SCALE: An Efficient Framework for Secure Dynamic Skyline Query Processing in the Cloud*

Weiguo Wang[1], Hui Li[1][0000−0003−2382−6289], Yanguo Peng[2], Sourav S Bhowmick[3], Peng Chen[1], Xiaofeng Chen[1], and Jiangtao Cui[2]

[1] School of Cyber Engineering, Xidian University, China
{wgwang,pchen97}@stu.xidian.edu.cn, {hli,xfchen}@xidian.edu.cn
[2] School of Computer Science and Technology, Xidian University, China
{ygpeng,cuijt}@xidian.edu.cn
[3] School of Computer Science and Engineering, Nanyang Technological University, Singapore
assourav@ntu.edu.sg

Abstract. It is now cost-effective to outsource large dataset and perform query over the cloud. However, in this scenario, there exist serious security and privacy issues that sensitive information contained in the dataset can be leaked. The most effective way to address that is to encrypt the data before outsourcing. Nevertheless, it remains a grand challenge to process queries in ciphertext efficiently. In this work, we shall focus on solving one representative query task, namely *dynamic skyline query*, in a secure manner over the cloud. However, it is difficult to be performed on encrypted data as its dynamic domination criteria require both subtraction and comparison, which cannot be directly supported by a single encryption scheme efficiently. To this end, we present a novel framework called SCALE. It works by transforming traditional dynamic skyline domination into pure comparisons. The whole process can be completed in single-round interaction between user and the cloud. We theoretically prove that the outsourced database, query requests, and returned results are all kept secret under our model. Empirical study over a series of datasets demonstrates that our framework improves the efficiency of query processing by nearly **three orders of magnitude** compared to the state-of-the-art.

Keywords: skyline, secure, cloud, query

1 Introduction

With the rapid expansion in data volumes, many individuals and organizations are increasingly inclined to outsource their data to public cloud services since they provide a cost-effective way to support large-scale data storage and query processing. As a major type of query and fundamental building block for various applications, *skyline query* [6] has become an important issue in database research for extracting interesting objects from multi-dimensional datasets. The skyline query processing is widely adopted in many applications that require multi-criteria decision making such as market research [12], location based systems [14], web services study [2], etc. The *skyline*

* corresponding author: Hui Li, hli@xidian.edu.cn.

operator filters out a set of interesting points based on a group of evaluation criteria from a large set of points. A point is considered as interesting, if there does not exist a point that is at least as good in all criteria and better in at least one criteria. However, similar to other types of query, outsourcing skyline query workload to a public cloud will inevitably raise privacy issues. Since a real-world database may often contain sensitive information such as personal electronic mails, health records, financial transactions, etc., a cloud service provider may illegally spy on the data and invade the privacy of the data owner and users.

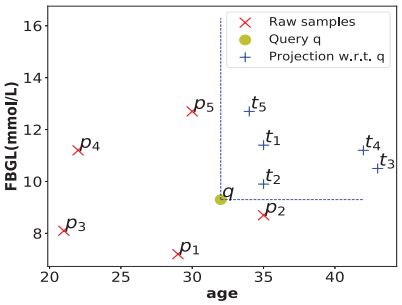
In this paper, we focus on the problem of *secure* skyline querying on the cloud aiming to protect the security of outsourced data, query request and results. Secure query processing on encrypted data has been extensively studied during recent years [17, 24]. For instance, fully homomorphic encryption schemes [13] ensure strong security while enabling arbitrary computations on the encrypted data. Modular Order-preserving encryption [4, 8] provides an intuitive security model which supports comparison over the ciphertext without decryption. Despite the promising achievements in the area of secure query processing, it remains a grand challenge for processing *dynamic* skyline queries over ciphertext, where the *skyline operator* is executed with respect to some query point [19] and is adopted in many applications [26]. The main reason for the problem is as follows. Given a query request, a dynamic skyline query requires performing *both* comparison and distance evaluation online simultaneously. Unfortunately, accomplishing this task over ciphertext cannot be realized efficiently via existing encryption schemes.

For instance, suppose that a medical institution wishes to outsource its electronic diabetes records to some public cloud service. Naturally, the medical institution would like to prevent any leak of the contents of the records to the cloud server. An electronic diabetes record consists of a series of attributes, including *ID*, *age*, *FBGL (fasting blood glucose level)*, etc. Let $P = \{p_1, \ldots, p_n\}$ denote a set of electronic diabetes records. When the medical institution receives a new record (i.e., patient) q, it expects the cloud server to retrieve a similar record to enhance and personalize the treatment for the new patient q. However, it is usually difficult or even impossible to uniformly assign weights to all the attributes to return the nearest neighbor (*e.g.,* p_1 is the nearest if only *age* is involved while p_2 is the nearest if only *FBGL* is taken into account). In light of that, dynamic skyline query provides all possible Pareto records that are not dominated by any other ones. Given a query q, we can compute the difference between each attribute for p_i and q. Let t_i be the difference tuple between p_i and q, and $t_i[j] = |p_i[j] - q[j]|$ for each dimension j. An object t_i dominates t_j if it is better than t_j in at least one dimension and not worse in every other dimensions. If an object cannot be dominated by any other object, this object is one of the skyline points that needs to be returned. As shown in Fig. 1, there are five patient records $p_1, \ldots, p_5$. Given a query record q, we calculate $t_1, \ldots, t_5$ and can easily identify the skyline points as t_5 and t_2. Therefore, p_5 and p_2 are the results for the dynamic skyline query w.r.t q.

Notably, in the above example, a dynamic skyline query requires performing both subtraction and comparison online. As there is no practical encryption scheme supporting both operators over ciphertext, existing model employs secure multiparty computation over at least two third-party non-collusion clouds and processes the query with

Records	Age	FBGL	Projected Age (w.r.t. q)	Projected FBGL (w.r.t. q)
p_1	29	7.2	35	11.4
p_2	35	8.7	35	9.9
p_3	21	8.1	3	10.5
p_4	22	11.2	42	11.2
p_5	30	12.7	34	12.7
q	32	9.3		

(a) Original and projected samples



(b) Dynamic skyline

Fig. 1: Dynamic Skyline Query Example.

multiple rounds of interactions. In this work, we present a novel framework called SCALE (**S**e**C**ure dyn**A**mic sky**L**ine qu**E**rying) by transforming traditional skyline domination criteria, which requires both subtraction and comparison, into comparison only. In this way, we are able to present a new scheme that can support dynamic skyline query over ciphertext without any help from a second cloud and can be completed in a single-round interaction between a user and the cloud. We theoretically prove that the outsourced database, query requests, and returned results are all kept secret under our model. Empirical study over four datasets including both synthetic and real-world ones demonstrate that our framework outperforms the state-of-the-art method by nearly **three orders of magnitude**. Notably, as a special case of dynamic skyline query, skyline computation can also be processed securely and efficiently under our model (with trivial modifications). In summary, this work makes the following contributions.

- We propose a new scheme to encrypt the outsourced database and query request. Based on the scheme, dynamic skyline query can be answered without decrypting the database or the query. Within the scheme, the cloud server and data user need only one interaction during the query.
- We theoretically prove that our model is secure if the cloud is curious-but-honest.
- Empirical study over both synthetic and real-world datasets justify that our model is superior to the state-of-the-art w.r.t the query response time.

The rest of this paper is organized as follows. In Section 2 we review related works. In Section 3 we formally present the problem definition and system model for this work. The detailed designs of encryption scheme and query framework are discussed in Section 4. Empirical study and corresponding results are shown in Section 5. In Section 6, we conclude this work.

2 Related Work

The skyline query is particularly important for several applications involving multi-criteria decision making. The computation of the skyline is equivalent to determining the maximal vector problem in computational geometry [15, 23], or equivalently the Pareto optimal set [23] problem. *static* skyline query has been extensively studied in the database field [6, 11, 21, 27]. A *dynamic* skyline query is a variation of skyline computation that was first introduced in [19, 20]. Instead of computing the skyline points

Table 1: Summary of notations

Notation	Definition
$\mathsf{Enc}(q)$ ($\mathsf{Enc}(2q)$)	Ciphertext of the query (doubled query) tuple
$P = \{p_1, \ldots, p_n\}$	A database with n tuples
$E(P)$	Ciphertexts of tuples for P
$E(\Phi)$	Ciphertexts of the pairwise sums for tuples in P
$p_i[j]$	The $j-th$ attribute of p_i
$keys[\cdot]$	The set of private keys

purely from the given dataset, dynamic skyline query returns series of points that are not dominated by any others with respect to q. In another word, skyline computation can be viewed as a special case of dynamic skyline query where q is fixed as origin point and only the comparison (without distance evaluation) is required.

With the development of cryptography, Encryption technology is gradually applied in the database field. Bothe *et al.* [7] presented an approach for skyline computation over Encrypted Data. It provided efficiency analysis and empirical study for computing skyline points and decrypting the results. However, it failed to provide any formal security guarantee. Another work [9] proposed three novel schemes that enable efficient verification of skyline query results returned by an unauthentic cloud server. This work focuses on the verification but not privacy issues, and does not work on ciphertext. It is orthogonal to the scope of this paper. Liu *et al.* [17] proposed the first semantically secure protocol for dynamic skyline query over the cloud platform. The scheme adopts both Paillier cryptosystem [18] and Secure Multi-party Computation (SMP) as building blocks. Although it is proved to be semantically secure, the protocol suffers from huge computation cost and strict system model. In fact, as a query framework, the response time is the most important issue for the success of the application, but the performance of [17] is far from satisfactory in this aspect.

Order-Preserving Encryption (OPE) scheme [1], whose ciphertext preserve the original ordering of the plaintexts, has been extensively applied in range query over encrypted databases. The ideal security goal for an order-preserving scheme, IND-OCPA [3], is to reveal no additional information about the plaintext values besides their order. Following that, a series of schemes have been proposed in literature [5, 22]. Chenette *et al.* [10] built efficiently implementable order-revealing encryption based on pseudorandom functions. Lewi *et al.* [16] improved the above scheme. The ORE scheme in [16] is adopted for this work. We will discuss it further in Section 4.

3 Problem Definition

In this section, we shall first introduce a group of key concepts for skyline query, then describe the system and security models utilized in this paper. For ease of discussion, the key notations used throughout this paper are summarized in Table 1.

3.1 Skyline Query Definition

In this part, we shall introduce a series of key concepts for skyline problem that is important for our subsequent discussions.

Definition 1 (Dynamic Domination). *Given two points p_α, p_β and a query point q in d-dimensional space, we say p_α dynamically dominates p_β with respect to q (denoted by $p_\alpha \prec_q p_\beta$), if $\forall i \in \{1, \ldots, d\}$, $|p_\alpha[i] - q[i]| \leq |p_\beta[i] - q[i]|$, and $\exists i \in \{1, \ldots, d\}$, $|p_\alpha[i] - q[i]| < |p_\beta[i] - q[i]|$.*

Definition 2 (Dynamic Skyline Query). *Given a dataset $P = \{p_1, \ldots, p_n\}$ and a query q in d-dimensional space, dynamic skyline query returns the set $S \subseteq P$, such that $\forall p \in S$, $\nexists p' \in P$ such that $p' \prec_q p$ (*i.e., *$\forall p \in S, p' \in P$, p' cannot dynamically dominate p with respect to q).*

A common algorithm (*i.e.,* BNL [6]) for dynamic skyline query is shown in Algorithm 1. It first calculates the differences (*i.e.,* t_i) between each tuple (*i.e.,* p_i) and the query request (*i.e.,* q) in every dimension (Lines 1-3). When a tuple p_i is read from P, it is added to S if S is empty (Lines 5-6). Otherwise, we shall compare p_i's corresponding difference tuple with respect to q, namely t_i, with that of each tuple in S. In case $t_i \prec t_j$, where $p_j \in S$, we shall delete p_j from S. If there is no $p_j \in S$ such that $t_j \prec t_i$, we shall add p_i to S (Lines 10-11, 16-18). The algorithm repeats this process for the remaining tuples in P, and finally returns S (Line 21). We shall use this as the basis for our secure skyline model. Notably, this is not the most efficient algorithm for plaintext skyline query. We select this method as our building block for the following reasons. Firstly, the state-of-the-art solution for secure dynamic skyline is [17], which adopts BNL [6] as the basic building block. In line with [17] and to make a fair comparison, our solution is constructed according to the same query framework. Secondly, BNL is a common and popular iterative algorithm for answering dynamic skyline query in plaintext. Thirdly, as discussed in Section 1, the key challenge in secure dynamic skyline query lies in the solution for performing both subtraction and comparison over ciphertext. A secure model building on any other (plaintext) dynamic skyline query algorithm inevitably has to address that. In other words, although our solution in this work adopts Algorithm 1 as the foundation, it can be easily adapted to other (plaintext) dynamic skyline query algorithms.

3.2 System Model and Design Goals

Our system model involves three types of participants: a data owner, a cloud server and a group of query users. The cloud server is assumed to have large storage and computation ability, and provide outsourcing storage and computation services. As Fig. 2 shows, the data owner employs the cloud service to store his private database. To preserve data privacy, the data owner will encrypt his dataset, and only outsource the encrypted dataset to the cloud. Every query user may submit a query point (*i.e.,* q) to the system. The query request may be locally encrypted before sending to the cloud server. Then, the cloud server will perform dynamic skyline query over encrypted database and query request without decryption. Afterwards, it returns the encrypted results to the user. Finally, the user decrypts these results using his own private keys.

Security model. We parameterize the security model by a collection of leakage functions $\mathcal{L} = (\mathcal{L}_{Encrypt}, \mathcal{L}_{Query}, \mathcal{L}_{Insert}, \mathcal{L}_{Delete})$. These functions describe what information the protocol leaks to the adversary $\mathcal{A}$. Our model ensures that the scheme does not reveal any information beyond what can be inferred from the leakage functions.

We define two games $\mathrm{Game}_{\mathcal{R},\mathcal{A}}$ and $\mathrm{Game}_{\mathcal{S},\mathcal{A}}$ as follows. The adversary repeatedly encrypts data and queries skyline points, and receives the transcripts generated

Algorithm 1 Basic Skyline Query Algorithm

Require: The dataset P and a query tuple q
Ensure: The result set of skyline points S

for i in $1, \ldots, n$ and j in $1, \ldots, d$ **do**
  let $t_i[j] = |p_i[j] - q[j]|$
end for
for i in $1, \ldots, n$ **do**
  if S is empty **then**
    add p_i to S
  else
    $flag \leftarrow True$
    for each $p_j \in S$ **do**
      if $t_j \prec t_i$ **then**
        $flag \leftarrow False$
      else if $t_i \prec t_j$ **then**
        delete p_j from S
      end if
    end for
    if $flag == True$ **then**
      add p_i to S
    end if
  end if
end for
return S

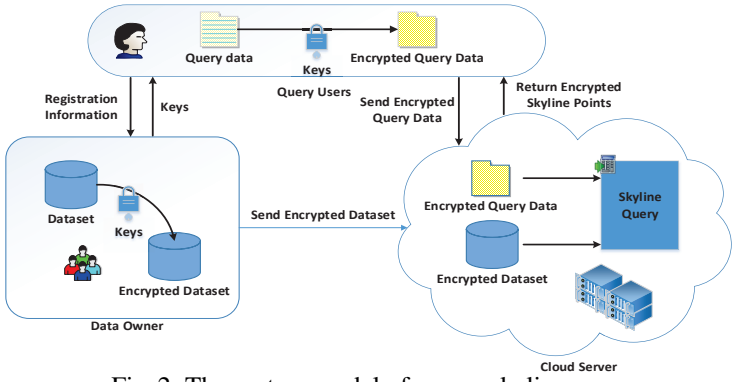


Fig. 2: The system model of secure skyline query

from $Encrypt()$ and $Query()$ algorithms in the real game $\mathrm{Game}_{\mathcal{R},\mathcal{A}}$ or receives the transcripts generated by the simulator $\mathcal{S}(\mathcal{L}_{Encrypt})$ and $\mathcal{S}(\mathcal{L}_{Query})$ in the ideal game $\mathrm{Game}_{\mathcal{S},\mathcal{A}}$. Eventually, $\mathcal{A}$ outputs a bit 0 ($\mathrm{Game}_{\mathcal{R},\mathcal{A}}$) or 1 ($\mathrm{Game}_{\mathcal{S},\mathcal{A}}$).

Definition 3 (Adaptively secure). *A scheme is $\mathcal{L}$-adaptively-secure if for all probabilistic polynomial-time algorithm $\mathcal{A}$, there exists an efficient simulator $\mathcal{S}$ such that:* $\left|Pr[\mathrm{Game}_{\mathcal{R},\mathcal{A}}(\lambda) = 1] - Pr[\mathrm{Game}_{\mathcal{S},\mathcal{A}}(\lambda) = 1]\right| \leq negl(\lambda)$.

Design goals. Our design goals contain both efficiency and privacy, including database privacy, query privacy, and result privacy. The details are as follows.

- Data owners need to encrypt the database before it is sent to the cloud server. Meanwhile, the content in the database is not leaked to the cloud server.
- Query request, as well as the results, should not be revealed to the cloud server throughout query processing.

– As a query processing framework, efficiency should be considered as one of the most important issue for measuring its success. Although the entire query processing is performed in ciphertext here, it should minimize the additional cost associated with it.

4 The SCALE Framework

As discussed above, processing dynamic skyline query given a query point q requires performing both subtraction and comparison. Addressing both tasks in ciphertext form is challenging as there is no practical encryption scheme that supports both operations simultaneously. To address this challenge, we re-investigate the entire dynamic skyline query workflow described in Definition 3.1 and Algorithm 1. Our investigation revealed an important fact that may lead to an effective solution. Notably, to answer a dynamic skyline query given a request q, quantifying the differences between each point p_i and q through all dimensions is not mandatory. Instead, what we need is the relative order of such differences for a group of different p_i.

Observation 1 *In order to evaluate whether p_α dynamically dominates p_β with respect to q, we do not need to know the exact values for the difference vectors T_α and T_β, where $T_i[j] = |p_i[j] - q[j]|$ for $j \in [1, \ldots, d]$. In fact, what we really need to know is whether $T_\alpha[j] \leq T_\beta[j]$ or $T_\alpha[j] < T_\beta[j]$ for $j \in [1, \ldots, d]$. For simplicity, for an arbitrary dimension j, we need to know whether $p_\alpha[j]$ or $p_\beta[j]$ is close to $q[j]$. To answer that, we have to consider two possible cases depending on whether $q[j]$ falls in the interval between $p_\alpha[j]$ and $p_\beta[j]$. Fig. 3a and Fig. 3b depict the cases. In Fig. 3a, the order between $T_\alpha[j]$ and $T_\beta[j]$ can be interpreted as the relationship between $p_\alpha[j]$ and $p_\beta[j]$. In the case of Fig. 3b, the order between $T_\alpha[j]$ and $T_\beta[j]$ can be interpreted as the relationship between $p_\alpha[j] + p_\beta[j]$ and $q[j] + q[j]$.*

In the aforementioned study, we notice that the multi-type-operation requirement (*i.e.,* with both subtraction and comparison) in dynamic skyline query can be transformed to **uni-type-operation involving only comparison**. Inspired by this critical point, current encryption schemes that support comparison over ciphertext can be adopted in our framework to realize our design goals.

4.1 Database Encryption

In our scheme, we adopt a state-of-the-art encryption scheme that supports comparison, namely *order-revealing encryption* [16]. We first present the formal definition of order-revealing encryption.

Definition 4 (Order-Revealing Encryption). *An order-revealing encryption (ORE) scheme [16] is a tuple of three algorithms including* Setup*,* Encrypt *and* Compare *defined over a well-ordered domain D with the following properties:*

- $\mathsf{Setup}(1^\lambda) \rightarrow sk$*: On input a security parameter λ, the setup algorithm outputs a secret key sk.*
- $\mathsf{Encrypt}(sk, m) \rightarrow ct$*: On input a secret key sk and a message $m \in D$, the encryption algorithm outputs a ciphertext ct.*

(a) Case 1. (b) Case 2.

Fig. 3: Cases for the relationship between q and (p_α, p_β)

Algorithm 2 SecureCompare Algorithm

Require: The ORE ciphertext for $\mathsf{Enc}(p_\alpha[j])$, $\mathsf{Enc}(p_\beta[j])$, $\mathsf{Enc}(q[j])$, as well as $\mathsf{Enc}(p_\alpha[j] + p_\beta[j])$, $\mathsf{Enc}(2q[j])$.
Ensure: The comparison result as $-1, 0, 1$ denoting that $p_\alpha[j]$ is closer to (*resp.*, equivalent with, farther from) $q[j]$ than $p_\beta[j]$.

if ORE.Compare($\mathsf{Enc}(p_\alpha[j]), \mathsf{Enc}(p_\beta[j])) == 0$ **then**
  return 0
else if ORE.Compare($\mathsf{Enc}(p_\alpha[j]), \mathsf{Enc}(p_\beta[j])) == -1$ **then**
  if $\mathsf{Enc}(q[j])$ falls outside the interval **then**
    return ORE.Compare($\mathsf{Enc}(q[j]), \mathsf{Enc}(p_\alpha[j])$)
  else
    return ORE.Compare($\mathsf{Enc}(2q[j]), \mathsf{Enc}(p_\alpha[j] + p_\beta[j])$)
  end if
else
  if $Enc(q[j])$ falls outside the interval **then**
    return ORE.Compare($\mathsf{Enc}(q[j]), \mathsf{Enc}(p_\beta[j])$)
  else
    return ORE.Compare($\mathsf{Enc}(p_\alpha[j] + p_\beta[j]), \mathsf{Enc}(2q[j])$)
  end if
end if

– $\mathsf{Compare}(ct_1, ct_2) \rightarrow b$: *On input two ciphertexts ct_1, ct_2, the compare algorithm outputs a bit $b \in \{-1, 0, 1\}$.*

With the help of the ORE scheme, evaluating the dynamic domination relation between p_α and p_β can be carried out securely in ciphertext form as outlined in Algorithm 2. For ease of subsequent discussion, we shall denote $\mathsf{Enc}(x)$ as the ORE ciphertext for the original message x.

Minimizing the number of keys. Following Observation 1, a data owner needs to encrypt database P and the sum of any two tuples in P in each dimension, namely $p_\alpha[j] + p_\beta[j]$, where $\alpha \neq \beta, \alpha, \beta \in [1, n], j \in [1, d]$. The above two ciphertexts are denoted as $E(P)$ and $E(\Phi)$, respectively. In this step, if we use the same private key on both $E(P)$ and $E(\Phi)$, the sum of paired tuples in $E(\Phi)$, although encrypted, will leak more message about plaintext beyond the order.

For example, assume that P contains five tuples, whose values in a particular dimension are a, b, c, d, e, respectively. Suppose that after sorting the values in ascending order, we get b, c, a, e, d. Then their sums can be listed as $b+c, b+a, b+e, b+d, c+a, c+e, c+d, a+e, a+d, e+d$. For ease of discussion, in the following we shall refer to these values as *pairs of sums*. If we encrypt the results for these pairs of sums using the same key as $E(P)$, an attacker can get the ordering of plaintexts. Therefore, he may possibly know $b+e \leq c+a$, and then infer that $e-a \leq c-b$. In this way, besides the order, the distribution of values in plaintext tuples is also leaked.

However, according to the security model in this work, except the order of tuples in some dimensions, the cloud should not be able to infer the content of the tuples.

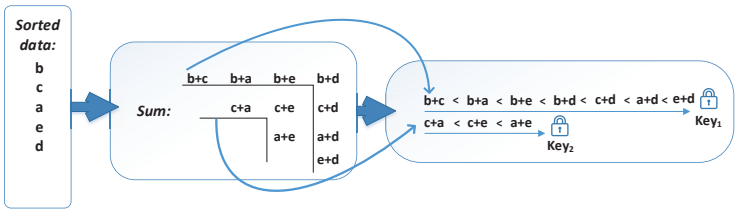


Fig. 4: A novel encryption scheme for pairs of tuples

Therefore, we have to avoid leaking the distribution of data by adopting different keys in ORE. Intuitively, an ideal method is to encrypt each pair of sums using a different key, as it is not required to perform comparison among any pair of $p_\alpha[j], p_\beta[j]$ according to Algorithm 2. However, the increased number of keys will further introduce key management and storage problems. We propose a novel method to address this problem. As shown in Fig. 4, b, c, a, e, d are the sorted values for five tuples in P on a particular dimension. According to Algorithm 2, these values should be encrypted using the same key as comparisons over their ciphertext are required. As a result, given that $\mathsf{Enc}(b), \ldots, \mathsf{Enc}(d)$ are encrypted using the same key under ORE, any adversary can easily infer that $b + c < b + a < b + e < b + d$ regardless that $b + c, \ldots, b + d$ are encrypted with different keys or not. Therefore, it is not beneficial to use multiple keys for such a group of sums.

Definition 5 (Order-Obvious Class). *Given the order of n elements, whose exact values are unknown, if the order of two summations over paired elements can be inferred, we call them Order-Obvious. All the $n(n-1)/2$ paired summations can be divided into several disjoint subsets accordingly, such that all the summations in each subset are Order-Obvious. We refer to each subset as an Order-Obvious Class (abbrev. OOC).*

Generally, we can find all *OOC*s, which is classified using the solid lines in Fig. 4. The relations for sums in the same *OOC* (*e.g.,* line) can be inferred easily purely from $E(P)$. In light of that, we can use the same key to encrypt the sums in the same *OOC*, and adopt different keys across *OOC*s. In this way, any adversary cannot get additional information over the ciphertexts besides the order, and we can effectively minimize the number of keys. In particular, the minimum number of keys, denoted as κ, (*e.g.,* the number of lines in Fig. 4) must satisfy the following theorem.

Theorem 1. *In order to satisfy the predefined security model, the minimum number of encryption keys in a dimension should be $\kappa = \lceil \frac{2*n-3}{4} \rceil$.*[4]

Remark. Through the above strategy, we have minimized the required number of encryption keys. In spite of that, κ is still linear to n, which may introduce key management burden if n is very large. To address this, we suggest the following implementations. For each row in Fig. 4, we assign it a random Id_i. The data owner only needs to store one master key mk and a series of random Id_i. Then, key_i for encrypting each row is generated by $mk \oplus Id_i$. In this way, we can effectively generate κ different keys based on mk.

[4] The proof for all theories can be found in our Technical Report [25]

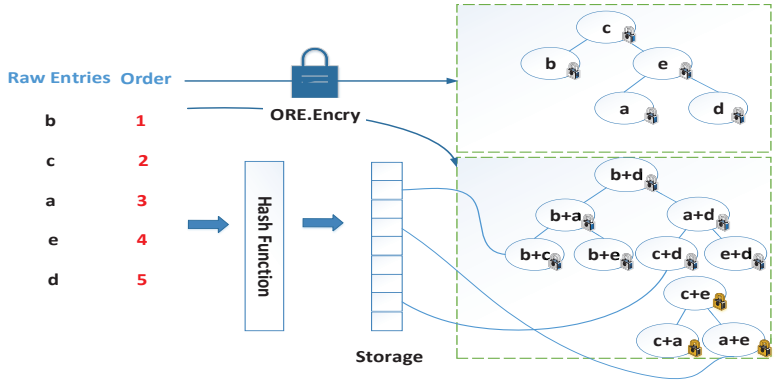


Fig. 5: The complete ciphertext storage structure

Accessing the pairs of sums. As required by Algorithm 2, in order to compare $t_\alpha[j]$ and $t_\beta[j]$, it is always required to retrieve the ciphertext of $p_\alpha[j] + p_\beta[j]$. Therefore, it is necessary to build a map between the elements of $E(P)$ with the corresponding sums in $E(\Phi)$. That is, we need to build an index that maps $\mathsf{Enc}(p_\alpha[j])$ and $\mathsf{Enc}(p_\beta[j])$ to $\mathsf{Enc}(p_\alpha[j] + p_\beta[j])$. To this end, we present an index based on hash function. Formally, we define a hash function as $h : \mathbb{N}^2 \rightarrow \mathbb{N}$, where $\mathbb{N}$ denote the set of natural numbers. The hash function h should satisfy the following property, $\forall x_1, y_1, x_2, y_2 \in \mathbb{N}$, $h(x_1, y_1) = h(x_2, y_2)$ if and only if $x_1 = x_2$ and $y_1 = y_2$.

Assume the indices for $\mathsf{Enc}(p_\alpha[j])$ and $\mathsf{Enc}(p_\beta[j])$ in $E(P)$ are denoted as α and β, respectively. Then the index of $\mathsf{Enc}(p_\alpha[j] + p_\beta[j])$ in $E(\Phi)$ can be easily acquired as $h(\alpha, \beta)$. Fig. 5 presents an example for the hash function. There are five encrypted values in $E(P)$, namely $a, \ldots, e$. The hash function in this example is simply designed as a regular traversal order for the corresponding sums. In fact, any hash function that satisfies the aforementioned property can be adopted here.

Indexing the pairs of sums. Additionally, as all the pairs of sums within a particular *OOC* are encrypted by ORE using the same key, we need to exploit additional index structures for efficient retrieval of corresponding entries for these *pairs of sums*. Therefore, we also design an index scheme for management of these ORE encrypted *pairs of sums*. In SCALE, we adopt AVL-Tree based structure to construct the indexing structure, as it provides excellent efficiency when querying for a particular range. Specifically, it is possible for us to build an AVL-Tree to index all these encrypted sums in the same *OOC*. Notably, each AVL-Tree is rooted at the median of each *OOC* and all the nodes in an AVL-Tree are the corresponding ciphertexts for *pairs of sums* in the same *OOC*.

For instance, given the records in Fig. 4, there are two *OOC*s. We shall build two different AVL-Trees for indexing the corresponding ciphertexts for each *OOC*, respectively. That is, the first *OOC* centered at $b + d$ corresponds to an AVL-Tree rooted at $\mathsf{Enc}(b + d)$; another *OOC* centered at $c + e$ corresponds to another AVL-Tree rooted at $\mathsf{Enc}(c + e)$ (as shown in Fig. 5).

In fact, data structures other than AVL-Tree can also be adopted to index the ORE ciphertexts for each *OOC*. We select AVL-Tree as the default setting in SCALE as it provides the best query response time among alternative choices.

Algorithm 3 Dataset Encryption

Require: The dataset P
Ensure: The ciphertexts sets $E(P)$, $E(\Phi)$
1: generate $d + \lceil \frac{2*n-3}{4} \rceil * d$ keys with ORE.Setup as $keys[]$
2: **for** $p \in P$ and j in $1, \ldots, d$ **do**
3: $\mathsf{Enc}(p[j]) \leftarrow \mathsf{ORE.Encrypt}(keys[j], p[j])$
4: let $\mathsf{Enc}(p) = \{\mathsf{Enc}(p[1]), \ldots, \mathsf{Enc}(p[d])\}$ and add $\mathsf{Enc}(p)$ to $E(P)$
5: **end for**
6: let $m = 1$
7: **for** j in $1, \ldots, d$ **do**
8: $\Lambda = (p^{(1)}[j], \ldots, p^{(n)}[j]) \leftarrow$ sort $p_1[j], \ldots, p_n[j]$ in ascending order
9: **while** Λ is not empty **do**
10: **for** i in $2, \ldots, len(\Lambda)$ **do**
11: add $\mathsf{ORE.Encrypt}(keys[d+m], p^{(1)}[j] + p^{(i)}[j])$ to $E(\Phi)$
12: **end for**
13: **for** i in $2, \ldots, len(\Lambda) - 1$ **do**
14: add $\mathsf{ORE.Encrypt}(keys[d+m], p^{(n)}[j] + p^{(i)}[j])$ to $E(\Phi)$
15: **end for**
16: remove the first and last elements in Λ, let $m = m + 1$
17: **end while**
18: **end for**
19: **return** $E(P)$, $E(\Phi)$

Algorithm 4 Query Request Encryption

Require: The query data q, keys from data owner $keys[]$
Ensure: The ciphertexts $\mathsf{Enc}(q)$, $\mathsf{Enc}(2q)$
1: **for** j in $1, \ldots, d$ **do**
2: $\mathsf{Enc}(q[j]) \leftarrow \mathsf{ORE.Encrypt}(keys[j], q[j])$
3: **for** m in $1, \ldots, \lceil \frac{2*n-3}{4} \rceil$ **do**
4: let $key\_num = d + (j-1) * \lceil \frac{2*n-3}{4} \rceil + m$
5: $\mathsf{Enc}(2q_m[j]) \leftarrow \mathsf{ORE.Encrypt}(keys[key\_num], 2q[j])$
6: **end for**
7: **end for**
8: **return** $\mathsf{Enc}(q)$, $\mathsf{Enc}(2q)$

Database encryption. We have now all the ammunitions in place to demonstrate the entire process of encrypting the database (Algorithm 3). First, the data owner generates $d + \lceil \frac{2*n-3}{4} \rceil * d$ keys (Line 1), and for each column (*i.e.,* attribute) in P we encrypt the entries using the same key (Lines 3-5), resulting in $E(P)$. Then, the data owner sorts the entries (Line 8) in each column (*i.e.,* attribute) and computes the sums for pairs of entries in each dimension. Afterwards, the sums are then encrypted using the corresponding keys as shown in Fig. 4 (Lines 9-17), resulting in $E(\Phi)$. Finally, the data owner sends $E(P)$, $E(\Phi)$ to the cloud server.

Besides, the data owner also creates a hash function h that maps each pair of elements in $E(P)$ and the corresponding sums in $E(\Phi)$, and sends h to the cloud server. It is now possible for the cloud to quickly locate the ciphertext of the corresponding sums for each pair $(p_\alpha[j], p_\beta[j])$.

4.2 Query Processing

A data user needs to register their information to the Data owner and securely get the keys. Then the query user encrypts the request according to Algorithm 4 before sending it to the cloud server.

Algorithm 5 Secure Skyline Query Algorithm

Require: The ciphertext for dataset $E(P)$ and $E(\Phi)$, query request $\mathsf{Enc}(q)$ and $\mathsf{Enc}(2q)$
Ensure: The encrypted result set of skyline points S

```
1: for i in 1, ..., n do
2:     if S is empty then
3:         add Enc(p_i) to S
4:     else
5:         flag_cur ← True
6:         for each Enc(p_j) ∈ S do
7:             for m in 1, ..., d do
8:                 flag[m] ←SecureCompare(Enc(p_i[m]), Enc(p_j[m]), Enc(q[m]), Enc(p_i[m] + p_j[m]),
                   Enc(2q[m]))
9:             end for
10:            if ∀m, flag[m] ≥ 0, and ∃k such that flag[k] > 0 then
11:                flag_cur ← False
12:            else if ∀m, flag[m] ≤ 0, and ∃k such that flag[k] < 0 then
13:                delete Enc(p_j) from S
14:            end if
15:        end for
16:        if flag_cur is True then
17:            add Enc(p_i) to S
18:        end if
19:    end if
20: end for
21: return S
```

As shown in Algorithm 4, user encrypts each dimension of the query tuple using corresponding keys (Line 2) and encrypts the doubled entries for the query tuple using other keys (Lines 4-5). Finally, the user sends $\mathsf{Enc}(q)$, $\mathsf{Enc}(2q)$ to the cloud server. As mentioned in Algorithm 1, given an encrypted query q as shown Algorithm 4, the cloud server needs to perform comparisons and computations over encrypted data. According to the approach shown in Fig. 3, the cloud server can perform skyline query via the comparison relationship with encrypted tuples, encrypted query request, encrypted sums, and encrypted doubled request. As a result, the process described in Algorithm 1 can be now performed in ciphertext without decryption, which is shown in Algorithm 5. To illustrate the entire protocol, we provide a running example in the following.

Example 1. Assuming that P contains five tuples, whose entries in dimension 1 are sorted as $7, 13, 21, 32, 53$. For simplicity, hereby we show only one dimension. According to Algorithm 3, we shall first compute the sums for all pairs of values, *e.g.,* $7+13=20$, $7+21=28$, $7+32=39$, $7+53=60$, $13+21=34$, $13+32=45$, $13+53=66$, $21+32=53$, $21+53=74$, $32+53=85$. As shown in Theorem 1, the number of encryption keys required for these sums can be calculated as $\lceil \frac{2*5-3}{4} \rceil = 2$. Therefore, we use two keys to encrypt the above sums, resulting in $\mathsf{Enc}_1(20)$, $\mathsf{Enc}_1(28)$, $\ldots$, $\mathsf{Enc}_1(85)$ and $\mathsf{Enc}_2(34)$,$\mathsf{Enc}_2(45)$,$\mathsf{Enc}_2(53)$.

Besides, we also need to use another key to encrypt the original tuples, *e.g.,* $\mathsf{Enc}_3(7)$, $\mathsf{Enc}_3(13), \ldots, \mathsf{Enc}_3(53)$. Suppose that a user submits a query with $q[1]=23$. Then q and $2q$ need to be encrypted according to our scheme, resulting in $\mathsf{Enc}_1(46), \mathsf{Enc}_2(46)$, $\mathsf{Enc}_3(23)$. These ciphertexts are then sent to the cloud server. The cloud server compares ciphertexts one by one according to the protocol. Through $\mathsf{ORE.Compare}$ (Definition 4.1) and Algorithm 2, the cloud server can easily determine that $\mathsf{Enc}_3(32)$ dominates $\mathsf{Enc}_3(53)$ following the case shown in Fig. 3a. Similarly, $\mathsf{Enc}_3(21)$ dominates $\mathsf{Enc}_3(7)$ and $\mathsf{Enc}_3(13)$. In the case shown in Fig. 3b, $\mathsf{Enc}_3(21)$ dominates $\mathsf{Enc}_3(32)$

because $\mathsf{ORE.Compare}(\mathsf{Enc}_2(53), \mathsf{Enc}_2(46)) = 1$. Algorithm 5 will iteratively repeat this process for all dimensions and remaining tuples. ■

4.3 Security Analysis

The presented SCALE framework is constructed based on ORE scheme proposed in [16], which is secure with leakage function $\mathcal{L}_{BLK}$.

Lemma 1. *The ORE scheme is secure with leakage function $\mathcal{L}_{BLK}$ assuming that the adopted* pseudo random function *(PRF) is secure and the adopted hash functions are modeled as random oracles. Here, $\mathcal{L}_{BLK}(m_1, \ldots, m_t) = \{(i, j, BLK(m_i, m_j))|1 \leq i < j \leq t\}$ and $BLK(m_i, m_j) = (ORE.Compare(m_i, m_j), \mathtt{ind}_{\mathtt{diff}}(m_i, m_j))$, in which $\mathtt{ind}_{\mathtt{diff}}$ is the first differing block function that is the first index $i \in [n]$ such that $x_i = x_j$ for all $j < i$ and $x_i \neq x_j$. (The proof of this lemma is in Appendix 4.1 in [16])*

In order to formally prove the security of SCALE, we extend $\mathcal{L}$-adaptively-secure model for keyword searching scheme as shown in Definition 3.

Theorem 2. *Let the adopted PRF in ORE is secure. The presented* SCALE *framework is $\mathcal{L}$-adaptively-secure in the (programmable) random oracle model, where the leakage function collection $\mathcal{L} = (\mathcal{L}_{Encrypt}, \mathcal{L}_{Query}, \mathcal{L}_{Insert}, \mathcal{L}_{Delete})$ is defined as follows,*

$$\mathcal{L}_{Encrypt} = \mathcal{L}_{BLK}(\cup_{k=1}^{d} X_k^{(n)}), \mathcal{L}_{Query} = \mathcal{L}_{BLK}(\cup_{k=1}^{d} X_k^{(n)'}),$$
$$\mathcal{L}_{Insert} = \mathcal{L}_{BLK}(\cup_{k=1}^{d} X_k^{(n+1)}), \mathcal{L}_{Delete} = \mathcal{L}_{BLK}(\cup_{k=1}^{d} X_k^{(n)})$$

where $X_k^{(n)} = \cup_{t=1}^{(\lceil \frac{2*n-3}{4} \rceil)}(Y_t^k)$, $X_k^{(n)'} = \cup_{t=1}^{(\lceil \frac{2*n-3}{4} \rceil)}(Y_t^k \cup \mathsf{Enc}_t(q) \cup \mathsf{Enc}_t(2q))$ and $Y_t^k = \{\mathsf{Enc}(p_t[k] + p_j[k])|t < j < n - t + 1\} \cup \{\mathsf{Enc}(p_j[k] + p_{(n-t+1)}[k])|t < j < n - t + 1\}$.

The advantage for any probabilistic polynomial-time adversary is,

$$|\Pr[\mathrm{Game}_{\mathcal{R},\mathcal{A}(\lambda)} = 1] - \Pr[\mathrm{Game}_{\mathcal{S},\mathcal{A}(\lambda)} = 1]|$$
$$\leq \mathrm{negl}(\lambda) = d \cdot (\mathrm{negl}^{ORE}(\lambda) + (2n-1)\mathrm{poly}(\lambda)/2^{\lambda}).$$

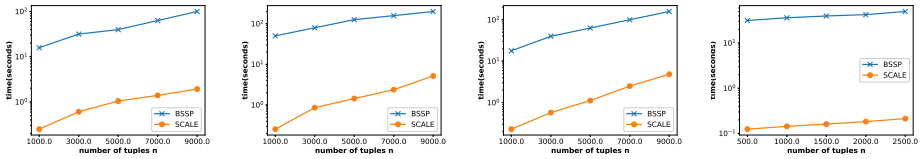
4.4 Complexity Analysis

In the encryption phase, the plaintext data from the data owner can be sorted and encrypted in advance. We need $O(d + \lceil \frac{2*n-3}{4} \rceil)$ encryption operations every time when a user submits a query following Algorithm 4.

During the querying phase, our scheme replaces the original plaintext subtraction and comparison operations with a limited number of comparisons over ciphertext. The time taken for encryption and ciphertext comparisons is only affected by the block size in ORE, key length in AES, and plaintext length. Therefore, the main logic for dynamic skyline query processing is unchanged. Hence, the complexity for query processing in our scheme is consistent with that of [6], *i.e.,* $O(n^2)$ for the worst case.

5 Experimental Study

In this section, we evaluate the performance and scalability of SCALE under different parameter settings over four datasets, including both real-world and synthetic ones. We also compare our model with another baseline, namely BSSP [17], which is the only solution for secure dynamic skyline query.



(a) CORR (b) ANTI (c) INDE (d) NBA

Fig. 6: Response time by varying n (with $d = 3, block = 16, K = 256$)

5.1 Experiment Settings

All algorithms are implemented in C, and tested on the platform with a 2.7GHz CPU, 8GB memory running MacOS. We use both synthetic and real-world datasets in our experiments. In particular, we generated independent (INDE), correlated (CORR), and anti-correlated (ANTI) datasets following the seminal work in [6]. In line with [17], we also adopt a dataset that contains 2500 NBA players who are league leaders of playoffs[5]. Each player is associated with six attributes that measure the player's performance: Points, Offensive Rebounds, Defensive Rebounds, Assists, Steals, and Blocks.

5.2 Performance Results

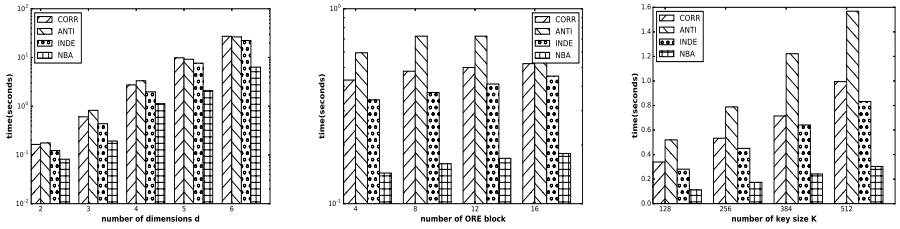
In this subsection, we evaluate our protocols by varying the number of tuples (n), the number of dimensions (d), the ORE block setting, and the length of key (K).

Varying the number of tuples. Fig. 6 shows the time cost by varying the number of tuples, namely n, on the four datasets. In this group of experiments, we fix the number of dimensions, ORE block size and key length as 3, 16 and 256, respectively. We observe that for all datasets, the time cost increases almost linearly with respect to n. This phenomenon is consistent with our complexity study in Section 4.4. Notably, for the real-world dataset (*i.e.,* NBA), the query response time is less than 0.2 seconds, which is efficient enough in practice. Compared to the state-of-the-art [17], SCALE is more than 3 orders of magnitude faster. In the following, we shall fix $n = 2500$ and focus on evaluating the effects of the three parameters in our scheme.

Impact of d. Fig. 7a shows the time cost for different d on the four datasets, where we fix the ORE block size and key length as 16 and 256, respectively. For all datasets, as d increases from 2 to 6, the response time in all four datasets increases almost exponentially as well. This fact is consistent with the ordinary dynamic skyline querying in plaintext. This is because an increase in d leads to more comparison operations for the decision of dynamic dominance criteria.

Impact of ORE block. Encrypting plaintext based on block cipher, different block sizes may take different time. Fig. 7b plots the time cost by varying the block sizes used in the ORE scheme, where d and K are fixed as 3 and 256, respectively. As mentioned in [16], this ORE scheme leaks the first block of δ-bits that differs, therefore, increasing the block size brings higher security. Observe that the response time increases slightly with respect to the size of ORE block. That is, higher security level in ORE has to sacrifice some response time.

[5] `https://stats.nba.com/alltime-leaders/?SeasonType=Playoffs`.



(a) Effect of d (with $block = 16, K = 256$)

(b) Effect of block size (with $d = 3, K = 256$)

(c) Effect of key length (with $block = 16, d = 3$)

Fig. 7: The effects of different parameters ($n = 2500$)

Impact of K. Fig. 7c shows the time cost by varying the lengths for the keys in the ORE scheme. This ORE scheme uses AES as the building block, therefore, increasing the encryption key size brings in higher security. Similar to that of block size, the response time also increases linearly with respect to the size of encryption keys. Comparing Fig. 7c against 7b, we find the following phenomenons. First, increasing the security level for ORE will definitely sacrifice some efficiency. Second, the key length in AES exhibits more significant impact on the efficiency comparing to that of ORE block size.

6 Conclusions

In this paper, we have presented a new framework called SCALE to address the secure dynamic skyline query problem in the cloud platform. A distinguishing feature of our framework is the conversion of the requirement of both subtraction and comparison operations to only comparisons. As a result, we are able to use ORE to realize dynamic domination protocol over ciphertext. Based on this feature, we built SCALE on top of BNL. In fact, our framework can be easily adapted to other plaintext dynamic skyline query models. We theoretically show that the proposed scheme is secure under our system model, and is efficient enough for practical applications. Moreover, there is only one interaction between a user and the cloud, which minimizes the communication cost and corresponding threats. Experimental study over both synthetic and real-world datasets demonstrates that SCALE improves the efficiency by at least three orders of magnitude compared to the state-of-the-art method. As part of our future work, we plan to further enhance the security of our scheme and explore how the scheme can be adapted to support other variations of skyline query.

Acknowledgments

This work is supported by National Natural Science Foundation of China (No. 61672408, 61972309, 61702403, 61976168) and National Engineering Laboratory (China) for Public Safety Risk Perception and Control by Big Data (PSRPC). Sourav S Bhowmick is partially funded by Huawei grant M4062170.

References

1. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Order preserving encryption for numeric data. In: SIGMOD. pp. 563–574. ACM (2004)

2. Alrifai, M., Skoutas, D., Risse, T.: Selecting skyline services for qos-based web service composition. In: WWW. pp. 11–20 (2010)
3. Boldyreva, A., Chenette, N., Lee, Y., O'Neill, A.: Order-preserving symmetric encryption. In: EUROCRYPT. pp. 224–241 (2009)
4. Boldyreva, A., Chenette, N., O'Neill, A.: Order-preserving encryption revisited: Improved security analysis and alternative solutions. In: CRYPTO. pp. 578–595 (2011)
5. Boneh, D., Lewi, K., Raykova, M., Sahai, A., Zhandry, M., Zimmerman, J.: Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. In: EUROCRYPT. pp. 563–594 (2015)
6. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: ICDE. pp. 421–430 (2001)
7. Bothe, S., Cuzzocrea, A., Karras, P., Vlachou, A.: Skyline query processing over encrypted data: An attribute-order-preserving-free approach. In: PSBD@CIKM. pp. 37–43 (2014)
8. Chatterjee, S., Das, M.P.L.: Property preserving symmetric encryption revisited. In: ASIACRYPT. pp. 658–682 (2015)
9. Chen, W., Liu, M., Zhang, R., Zhang, Y., Liu, S.: Secure outsourced skyline query processing via untrusted cloud service providers. In: INFOCOM. pp. 1–9 (2016)
10. Chenette, N., Lewi, K., Weis, S.A., Wu, D.J.: Practical order-revealing encryption with limited leakage. In: FSE. pp. 474–493 (2016)
11. Chomicki, J., Godfrey, P., Gryz, J., Liang, D.: Skyline with presorting. In: ICDE. pp. 717–719 (2003)
12. Dellis, E., Seeger, B.: Efficient computation of reverse skyline queries. In: VLDB. pp. 291–302 (2007)
13. Gentry, C.: A fully homomorphic encryption scheme. Stanford University (2009)
14. Kriegel, H., Renz, M., Schubert, M.: Route skyline queries: A multi-preference path planning approach. In: ICDE. pp. 261–272 (2010)
15. Kung, H.T., Luccio, F., Preparata, F.P.: On finding the maxima of a set of vectors. J. ACM **22**(4), 469–476 (1975)
16. Lewi, K., Wu, D.J.: Order-revealing encryption: New constructions, applications, and lower bounds. In: CCS. pp. 1167–1178 (2016)
17. Liu, J., Yang, J., Xiong, L., Pei, J.: Secure skyline queries on cloud platform. In: ICDE. pp. 633–644 (2017)
18. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: EUROCRYPT. pp. 223–238 (1999)
19. Papadias, D., Tao, Y., Fu, G., Seeger, B.: An optimal and progressive algorithm for skyline queries. In: SIGMOD. pp. 467–478 (2003)
20. Papadias, D., Tao, Y., Fu, G., Seeger, B.: Progressive skyline computation in database systems. ACM Trans. Database Syst. **30**(1), 41–82 (2005)
21. Park, Y., Min, J., Shim, K.: Efficient processing of skyline queries using mapreduce. IEEE Trans. Knowl. Data Eng. **29**(5), 1031–1044 (2017)
22. Popa, R.A., Li, F.H., Zeldovich, N.: An ideal-security protocol for order-preserving encoding. In: SP. pp. 463–477 (2013)
23. Preparata, F.P., Shamos, M.I.: Computational Geometry - An Introduction. Springer (1985)
24. Sun, W., Zhang, N., Lou, W., Hou, Y.T.: When gene meets cloud: Enabling scalable and efficient range query on encrypted genomic data. In: INFOCOM. pp. 1–9 (2017)
25. Wang, W., Li, H., Peng, Y., Bhowmick, S.S., Chen, P., Chen, X., Cui, J.: An efficient secure dynamic skyline query model. **arXiv:2002.07511** (2020)
26. Wang, W.C., Wang, E.T., Chen, A.L.P.: Dynamic skylines considering range queries. In: DASFAA. pp. 235–250 (2011)
27. Zhou, X., Li, K., Zhou, Y., Li, K.: Adaptive processing for distributed skyline queries over uncertain data. IEEE Trans. Knowl. Data Eng. **28**(2), 371–384 (2016)