Authentication of Location-based Skyline Queries

Xin Lin†§, Jianliang Xu†, Haibo Hu†
†Department of Computer Science, East China Normal University, Shanghai, China
§Department of Computer Science, Hong Kong Baptist University, Kowloon Tong, Hong Kong
xlin@cs.ecnu.edu.cn  {xujl,haibo}@comp.hkbu.edu.hk

ABSTRACT

In outsourced spatial databases, the location-based service (LBS) provides query services to the clients on behalf of the data owner. However, if the LBS is not trustworthy, it may return incorrect or incomplete query results. Thus, authentication is needed to verify the soundness and completeness of query results. In this paper, we study the authentication problem for location-based skyline queries, which have recently been receiving increasing attention in LBS applications. We propose two authentication methods: one based on the traditional MR-tree index and the other based on a newly developed MR-Sky-tree. Experimental results demonstrate the efficiency of our proposed methods in terms of the authentication cost.

Categories and Subject Descriptors
H.2.4 [Database Management]: Query Processing

General Terms
Algorithm, Experimentation, Performance

Keywords
Data Outsourcing, Skyline Query, Authentication

1. INTRODUCTION

The past decade has seen tremendous amount of research efforts in spatial database technology. In practice, spatial databases from various sources (e.g., land surveys, traffic management, and environmental monitoring) are often outsourced to a service provider (i.e., LBS) because the agencies collecting such data (e.g., governments or non-profit organizations) are usually not able to support advanced query services [7]. However, such an outsourcing model brings a great challenge in query processing. Since the LBS is not the real owner of data, clients may want to authenticate the soundness and completeness of query results: soundness means that the original data is not modified by the LBS, while completeness means that no valid result is missing. This leads to a problem known as authenticated query processing [2, 3].

A general framework of authenticated query processing is shown in Fig. 1. Before outsourcing a spatial dataset to the LBS, the data owner (DO) builds an authenticated data structure (ADS) of the dataset. To support efficient query processing, the ADS is often a tree-like index structure, where the root is signed by the DO using her private key. The LBS server keeps the spatial dataset, as well as the ADS and its root signature. For each user query, the LBS server returns the query results, the root signature, as well as a verification object (VO) that is constructed based on the ADS. The correctness of the query results can be verified by the client using the returned VO, the root signature, and the DO’s public key.

There have been several studies that focused on authenticated processing of location-based queries [7, 8, 9]. However, the existing approaches are confined to proximity-based queries, including range queries [7], nearest-neighbor (NN) queries [9], and shortest-path queries [8]. These queries are not sufficient to the LBS applications that need to consider both spatial and non-spatial attributes of queried objects. A typical example is finding nearby car parks with cheap parking fees, where the distance is a spatial attribute and the parking fee is a non-spatial attribute. In this scenario, a multi-criterion query is clearly more appealing than a conventional spatial query that considers the distance only.

Among various multi-criterion queries, location-based skyline is considered one of the most useful queries and has received considerable attention in LBS research (e.g., [6, 10]). The dynamic nature of spatial attributes makes location-based skyline queries unique and challenging. Take the above carpark-finding scenario for example. The distance from a client to a carpark varies with the location of the client. Therefore, the skyline results would be different with respect to different query locations. In the literature, several algorithms have been proposed to answer location-based skyline queries [6, 10]. However, no existing work has considered authenticated processing of such queries.

In this paper, we study the authenticated location-based
skyline problem and propose two methods. The first method is based on a generic ADS, namely MR-tree [7], that indexes spatial objects with an authenticated R-tree structure. We develop efficient algorithms to construct the VO for each location-based skyline query on the LBS server and to verify the query results on the client. In the second method, we observe that the solution space of spatial objects can be pre-computed in a form of skyline scopes. Thus, we propose a new ADS, called MR-Sky-tree, which indexes the skyline scopes of spatial objects (rather than the objects themselves). By doing so, both the VO construction time and the VO size can be reduced notably. Although MR-Skytree costs more time in building the index structure than MR-tree, its run-time performance is attractive for static or infrequently updated datasets (e.g., carparks, hotels, and points-of-interest). On the whole, our contributions made in this paper can be summarized as follows:

- We identify the problem of authenticated location-based skyline queries that arises in outsourced spatial databases. To the best of our knowledge, this is the first work that studies this problem.
- We propose a basic MR-tree-based method for authenticated location-based skyline queries. We also propose a solution-based MR-Sky-tree to index the skyline scope of each object. Given a location-based skyline query, by employing the MR-Skytree, not only the VO can be quickly constructed but also its size is reduced notably.
- We conduct extensive experiments to evaluate the performance of the proposed methods and algorithms. The results show that our methods and algorithms perform well under various system settings.

The rest of this paper proceeds as follows. Section 2 gives some background and preliminaries of the problem to be studied. We present the MR-tree-based method in Sections 3. In Section 4, we propose the MR-Sky-tree method. The proposed methods and algorithms are experimentally evaluated in Section 5. Finally, this paper is concluded in Section 6.

2. BACKGROUND AND PRELIMINARIES

In this section, we give the formal definition of location-based skyline queries (LSQs) and present some background on the related authentication problem.

2.1 Location-based Skyline Queries (LSQs)

We consider a set of data objects $O$. Each object $o \in O$ is associated with a 2D spatial location attribute (denoted by $o.x$ and $o.y$) and several non-spatial attributes (e.g., parking fee and service quality, denoted by $o.A$, for the $i$-th non-spatial attribute). In this paper, we employ the Euclidean distance metric to measure the spatial proximity.

**Definition 1.** (Non-Spatial Domination) Given two objects $o$ and $o'$, if $o'$ is no worse than $o$ on all non-spatial attributes, then we say $o'$ non-spatially dominates $o$, and $o'$ is a non-spatial dominator of $o$. Formally, it is denoted as $o' \prec_{nS} o$. The set of $o$’s non-spatial dominator objects is denoted as $\text{Dom}(o)$.

**Definition 2.** (Dominance) Given a query point $q$ and two objects $o$ and $o'$, if (1) $o'$ non-spatially dominates $o$, and

<table>
<thead>
<tr>
<th>Carpark</th>
<th>Park fee</th>
<th>Service quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>15</td>
<td>4</td>
</tr>
<tr>
<td>b</td>
<td>13</td>
<td>5</td>
</tr>
<tr>
<td>c</td>
<td>14</td>
<td>5</td>
</tr>
<tr>
<td>d</td>
<td>18</td>
<td>3</td>
</tr>
</tbody>
</table>

Fig. 2 shows an example of LSQ about the aforementioned carpark-finding scenario. Suppose we have four carparks (i.e., $a$, $b$, $c$, and $d$) and two query points (i.e., $q_1$ and $q_2$). The non-spatial attributes of the four carparks are shown in the table, where low parking fee and high service quality are preferred. If the query is issued at $q_1$, the LSQ result is $\{a, b\}$ because $c$ is dominated by $b$ and $d$ is dominated by both $a$ and $b$; if the query is issued at $q_2$, the LSQ result is $\{b, c\}$ because they dominate $a$ and $d$ but not each other.

Given an LSQ query, the authentication problem is how to construct the VO for the client to verify the correctness of the query results. It involves three related issues (refer to Fig. 1): i) ADS design on the data owner; ii) dynamic VO construction for each LSQ query on the LBS server; iii) result verification based on the received VO on the client.

2.2 Query Authentication and MR-Tree

Authenticated query processing has been studied for a long time. Most prior work on query authentication is based on an ADS called Merkle B-tree (MB-tree) [2]. Yang et al. [7] first introduced this problem to the spatial database domain and studied the authentication of spatial range queries. They proposed an authenticated index structure called MR-tree, which combines the ideas of MB-tree and R-tree. Fig. 3
exemplifies the index structure of MR-tree. The data objects are shown in Fig. 3(a). As shown in Fig. 3(b), each leaf node in the MR-tree is identical to that of R-tree, which stores pointers pointing to actual data objects. The digest of a leaf node is obtained by hashing the concatenation of the binary representations of all objects in the node. Each internal node contains a number of entries in the form of $(ptr_i, MBR_i, H_i)^1$ where $ptr_i$, $MBR_i$, and $H_i$ are the pointer, the minimum bounding rectangle, and the digest of the $i$-th child, respectively. The digest of an internal node summarizes the MBRs and digests of all children nodes (e.g., $H_1$ and $H_2$ in Fig. 3(b)). The use of digests makes possible the pruning of index nodes in the VO while being able to verify the correctness of query results.

In [7], the authors also provided a baseline solution to authenticating general skyline queries. The basic idea is to convert a skyline query to a range query. Assuming the skyline results are $p_1, p_2,$ and $p_9$ as shown in Fig. 3(a). To construct the VO, the LBS server only needs to process a range query with the shaded area as the query range because all objects in the blank area are dominated by at least one skyline object. However, this solution is effective only for static skyline queries, but not for LSQs where query points are dynamic. Although it can be adapted to LSQs, it is less efficient than our two proposed methods, as will be shown in the performance evaluation (Section 5). In the following sections, we present the proposed methods for the LSQ authentication problem.

3. BASIC MR-TREE-BASED METHOD

In this section, we develop a basic authentication method by leveraging the existing MR-tree as the ADS. A naive approach is to return the whole MR-tree (see Fig. 3 for an example) excluding the digests as the VO, along with the signature of the root node. Thus, the client can authenticate the integrity of the received MR-tree by the root’s signature, and then perform local query processing over the MR-tree to verify the correctness of the skyline results. However, the drawback of this approach is that the VO size is too large. In the following, we propose to reduce the VO size by pruning some index nodes in the MR-tree, without affecting the result verification ability.

Before presenting the detailed algorithm, we introduce a notion of mindist between a data object/index node and a query point. Without loss of generality, we assume that a lower value is preferred for any non-spatial attribute by a skyline query.

**Definition 4. (MINDIST)** For an object $o$ and a query point $q$, the mindist is defined as: $\text{mindist}(o, q) = \text{dist}_S(o, q) + \sum_i o.A_i$, where $\text{dist}_S(o, q)$ is the distance between $o$ and $q$ in the spatial dimension. For an index node $m$ and a query point $q$, the mindist is defined as: $\text{mindist}(m, q) = \min_{o \in m} \text{mindist}(m, q) + \sum_{i \in m} o.A_i$, where $\min_{o \in m} \text{mindist}(m, q)$ is the minimum distance between $m$ and $q$ in the spatial dimension (see Fig. 4 for an illustration), and $\sum_{i \in m} o.A_i$ is the minimum value in the $i$-th non-spatial dimension over all objects covered by $m$.

Given an object $o$ and a region $R$, if and only if $o$ dominates all objects covered by $R$, we say $o$ dominates $R$. Then, the following lemmas are obtained [4]:

**Lemma 1.** Given a query point $q$, if an object $o$ dominates another object $o'$, we must have $\text{mindist}(o, q) < \text{mindist}(o', q)$; if an object $o$ dominates a region $R$, we must have $\text{mindist}(o, q) < \text{mindist}(R, q)$.

**Lemma 2.** Any data object added to a candidate skyline set based on a traversal of MR-tree in ascending order of mindist is guaranteed to be a final skyline result.

Lemma 2 implies that only a single scan of objects is needed to find the skyline results if the MR-tree is traversed in ascending order of mindist. Based on these two lemmas, the VO construction algorithm is developed in Algorithm 1. The VO and skyline objects are recorded in the data structures voTree and $S$, respectively. Initially, $S$ is empty, and voTree contains only the root (excluding the digest) of the MR-tree (Lines 1–2). We also maintain a min-heap $H$ to keep the index nodes and objects to be scanned in the order of mindist (Line 3). In the loop iteration, each time the top element $e$ is popped up from $H$ (Lines 4–5). In the case where $e$ is an index node, if it is dominated by some object in $S$, then this node will be pruned from voTree and we only keep its digest in the parent node (Lines 6–8); otherwise all children of this node will be inserted into $H$ and voTree (Lines 9–10). In the case where $e$ is a data object, if $e$ is not dominated by any object in $S$, then $e$ is added to $S$ (Lines 11–13).

We use a simple example to illustrate the process of this algorithm. Suppose we have a number of objects $a - l$ as listed in Table 1. Each object includes one spatial ($x$ and $y$) and two non-spatial ($A_1$ and $A_2$) attributes. Assume the

![Figure 4: distS(o, q) and min_distS(m, q)](image)

**Algorithm 1 VO Construction in the Basic Method**

INPUT: Root of MR-tree $mrRoot$; Query Point $q$

OUTPUT: Skyline Set $S$; VO Tree $voTree$

1: $S \leftarrow \emptyset$
2: initialize $voTree$ with $mrRoot$ (excluding the digest)
3: insert $mrRoot$ into a min-heap $H$ ordered by mindist
4: while $H$ is not empty do
5: get the top element $e$ from $H$
6: if $e$ is an index node do
7: if $e$ is dominated by some object in $S$ do
8: prune $e$ and keep its digest in the parent node
9: else
10: insert $e$’s children into $H$ and voTree
11: if $e$ is an object do
12: if $e$ is not dominated by any object in $S$ do
13: $S \leftarrow S \cup \{e\}$
Table 1: An Object Set

<table>
<thead>
<tr>
<th>Obj.</th>
<th>x</th>
<th>y</th>
<th>A1</th>
<th>A2</th>
<th>Obj.</th>
<th>x</th>
<th>y</th>
<th>A1</th>
<th>A2</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>18</td>
<td>8</td>
<td>14</td>
<td>q</td>
<td>16</td>
<td>43</td>
<td>74</td>
<td>80</td>
</tr>
<tr>
<td>b</td>
<td>24</td>
<td>20</td>
<td>16</td>
<td>31</td>
<td>h</td>
<td>43</td>
<td>23</td>
<td>41</td>
<td>61</td>
</tr>
<tr>
<td>c</td>
<td>33</td>
<td>13</td>
<td>28</td>
<td>39</td>
<td>i</td>
<td>45</td>
<td>18</td>
<td>89</td>
<td>16</td>
</tr>
<tr>
<td>d</td>
<td>38</td>
<td>17</td>
<td>88</td>
<td>22</td>
<td>j</td>
<td>56</td>
<td>11</td>
<td>76</td>
<td>11</td>
</tr>
<tr>
<td>e</td>
<td>34</td>
<td>24</td>
<td>75</td>
<td>68</td>
<td>k</td>
<td>46</td>
<td>22</td>
<td>22</td>
<td>55</td>
</tr>
<tr>
<td>f</td>
<td>19</td>
<td>17</td>
<td>32</td>
<td>38</td>
<td>l</td>
<td>12</td>
<td>23</td>
<td>17</td>
<td>25</td>
</tr>
</tbody>
</table>

Table 2: Changes of Data Structures

<table>
<thead>
<tr>
<th>Min-Heap $H$</th>
<th>Skyline Set $S$</th>
<th>VOTree (Fig. 6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_1, N_2$</td>
<td>$\emptyset$</td>
<td>(1)</td>
</tr>
<tr>
<td>$N_3, N_4, N_5, N_6$</td>
<td>$\emptyset$</td>
<td>(2)</td>
</tr>
<tr>
<td>$a, N_4, l, N_2, N_5$</td>
<td>$\emptyset$</td>
<td>(3)</td>
</tr>
<tr>
<td>$l, b, f, c, N_2, N_5$</td>
<td>${a}$</td>
<td>(4)</td>
</tr>
<tr>
<td>$N_2, N_5$</td>
<td>${a, b, f, l}$</td>
<td>(4)</td>
</tr>
<tr>
<td>$N_4, N_7, N_6$</td>
<td>${a, b, f, l}$</td>
<td>(5)</td>
</tr>
<tr>
<td>$N_7, N_6$</td>
<td>${a, b, f, l}$</td>
<td>(6)</td>
</tr>
<tr>
<td>$j, d, i, N_6$</td>
<td>${a, b, f, l}$</td>
<td>(7)</td>
</tr>
<tr>
<td>$N_6$</td>
<td>${a, b, d, f, j, l}$</td>
<td>(7)</td>
</tr>
<tr>
<td>$\emptyset$</td>
<td>${a, b, d, f, j, l}$</td>
<td>(8)</td>
</tr>
</tbody>
</table>

Figure 5: MR-Tree Structure of Our Example

Figure 6: Changes of VOTree

4. MR-SKY-TREE METHOD

In essence, the MR-tree-based method computes the location-based skyline $LSQ(O, q)$ on the fly. For static datasets, the data owner may alternatively precompute the skyline or at least its outline for the entire space. In this section, we propose such a solution-based method.

First, we introduce a notion of skyline scope for each queried object. The skyline scope of each object $o$ is defined as the spatial area in which $o$ will contribute to the skyline results.

Definition 5. (Skyline Scope) For an object $o \in O$, its skyline scope in a 2D plane $P$ is denoted as $SS(o) = \{q \mid q \in P \land o \in LSQ(q, O), \} \text{where } o \in LSQ(q, O) \text{ means } \forall m \in Dom(o), dist_S(o, q) < dist_S(m, q)$.

If $Dom(o)$ is empty, i.e., $o$ has no non-spatial dominators, then $o$ must be a skyline result of any query point $q$. That is, $o$’s skyline scope is the entire space. Otherwise, $o$ will only be a skyline result of $q$ if it is closer to $q$ than all of its non-spatial dominators. The skyline scope of the latter is essentially the Voronoi cell of $o$ in the object set $\{o\} \cup Dom(o)$. Fortunately, the Voronoi cell can be computed using a divide-and-conquer algorithm, with a time complexity of $O(|O|\log |O|)$ [1].

With skyline scopes computed, a location-based skyline
Algorithm 2 VO Construction in the MR-Sky-tree Method

INPUT: Root of MR-Sky-Tree \( mrRoot \); Query Point \( q \);
OUTPUT: Skyline Set \( S \); VO Tree \( voTree \)

1: \( S \leftarrow \emptyset \)
2: initialize \( voTree \) with \( mrRoot \) (excluding the digest)
3: insert \( mrRoot \) into a queue \( \mathcal{Q} \)
4: while \( \mathcal{Q} \) is not empty do
5: \( e \) ← get the top element from \( \mathcal{Q} \)
6: insert \( e \)’s children into \( voTree \)
7: if \( e \) is an index node do
8: \( c \) ← for each child \( c \) of \( e \) do
9: if \( c \) covers \( q \) do
10: insert \( c \) to \( \mathcal{Q} \)
11: else
12: prune \( c \) and keep its digest in the parent node
13: else // \( e \) is a leaf node
14: \( c \) ← for each child \( c \) of \( e \) do
15: if \( c \) covers \( q \) do
16: \( S \leftarrow S \cup \{ c \} \)

Figure 7: Skyline Scopes

The algorithm can be computed by finding the objects whose skyline scopes cover the query point \( q \). Fig. 7 shows an example, where \( a \), \( d \), and \( e \) have no non-spatial dominators, \( a < b < c \), \( d < b < c \), \( c < b < c \). Hence, the skyline scopes of \( a \), \( d \), and \( e \) are the whole space, and the skyline scopes of \( b \) and \( c \) are enclosed by the corresponding perpendicular bisectors as shown in the figure. Since the query point \( q \) is covered by the skyline scopes of \( a \), \( b \), \( d \), and \( e \), the skyline result set is \( \{ a, b, d, e \} \).

To support query authentication, all skyline scopes should be inserted into an MR-tree as data points. We call this special solution-based index as MR-Sky-tree. Then, the original location-based skyline query is reduced to a point-location query on the MR-Sky-tree. Algorithm 2 shows the VO construction procedure. Specifically, it checks, from root and downwards, whether each child of an MR-Sky-tree node covers the query point. If it does, the child is inserted to a queue \( \mathcal{Q} \) for further checking (Lines 9–10). Otherwise, the child is pruned and its MBR and digest are inserted into the final VO for verification (Lines 11–12).

Fig. 8 shows an example. The MR-Sky-tree structure is shown in Figs. 8(a) and 8(b). Assume that a query point \( q \) is only inside of \( a \) and \( c \)’s skyline scopes. The root node is visited first, followed by its children nodes \( N_1 \) and \( N_2 \). As \( N_1 \) covers \( q \), while \( N_2 \) does not, \( N_1 \)’s children \( N_3 \) and \( N_4 \) are visited next, while \( N_2 \)’s MBR and digest (\( N_2, H_2 \)) are inserted to \( voTree \). At Level 1, \( N_3 \) and \( N_4 \) both cover the query point \( q \), so their children, objects \( a, c, b, d \) are visited. While \( \{ a, c \} \) are the final skyline objects, the final \( voTree \) is shown in Fig. 8(c).

To verify this result, the client checks the following three facts: 1) the skyline scopes of all objects in the result set should cover the query point \( q \); 2) no MBRs of pruned nodes and skyline scopes of non-result objects cover \( q \); 3) the root signature matches the digest computed from \( voTree \). The fact 3 ensures the soundness of results, and facts 1-3 ensure the completeness of results.

5. EXPERIMENTS

5.1 Experiment Setup

In this section, we evaluate the performance of our proposed methods and algorithms through experiments. The spatial object set used in the experiments contains 2,249,727 objects representing the centroids of the street segments in California [5]. All testing datasets draw objects randomly from this set. The data space is normalized to a 100,000 Unit \( \times \) 100,000 Unit square, where 1 Unit represents about 1 meter. The non-spatial attribute values of these objects are synthesized with a uniform distribution in the interval \([0, 100,000]\). The page size is 4K bytes and the size of each object is 320 bytes. The hashing function we choose is SHA-512, so the size of each digest is 64 bytes.

We measure the performance with three metrics: index construction time, query processing time, and client verification time. We compare the basic MR-tree-based method (denoted as Basic) and the MR-Sky-tree method (denoted as Sky). In addition, we implement a baseline method (denoted as Baseline), which is a modification of ARQ [7]. It finds the skyline results by the dynamic BBS algorithm [4] and then uses the shaded area like in Fig. 3 as a query range to obtain the VO. In theory, the VO size and verification time of the baseline algorithm are the same as those of the MR-tree-based method, while their query processing times are different since the MR-tree-based method integrates skyline result computation and VO construction.

We conducted our experiments on a workstation (Intel Xeon E5440 2.83GHz CPU) running on Ubuntu Linux Operating System. The simulation codes were written in Java (JDK 1.6). Each measurement is the average result over 100 queries.

5.2 Index Construction Time

Firstly, we measure the ADS index construction time of
its index construction can finish within 5 hours. Nonetheless, even when the cardinality reaches 1,000,000, index construction time for Basic and Baseline are lower than that of Sky because it incurs the computation of the skyline scopes of all objects. Nonetheless, even when the cardinality reaches 1,000,000, its index construction can finish within 5 hours.

5.3 Query Processing Time

Fig. 10 shows the results for query processing time, which includes the time for both skyline result computation and VO construction. We observe that Sky performs the best, especially as the non-spatial dimensionality increases. This is because the cardinality of the skyline result set soars as non-spatial dimensionality increases (from 39 for 2 non-spatial attributes to about 28,000 for 8 non-spatial attributes). The query processing time sharply increases in the Basic and Baseline methods, as the skyline objects distribute dispersely in all dimensions. In contrast, Sky already precomputes the solution in the non-spatial dimensions during index construction, and the skyline scopes of data objects are always 2D regardless of the non-spatial dimensions. Besides, Sky is faster than Baseline because it integrates VO construction into skyline result computation.

5.4 Client Verification Time

Fig. 11(a) shows Basic and Baseline incur a slightly longer verification time than Sky for all dataset cardinality settings. Regarding the non-spatial dimensionality, Fig. 11(b) shows that the costs of Basic and Baseline increase significantly with increasing dimensionality, due to the larger number of skyline results. On the other hand, Sky keeps a relatively stable verification time. This is because the skyline scopes are already precomputed and thus not much affected by the dimensionality.

6. CONCLUSIONS

In this paper, we have studied the problem of authenticated processing of location-based skyline queries. We have proposed two methods for VO construction and result verification: one is based on the existing MR-tree and the other is based on a novel solution-based MR-Sky-tree. The experiments show that the MR-Sky-tree method performs the best in terms of the authentication cost. However, as it needs more time to build/maintain the MR-Sky-tree index, this method is more suitable for static or infrequently updated datasets. For dynamic datasets, the MR-tree-based method is preferred. As for future work, we will extend the authentication problem to continuous skyline queries and road networks.

Acknowledgments

This work is supported by GRF Grants HKBU211510 & HKBU210811 and NSFC Grant 60903169. Lin Xin’s research is also supported by the Fundamental Research Funds for the Central Universities and the Opening Project of Shanghai Key Laboratory of Integrated Administration Technologies for Information Security (No. AGK2008004).

7. REFERENCES