EasyChair Preprint

№ 13049

Secure Skyline Group Queries Based on Encrypted Dominance Graphs in Cloud Environments

Xiyu Liu, Yiping Teng, Jiawei Qi, Siyu Duan, Bingfeng Yu and Chunlong Fan

April 19, 2024

Secure Skyline Group Queries Based on Encrypted Dominance Graphs in Cloud Environments

Xiyu Liu
School of Computer
Shenyang Aerospace University
Shenyang, China
liuxiyu@stu.sau.edu.cn

Yiping Teng
School of Computer
Shenyang Aerospace University
Shenyang, China
typ@sau.edu.cn

Jiawei Qi
School of Computer
Shenyang Aerospace University
Shenyang, China
qijiawei@stu.sau.edu.cn

Siyu Duan
School of Computer
Shenyang Aerospace University
Shenyang, China
3123655850@qq.com

Bingfeng Yu
School of Computer
Shenyang Aerospace University
Shenyang, China
2034086967@qq.com

Chunlong Fan
School of Computer
Shenyang Aerospace University
Shenyang, China
fanchl@sau.edu.cn

***Abstract*—In this paper, we define and investigate the secure skyline group query problem and propose a method named Secure Skyline Group Query processing based on Dominance Graph (DGSSGQ), which aims to efficiently find the skyline groups on encrypted datasets. In DGSSGQ method, we first present the data preprocessing and encryption mechanism based on dominance graphs, which record the dominance relationships between points in the dataset. To save the query processing costs, we further present the maintenance of the encrypted dominance graph. To achieve secure calculation on the encrypted DG, we design Secure Inclusion Protocol (SIP), which enables the determination of the inclusion of the nodes in the DG in a group. Based on the encrypted DG and SIP presented, we propose the secure skyline group query processing method to find the skyline groups. Thorough analysis shows the security and complexity of the proposed query methods, and the results of extensive experiments on real and synthetic datasets illustrate the performance of our proposed methods.**

***Index Terms*—Skyline groups, Encrypted data, Dominance graph, Cloud computing**

I. Introduction

During the past decade, the skyline query [1] has garnered widespread attention within the database domain, particularly for their unique value in the processing of multi-preference data analysis and decision-making. In recent years, the concept of the skyline query has undergone further expansion, with various variants [2], [3] gradually drawing more attention from the academic community. Specifically, skyline group query, as one of these variants, aims to identify fixed-sized groups of tuples that cannot be dominated by any other group of tuples in the same size. This problem was initially introduced in [4], [5] and has been extensively explored and expanded in a series of studies [2], [6]–[11].

To optimize costs and flexibility, data owners prefer to outsource data storage and query processing tasks to public cloud services, thereby alleviating the pressure of big data management on themselves. However, directly outsourcing data processing tasks to the public cloud can lead to serious privacy protection issues [12]–[14]. Therefore, securely processing skyline group queries on public cloud platforms, to prevent the leakage of sensitive information, is particularly important.

To prevent cloud service providers and other unauthorized entities from accessing sensitive data, a direct approach is to perform skyline group queries on encrypted data. The most related study [15], our previous work, focused on skyline group queries over encrypted data based on aggregated functions, which can only support poor skyline query semantics, while other existing research focused mainly on traditional skyline queries on encrypted data [3], [16], [17]. Besides, few studies can facilitate incremental updates for skyline group queries in ciphertext, while reconstructing the whole encrypted dataset might lead low query performance. Therefore, it becomes crucial to develop an effective and secure skyline group query method on encrypted data that supports incremental updates in public cloud environments.

In this paper, we define and explore a novel secure skyline group query adopting the encrypted Dominance Graph (DG) in support of the optimal group skyline semantics. Based on DG, we first introduce a data preprocessing and encryption mechanism, in which the dominance relationships between points in the dataset are extracted in DG and the points with the numbers of dominating points are added to the corresponding buckets. After that, both the dominance graph and the buckets are encrypted. To save the operating costs of query processing, we further present the maintenance of the encrypted dominance graph. Achieving secure calculation on the encrypted DG, we design a novel secure computation protocol, i.e., Secure Inclusion Protocol (SIP), which enables the inclusion determination of the nodes on DG in a given group. Based on the encrypted DG and SIP presented, we propose a secure skyline group query processing method based

on the encrypted dominance graph, which applies a depth-first-search on encrypted DG to find the skyline groups. Thorough analysis shows the security and complexity of the proposed query methods, and the results of extensive experiments on real and synthetic datasets illustrate the performance of our proposed methods.

Overall, our contributions can be summarized as follows.

- We define and study the secure skyline group query problem based on the dominance graph on encrypted data.
- To address this problem, we propose a method of Secure Skyline Group Query processing based on Dominance Graph (DGSSGQ), where we present the construction and maintenance of the encrypted dominance graph and a novel Secure Inclusion Protocol to facilitate the secure computation on the encrypted dominance graph.
- We provide analysis to show the security and complexity of the proposed query method and conduct extensive experiments on real dataset and synthetic datasets, of which the results illustrate the performance of our proposed method.

The remainder of the paper is structured as follows. Section II provides an overview of related work in the field. In Section III, we present the problem definition and the preliminary concepts used in this paper. Section IV describes both the proposed dominance-graph-based secure skyline group query method and discusses incremental updates for secure skyline group queries. The security and computational complexity of the proposed methods are analyzed in Section V, followed by the presentation of the experimental results in Section VI. Finally, Section VII offers concluding remarks.

II. Related Work

[15] represents the content most closely related to this paper. The paper defined the secure skyline group query problem and developed two secure query methods: the Basic Secure Skyline Groups Query (BSSGQ) method and the Dynamic-programming-based Secure Skyline Groups Query (DSSGQ) method. The BSSGQ method generates candidate groups and their aggregate tuples in an encrypted environment and introduces a secure group dominance protocol. To address the efficiency challenges in the BSSGQ method, the DSSGQ method dynamically constructs and calculates the skyline subgroups in ciphertext, effectively reducing the generation of candidate groups.

A. Skyline Groups Queries

Unlike traditional skyline queries over points, the goal of skyline groups queries is to find combinations of points that are not dominated by any other group of equal size. Liu et al. [6], [11] introduced an innovative structure that retrieves k-point G-skyline groups by representing points in the form of a directed skyline graph based on the first k skyline layers. This directed skyline graph demonstrates that computing k-point G-skyline relies only on points within the first k layers rather than all points in the entire input dataset. Wang et al. [2] proposed an efficient method named Minimum Dominating Search (MDS) to address the G-skyline problem, by constructing a Minimum Dominating Graph (MDG) and adopting optimization strategies based on skyline groups. Yu et al. [18], [19] introduced the computation of skylines at the group level, proposing efficient methods for finding Group-based Skylines (G-skyline). By employing novel structures and Group-based Clustering (G-clustering) algorithms, they further introduced the concept of Representative G-skyline (RG-skyline). Zhu et al. [7], [9] developed pruning techniques to handle the TKD (top-k dominate) query of group skylines under all existing skyline group definitions. With these techniques, it is possible to return top-k dominated skyline groups. By employing aggregate functions frequently utilized in database applications, such as SUM, MAX, and MIN, research studies [4], [5], [8], [20] explored the skyline group query issue through the computation of representative points. The secure methods proposed in this paper draw inspiration from these investigations. Sun et al. [21] addresses the spatio-textual group skyline query problem by proposing two query methods, including a basic method and an efficient index-based method.

B. Secure Skyline Queries

Liu et al. [16] pioneered secure skyline queries on encrypted data with semantic security, introducing a comprehensive secure dominance protocol foundational for various queries. They also developed two secure skyline query protocols using partial homomorphic properties and novel permutation and perturbation techniques to ensure accurate results while maintaining privacy. Subsequently, Liu et al. [17] delved into the issue of secure skyline queries on semantically secure encrypted data and presented a comprehensively secure query protocol. As a core subroutine, they introduced a novel secure comparison protocol, which is also suitable for constructing other query processes. Wang et al. [3] presented DynPilot, a novel scheme for privacy-preserving verifiable location-based skyline queries on dynamic and encrypted data, designed the dynamic and efficient secure verifiable tree DSV-tree, and adopted a fuzzy updating strategy. And in [22], they proposed a secure indexing scheme aimed at protecting privacy in location-based skyline queries on static or infrequently updated datasets and introduced the SR-tree index and designed the BasSky and SecSky protocols.

III. Problem Definitions and Preliminaries

A. Problem Definitions

Definition 1: **(Group Dominance).** *Given two groups, let* $G = \{p_1, p_2, \ldots, p_g\}$ *and* $G' = \{p'_1, p'_2, \ldots, p'_g\}$ *be two distinct groups, each containing* g *points. If there exist two groups* $G = \{p_{u1}, p_{u2}, \ldots, p_{ug}\}$ *and* $G' = \{p'_{u1}, p'_{u2}, \ldots, p'_{ug}\}$ *such that, after sorting the points in each group in ascending order, one group dominates the other in every attribute of every point, for example,* $p_{u1} \prec p'_{u1}$*, then we say group* G *dominates group* G'*, denoted as* $G \prec G'$*.* □

Definition 2: **(Dominate Graph).** *Given a dataset* $DS = \{p_1, p_2, \ldots, p_n\}$ *of* n *tuples in an* m*-dimensional space, the dominance graph (DG) of* DS *is a directed acyclic graph*

TABLE I: Notations

DS	dataset of n tuples
$p_i[j]$	the j^{th} attribute of p_i
m	number of dimensions
g	number of group size
G	a g-tuple group
key	key size
gr	the current group
$G_a \prec G_b$	G_a dominates G_b
S_g^n	all g-tuple skyline groups
$DG =< V, E >$	dominate graph
B	buckets
$p.parents$	the parents node set of p
$p.children$	the children node set of p
GSG	all skyline groups of different sizes
p_{ins}	an inserted tuple
p_{del}	a deleted tuple
g_{exp}	the expanded group size
g_{red}	The reduced group size

where each node represents a point in DS, and each edge represents a dominance relationship between two points. □

Definition 3: **(G-Skyline).** *Let DS as a collection of data points in m dimensions and A as a group comprising k points from DS. A qualifies as a g-skyline group if there is no other k-point group that g-dominates A. The set of all g-skyline groups containing k points within DS forms the k-point g-skyline.* □

Definition 4: **(Skyline Group).** *Given a dataset $DS = \{p_1, p_2, \ldots, p_n\}$ of n tuples in an m-dimensional space, $G = \{p_1, p_2, \ldots, p_g\}$ is a group from DS. G is a g-skyline group if and only if there does not exist any g-group that dominates G.* □

Definition 5: **(Secure Skyline Groups Query(SSGQ)).** *Given an encrypted dataset $E_{pk}(DS)$ and a query group size g, the purpose of SSGQ is to discover groups in $E_{pk}(DS)$ that are not dominated by any other groups of equal size g. The result of SSGQ $(E_{pk}(DS), g)$ is $\{E_{pk}(G_1), \ldots, E_{pk}(G_l)\}$, with $\{E_{pk}(G_1), \ldots, E_{pk}(G_l)\}$ being the encrypted outcomes.* □

B. System Framework

In this paper, we employ a widely recognized cloud computing framework that has been utilized in numerous related studies [13], [14], [17]. This framework comprises three types of entities: data owner, user, and cloud servers. Besides the cloud server C_1 responsible for computing and searching on encrypted data, our framework introduces an additional non-colluding cloud server C_2 to ensure the correctness and efficiency of cloud computing, as well as to reduce interaction costs between data owners and users. C_2 provides encryption and decryption services by holding secret keys shared with the data owner. The system framework depicted in Fig. 1 is described as follows.

Data Owner. The data owner first generates a public key and a private key (denoted as pk and sk, respectively). Then, the data owner encrypts the dataset, obtaining $E_{pk}(DS)$ and $E_{pk}(DG)$, and sends pk, $E_{pk}(DS)$, and $E_{pk}(DG)$ to C_1. Lastly, the data owner sends pk and sk to C_2, and also sends

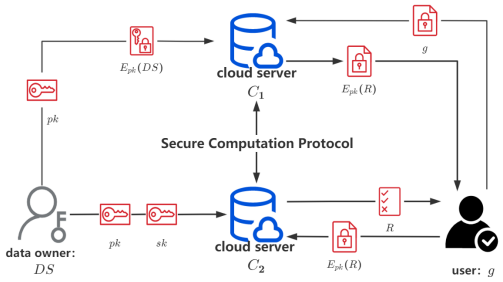


Fig. 1: System framework

pk to the user.

User. Using the same pk, the user specifies the query group size g and uploads it to C_1 to initiate a secure skyline group query search. After the search process, the user receives random information R about the result from C_1 and recovers the search result R from C_2.

Cloud Server. Upon receiving $E_{pk}(DS)$ and $E_{pk}(DG)$, C_1, with the assistance of C_2 and through secure computation protocols, performs the skyline group query processing and sends R to the user.

C. Security Model

This paper follows the footsteps of numerous prior researches [16], [17], adopting the semi-honest security model as the foundation. In this model, cloud service providers are expected to process user data strictly according to the preset algorithms and protocols, while they might attempt to extract valuable information. The model presumes that the two cloud servers C_1 and C_2 will not collude, thus not jointly infringe upon the privacy of user data. Under this security framework, the following privacy protection requirements need to be met:

- **Data Privacy.** Neither the user, C_1, nor C_2 should obtain any unencrypted information of the dataset DS.
- **Result Privacy.** Apart from the user, no other parties should know the specifics of the result set R.
- **Access Pattern Privacy.** At any computation stage, C_1 and C_2 should not identify any specific characteristics of any intermediate results, such as the correlation between the computed data and the original data.

D. Preliminaries

Dominance Graph. The construction algorithm for the Dominance Graph (DG) is formulated based on Definition 2 and [2], focusing solely on identifying the dominance relations among all points within the dataset DS. For every point within DS, the algorithm incorporates this point into DG by comparing it against all currently included points in DS. The specific operational steps are detailed in Alg.1. **Paillier Cryptosystem.** In this paper, we adopt the semantically secure Paillier cryptosystem [23] for encryption, notable for its support of homomorphic addition and scalar multiplication operations.

- Homomorphic addition:
  $E_{pk}(x_1) \times E_{pk}(x_2) \bmod N^2 = (x_1 + x_2) \bmod N$
- Homomorphic multiplication:
  $E_{pk}(x_1)^{x_2} \bmod N^2 = x_1 \times x_2 \bmod N$

Algorithm 1: Dominance Graph Construction Algorithm

Input: Dataset DS;
Output: DG;
Initialize $DG = (V, E)$;
Sort the data points in DS in ascending order based on the sum of each dimension;
for $i = 0$ *to* $n - 1$ **do**
  $B[i] = \emptyset$;
for $p \in DS$ **do**
  $p.parents = \emptyset$;
  for $i = n - 1$ *to* 0 **do**
    for $p' \in B[i]$ **do**
      if $p' \prec p$ **then**
        $p.parents = p.parents \cup \{p'\} \cup p'.parents$;
  for $p' \in p.parents$ **do**
    Add an edge from node p' to node p in the E;
  $B[|p.parents|] = B[|p.parents|] \cup \{p\}$;
return DG;

Secure Multiplication (SM) Protocol. SM protocol [12] is that the server C_1 with input $E_{pk}(x_1)$ and $E_{pk}(x_2)$, and performs $E_{pk}(x_1 \times x_2)$ operations on $E_{pk}(x_1)$ and $E_{pk}(x_2)$ under encrypted conditions.

Secure Less (SLESS) Protocol. [17] Cloud server C_1 with input $E_{pk}(x_1)$ and $E_{pk}(x_2)$ and Cloud server C_2 has sk. C_1 and C_2 securely compute the encryption boolean output $E_{pk}(Bool(x_1 < x_2))$.

Secure Equal (SEQ) Protocol. [17] Cloud server C_1 has encrypted inputs $E_{pk}(x_1)$ and $E_{pk}(x_2)$, while cloud server C_2 holds sk. Neither server knows the values of x_1 and x_2. The SEQ protocol's aim is to securely calculate the encrypted boolean result $E_{pk}(Bool(x_1 == x_2))$.

IV. Secure Skyline Group Query

This section introduces a secure skyline group query processing based on dominance graph (DGSSGQ) using an encrypted Dominance Graph (DG) to support optimal group skyline semantics. It employs a Secure Inclusion Protocol (SIP) for safe computations on the encrypted DG. Based on the encrypted DG and SIP, a secure skyline group query processing method is proposed, utilizing depth-first search to identify skyline groups on the encrypted DG. Additionally, it discusses dynamic adjustments on the encrypted dominance graph for incremental updates to calculate skyline groups. These algorithms are based on the basic security protocols mentioned in Section III-D.

A. Encryption of Dominance Graph

To ensure the privacy of the values of each point in the dataset DS, the dominance relations described in the dominance graph DG, and the points in the buckets B, this paper adopts the Paillier homomorphic encryption technique to encrypt the dataset, the constructed dominance graph, and the buckets. The encryption process for the dataset DS, the dominance graph DG, and the buckets B is described as follows:

1) **Key Generation:** First, generate the public key pk and the private key sk for the Paillier encryption system.
2) **Encryption of Data Points:** For the values of each point p_i in the dataset, dominance graph and buckets, encrypt p_i using the public key pk to obtain the encrypted data $E_{pk}(p_i)$.
3) **Encryption of Dominance Relations:** For the dominance relations of each point's value p_i in the dominance graph and buckets, specifically $p_i.parents$ and $p_i.children$, use the public key pk to encrypt $p_i.parents$ and $p_i.children$, obtaining the encrypted data $E_{pk}(p_i.parents)$ and $E_{pk}(p_i.children)$.
4) **Construct the Encrypted Information:** Reassemble all encrypted points in their original order to form an encrypted information set $E_{pk}(DG)$ and $E_{pk}(B)$.

B. Secure Inclusion Protocol

Algorithm 2: Secure Inclusion Protocol

Input: C_1 has $E_{pk}(U)$, $E_{pk}(V)$, $E_{pk}(|V|)$, g; C_2 has sk;
Output: α';
C_1:
$\alpha = E_{pk}(0)$;
for $i \in E_{pk}(U)$ **do**
  for $j \in E_{pk}(V)$ **do**
    $\alpha = \alpha * \mathbf{SEQ}(i, j)$;
$\alpha = \alpha * \mathbf{SEQ}(\alpha, E_{pk}(|V|))$;
return α' to C_1;

In the Secure Inclusion Protocol (SIP) detailed in Alg. 2, consider that C_1 contains $E_{pk}(U)$ (the parent nodes of the point), $E_{pk}(V)$ (the points within $E_{pk}(gr)$), and $E_{pk}(|V|)$ (the count of points in $E_{pk}(gr)$), while C_2 has sk. Initially, C_1 computes $SEQ(E_{pk}(U), E_{pk}(V))$ and updates α by setting $\alpha \leftarrow \alpha * SEQ(i, j)$. Subsequently, C_1 calculates α' as $\alpha' \leftarrow SEQ(\alpha, E_{pk}(|V|))$ and outputs α' to C_1 (lines 2-6).

C. Encrypted Dominance Graph Maintenance

In the maintenance of the encrypted dominance graph, we focus on securely updating a data point, particularly when inserting or deleting a new point into the dataset. This process involves dynamically adjusting the dominance relationship for the encrypted data point.

In Alg.3, we introduce the maintenance of the dominance graph when inserting a point in an encrypted environment. First, we initialize the set $E_{pk}(p_{ins}.parents)$ as an empty set (line 1). Then, we iterate through the points in the bucket $E_{pk}(B[i])$, determining the dominance relationship between $E_{pk}(p_i)$ and $E_{pk}(p_{ins})$ and saving it to the corresponding element of vector α and determining the dominance relationship between $E_{pk}(p_{ins})$ and $E_{pk}(p_i)$ and saving it to the corresponding element of vector β (lines 3-5). The results are sent to C_2 for decryption. C_2 decrypts and returns to C_1 the values in $D_{sk}(\alpha)$ and $D_{sk}(\beta)$ that are 1 (lines 8-9). C_1 receives α and β, adds the α's $E_{pk}(p_{ins}.parents)$ to $E_{pk}(p_{ins})$'s parent set, and based on the transitivity of dominance, adds the parents of $E_{pk}(p_i)$ (without duplicates) to $E_{pk}(p_{ins}.parents)$. Next, based on the number of $E_{pk}(p_{ins}.parents)$, $E_{pk}(p_{ins})$ is placed into the appropriate bucket. The β's $E_{pk}(p_i)$ parents,

Algorithm 3: Secure Point Insertion

Input: C_1 has $E_{pk}(DG)$, $E_{pk}(B)$, $E_{pk}(p_{ins})$, g; C_2 has sk;
Output: $E_{pk}(B)$;
C_1:
1 $E_{pk}(p_{ins}.parents) = \emptyset$;
2 $E_{pk}(B)$ is the first g buckets;
3 **for** $i = 1$ *to* $|E_{pk}(B)|$ **do**
4 $\quad \alpha_i = \mathbf{SDMN}(E_{pk}(p_i), E_{pk}(p_{ins}))$;
5 $\quad \beta_i = \mathbf{SDMN}(E_{pk}(p_{ins}), E_{pk}(p_i))$;
6 send α, β to C_2;
7 C_2:
8 receive α, β from C_1;
9 decrypt α, β and send 1 to C_1;
C_1 and C_2:
10 **for** $\alpha_i \in \alpha$ **do**
11 $\quad E_{pk}(p_{ins}.parents) = E_{pk}(p_{ins}.parents) \cup E_{pk}(p_i) \cup E_{pk}(p_i.parents)$;
12 **for** $\beta_i \in \beta$ **do**
13 $\quad E_{pk}(p_i.parents) = E_{pk}(p_i.parents) \cup E_{pk}(p_{ins}) \cup E_{pk}(p_{ins}.parents)$;
14 $\quad E_{pk}(B[|p_{ins}.parents|]) = E_{pk}(B[|p_{ins}.parents|]) \cup E_{pk}(p_{ins})$;
15 **for** $i = 1$ *to* $|p_{ins}.children|$ **do**
16 $\quad$ use $\mathbf{SM}(E_{pk}(w_i), E_{pk}(0))$ to calculate in the $B[|c_i.parents|]$ of $E_{pk}(w_i)$;
17 $\quad E_{pk}(B[|c_i.parents|]) = E_{pk}(B[|c_i.parents|]) \cup E_{pk}(c_i)$;
18 return $E_{pk}(B)$;

along with $E_{pk}(p_{ins})$ and $E_{pk}(p_{ins}.parents)$, are added to $E_{pk}(p_{ins}.parents)$(lines 10-14). Finally, we iterate through the points in $E_{pk}(p_{ins}.children)$, the points $E_{pk}(c_i)$, where the corresponding bucket index changes, thus C_1 uses the Secure Multiplication Protocol $\mathbf{SM}(E_{pk}(c_i), E_{pk}(0))$ to effectively remove $E_{pk}(c_i)$ from $E_{pk}(B[|c_i.parents-1|])$. Finally, based on the number of $c_i.parents$, $E_{pk}(c_i)$ is placed into the appropriate bucket (lines 15-17).

Algorithm 4: Secure Point Deletion

Input: C_1 has $E_{pk}(DG)$, $E_{pk}(B)$, $E_{pk}(p_{del})$, g; C_2 has sk;
Output: $E_{pk}(B)$;
C_1:
1 use **SM** $(E_{pk}(p_{del}), E_{pk}(0))$ to calculate in the $B[|c_i.parents|]$ to $E_{pk}(p_{del})$;
2 **for** $i = 1$ *to* $E_{pk}(p_i.parents)$ **do**
3 $\quad$ use **SM** $(E_{pk}(\omega_i), E_{pk}(0))$ to calculate in the $B[|c_i.parents|]$ to $E_{pk}(\omega_i)$;
4 $\quad B[|c_i.parents-1|] = B[|c_i.parents-1|] \cup \mathrm{E}_{pk}(c_i)$;
5 return $E_{pk}(B)$;

In Alg.4, we introduce the maintenance of the dominance graph for deleting a point under encrypted condition. C_1 holds $E_{pk}(B)$, g, $E_{pk}(p_{del})$, and $E_{pk}(GSG)$, while C_2 has sk. First, C_1 removes $E_{pk}(p_{del})$ from $E_{pk}(B[|p_{del}.parents|])$ using the Secure Multiplication Protocol $SM(E_{pk}(p_{del}), E_{pk}(0))$(line 1). Then, we iterate through $E_{pk}(p_{del}.children)$, removing $E_{pk}(c_i)$ from $E_{pk}(B[|c_i.parents|])$ and adding it to $E_{pk}(B[|c_i.parents-1|])$ using $SM(E_{pk}(c_i), E_{pk}(0))$ (lines 2-4).

D. Secure Skyline Group Query Processing Based on Dominance Graph

Algorithm 5: Secure Skyline Group Query processing based on Dominance Graph

Input: C_1 holds $E_{pk}(DG)$, bucket $E_{pk}(B)$, query group size g; C_2 has sk;
Output: $E_{pk}(S_g^n)$;
C_1 and C_2:
1 $E_{pk}(S_g^n) = \emptyset$;
2 $E_{pk}(gr) = \emptyset$;
3 Sort the dataset in ascending order by bucket;
4 **for** $E_{pk}(p) \in E_{pk}(B[0])$ **do**
5 $\quad E_{pk}(gr) = E_{pk}(gr) \cup E_{pk}(p)$;
6 $\quad E_{pk}(GSG) = E_{pk}(GSG) \cup E_{pk}(gr)$;
7 $\quad$ search single point $(E_{pk}(gr), E_{pk}(p), g, E_{pk}(S_g^n))$;
8 return $E_{pk}(S_g^n)$;

In this section, we propose a secure skyline group query processing based on dominance graph, aimed at querying skyline groups according to group size.

In alg.5, we generate candidate groups by adding appropriate single points to the current candidate group $E_{pk}(gr)$. First, we initialize the result set $E_{pk}(S_g^n)$ and the current candidate group $E_{pk}(gr)$ as empty sets (lines 1-2). Next, we arrange all points in the multidimensional data dominance graph $E_{pk}(DG)$ in ascending order based on the number of their parent nodes, in other words, according to which bucket $E_{pk}(B)$ they are in (line 3). The advantage of this sorting strategy is that it avoids generating duplicate candidate groups by always choosing to add points that come after the last added point in the current group $E_{pk}(gr)$. Then, we begin a depth-first-search (DFS) by adding skyline points from the bucket $E_{pk}(B[0])$ to the current empty group $E_{pk}(gr)$ (lines 4-8). Here, we add the updated $E_{pk}(gr)$ with skyline points to $E_{pk}(GSG)$ for use in later encrypted dominance graph maintenance (line 6).After completing the depth-first-search (DFS), the final set of skyline groups $E_{pk}(S_g^n)$ is obtained. This DFS process is

Algorithm 6: SearchSinglePoint secure query

Input: C_1 has $E_{pk}(gr)$, $E_{pk}(p)$, $E_{pk}(S_g^n)$, the last point $E_{pk}(p_{last})$ in $E_{pk}(gr)$; C_2 has sk;
C_1 and C_2:
1 **if** $|E_{pk}(gr)| = g$ **then**
2 $\quad E_{pk}(S_g^n) = E_{pk}(S_g^n) \cup E_{pk}(gr)$;
3 **for** $E_{pk}(p) \in E_{pk}(DG)$ *and* $E_{pk}(p)$ *is the last point* $E_{pk}(p_{last})$ **do**
4 $\quad \delta = \mathbf{SIP}(E_{pk}(p.parents), E_{pk}(gr), E_{pk}(|gr|))$;
5 $\quad E_{pk}(gr) = \mathbf{SM}(\mathrm{E}_{pk}(gr) \cup \delta)$;
6 $\quad E_{pk}(GSG) = E_{pk}(GSG) \cup E_{pk}(gr)$;
7 $\quad$ search single point $(E_{pk}(gr), E_{pk}(p), g, E_{pk}(S_g^n))$;

based on a recursive function named SearchSinglePoint Alg.6. First, we check if the current group $E_{pk}(gr)$ contains g points; if the number of points in the current group equals g, then this group is considered a complete g-skyline group and is added to $E_{pk}(S_g^n)$. Next, we consider adding a new candidate point

$E_{pk}(p)$ to $E_{pk}(gr)$ and recursively call the Single-point secure query function. Using the Secure Inclusion Protocol (SIP), we compute whether all parent nodes in $E_{pk}(p)$ are in $E_{pk}(gr)$, and use the Secure Multiplication Protocol (SM) to compute $E_{pk}(gr) \leftarrow SM(\delta, E_{pk}(gr) \cup E_{pk}(p))$. Finally, we recursively call the SearchSinglePoint function and add the current group $E_{pk}(gr)$ to $E_{pk}(GSG)$ (lines 3-7).

1) This section calculates skyline groups through the maintenance of encrypted dominance graph and query algorithms of the dominance graph: We will compute the skyline groups based on the updated bucket $E_{pk}(B)$ Alg.7. C_1 holds $E_{pk}(B)$, g, and $E_{pk}(GSG)$, while C_2 has sk. First, we initialize the result set $E_{pk}(S_g^{n+1})$ as an empty set (line 1). Following that, we perform an ascending order sort of the nodes in the first g buckets (line 2). As we iterate through the skyline groups in $E_{pk}(GSG)$, C_1 uses a Secure Equal Protocol(SEQ) to calculate whether $E_{pk}(\omega_i)$ has children nodes of $E_{pk}(p_{ins})$, then $\Phi_i' = E_{pk}(1)^{-\Phi_i}$ and uses the Secure Multiplication Protocol (SM) to compute $E_{pk}(\omega_i) \leftarrow SM(\Phi_i', E_{pk}(\omega_i))$ (line 8). If the number of points in $E_{pk}(\omega_i)$ is less than the group size g, using the Secure Inclusion Protocol (SIP), it calculates whether all parent nodes in $E_{pk}(p_{ins})$ are in $E_{pk}(\omega_i)$, and uses the Secure Multiplication Protocol (SM) to compute $E_{pk}(\omega_i) \leftarrow SM(\delta, E_{pk}(\omega_i) \cup E_{pk}(p_{ins}))$, finally recursively calling the search single point secure query (line 12). If the number of points in $E_{pk}(\omega_i)$ equals the group size g, then it computes $E_{pk}(S_g^{n+1}) \leftarrow E_{pk}(S_g^{n+1}) \cup E_{pk}(\omega_i)$ (line 13-14). In Alg.8, we introduce the computation algorithm for deleting a point under encrypted conditions. C_1 holds $E_{pk}(B)$, g, $E_{pk}(p_{del})$, and $E_{pk}(GSG)$, while C_2 has sk. First, we initialize the result set $E_{pk}(S_g^{n-1})$ as an empty set (line 1). Next, we sort the nodes in the first g buckets in ascending order (line 2). Next, we check if skyline groups in the intermediate result $E_{pk}(GSG)$ can serve as candidate groups. We determine if the current tail node $E_{pk}(p_{last})$ is the node to be deleted $E_{pk}(p_{del})$, with $\varphi = SEQ(E_{pk}(p_{last}), E_{pk}(p_{del}))$. If equal, it indicates the current group will not become a skyline group, and we compute $E_{pk}(gr) = SM(E_{pk}(gr), \varphi)$. After removing the tail node from the current group, we iterate through $E_{pk}(p_{del}.children)$ denoted as $E_{pk}(\omega_i)$ and use the Secure Inclusion Protocol (SIP) to check if all parent nodes in $E_{pk}(p_{del}.children)$ are in $E_{pk}(gr)$, and use the Secure Multiplication Protocol (SM) to compute $E_{pk}(gr) \leftarrow SM(\delta, E_{pk}(\omega_i) \cup E_{pk}(gr))$, finally recursively calling the Single-point secure query (lines 7-10). We verify if any point in $E_{pk}(gr)$ equals $E_{pk}(p_{del})$ and compute $\delta = E_{pk}(0)$, then iterate through points in $E_{pk}(gr)$ using homomorphic addition and the Secure Equal Protocol (SEQ) to compute $\delta \leftarrow \delta * SEQ(\omega_i, E_{pk}(p_{del}))$. Then $\delta' = E_{pk}(1)^{-\delta}$, and finally $E_{pk}(S_g^{n-}) \leftarrow E_{pk}(S_g^{n-1}) \cup SM(\delta', E_{pk}(gr))$ (lines 11-16).

Algorithm 7: Secure Single-Point Insertion Query

Input: C_1 holds $E_{pk}(B)$,$E_{pk}(p_{ins})$,$E_{pk}(GSG)$; C_2 has sk;
Output: $E_{pk}(S_g^{n+1})$;
C_1 and C_2:
1 $E_{pk}(S_g^{n+1}) = \emptyset$;
2 Sort the dataset in ascending order by bucket;
3 **for** $i = 1$ *to* $|E_{pk}(GSG)|$ **do**
4 **for** $j = 1$ *to* $|E_{pk}(\omega_i)|$ **do**
5 **for** $k = 1$ *to* $|E_{pk}(p_{ins}.children)|$ **do**
6 $\phi_i = \phi_i * \mathbf{SEQ}(E_{pk}(\varphi_j), E_{pk}(\lambda_k))$;
7 $\phi_i^{'} = E_{pk}(1)^{-\phi_i}$;
8 $E_{pk}(\omega_i) = \mathbf{SM}(\phi_i^{'}, E_{pk}(\omega_i))$;
9 **if** $|E_{pk}(\omega_i)| < g$ **then**
10 $\delta =$ **SIP**$(E_{pk}(p_{ins}.parents), E_{pk}(\omega_i), E_{pk}(|p_{ins}.parents|))$;
11 $E_{pk}(\omega_i) = \mathbf{SM}(\delta, E_{pk}(\omega_i) \cup (p_{ins}))$;
12 search single point $(E_{pk}(gr), E_{pk}(p_{ins}), g, E_{pk}(S_g^{n+1}))$;
13 **if** $|E_{pk}(\omega_i)| = g$ **then**
14 $E_{pk}(S_g^{n+1}) = E_{pk}(S_g^{n+1}) \cup E_{pk}(\omega_i)$;

Algorithm 8: Secure Single-Point Deletion Query

Input: C_1 holds $E_{pk}(DG)$, $E_{pk}(B)$, $E_{pk}(p_{del})$, $E_{pk}(GSG)$; C_2 has sk;
Output: $E_{pk}(S_g^{n-1})$;
C_1 and C_2:
1 $E_{pk}(S_g^{n-1}) = \emptyset$;
2 Sort the dataset in ascending order by bucket;
3 **for** $E_{pk}(gr) \in E_{pk}(GSG)$ *&* $E_{pk}(p_{last})$ *is the tail node of* $E_{pk}(gr)$ **do**
4 $\varphi = \mathbf{SEQ}(E_{pk}(p_{last}), E_{pk}(p_{del}))$;
5 $E_{pk}(gr) = \mathbf{SM}(E_{pk}(gr), \varphi)$;
6 Remove the tail node of $E_{pk}(gr)$;
7 **for** $i = 1$ *to* $|E_{pk}(p_{del}.children)|$ **do**
8 $\delta = \mathbf{SIP}(E_{pk}(\omega_i.parents), E_{pk}(gr), E_{pk}(|gr|))$;
9 $E_{pk}(gr) = \mathbf{SM}(E_{pk}(gr) \cup (\omega_i), \delta)$;
10 search single point $(E_{pk}(gr), E_{pk}(\omega_i), g, E_{pk}(S_g^{n-1}))$;
11 **if** $|E_{pk}(\omega_i)| = g$ **then**
12 $\delta = E_{pk}(0)$;
13 **for** $i = 1$ *to* $|E_{pk}(gr)|$ **do**
14 $\delta = \delta * \mathbf{SEQ}(E_{pk}(\omega_i), E_{pk}(p_{del}))$;
15 $\delta^{'} = E_{pk}(1)^{-\delta}$;
16 $E_{pk}(S_g^{n-1}) = E_{pk}(S_g^{n-1}) \cup \mathbf{SM}(\delta^{'}, E_{pk}(gr))$;

V. Analysis

A. Security Analysis.

Formal theories in [14], [17] are applied to facilitate the security analysis of the proposed methods as follows.

Theorem 1: **(Composition Theorem) [24].** If a protocol consists of some secure subprotocols and all intermediate processes and results are random or pseudo-random, the protocol is considered secure. □

Analysis of Privacy Requirements. As for the privacy requirements listed in Section III-C, we make a particular security analysis as follows.

Data Privacy. In DGSSGQ, the data owner's dominance graph is encrypted with Paillier encryption, producing $E_{pk}(DG)$ which is sent to node C_1. This safeguards data privacy against breaches, as C_1 cannot decrypt without the key, and C_2 does not access raw data. Clients access only the final skyline results, ensuring data privacy and security.

Result Privacy. User In the result return phase, we use random perturbation techniques to protect the privacy of query results. The actual dimension values of each point in the skyline group are perturbed by computing node C_1 before being sent to the client, preventing C_2 from accessing their true information. The client then processes these noisy results to recover the accurate skyline group data. This approach ensures that neither C_1 nor C_2 can discern the true values, effectively safeguarding result privacy.

B. Complexity Analysis

The construction of DG can be divided into two parts. In the first part, sorting all points requires $O(n \log n)$ times. In the second part, for each point p in DS, we identify the points that dominate p in DG. For the basic algorithm shown in Alg.1, the time complexity of this part is $O(n^2 n_m)$, because there are nearly $n^2 n_m$ dominance checks. Therefore, the total time complexity of Alg.1 is $O(n \log n + n^2 n_m)$. For the basic algorithm in Alg.2, the secure equality protocol uses $|U| \times |V| + 1$ encryptions and $2|U| \times 2|V| + 2$ decryptions. In Alg.3, $8|B| + |p_{ins}.children|$ encryptions and $16|B| + 2|p_{ins}.children|$ decryptions are performed. In Alg.4, in the worst case, $1+|E_{pk}(p_{del}.children)|$ encryptions and $2 + 2|E_{pk}(p_{del}.children)|$ decryptions are performed. In Alg.5, $|B[0]|$ secure single point queries are used, involving $|B[0]| \times n^2$ encryptions and $|B[0]| \times 2n^2$ decryptions. In Alg.6, the time complexity of traversing almost every node is $O(n^2)$, applying n^2 secure multiplication protocols, totaling n^2 encryptions and $2n^2$ decryptions. In Alg.7, $|E_{pk}(p_{del}.children)|$ encryptions and $2|E_{pk}(p_{del}.children)|$ decryptions are executed. In Alg.8, in the worst case, $1 + |E_{pk}(GSG)|(3 + |E_{pk}(p_{del}.children)| \times n^2)$ encryptions and $1+|E_{pk}(GSG)|(7+|E_{pk}(p_{del}.children)| \times 2n^2)$ decryptions are performed.

VI. Experiments

A. Experimental Setups

Setup. All the mentioned algorithms were implemented in Python and tested on a Tower Server (Dell Precision 7920) equipped with dual 40-core Intel(R) Xeon(R) Bronze 3204 CPUs at 1.90GHz and 256GB of RAM, operating on Ubuntu 20.04. In this experimental environment, both C_1 and C_2 were executed on the same server. The experiments did not account for data transmission times, and the response time is presented as the cumulative computation time for C_1 and C_2.

Datasets. We also generated several types of datasets using a data generator [1], including anti-correlated datasets (ANTI), correlated datasets (CORR), and independent datasets (INDE). Regarding real NBA datasets, we selected data from the official NBA website [25], [26], which contains information on 5000 top players who participated in the playoffs. We evaluated these players based on five metrics: points (PTS), rebounds (REB), assists (AST), steals (STL), and blocks (BLK). During data processing, we operated under the assumption that "the lower the parameter value, the better". When evaluating the DGSSGQ method, we set initial parameters beyond variables to include the number of tuples $n = 1000$, group size $g = 3$, and tuple dimensions $m = 2$ among various settings.

B. Experimental Evaluation

1) Evaluation on Secure Tuple Insertion: This section focuses on comparing the efficiency of recalculating skyline groups from scratch after an update request involving insertions, with the efficiency of computing skyline groups using our method based on pre-existing results.

Impact of number of tuples n. As shown in Fig. 2, the experiment evaluates DGSSGQ for inserting a single point across different tuple counts n. Figure 2 shows computation time increases with n. Insertion operations in existing skyline groups (INS_DGSSGQ) perform better than recalculating the entire skyline group (DGSSGQ) from scratch, demonstrating improved efficiency.

Impact of the group size k. As shown in Fig. 3, insertions were conducted via DGSSGQ under varying query group sizes g. As Figure 3 illustrates, DGSSGQ's computation time rises linearly with increasing g. Comparing full recalculation of skyline groups (DGSSGQ) to insertion in existing groups (INS_DGSSGQ), the latter is notably more time-efficient.

Impact of number of dimensions m. As shown in Fig. 4, insertions were performed through DGSSGQ across dimensions m. The computation time increases linearly with dimension m. Insertions on existing skyline groups (INS_DGSSGQ) proved more efficient than full recalculations.

2) Evaluation on Tuple Deletion: This section evaluates the efficiency of fully recalculating skyline groups following a deletion update request against the efficiency of our method, which calculates skyline groups based on existing results.

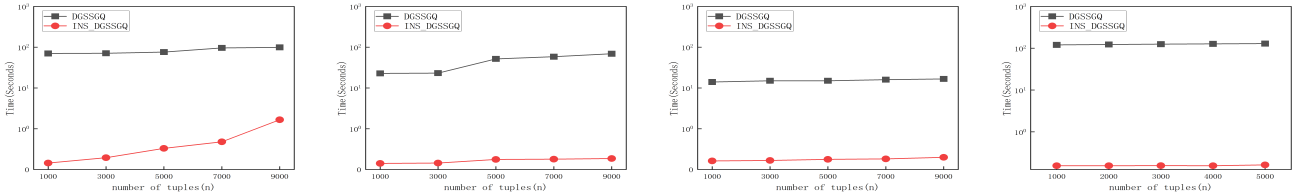
Impact of number of tuples n. Fig. 5 shows the deletion of a single point through DGSSGQ under different tuple counts n. The experimental analysis reveals that computation time increases with n. Comparing recalculating skyline groups from scratch (DGSSGQ) and deleting from existing groups (DEL_DGSSGQ), the latter is more time-efficient.

VII. Conclusion

In this paper, we propose a secure skyline group query scheme based on dominance graphs to address the secure skyline group query problem. In DGSSGQ method, we present the data preprocessing and encryption mechanism based on dominance graphs as well as the maintenance of the encrypted dominance graph. To achieve secure calculation on the encrypted DG, we design a novel secure inclusion protocol to support the determination of the node inclusion in a group. Based on the encrypted DG and SIP presented, we propose the secure skyline group query processing method to find the skyline groups. Through theoretical analysis, we show the security and computational complexity of the proposed methods, and conduct extensive experiments on real and synthetic datasets to demonstrate the performance of our proposed methods. For the future work, we plan to effectively improve the query efficiency while maintaining the data security on large-scale encrypted data.

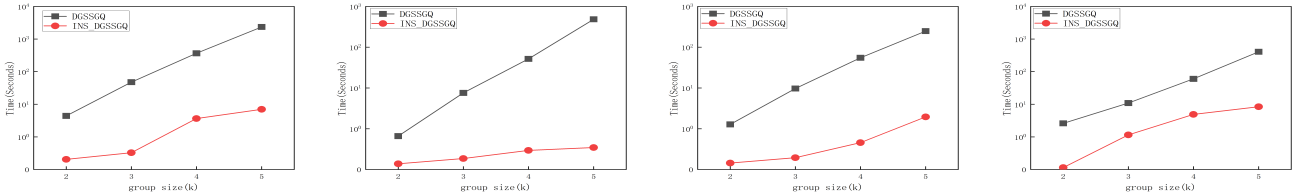
Acknowledgment

The work is supported by the Young and Middle-aged Science and Technology Innovation Talent Support Plan of Shenyang under Grant RC230832.



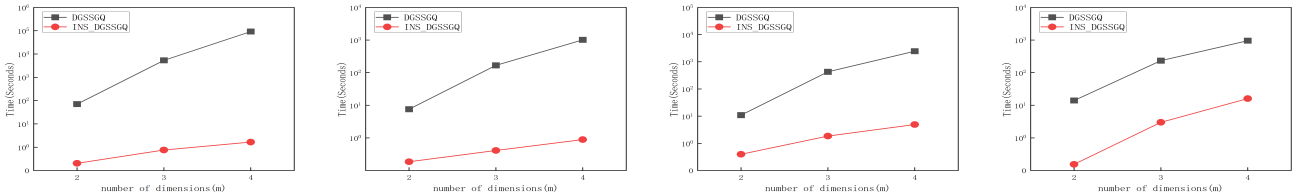
(a) ANTI (b) CORR (c) INDE (d) NBA

Fig. 2: Performance evaluation on tuple insertion when varying n ($g = 3$, $m = 2$, $keysize = 512$)



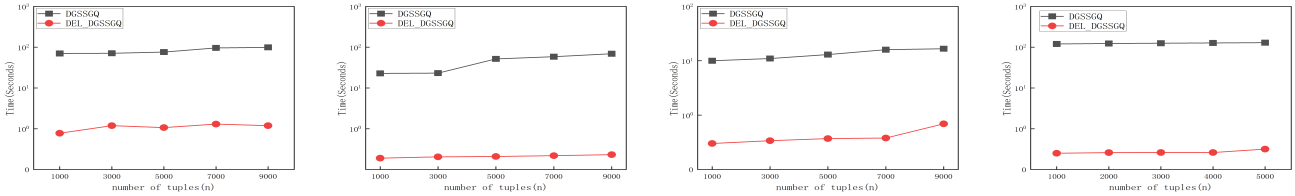
(a) ANTI (b) CORR (c) INDE (d) NBA

Fig. 3: Performance evaluation on tuple insertion when varying g ($m = 2$, $keysize = 512$)



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

Fig. 4: Performance evaluation on tuple insertion when varying m ($g = 3$, $keysize = 512$)



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

Fig. 5: Performance evaluation on tuple deletion when varying n ($g = 3$, $m = 2$, $keysize = 512$)

REFERENCES

[1] S. Börzsönyi, D. Kossmann, and K. Stocker, "The skyline operator," in *IEEE ICDE*, 2001, pp. 421–430.

[2] C. Wang, C. Wang, G. Guo, and et al., "Efficient computation of g-skyline groups," *IEEE TKDE*, vol. 30, no. 4, pp. 674–688, 2018.

[3] Z. Wang, X. Ding, J. Lu, L. Zhang, P. Zhou, K. R. Choo, and H. Jin, "Efficient location-based skyline queries with secure r-tree over encrypted data," *IEEE TKDE*, vol. 35, no. 10, pp. 10 436–10 450, 2023.

[4] C. Li, N. Zhang, N. Hassan, and et al., "On skyline groups," in *ACM CIKM*, 2012, pp. 2119–2123.

[5] N. Zhang, C. Li, N. Hassan, and et al., "On skyline groups," *IEEE TKDE*, vol. 26, no. 4, pp. 942–956, 2014.

[6] J. Liu, L. Xiong, J. Pei, and et al., "Finding pareto optimal groups: Group-based skyline," *ACM VLDB*, vol. 8, no. 13, pp. 2086–2097, 2015.

[7] H. Zhu, X. Li, Q. Liu, and et al., "Top-k dominating queries on skyline groups," *IEEE TKDE*, vol. 32, no. 7, pp. 1431–1444, 2020.

[8] H. Im and S. Park, "Group skyline computation," *Inf. Sci.*, vol. 188, pp. 151–169, 2012.

[9] H. Zhu, P. Zhu, X. Li, and et al., "Top-k skyline groups queries," in *EDBT*, 2017, pp. 442–445.

[10] K. Zhang, H. Gao, X. Han, and et al., "Finding k-dominant g-skyline groups on high dimensional data," *IEEE Access*, vol. 6, pp. 58 521–58 531, 2018.

[11] J. Liu, L. Xiong, J. Pei, and et al., "Group-based skyline for pareto optimal groups," *IEEE TKDE*, vol. 33, no. 7, pp. 2914–2929, 2021.

[12] Y. Elmehdwi, B. K. Samanthula, and W. Jiang, "Secure k-nearest neighbor query over encrypted data in outsourced environments," in *IEEE ICDE*, 2014, pp. 664–675.

[13] A. Liu, K. Zheng, L. Li, and et al., "Efficient secure similarity computation on encrypted trajectory data," in *IEEE ICDE*, 2015, pp. 66–77.

[14] N. Cui, X. Yang, B. Wang, and et al., "Svknn: Efficient secure and verifiable k-nearest neighbor query on the cloud platform*," in *IEEE ICDE*, 2020, pp. 253–264.

[15] Y. Teng, Y. Sun, Z. Shi, D. Jiang, L. Zhao, and C. Fan, "Secure skyline groups queries on encrypted data on cloud platform," in *HPCC/DSS/SmartCity/DependSys*. IEEE, 2021, pp. 551–560.

[16] J. Liu, J. Yang, L. Xiong, and J. Pei, "Secure skyline queries on cloud platform," in *IEEE ICDE*, 2017, pp. 633–644.

[17] J. Liu, J. Yang, L. Xiong, and et al., "Secure and efficient skyline queries on encrypted data," *IEEE TKDE*, vol. 31, no. 7, pp. 1397–1411, 2019.

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

[19] W. Yu, J. Liu, J. Pei, L. Xiong, X. Chen, and Z. Qin, "Efficient contour computation of group-based skyline," *IEEE TKDE*, vol. 32, no. 7, pp. 1317–1332, 2020.

[20] Y. Chung, I. Su, and C. Lee, "Efficient computation of combinatorial skyline queries," *Inf. Syst.*, vol. 38, no. 3, pp. 369–387, 2013.

[21] M. Sun, Y. Teng, F. Zhao, J. Qi, D. Jiang, and C. Fan, "Spatio-textual group skyline query," in *DASFAA*, vol. 13922. Springer, 2023, pp. 34–50.

[22] Z. Wang, L. Zhang, X. Ding, K. R. Choo, and H. Jin, "A dynamic-efficient structure for secure and verifiable location-based skyline queries," *IEEE Trans. Inf. Forensics Secur.*, vol. 18, pp. 920–935, 2023.

[23] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *EUROCRYPT*, vol. 1592, 1999, pp. 223–238.

[24] A. C. Yao, "How to generate and exchange secrets (extended abstract)," in *IEEE FOCS*, 1986, pp. 162–167.

[25] ESPN Enterprises Inc., "ESPN NBA Stats," https://www.espn.com/nba/stats, 2023, accessed: September 1, 2023.

[26] NBA.com, "NBA.com Website," https://stats.nba.com, 2023, accessed: September 1, 2023.