsigned integer. It is also considered that there are 30 partitions per attribute.

Figure 4.10 shows the time required for skyline computation with variation in the number of participating parties. The number of parties varied from 2–8. Since the 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 the proposed method

in Figure 4.1(

databases.

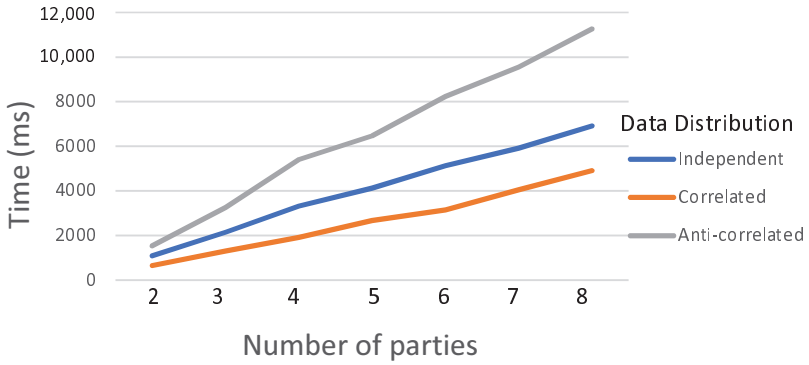


Figure 4.10: Running time varies with participating parties (attribute: 2, partition: 30/attribute).

4.7 Conclusions

In the proposed system, it efficiently manipulated multi-party data without revealing the real attribute values throughout the calculation of the secure skyline. It concurrently performs local skyline calculation and encryption of local skyline objects in each party. Furthermore, Coordinator 1 and Coordinator 2 parallelly conducted the processes for calculating the global skyline. It also kept a limited interchange of data among other participants throughout the calculation of the skyline. Therefore, the proposed system gave better performance. Both of the coordinators applied the MapReduce framework; accordingly, the method can process a big amount of data from multi-party with fault tolerance and cost-effectively.

Chapter 5

Conclusion

In this chapter, the author first presented the application of proposed system in Section 5.1. Then in Section 5.2, the author explained the critical contribution. And finally, in Section 5.3, the the author illustrates the future direction for improving the current systems.

5.1 Applications of proposed models

The proposed model (in Chapter 3) provides the process to securely store and manipulate data in an efficient way without disclosing the values and the orders of original data. In many practical scenarios like student merit position in an educational institute, the tender evaluation results etc. are sensitive to reveal the order information. The order information can be securely stored and manipulated by this method. For the last several decades, the skyline query is recognized to be a widespread query method for collecting valuable objects from datasets. The proposed model (in Chapter 4) capable of computing the skyline query without disclosing domain values from multiple datasets. An opportunity is created for collecting interesting objects where the datasets are associated with multiparty, and individuals are not interested in revealing the domain values at all. The proposed scheme is designed by utilizing Google's *MapReduce* framework, which provides the users to apply

the model in a position where traditional single-core algorithms are not efficient.

5.2 Contribution

Computational performance, as well as the secrecy of data, have been acquired significant awareness from the database analysis area for decades. In this study, the author analyzed two sophisticated features of secure data storage and skyline queries: (i) Semi-order Preserving Encryption (ii) Secure skyline query

5.2.1 Contribution Of SOPE

There are many circumstances in which no one expects to disclose order information. In such cases, the proposed advanced encryption method successfully protects the order information. It also gives an additional security layer to the OPES approach. Order-preserving encryption (OPE) produces ciphertexts that preserve the relative order of the underlying plaintexts. Thus, it is very suitable for range queries over encrypted outsourced data, as it is a famous case in cloud database scenarios. However, the order information in the ciphertext is also mater of concern for the organizations, especially for merit order in an educational institute, the salary of the employees in other organization, the order of tender evaluation, quotation serial etc. The proposed SOPE method can successfully hide the order information and also overcome the performance degradation issues. Execution time for the proposed methods is acceptable, as shown by experiments. We can execute the queries using different comparison operators without decrypting the original value. Therefore, our technique is suitable for practical use.

5.2.2 Contribution of Secure Skyline

The secure MapReduce based skyline query from multiple datasets owned by different parties is illustrated in Chapter 4. In the last several decades, the database researcher

community consider the skyline query as one of the popular and useful queries to analyze massive data. On the other hand, the conventional skyline computation method incapable of computing any outcome if the attribute values in the database entity are sensitive or non-revealable. In many cases, the researcher only wants to know the skyline objects without knowing all objects values in the database domain. The computation becomes more critical when data are related to multiple organizations, and they do not want to disclose the objects attribute values to others. To process the massive data comes from the different organization is also a matter of concern.

The author proposed a novel method that can process the skyline query without disclosing the attribute values of the objects in the database domain. MapReduce is a distributed framework designed by Google. The Apache Hadoop is open-source software for reliable, scalable, distributed computing to run MapReduce based algorithms. The author develops a MapReduce based secure skyline computation method; that can compute skyline from massive data of various organization without disclosing the attribute values of objects in the database domain. Each party concurrently execute the operations in their local data. The coordinators distributively and simultaneously perform the processes in each phase during the computation of the skyline. Therefore, the experiment results show that the proposed method computationally efficient enough compare to other methods.

5.3 Future Direction

This dissertation the authors incorporates various direction to extends the current work for future research.

5.3.1 Semi-order preserving Encryption

As discussed in the Introduction section, there are many situations in which people do not want to reveal order information. In such cases, our proposed encryption technique

successfully hides the order information. It also provides an extra protection layer to the OPES method. Execution time for the proposed methods is acceptable, as shown by experiments. We can execute the queries using different comparison operators without decrypting the original value. Therefore, our technique is suitable for practical use.

So far, the feasibility of the proposed schemes is confirmed. The author is also considering the distributed computation of the comparison operators of the proposed schemes in the MapReduce framework.

5.3.2 Secure MR skyline query

The secure skyline query is one of the essential tools to select representative objects from a large dataset without disclosing the attribute values of multiple organizations. The proposed encryption technique can compute the skyline query in the MapReduce framework. It is a very efficient method to perform secure skyline query in a distributed manner, but it needs two coordinators. As a future direction, the author will consider the skyline query using one coordinator to compute the skyline query. Moreover, the author also considers securely compute other variations of skyline query such as top-k query and k-skyband query in a distributed manner on a MapReduce framework.

Reference

[1] Apache. welcome to apachetmhadoop. `http://hadoop.apache.org`. Accessed: 2019-05-01.

[2] F. N. Afrati, P. Koutris, D. Suciu, and J. D. Ullman. Parallel skyline queries. In *International Conference on Database Theory (ICDT)*, page 274284, 2012.

[3] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order-preserving encryption for numeric data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*, pages 563–574, 2004.

[4] Louai Alarabi, Mohamed F. Mokbel, and Mashaal Musleh. St-hadoop: a mapreduce framework for spatio-temporal data. *GeoInformatica*, 22(4):785–813, Oct 2018.

[5] Apache. Apache hadoop. In *http://hadoop.apache,org*. 2010.

[6] Mohammad Shamsul Arefin and Yasuhiko Morimoto. Privacy aware parallel computation of skyline sets queries from distributed databases. *2013 International Conference on Computing, Networking and Communications (ICNC)*, pages 186–192, 2011.

[7] Wolf-Tilo Balke, Ulrich Güntzer, and Jason Xin Zheng. *Efficient Distributed Skylining for Web Information Systems*, pages 256–273. Springer Berlin Heidelberg, 2004.

[8] G. Bebek. Anti-tamper database research: Inference control techniques. *Technical Report EECS 433 Final Report, Case Western Reserve University*, 433, 2002.

[9] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'Neill. Order-preserving symmetric encryption. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009*, pages 224–241, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[10] Alexandra Boldyreva, Nathan Chenette, and Adam O'Neill. *Order-Preserving Encryption Revisited: Improved Security Analysis and Alternative Solutions*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[11] S. Borzsonyi, D. Kossmann, and K. Stocker. The skyline operator. In *Proceedings of IEEE International Conference on Data Engineering (ICDE)*, pages 421–430, 2001.

[12] Nathan Chenette, Kevin Lewi, Stephen A. Weis, and David J. Wu. Practical order-revealing encryption with limited leakage. In *Fast Software Encryption (FSE)*, 2016.

[13] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang. Skyline with presorting. In *Proceedings of IEEE International Conference on Data Engineering (ICDE)*, pages 717–719, 2003.

[14] E. Dellis and B. Seeger. Efficient computation of reverse skyline queries. In *Proceedings of VLDB*, pages 291–302, 2007.

[15] Lakhmi C. Jain Editors: Barbara Catania. *Intelligent Systems Reference Library Volume 36 2013*. Springer Berlin Heidelberg, 2013.

[16] H. Hacigümüş, B.R. Iyer, C. Li, and S. Mehrotra. Executing sql over encrypted data in the databaseservice-provider model. *Proc. of the ACM SIGMOD Conf. on Management of Data, Madison,Wisconsin*, pages 216–227, 2002.

[17] Koki Hamada, Dai Ikarashi, Koji Chida, and Katsumi Takahashi. Oblivious radix sort: An efficient sorting algorithm for practical secure multi-party computation. *IACR Cryptology ePrint Archive*, 2014:121, 2014.

[18] Carmit Hazay and Yehuda Lindell. *Definitions*, pages 19–49. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[19] Katja Hose and Akrivi Vlachou. A survey of skyline processing in highly distributed environments. *The VLDB Journal*, 21(3):359–384, Jun 2012.

[20] J. Hua, H. Zhu, F. Wang, X. Liu, R. Lu, H. Li, and Y. Zhang. Cinema: Efficient and privacy-preserving online medical primary diagnosis with skyline query. *IEEE Internet of Things Journal*, pages 1–1, 2018.

[21] Yuan-Ko Huang and Zong-Han He. Processing continuous k-nearest skyline query with uncertainty in spatio-temporal databases. *Journal of Intelligent Information Systems*, 45(2):165–186, Oct 2015.

[22] Vincent Rijmen Joan Daemen. *The Design of Rijndael AES The Advanced Encryption Standard*. Springer International Publishing, 2002.

[23] Hua Lu Kasper Mullesgaard, Jens Laurits Pedersen and Yongluan Zhou. Efficient skyline computation in mapreduce. In *International Conference on Extending Database Technology (EDBT)*, pages 37–48, 2014.

[24] Donald Kossmann, Frank Ramsak, and Steffen Rost. Shooting stars in the sky: An online algorithm for skyline queries. In *Proceedings of the 28th International Conference on Very Large Data Bases*, VLDB '02, pages 275–286. VLDB Endowment, 2002.

[25] Yuxuan Liang, Kun Ouyang, Lin Jing, Sijie Ruan, Ye Liu, Junbo Zhang, David S. Rosenblum, and Yu Zheng. Urbanfm: Inferring fine-grained urban flows. *CoRR*, abs/1902.05377, 2019.

[26] J. Liu, J. Yang, L. Xiong, and J. Pei. Secure skyline queries on cloud platform. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 633–644, April 2017.

[27] Ximeng Liu, Rongxing Lu, Jianfeng Ma, Le Chen, and Haiyong Bao. Efficient and privacy-preserving skyline computation framework across domains. *Future Generation Computer Systems*, 62:161 – 174, 2016.

[28] Ye Liu, Yu Zheng, Yuxuan Liang, Shuming Liu, and David S. Rosenblum. Urban water quality prediction based on multi-task multi-view learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 2576–2582. AAAI Press, 2016.

[29] Zheli Liu, Xiaofeng Chen, Jun Yang, Chunfu Jia, and Ilsun You. New order preserving encryption model for outsourced databases in cloud environments. *Journal of Network and Computer Applications*, 59:198 – 207, 2016.

[30] Kasper Mullesgaard, Jens Laurits Pederseny, Hua Lu, and Yongluan Zhou. Efficient skyline computation in mapreduce. In *EDBT*, 2014.

[31] G Ozsoyoglu, D Singer, and SS Chung. Anti-tamper databases: Querying encrypted databases. *Proc. of the 17th Annual IFIP WG 11.3 Working Conference on Database and Applications Security, Estes Park, Colorado*, pages 133–146, 2006.

[32] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Proceedings of Advances in Cryptology - Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT) '99*, pages 223–238, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[33] Dimitris Papadias, Yufei Tao, Greg Fu, and Bernhard Seeger. Progressive skyline computation in database systems. *ACM Trans. Database Syst.*, 30(1):41–82, March 2005.

[34] Yoonjae Park, Jun-Ki Min, and Kyuseok Shim. Parallel computation of skyline and reverse skyline queries using mapreduce. In *Proceedings of International Conference on Very Large Data Bases (VLDB) Endowment*, volume 6-14, pages 2002–2013, August 2013.

[35] Yoonjae Park, Jun-Ki Min, and Kyuseok Shim. Parallel computation of skyline and reverse skyline queries using mapreduce. *Proc. VLDB Endow.*, 6(14):2002–2013, September 2013.

[36] Mahboob Qaosar, Asif Zaman, Md. Anisuzzaman Siddique, Annisa, and Yasuhiko Morimoto. Privacy-preserving secure computation of skyline query in distributed multiparty databases. *Information*, 10(3), 2019.

[37] João B. Rocha, Akrivi Vlachou, Christos Doulkeridis, and Kjetil Nørvåg. *AGiDS: A Grid-Based Strategy for Distributed Skyline Query Processing*, pages 12–23. Springer, 2009.

[38] Hyeong-Cheol Ryu and Sungwon Jung. Mapreduce-based skyline query processing scheme using adaptive two-level grids. *Cluster Computing*, 20(4):3605–3616, December 2017.

[39] Anthony Vinay Kumar S and A. Arya. Fastbit-radix sort: Optimized version of radix sort. In *2016 11th International Conference on Computer Engineering Systems (ICCES)*, pages 305–312, 2016.

[40] Bharath K. K. Samanthula, Hu Chun, and Wei Jiang. An efficient and probabilistic secure bit-decomposition. In *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, ASIA CCS '13, pages 541–546, New York, NY, USA, 2013. ACM.

[41] M. A. Siddique, H. Tian, and Y. Morimoto. Distributed skyline computation of vertically splitted databases by using mapreduce. In *DASFAA*, pages 33–45, 2014.

[42] M. A. Siddique, H. Tian, and Y. Morimoto. k-dominant skyline query computation in mapreduce environment. *IEICE Transactions on Information and Systems*, pages 1745–1361, 2015.

[43] Douglas R. Stinson. *Cryptography : theory and practice*. Chapman & Hall/CRC, Boca Raton, 2005.

[44] T. Veugen, F. Blom, S. J. A. de Hoogh, and Z. Erkin. Secure comparison protocols in the semi-honest model. *IEEE Journal of Selected Topics in Signal Processing*, 9(7):1217–1228, Oct 2015.

[45] S. Wang, B. C. Ooi, A. K. H. Tung, and L. Xu. Efficient skyline query processing on peer-to-peer networks. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 1126–1135, April 2007.

[46] Liangliang Xiao, I-Ling Yen, and Dung T Huynh. Extending order preserving encryption for multi-user systems. *IACR Cryptology ePrint Archive*, 2012:192, 2012.

[47] Ce Yang, Weiming Zhang, and Nenghai Yu. Semi-order preserving encryption. *Information Sciences*, 387:266 – 279, 2017.

[48] Asif Zaman, Md. Anisuzzaman Siddique, Annisa, and Yasuhiko Morimoto. Secure computation of skyline query in mapreduce. In Jinyan Li, Xue Li, Shuliang Wang, Jianxin Li, and Quan Z. Sheng, editors, *Advanced Data Mining and Applications*, pages 345–360, Cham, 2016. Springer International Publishing.

[49] Boliang Zhang, Shuigeng Zhou, and Jihong Guan. Adapting skyline computation to the mapreduce framework: Algorithms and experiments. In *Proceedings of DASFAA'11*, pages 403–414. Springer-Verlag, 2011.

[50] J. Zhang, X. Jiang, W. Ku, and X. Qin. Efficient parallel skyline evaluation using mapreduce. *IEEE Transactions on Parallel and Distributed Systems*, 27(7):1996–2009, July 2016.

List of Referred Publications

Referred Journals

J-1 Saleh Ahmed, Mahboob Qaosar, Asif Zaman, Md. Anisuzzaman Siddique, Chen Li, Kazi Md. Rokibul Alam and Yasuhiko Morimoto, "*Privacy-Aware MapReduce Based Multi-Party Secure Skyline Computation*", Information, MDPI, Switzerland,10(6):207(1-19), DOI: 10.3390/info10060207.

J-2 Saleh Ahmed, Annisa, Asif Zaman, Zhan Zhang, Kazi Md. Rokibul Alam and Yasuhiko Morimoto, *"Semi-Order Preserving Encryption Technique for Numeric Database"*, International Journal of Networking and Computing (IJNC), ISSN 2185-2847, Vol 9, Issue 1, Pages 111-129, January 2019.

Referred International Conferences

C-1 Saleh Ahmed, Annisa, Asif Zaman, Zhan Zhang, Kazi Md. Rokibul Alam, and Yasuhiko Morimoto *"Semi-Order Preserving Encryption Technique forNumeric Data to Enhance Privacy"*, Proceedings of the Fifth International Symposium on Computing and Networking (CANDAR, 17), pp:68-74, Aomori, Japan, November 19-22, 2017, DOI 10.1109/CANDAR.2017.39

Other Publications **(not in dissertation)**

Referred Journals

J-3 Chen Li, Annisa Annisa, Asif Zaman, Mahboob Qaosar, Saleh Ahmed and Yasuhiko Morimoto, "*MapReduce Algorithm for Location Recommendation by Using Area Skyline Query*", Algorithms, MDPI, Switzerland,Vol: 11, Issue 6, Page 191(1-15), doi:10.3390/a11120191.

Referred International Conferences

C-2 Saleh Ahmed, Mahboob Qaosar, Rizka Wakhidatus Sholikah and Yasuhiko Morimoto, *"Early Dementia Detection through Conversations to Virtual Personal Assistant"*, The 2018 AAAI Spring Symposium Series Technical Report on Beyond Machine Intelligence: Understanding Cognitive Bias and Humanity for Well-Being AI, pp. 198-203, Palo Alto, California, USA, March 26-28, 2018.

C-3 Mahboob Qaosar, Saleh Ahmed, Chen Li and Yasuhiko Morimoto, *"Hybrid Sensing and Wearable Smart Device for Health Monitoring and Medication: Opportunities and Challenges"*, The 2018 AAAI Spring Symposium Series Technical Report on Beyond Machine Intelligence: Understanding Cognitive Bias and Humanity for Well-Being AI, pp. 269-274, Palo Alto, California, USA, March 26-28, 2018.