Received: 27 August 2019 | Revised: 9 February 2020 | Accepted: 22 February 2020
DOI: 10.1002/cpe.5747

WILEY

SPECIAL ISSUE PAPER

On authenticated skyline query processing over road networks

Xiaoyu Zhu[1] | Jie Wu[2] | Wei Chang[3] | Md Zakirul Alam Bhuiyan[4] | Kim-Kwang Raymond Choo[5] | Fang Qi[1] | Qin Liu[6] | Guojun Wang[7]

[1]School of Computer Science and Engineering, Central South University, Changsha, China
[2]Center for Networked Computing, Temple University, Philadelphia, Pennsylvania, USA
[3]Department of Computer and Information Sciences, Saint Joseph's University, Philadelphia, Pennsylvania, USA
[4]Department of Computer Science, Fordham University, New York City, New York, USA
[5]Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, Texas, USA
[6]School of Computer Science and Electronic Engineering, Hunan University, Changsha, China
[7]School of Computer Science, Guangzhou University, Guangzhou, China

Correspondence
Guojun Wang, School of Computer Science, Guangzhou University, Guangzhou, 510006, China.
Email: csgjwang@gmail.com

Funding information
High-Level Talents Program of Higher Education in Guangdong Province, Grant/Award Number: 2016ZJ01; Hunan Provincial Natural Science Foundation, Grant/Award Number: 2017JJ2333; National Natural Science Foundation of China, Grant/Award Numbers: 61472451, 61632009; Natural Science Foundation of Guangdong Province, Grant/Award Number: 2017A030308006

Summary
In recent times, many location-based service providers (LBSPs) choose to outsource data query services to third-party cloud service providers (CSPs). This allows users to easily search for points of interests (POIs), such as restaurants and parking lots in their vicinity, using their mobile devices and in-vehicle infotainment units. Skyline query is one potential technique to be deployed for road networks. However, the untrusted CSPs may forge or omit query results, intentionally or not. Therefore, in this article, we posit that by observing the unique properties of skyline query results in road networks, we can bind each POI with four nearby POIs with special properties using signature chain technology. Our proposed approach not only provides users with skyline query result authentication ability over the road network, but also have low communication overhead. Specifically, the overhead analysis and experimental results show that our proposed approach decreases the communication overhead.

KEYWORDS
data outsourcing, LBSP, query authentication, road network, skyline query

1 | INTRODUCTION

Recent advances in mobile computing and positioning technology have partly contributed to the increasingly popularity of location-based services (LBSs),[1] for example, on mobile devices (eg, Android and iOS phones and tablets) and in-vehicle infotainment units. On these Internet-connected devices, users can initiate various queries to search for nearby points of interests (POIs), such as parks, restaurants, and shopping malls. Based on the distance to the user and the numerical attribute (eg, price), the queries return POIs that are closer to the query user and/or preferred in numerical value—these are also referred to as location-based skyline queries (LBSQs) in the literature.[2] For example, suppose a tourist wants to find a nearby restaurant that is in the lower price range, a set of restaurants is returned if there is no other restaurant with shorter distance to the tourist query and price preference. Figure 1 shows a representative road network, where the restaurants are distributed along the streets. The LBSQ results can be unpredictable because as the tourist submits query requests at different locations, the restaurants' distance to the tourist differs.

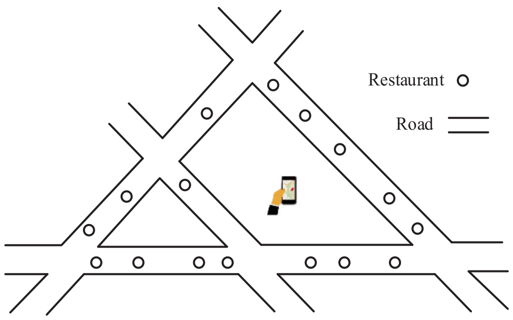


FIGURE 1 An example road network

Cloud computing[3,4] is often used to facilitate LBSs, where LBS provider (LBSP) outsources POI dataset and queries to the third-party cloud service providers (CSPs) in order to minimize storage overhead and improve query quality. Clearly, not all CSPs can be fully trusted and may return results favoring paying POIs or for other reasons (eg, minimizing computing resources). The CSPs may be malfunctioned, or attacked by the virus or hackers, for example, the data breach of dropbox accounts[5] and the healthy data breach on Amazon.[6] Therefore, we need to protect the integrity of the query results as the LBSPs will suffer the consequences of incorrect results, such as decreasing users and hence decreasing revenue. In other words, we need to enable users to verify the soundness and completeness of query results. Specifically, if each query result comes from the original POI dataset, then the query results are correct, and if each skyline result is contained in the query results, then the query results are complete.

Chen et al[7] proposed a location-based skyline query authentication solution to authenticate skyline query results over road networks. However, their approach has several limitations. First, the communication overhead from LBSP to CSP is large, as the LBSP needs to construct Merkle Hash Tree (MHT), which is very large. Second, the communication overhead between CSP and user is high, since each skyline result contains an auxiliary set for authentication. Finally, the dataset update process is inefficient. When the data owner wants to add or delete a data record, the data owner needs to process the data and regenerate the data structure. Thus, we propose a skyline query authentication solution to solve the LBSQ authentication problem. The main contributions of this article are summarized as follows.

- We propose a novel location-based skyline query authentication scheme that can verify the soundness and completeness of LBSQ results, in which each POI is bounded with its neighboring POIs with special attributes through signature chain technique.
- Our proposed solution is efficient in terms of communication cost and update cost. Specifically, the VO size and the update signatures can be reduced notably due to the unique design of our binding technique.
- The security analysis show that our solution can achieve the security goals, and the extensive experimental results show that our solution can largely decrease the communication cost by up to 75% compared with the baseline method.

The rest of the article is organized as follows: Sections 2 and 3, respectively, introduce the related work, the models, and problem. Section 4 introduces the proposed skyline query authentication method, and in Section 5 we extend the method over multiple road segments. The performance analysis and experimental results are presented in Sections 6 and 7, respectively. Specifically, the comparative summary of the performance shows that our proposed approach requires smaller communication overhead. We then conclude this article in Section 8.

2 | RELATED LITERATURE

The Authentication Data Structure (ADS) is proposed to authenticate the query results; there are mainly two ADSs: MHT and signature chain. A large number of authentication solutions based on MHT[8-11] and signature chain[12-15] have been presented in the literature. In signature chain-based approaches, the dataset are ordered and each data are bound with its predecessor and successor in a signature. After receiving a query request from the user, the server returns the query results with a verification object (VO), containing the boundaries and the results' signatures. A number of secure solutions focusing on different kinds of queries have been presented, such as those focusing on range queries,[16,17] top-*k* queries,[18-21] and other query types.[22] In addition, Kumari et al[24,25] focused on user authentication[23] and authenticated key agreement.

The skyline query problem[26-28] has also been widely studied as it can be deployed in a wide range of applications, such as location-based services,[29,30] social network services,[31,32] network storage,[33,34] and on encrypted data.[35] A number of methods that can authenticate skyline query results have been presented in the literature. Yang et al[36] constructed MR-tree based on MHT and R-tree. The MR-tree is usually used in the authentication of spatial queries such as skyline query. The extension version of MR-tree is MR*-tree,[16] which can reduce the VO size, but it needs higher construction cost. Lin et al[37] proposed the location-based skyline query authentication algorithm; the ADS for POIs are constructed using the general data structure MR-tree. The authors constructed a novel data structure MR-Sky-tree based on the unique properties of the spatial data, which can reduce the construction time and VO size. Furthermore, in a later work, Lin et al[38] proposed three query authentication solutions based on the MR-Sky-tree, aiming at continuous location-based skyline query. Their solution can decrease the computation and communication costs using effective range; the visible range was used to let user obtain the query results locally using visible area; and incremental VO was used to avoid transmitting all the VO when the query point moves out of the visible area, which can decrease the communication cost.

Lo and Ghinita[39] observed the property of skyline query authentication: if the query range has some characteristics in the VO, which can largely reduce the communication cost. The authors proposed a skyline query authentication solution based on the domain area, which can decrease the communication cost effectively compared with traditional authentication solution based on MR-tree. Lin et al[40] defined a novel query, which is location-based arbitrary skyline queries (LASOs), and they constructed a novel data structure Partial-S4-tree, which can decrease the query time of the server and decrease the VO size. However, this solution can only be applied to 2D plane, but in actual life, the POIs are distributed in the road network generally. Bothe et al[41] aimed at the challenge of performing skyline query in the encrypted data and proposed a prototype system and query interface eSkyline, which can support skyline query. Their solution did not need to protect the order of each attribute, which also provided a dominate relationship evaluation method for the skyline query. Chen et al[7] constructed a skyline query authentication solution based on MHT, aiming at the skyline query in the road network, which observed the neighbor relationship between POIs and bind the POI with its neighbors. However, as the POI are bound with its neighbors in the MHT, it will cause large update cost.

Liu et al[35] proposed a secure skyline query encryption scheme based on semantic security, which can preserve the data privacy, search privacy, and search results privacy. In addition, the authors also proposed a secure domination protocol, but this solution incurs high cost. Furthermore, Liu et al[42] implemented the secure skyline query in encrypted data based on the homomorphic encryption and achieved semantic security. At the same time, they used data division and lazy merge methods to decrease the computation cost furtherly and proposed the implementation solution using linear and parallel versions. Qaosar et al.[43] proposed a multiparty skyline query scheme while preserving the privacy; the solution integrated several privacy preserving techniques. Hua et al[44] aimed at the medical application scenarios and proposed a privacy preserving solution with skyline query; the user can use their solution to perform accurate skyline diagnosis while preserving the diagnosis model. However, none of these works can be applied in the location-based application scenarios. In our previous work,[45] we designed an efficient skyline query authentication scheme for only one road segment. Hence, in this article, we extend our skyline query authentication method over multiple road segments, which has small communication overhead and low update overhead.

3 | PRELIMINARIES

3.1 | System model

As illustrated in Figure 2, our system model includes LBSPs, CSPs, and some users. The LBSP first generates the signatures over the given dataset, prior to outsourcing the skyline query service to the CSP. The CSP stores the dataset and the signatures. The CSP will compute the skyline results and VO after receiving a skyline query from the user. The user authenticates the skyline results using VO and the LBSP's public key.

3.2 | Security goal

We assume that LBSP is trustworthy. However, we do not assume CSP to be trusted, in the sense that the CSP may return incorrect or incomplete skyline results. Thus, in this article, we aim to authenticate the skyline results over road networks. Specifically, using our proposed approach, we will authenticate the LBSQ results and achieve both soundness and completeness.

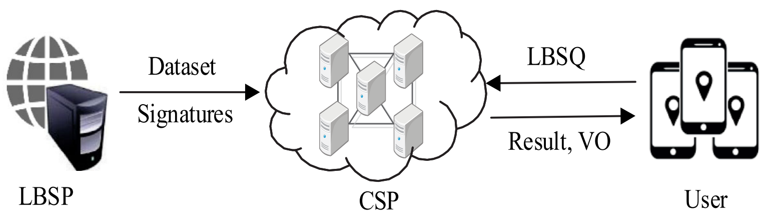


FIGURE 2 System model

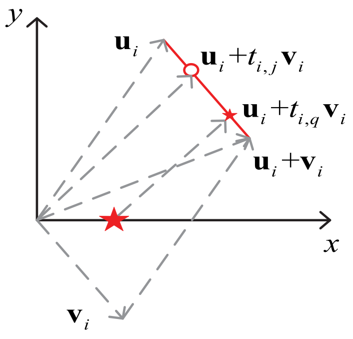

y
$\mathbf{u}_i$
$\mathbf{u}_i+t_{i,j}\mathbf{v}_i$
$\mathbf{u}_i+t_{i,q}\mathbf{v}_i$
$\mathbf{u}_i+\mathbf{v}_i$
x
$\mathbf{v}_i$

FIGURE 3 Representation of a road segment

Soundness: All returned POIs come from the original POI dataset, and they have not been tampered with.

Completeness: All POIs satisfying the skyline query are returned to the user.

3.3 | Definitions

The road networks are represented by a planar graph $G = (\mathbb{V}, \mathbb{E})$, where $\mathbb{V}$ represents the vertices, and $\mathbb{E} = \{e_1, \dots, e_m\}$ represents the road segments. We use $\{e_i = \boldsymbol{u}_i + t\boldsymbol{v}_i\}$ to denote the segment, where $\boldsymbol{u}_i, \boldsymbol{v}_i \in \mathbb{R}^2$ are two reference vectors, and $\boldsymbol{u}_i$ and $\boldsymbol{u}_i + \boldsymbol{v}_i$ are two end points. The road segment is shown in Figure 3.

The POI dataset is represented as $\mathcal{O} = \bigcup_{i=1}^{m} \mathcal{O}_i$, where $\mathcal{O}_i$ denotes the POIs over road segment e_i. Each POI $o_{i,j} = \{t_{i,j}, \lambda_{i,j}\}$ denotes the jth POI over e_i, where $t_{i,j}$ is $o_{i,j}$'s relative position, and $\lambda_{i,j}$ is the numeric attribute of interest.

Suppose there is a skyline query request $\langle q, I \rangle$. $q \in \mathbb{R}^2$ is the location of the query, and $I \subseteq \{1, \dots, m\}$ is the indexes of $\mathbb{E} = \{e_1, \dots, e_m\}$. The query results are denoted as $\text{sky}(\bigcup_{i \in I} \mathcal{O}_i | q)$.

The query q can be projected on segment e_i and its relative position is denoted as $t_{i,q}$. We assume that a POI only belongs to only one road segment, and the POIs are at different positions and contain different numerical values.

The following are the definitions of LBSQ, in which a lower price is preferred.

Definition 1 (Distance). For any two POIs $o_{i,j}$ and $o_{i',j'}$ in one road segment e_i, the distance between $o_{i,j}$ and $o_{i',j'}$ is denoted as $d(o_{i,j}, o_{i',j'}) = |t_{i,j} - t_{i',j'}|$.

Definition 2 (Query distance). For a POI $o_{i,j}$ in road segment e_i, the query distance between query position q and POI $o_{i,j}$ is denoted as $d(q, o_{i,j}) = |t_{i,j} - t_{i,q}|$.

Definition 3 (Dominance). For any two POIs $o_{i,j}$ and $o_{i',j'}$ in one road segment e_i, we say $o_{i,j}$ dominates $o_{i',j'}$ iff $d(q, o_{i,j}) < d(q, o_{i',j'})$ and $\lambda_{i,j} \le \lambda_{i',j'}$, or $d(q, o_{i,j}) \le d(q, o_{i',j'})$ and $\lambda_{i,j} < \lambda_{i',j'}$.

Definition 4 (LBSQ). A LBSQ retrieves the skyline results $\text{sky}(\bigcup_{i \in I} \mathcal{O}_i | q)$, each of which is not dominated by any other POI in $\bigcup_{i \in I} \mathcal{O}_i$.

3.4 | General design of the binding relations

For simplicity, in Section 4, we consider a simple case and give a basic solution, where there is only one road segment. The general case and the extended solution are discussed in Section 5. In order to guarantee that CSP will not be able to insert/delete POIs from a skyline query result, the LBSP has to bind the corresponding POIs in advance. However, for the LBSP, it is impossible to foresee future users' query locations. It is also impracticable to enumerate all skyline query results. Since all POIs are in the same road segment, if a POI o_i is included in a skyline query result, then the next selected one must come from o_i's left or right side, which is either closer to the query location or has a numerical value smaller than o_i. Based on this observation, for each POI o_i, our proposed approach finds four nearby POIs with special properties, and binds o_i with them. More specifically, these four POIs are called the distance neighbors and skyline neighbors of o_i.

4 | BASIC SOLUTION

We present the basic solution to authenticate the skyline query results. We only consider the simplest and basic condition in this section, where there is only one road segment in the road network, distributed with n POIs, and the user can issue a LBSQ to the CSP. In the following subsections, we first summarize our observations of the LBSQ results, before introducing the notions of distance and skyline neighbors. Finally, we introduce our solution.

FIGURE 4 An example of POIs on a road segment

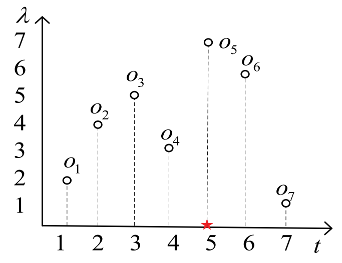
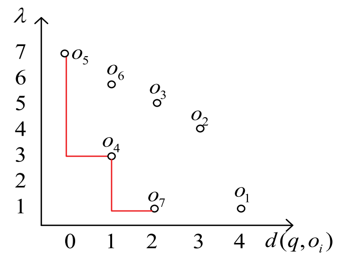


FIGURE 5 LBSQ for query q



4.1 | Properties of LBSQ

We denote $\mathcal{O}$ as the POI dataset over $e = \{\boldsymbol{u} + t\boldsymbol{v}\}$, and the location-based skyline query results $\text{sky}(\mathcal{O}|q)$ is determined by t_q.

$$\text{sky}(\mathcal{O}|q) = \text{sky}(\mathcal{O}|t_q), \tag{1}$$

t_q is defined as

$$t_q = \frac{\boldsymbol{v}^T(q - \boldsymbol{u})}{||\boldsymbol{v}||_2^2}. \tag{2}$$

In Figure 4, each POI o_i is denoted as (t_i, λ_i), where t_i denotes the relative position and λ_i is the numerical attribute. Figure 5 presents the query results corresponding to the query q, in which $d(q, o_i) = |t_i - t_q|$ denotes the query distance and $\text{sky}(\mathcal{O}|q) = \{o_5, o_4, o_7\}$ denotes the results. The nonskyline POI is dominated by some skyline result, for instance, o_4 dominates o_6, o_3 is dominated by o_7.

We observe that the LBSQ results contain the following properties:

(1) The results can be sorted according to query distance $d(q, o_5) < d(q, o_4) < \ldots < d(q, o_7)$ and by numerical value $\lambda_5 > \lambda_4 > \lambda_7$. The query results are $\{o_5, o_4, o_7\}$. The first result o_5 is closest to the query position in all the POIs, and its query distance $d(q, o_5)$ is the smallest.

From the observation, the first query result has a unique property, in the sense that it is the only POI closer to the query position than its left and right neighbor POIs. For example, only o_5 is closer to the query position than its left neighbor o_4 and right neighbor o_6, o_3's right neighbor o_4 is closer to the query position than o_3, o_7's left neighbor o_6 is closer to the query position than o_7.

(2) If a POI o_i is included in the results, the next selected result o_{i+1} should be selected from the candidate POIs whose numerical value is smaller than o_i's numerical value, and o_{i+1} is closest to the query position in all the candidate POIs. For example, o_5 is the first skyline result, the next skyline result selects from the candidate POIs $\mathcal{O}_{\text{candidate}} = \{o_1, o_2, o_3, o_4, o_6, o_7\}$, o_4 is closest to the query position in the candidate POIs, and its query distance $d(q, o_4)$ is the smallest. o_4 is the second result; the next skyline result selects from the candidate POIs $\mathcal{O}_{\text{candidate}} = \{o_1, o_7\}$, o_7 is closest to the query position in the candidate POIs. As o_1 is dominated by o_7, thus the query results are $\{o_5, o_4, o_7\}$.

From the observation, if o_i is a skyline result, the next skyline result o_{i+1} has one special property. o_{i+1} is selected from the candidate POIs, whose numerical value are smaller than o_i's numerical value. o_{i+1} is chosen from o_i's left and right neighbors in the candidate POIs. For example, o_5 is a skyline result; the next skyline result o_4 is o_5's left neighbor chosen from the candidate POIs $\{o_1, o_2, o_3, o_4, o_6, o_7\}$. o_4 is a skyline result; the next skyline result o_7 is o_4's right neighbor chosen from the candidate POIs $\{o_1, o_7\}$.

Based on the two properties observed from the skyline results, we define distance neighbors, which are used to find the POI closest to the query position. We define the skyline neighbors, which are used to find the next skyline result from the candidate POIs.

(1) **Distance neighbors**

For any $o_i \in \mathcal{O}$, its left and right distance neighbors are defined as $N_l(o_i)$ and $N_r(o_i)$. o_i's left and right distance neighbors actually means o_i's left and right neighbors according to the numerical value.

(2) **Skyline neighbors**

For each $o_i \in \mathcal{O}$, its left and right skyline neighbors are defined as $N_l^-(o_i)$ and $N_r^-(o_i)$. o_i's left and right skyline neighbors actually means o_i's left and right neighbors in the candidate POIs.

4.2 | Data preprocessing

The POIs $\mathcal{O} = \{o_i | 1 \le i \le n\}$ are preprocessed by the LBSP; each POI is denoted as $o_i = (t_i, \lambda_i)$, where $t_1 < t_2 < \ldots < t_n$.

(1) Compute distance neighbors

For each POI o_i, find its distance neighbors $N_l(o_i)$ and $N_r(o_i)$. If the left or right distance neighbor not exists, then assign it as o_{null}. o_1's left and right distance neighbors are o_{null} and o_2, respectively. o_4's left and right distance neighbors are o_3 and o_5, respectively. o_7's left and right distance neighbors are o_6 and o_{null}, respectively.

(2) Compute skyline neighbors

For each POI o_i, find skyline neighbor $N_l^-(o_i)$ and $N_r^-(o_i)$. If the skyline neighbor not exists, then assign it as o_{null}. o_1's skyline neighbors are o_{null} and o_7, respectively. o_4's left and right neighbors are o_1 and o_7, respectively. o_7's left and right neighbors are o_{null} and o_{null}, respectively.

(3) Construct signatures

For each POI o_i, $i \in [1, n]$, generate a signature $s(o_i)$:

$$s(o_i) = \text{Sig}(H(H(o_i)|H(N_l(o_i))|H(N_r(o_i))|H(N_l^-(o_i))|H(N_r^-(o_i)))). \tag{3}$$

Here, $H(\cdot)$ is a hash function, and Sig is a signature generation algorithm. The total number of signatures is n. For example,

o_1's signature is $s(o_1) = \text{Sig}(H(H(o_1)|H(o_{\text{null}})|H(o_2)|H(o_{\text{null}})|H(o_7)))$.

o_4's signature is $s(o_4) = \text{Sig}(H(H(o_4)|H(o_3)|H(o_5)|H(o_1)|H(o_7)))$.

o_7's signature is $s(o_7) = \text{Sig}(H(H(o_7)|H(o_6)|H(o_{\text{null}})|H(o_{\text{null}})|H(o_{\text{null}})))$.

The POIs and signatures are sent to the CSP. Algorithm 1 shows the data processing details.

Algorithm 1. Data Preprocessing

Input: The POI dataset $\mathcal{O}$.

Output: The signatures.

```
1: for each record $o_i \in \mathcal{O}$ do
2:     Compute $o_i$'s left distance neighbor $N_l(o_i)$ and right distance neighbor $N_r(o_i)$
3:     Compute $o_i$'s left skyline neighbor $N_l^-(o_i)$ and right skyline neighbor $N_r^-(o_i)$
4:     Construct a signature $s(o_i)$ using Eqn. 3
5: end for
6: Return all the signatures $\{s(o_i)\}$
```

4.3 | Query processing

After receiving a LBSQ q from the user, the CSP first queries the POIs and obtains sky$(\mathcal{O}|t_q)$.

(1) Compute t_q from q.

(2) For each $o_i \in \mathcal{O}$, compute each POI's query distance $d(q, o_i)$ and select the POI with the minimum distance as the first query result. If there exist two POIs with minimum distance, select the POI with smaller numerical value, assign the first result as o_x, and put o_x into sky$(\mathcal{O}|t_q)$.

(3) According to the skyline result o_x, select the POI closer the query position as the next skyline result o_{x+1} from candidate POIs, where the two candidate POIs are o_x's skyline neighbors. If the two candidate POIs' distances are the same, select the POI with smaller numerical value.

(4) Repeat step (3) until o_{x+1}'s skyline neighbors are o_{null} and o_{null}.

In Figure 4, select o_5 as the first result because o_5 is closest to q. Then select the next skyline result from two candidate POIs o_4 and o_6. The next skyline neighbor is o_4 because $\lambda_4 < \lambda_6$. Then continue to select o_7 as the next skyline result, the LBSQ results are sky$(\mathcal{O}|t_5) = \{o_5, o_4, o_7\}$.

After computing the query results, the CSP computes the VO. For each $o_x \in$ sky$(\mathcal{O}|t_q)$, the CSP returns its neighbors $N_l^-(o_i)$, $N_r^-(o_i)$, $N_l(o_i)$, and $N_r(o_i)$ and its signature $s(o_i)$.

4.4 | Query result verification

Based on the query results and VO returned by the CSP, the user can authenticate the results furtherly. Assume that the query results are $\{o_1, \dots, o_u\}$, where $d(q, o_1) < d(q, o_2) < \dots < d(q, o_u)$. The VO includes the signatures, $\{s(o_1), \dots, s(o_u)\}$, and the neighbors, $\{N_l(o_1), N_r(o_1), N_l^-(o_1), N_r^-(o_1), \dots\}$.

(1) The user checks the correctness of the query results. For each $o_x \in \{o_1, \dots, o_u\}$, check whether

$$\text{Sig}^{-1}(s(o_x)) = H(H(o_x)|H(N_l(o_x))|H(N_r(o_x))|H(N_l^-(o_x))|H(N_r^-(o_x))). \quad (4)$$

Here, the user uses Sig^{-1} and the public key from LBSP to verify the signature.

(2) If the query results are correct, then continue to verify the completeness of the query results.

1. Check whether o_1 is the first query result.

If $d(q, o_1) < d(q, N_l(o_1))$ and $d(q, o_1) < d(q, N_r(o_1))$,

or $d(q, o_1) = d(q, N_l(o_1))$ and $\lambda_1 < \lambda(N_l(o_1))$,

or $d(q, o_1) = d(q, N_r(o_1))$ and $\lambda_1 < \lambda(N_r(o_1))$.

Then the verification is passed. Otherwise, the verification is failed.

In which, $\lambda(N_l(o_1))$ and $\lambda(N_r(o_1))$ are the numerical values of $N_l(o_1)$ and $N_r(o_1)$, respectively.

2. If o_x is the skyline result, check whether o_{x+1} is the actual next skyline result chosen from o_x's candidate skyline neighbors.

If $N_l^-(o_x) = o_{x+1}$ and $d(q, N_l^-(o_x)) < d(q, N_r^-(o_x))$,

or $N_l^-(o_x) = o_{x+1}$ and $d(q, N_l^-(o_x)) = d(q, N_r^-(o_x))$ and $\lambda(N_l^-(o_x)) < \lambda(N_r^-(o_x))$,

or $N_r^-(o_x) = o_{x+1}$ and $d(q, N_r^-(o_x)) < d(q, N_l^-(o_x))$,

or $N_r^-(o_x) = o_{x+1}$ and $d(q, N_r^-(o_x)) = d(q, N_l^-(o_x))$ and $\lambda(N_r^-(o_x)) < \lambda(N_l^-(o_x))$.

Then the verification is passed. Otherwise, the verification is failed.

In which, $\lambda(N_l^-(o_x))$ and $\lambda(N_r^-(o_x))$ are the numerical values of $N_l^-(o_x)$ and $N_r^-(o_x)$, respectively.

3. Check whether o_u is the last skyline result.

If o_u's skyline neighbors are o_{null} and o_{null}, then the verification is passed. Otherwise, the verification is failed.

If all the steps are passed, then it proves that the query results are correct and complete. Otherwise, the verification is failed. Algorithm 2 shows the query result verification details.

Algorithm 2. Query result verification

Input: Query q, results $\text{sky}(\mathcal{O}|t_q)$.

Output: success or fail.

```
1: for For each result $o_x \in \text{sky}(\mathcal{O}|t_q)$ do
2:     Find $o_x$'s signature $s(o_x)$
3:     Find $o_x$'s neighbors $N_l(o_x)$, $N_r(o_x)$, $N_l^-(o_x)$, and $N_r^-(o_x)$
4:     if Eqn. 4 is not satisfied then
5:         Return fail
6:     end if
7: end for
8: if $o_1$ is not the first query result then
9:     Return fail
10: end if
11: for each $o_x, x \in \{1, u-1\}$ in $\text{sky}(\mathcal{O}|t_q)$ do
12:     if $o_{x+1}$ is not the next skyline result then
13:         Return fail
14:     end if
15: end for
16: if $o_u$ is not the last skyline result then
17:     Return fail
18: end if
19: Return success
```

The results are $\{o_5, o_4, o_7\}$, the VO includes the signatures $\{s(o_5), s(o_4), s(o_7)\}$ and the neighbors of the results. For each $o_x \in \{o_5, o_4, o_7\}$, the user first verifies the signatures. Then check that whether o_5 is the first result, o_4 is the next result after o_5, o_7 is the next result after o_4, and o_7 is the last result.

As for data update, if the LBSP needs to insert or delete a POI record o_i, it first finds o_i's distance neighbors, then it finds the POI records whose skyline neighbors contains o_i. Finally, LBSP generates a new signature $s(o_i)$ for o_i, o_i's distance neighbors $N_l(o_i)$ and $N_r(o_i)$, and POIs whose skyline neighbor $N_l^-(o_i)$ or $N_r^-(o_i)$ is o_i. For example, in Figure 4, if o_3 is deleted, o_2 and o_4's signatures should be regenerated.

5 | EXTENDED SOLUTION

We present the extended solution that extends the scenario to multiple road segments. Under the extended solution, for POIs in each independent road segment, the preprocess is the same as the basic solution. The user submits an LBSQ $\langle q, I\rangle$ to ask for $\text{sky}(\cup_{i\in I}\mathcal{O}_i|q)$ and receives skyline results independently. Finally, it authenticates the skyline results.

5.1 | Data preprocessing

The POI dataset is denoted as $\{\mathcal{O}_i\}_{i=1}^m$, there are m road segments, and $\mathcal{O}_i = \{o_{i,j}\}_{j=1}^{n_i}$ denotes the POIs on the ith road. Each POI is denoted as $o_{i,j} = \langle t_{i,j}, \lambda_{i,j}\rangle$. For each road segment, preprocess the dataset as basic solution.

(1) Compute distance neighbors
For each POI $o_{i,j}, i \in [1, m], j \in [1, n_i]$, find its distance neighbors $N_l(o_{i,j})$ and $N_r(o_{i,j})$. If the left or right distance neighbor not exists, then assign it as o_{null}.
(2) Compute skyline neighbors
For each POI $o_{i,j}$, find skyline neighbor $N_l^-(o_{i,j})$ and $N_r^-(o_{i,j})$. If the skyline neighbor not exists, then assign it as o_{null}.
(3) Construct signatures
For each POI $o_{i,j}, i \in [1, m], j \in [1, n_i]$, generate a signature $s(o_{i,j})$:

$$s(o_{i,j}) = \text{Sig}(H(H(o_{i,j})|H(N_l(o_{i,j}))|H(N_r(o_{i,j}))|H(N_l^-(o_{i,j}))|H(N_r^-(o_{i,j})))). \tag{5}$$

The POIs and their signatures are sent to the CSP.

5.2 | Query processing

The CSP receives a query $\langle q, I\rangle$ from the user, where q is the query position, I is the indexes of the roads; the CSP queries the POIs $\{\mathcal{O}_i\}_{i=1}^m$ and obtains $\text{sky}(\bigcup_{i\in I}\mathcal{O}_i|q)$.

(1) For each road $e_i (i \in I)$, compute $t_{q,i}$ as q's relative position on the road segment.
(2) For each $o_{i,j} \in \mathcal{O}_i$, compute each POI's query distance $d(q, o_{i,j})$, select the POI with the minimum distance as the first query result. If there exist two POIs with minimum distance, select the POI with smaller numerical value, assign the first result as $o_{i,x}$, and put $o_{i,x}$ into $\text{sky}(\bigcup_{i\in I}\mathcal{O}_i|q)$.
(3) According to the skyline result $o_{i,x}$, select the POI closer the query position as the next skyline result $o_{i,x+1}$ from candidate POIs, where the two candidate POIs are $o_{i,x}$'s skyline neighbors. If the two candidate POIs' distances are the same, select the POI with smaller numerical value.
(4) Repeat step (3) until $o_{i,x+1}$'s skyline neighbor are o_{null} and o_{null}.

After computing the query results, the CSP computes the VO. For each $o_{i,x} \in \text{sky}(\bigcup_{i\in I}\mathcal{O}_i|q)$, the CSP returns its neighbors $N_l^-(o_{i,x}), N_r^-(o_{i,x}), N_l(o_{i,x})$, and $N_r(o_{i,x})$ and its signature $s(o_{i,x})$.

5.3 | Query result verification

Based on the query results and VO returned by the CSP, the user can furtherly authenticate the soundness and completeness of the query results. For each road segment, the user authenticate the results as the basic solution.

(1) Verify the soundness of the query result and check whether the signatures of the query results are correct or not.
(2) If the query results are correct, then the user continues to verify the completeness of the query results.

If both of the steps are passed, then it proves that the query results are sound and complete. Otherwise, the verification fails.

6 | PERFORMANCE ANALYSIS

6.1 | Security analysis

Theorem 1. *Let* $\text{sky}(\mathcal{O}|q) = \{o_1, o_2, \dots, o_u\}$ *be the set of LBSQ results. For each* $o_x \in \text{sky}(\mathcal{O}|q), x \in [1, u-1]$, *the next skyline result* o_{x+1} *should be selected from* o_x*'s skyline neighbors* $N_l^-(o_x)$ *and* $N_r^-(o_x)$.

Proof. We prove Theorem 1 via the following three steps:

1. The next result's numerical value is smaller than its previous neighbor's numerical value.
2. The query's relative position is between o_x's left and right skyline neighbors. The left and right skyline neighbors are the only two candidates closest to the query position.

Step 1: We prove that the next skyline result should be chosen from subsets with numerical value smaller than o_x's numerical value λ_x. From the definition of skyline neighbor, we can see that the left and right skyline neighbors are chosen from two subsets with numerical value smaller than λ_x.

Step 2: We prove that the query's project position is between o_x's left and right skyline neighbors. First, we prove that t_q cannot be at the left side of the left neighbor $N_l^-(o_x)$. If the query position is at the left side of $N_l^-(o_x)$, $t_q \le t(N_l^-(o_x))$, then o_x's query distance $d(q, o_x)$ is larger than $N_l^-(o_x)$'s query distance $d(q, N_l^-(o_x))$, $d(q, o_x) > d(q, N_l^-(o_x))$. Then o_x is dominated by $N_l^-(o_x)$ because o_x's query distance and numerical value are both larger than $N_l^-(o_x)$'s query distance and numerical value, respectively, which is in contradiction with that o_x is a skyline result. Similarly, we can prove that the query position cannot be at the right side of the right neighbor $N_r^-(o_x)$. Thus, t_q is between o_x's left and right skyline neighbors.

Step 3: We prove that for all POIs that have a smaller numerical value than o_x's numerical value, the skyline neighbors are the only two candidates which are closest to the query position. For the left skyline neighbor $N_l^-(o_x)$, the POIs on its left side have larger query distance than $t(N_l^-(o_x))$. For the right skyline neighbor $N_r^-(o_x)$, the POIs on its right side have larger query distance than $t(N_r^-(o_x))$. Meanwhile, there does not exist any POI between $N_l^-(o_x)$ and $N_r^-(o_x)$ because they are the closest POIs to o_x.

In summarize, $N_l^-(o_x)$ and $N_r^-(o_x)$ are the only two candidate POIs, which are closest to query position in all the candidate POIs, whose numerical value are smaller than λ_x. Thus, the next skyline result o_{x+1} should be selected from $N_l^-(o_x)$ and $N_r^-(o_x)$. ■

Theorem 2. *Our proposed basic and extended solutions can authenticate the skyline results.*

Proof. Assume that $\text{sky}(\mathcal{O}|t_q) = \{o_1, \dots, o_u\}$ are the skyline results over road networks, in which $d(q, o_1) < d(q, o_2) < \dots < d(q, o_u)$.

First, we prove that $\text{sky}(\mathcal{O}|t_q)$ are sound: As the adversary does not have the secret key of LBSP, thus he cannot generate a correct signature for the forged POI.

Next, we prove that $\text{sky}(\mathcal{O}|t_q)$ are complete:

Case 1: o_1 is the first skyline result. Because $d(q, o_1) < d(q, N_l(o_1))$ and $d(q, o_1) < d(q, N_r(o_1))$, only o_1 has the smallest query distance, thus o_1 is the first result.

Case 2: o_u is the last skyline result. The last result o_u's skyline neighbors are o_{null} and o_{null}.

Case 3: o_x and o_{x+1} in $\text{sky}(\mathcal{O}|t_q)$ are consecutive. If the adversary adds or deletes a result, the user can detect the behavior.

The extended solution is based on basic solution, thus it can also authenticate the skyline results. ■

6.2 | Overhead

We compare our proposed basic solution 1D* as the baseline solution 1D[7] from four aspects, namely, LBSP's computation overhead, communication overhead from LSBP to CSP, communication overhead from CSP to user, and user's computation overhead. n is the number of POIs, k and k' are the number of query results of 1D* and 1D, respectively. C_S, C_H, and C_V denotes the signature construction cost, hash computation cost, and signature verification cost. S_S and S_H denote the size of a signature and a digest.

1) LBSP's computation overhead. In 1D*, the LBSP preprocesses the dataset; each POI is bound with its four neighbors. Assume that the dataset has n POIs, the LBSP needs to construct n signatures, and the computation cost is $C_S(n)$. In 1D, the LBSP needs to construct the MHT as the ADS, and the computation cost is $C_S(1) + C_H(2n)$.
2) The communication overhead from LSBP to CSP. In 1D*, n POIs correspond to n signatures; the communication cost is $S_S(n)$. In 1D, the MHT contains one root signature and $2n$ digests, thus the communication cost is $S_S(1) + S_H(2n)$.
3) The communication overhead from CSP to user. In 1D*, each POI corresponds to one signature; the communication cost is $S_S(k)$. In 1D, each result corresponds to logn digest and one root signature, thus the communication cost is $S_S(1) + S_H(k' \text{logn})$, where k' is larger than k because SKY contains some nonskyline results.
4) User's computation overhead. In 1D*, each POI corresponds to one signature verification operation; the computation cost is $C_V(k)$. In 1D, the user needs to reconstruct the root hash for k' results, and one root signature verification. The computation cost is $C_V(1) + C_H(k' \text{logn})$.

7 | SIMULATION RESULTS

In this section, we evaluate the performance of our proposed solutions from the following four aspects: LBSP's computation overhead, communication overhead from LSBP to CSP, communication overhead from CSP to user, and user's computation overhead.

We adopt the similar road network setting as the baseline solution,[7] and the experiment data are generated through simulation. The road network contains 20 road segments; each segment contains 1000 POIs, and each POI contains two values: the relative position and numerical value. The relative position and the numerical value of each POI are all chosen from [0,1] randomly. We choose SHA-1 to generate the digest, RSA (512 bits) to generate the signatures. 1D* and 2D* are used to denote our basic and extended solutions, respectively, while 1D, 2D, and 2D+ are used to denote 1D, 2D, and extended 2D of baseline solutions,[7] respectively.

7.1 | LBSP's computation overhead

The LBSP's computation overhead comes from constructing signatures for each POI, which are mainly affected by the number of POIs n and the number of road segments m. In Figure 6A, we study the impact of n on the LBSP's computation overhead. n changes from 50 to 1000, and m is set as 20. The experiment results show that the LBSP's computation overhead increases with n because the constructed signatures increase as n increases, thus the LBSP's computation cost increases.

Figure 6B shows the impact of m on LBSP's computation overhead. We fix n as 1000, and m changes from 2 to 20. The experimental results show that the LSBP's computation overhead increases with m. When m equals 20, the LBSP's computation time of 1D* is 1.2 seconds, while 2D*'s cost is 25.1 seconds.

7.2 | The communication overhead from LBSP to CSP

The communication overhead from LBSP to CSP is mainly the signature size of the POIs, which are affected by n and m. Figure 7A shows the effect of n on the communication overhead from LBSP to CSP. The experimental results show that the communication overhead from LBSP to CSP increases with n because each POI's signature size is fixed, thus 1D* and 2D* linearly increase with the number of POIs n.

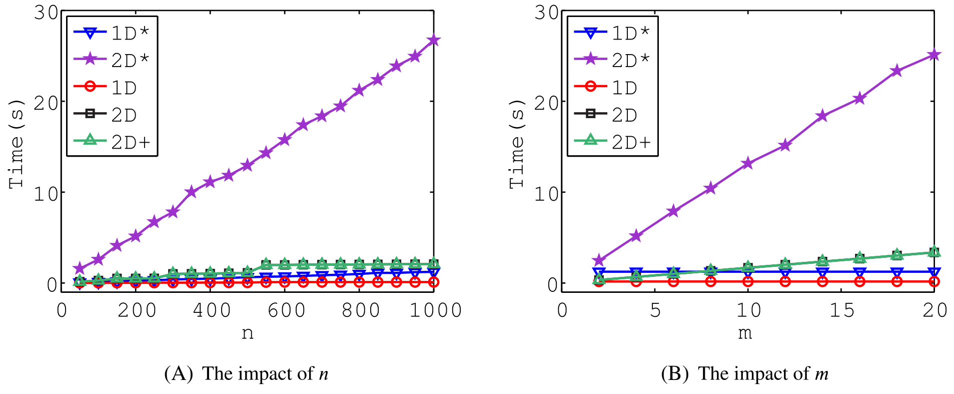
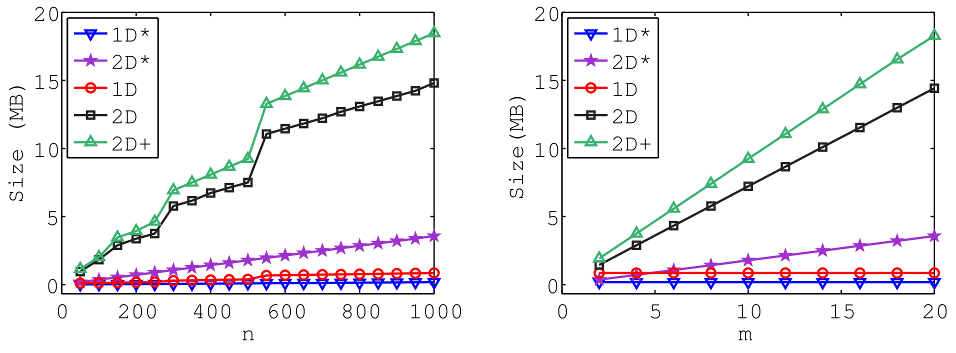


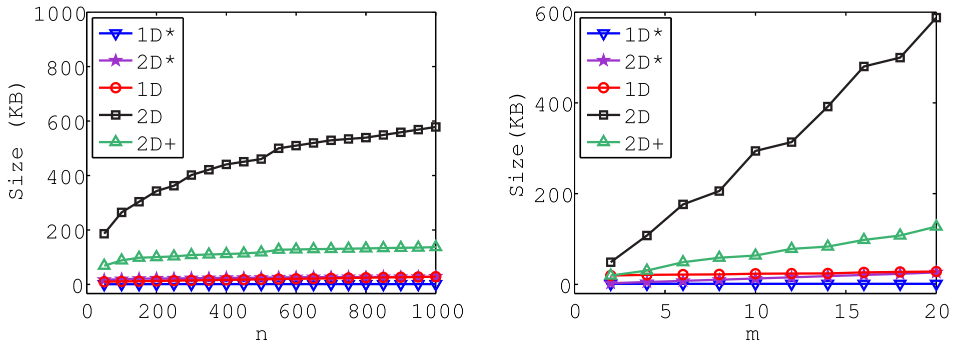
FIGURE 6 LBSP's computation overhead

FIGURE 7 LBSP-CSP communication overhead



(A) The impact of *n* (B) The impact of *m*

FIGURE 8 CSP-user communication overhead



(A) The impact of *n* (B) The impact of *m*

When $n = 1000$, the communication overhead of 1D* is 0.18 MB, 1D is 0.84 MB, 2D* is 3.57 MB, 2D is 14.8 MB, and 2D+ is 18.5 MB. Totally, the 1D*'s communication overhead from LBSP to CSP is smaller than 1D, and 2D*'s communication overhead is smaller than 2D and 2D+. For example, when $n = 1000$, 1D* decreases 79% communication overhead compared with 1D, while 2D* solution decreases 76% communication overhead compared with 2D, and 2D* solution decreases 80% communication overhead compared with 2D+. In 1D* and 2D*, each POI needs to construct a signature using signature chain. As the number of POIs in each road increases, the communication overhead linearly increases. However, in 1D, 2D and 2D+, the LBSP needs to construct a MHT for the POIs, the communication overhead increases with the number of POIs *n* on each road. Because the digest size of MHT is larger than the signature chain, thus the communication overhead of 1D is higher than 1D*; the communication overhead of 2D and 2D+ are higher than 2D*.

Figure 7B shows the impact of *m* on the communication overhead from LBSP to CSP. The experiment results show that the communication overhead of 1D* is unchanged. Because the number of POIs is fixed. The communication overhead of 2D* linearly changes with *m*. Because as the number of road segments *m* increases, the total number of POIs linearly increases. Totally, the 1D* solution decreases 79% communication overhead compared with 1D, 2D* decreases 75% communication overhead compared with 2D, and 2D* decreases 81% communication overhead compared with 2D+. Because the 1D* and 1D only aim at the single road segment, thus the communication overheads do not change with *m*. The 1D's communication overhead is higher than 1D*. 2D*, 2D, and 2D+ linearly increases with number of road segments, thus 2D and 2D+ in multiple road segments is higher than 2D*.

7.3 | The communication overhead from CSP to user

The communication overhead from CSP to user is mainly the VO constructed by the CSP for the query results, which are mainly affected by the number of POIs *n* and the number of road segments *m*. Figure 8A shows the effect of *n* on the communication overhead from CSP to user. The experiment results show that the communication overhead from CSP to user increases with *n*, and the communication overhead of 1D* and 2D* change slightly with the increase of *n* because the number of skyline results does not increase significantly with the increase of *n*. When $n = 1000$, the communication overhead of 1D* is 1.3 KB, 1D* is 28.4 KB, 2D* is 28.8 KB, 2D is 578.2 KB and 2D is 137.2 KB. Totally, the communication overhead from CSP to user of 1D* is smaller than 1D, and the communication overhead of 2D* is smaller than 2D. When $n = 50$, the 1D* decreases 92% communication overhead compared with 1D*, the 2D* decreases 91% communication overhead compared with 2D, and the 2D* decreases 75% communication overhead compared with 2D+.

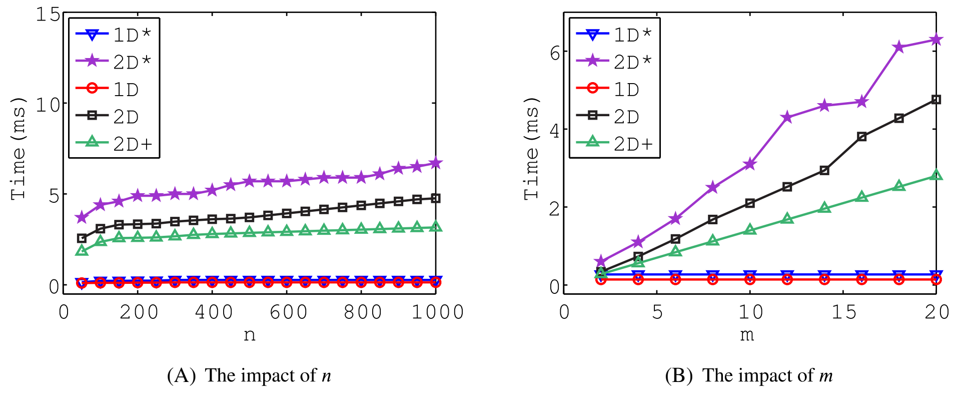


FIGURE 9 User's computation overhead

In 1D* and 2D*, the VO from CSP to user contains the query results and the corresponding signatures. In 1D and 2D, the query results contain a part of nonskyline POIs except the skyline POIs, which will result in the increase of the number of query results. In addition, the VO contains the auxiliary verification information corresponding to the query results. For each query result, the digests of the nonleaf nodes to construct the root node of the MHT are all returned, and the communication overhead is very high. Thus, in the communication overhead from CSP to user aspect, 1D's communication overhead is higher than 1D*, and the communication overhead of 2D is higher than 2D*.

Figure 8B shows the impact of *m* on the communication overhead from CSP to user. The experimental results show that the communication overhead of 1D* is unchanged, and the communication overhead of 2D* linearly changes with *m*. Totally, the 1D* decreases 94% communication overhead compared with 1D, the 2D* decreases 95% communication overhead compared with 2D solution, and the 2D* decreases 78% communication overhead compared with 2D+ solution.

7.4 | User's computation overhead

The user's computation overhead is the computation time that the user needs to authenticate the skyline results, which are mainly affected by the number of POIs *n* and the number of road segments *m*. Figure 9A shows the impact of *n* on the user computation cost. When *n* changes from 350 to 1000, the computation cost of 1D* is unchanged. When *n* changes from 500 to 1000, 1D*'s user computation cost slowly increases.

Figure 9B shows the impact of *m* on the user computation cost. The experiment results show that the 1D*'s user computation cost is unchanged, the 2D*'s computation cost increases with *m*. When $m = 20$, the computation cost of 1D* is 0.27 milliseconds, the communication cost of 2D* is 6.3 milliseconds, which are totally acceptable.

8 | CONCLUSION

LBS is likely to be increasingly popular as more things/devices are Internet- and interconnected, particularly in a smart city/nation setting. Seeking to mitigate some of the limitations in existing LBS schemes, we proposed an approach to authenticate the location-based skyline query results over road networks, in the sense that the users can detect cheating behavior of untrusted CSP, which may return incorrect or incomplete query results. Based on our observation of the unique properties of LBSQ results, we bind each POI with four neighbor POIs in a signature. The security analysis demonstrates that it is computationally hard for the CSPs to forge biased results and successfully pass the verification. The overhead analysis and experimental results suggest that our proposed solution achieves better performance in communication overhead, in comparison with the existing approach.

ACKNOWLEDGMENTS

This work is supported in part by the National Natural Science Foundation of China under Grant 61632009 and 61472451, in part by the Guangdong Provincial Natural Science Foundation under Grant 2017A030308006 and High-Level Talents Program of Higher Education in Guangdong Province under Grant 2016ZJ01, in part by Hunan Provincial Natural Science Foundation 2017JJ2333.

ORCID

Xiaoyu Zhu https://orcid.org/0000-0002-6815-3754
Kim-Kwang Raymond Choo https://orcid.org/0000-0001-9208-5336
Guojun Wang https://orcid.org/0000-0001-9875-4182

REFERENCES

1. Zhang S, Wang G, Bhuiyan MZA, Liu Q. A dual privacy preserving scheme in continuous location-based services. *IEEE Internet Things J*. 2018;5(5):4191-4200.
2. Goncalves M, Torres D, Perera G. Making recommendations using location-based skyline queries. Paper presented at: Proceedings of the 2012 23rd International Workshop on Database and Expert Systems Applications; 2012:111-115; IEEE.
3. Liu Y, Zeng Z, Liu X, Zhu X, Bhuiyan MZA. A novel load balancing and low response delay framework for edge-cloud network based on SDN. *IEEE Internet Things J*. 2019. https://doi.org/10.1109/JIOT.2019.2951857.
4. Wang N, Wu J. Optimal cloud instance acquisition via IaaS cloud brokerage with volume discount. Paper presented at: Proceedings of the 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS); 20181-10; IEEE.
5. Khandelwal Swati. Download: 68 million hacked dropbox accounts are just a click away! https://thehackernews.com/2016/10/dropbox-password-hack.html. Accessed October 4, 2016.
6. McGee Marianne Kolbasuk. Blood test results exposed in cloud repository. https://www.databreachtoday.com/blood-test-results-exposed-in-cloud-repository-a-10382. Accessed October 16, 2017.
7. Chen W, Liu M, Zhang R, Zhang Y, Liu S. Secure outsourced skyline query processing via untrusted cloud service providers. Paper presented at: Proceedings of the IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications; 2016:1-9; IEEE.
8. Devanbu P, Gertz M, Martel C, Stubblebine SG. Authentic data publication over the internet. *J Comput Secur*. 2003;11(3):291-314.
9. Li F, Hadjieleftheriou M, Kollios G, Reyzin L. Dynamic authenticated index structures for outsourced databases. Paper presented at: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data; 2006:121-132; ACM.
10. Ku WS, Hu L, Shahabi C, Wang H. Query integrity assurance of location-based services accessing outsourced spatial databases. Paper presented at: Proceedings of the International Symposium on Spatial and Temporal Databases; 2009:80-97; Berlin, Heidelberg / Germany, Springer.
11. Yiu ML, Lin Y, Mouratidis K. Efficient verification of shortest path search via authenticated hints. Paper presented at: Proceedings of the 2010 IEEE 26th International Conference on Data Engineering (ICDE 2010); 2010:237-248; IEEE.
12. Pang H, Jain A, Ramamritham K, Tan KL. Verifying completeness of relational query results in data publishing. Paper presented at: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data; 2005:407-418; ACM.
13. Narasimha M, Tsudik G. Authentication of outsourced databases using signature aggregation and chaining. Pape presented at: Proceedings of the International Conference on Database Systems for Advanced Applications; 2006:420-436; Springer.
14. Pang H, Zhang J, Mouratidis K. Scalable verification for outsourced dynamic databases. *Proc VLDB Endow*. 2009;2(1):802-813.
15. Zhu X, Wu J, Chang W, Wang G, Liu Q. Efficient authentication of multi-dimensional top-*k* queries. *IEEE Access*. 2018;7:4748-4762.
16. Yang Y, Papadopoulos S, Papadias D, Kollios G. Authenticated indexing for outsourced spatial databases. *VLDB J*. 2009;18(3):631-648.
17. Hu H, Xu J, Chen Q, Yang Z. Authenticating location-based services without compromising location privacy. Paper presented at: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data; 2012:301-312; ACM.
18. Zhu X, Wu J, Chang W, Wang G, Liu Q. Authentication of multi-dimensional top-*K* query on untrusted server. Paper presented at: Proceedings of the 2018 IEEE/ACM 26th International Symposium on Quality of Service; 2018:1-6.
19. Zhang R, Zhang Y, Zhang C. Secure top-k query processing via untrusted location-based service providers. Paper presented at: Proceedings of the 2012 IEEE INFOCOM; 2012:1170-1178; IEEE.
20. Chen Q, Hu H, Xu J. Authenticating top-k queries in location-based services with confidentiality. *Proc VLDB Endow*. 2013;7(1):49-60.
21. Zhang R, Sun J, Zhang Y, Zhang C. Secure spatial top-k query processing via untrusted location-based service providers. *IEEE Trans Depend Sec Comput*. 2014;12(1):111-124.
22. Liu Q, Tian Y, Wu J, Peng T, Wang G. Enabling verifiable and dynamic ranked search over outsourced data. *IEEE Trans Serv Comput*. 2019. https://doi.org/10.1109/TSC.2019.2922177.
23. Kumari S, Khan MK, Atiquzzaman M. User authentication schemes for wireless sensor networks: a review. *Ad Hoc Netw*. 2015;27:159-194.
24. Kumari S. Design flaws of "an anonymous two-factor authenticated key agreement scheme for session initiation protocol using elliptic curve cryptography". *Multimed Tools Appl*. 2016;76(11):1-3.
25. Kumari S, Chaudhary P, Chen C, Khan MK. Questioning key compromise attack on Ostad-Sharif et al.'s authentication and session key generation scheme for healthcare applications. *IEEE Access*. 2019;7:39717-39720.
26. Lee KCK, Lee W-C, Zheng B, Li H, Tian Y. Z-SKY: an efficient skyline query processing framework based on Z-order. *VLDB J*. 2010;19(3):333-362.
27. Liu J, Zhang H, Xiong L, Li H, Luo J. Finding probabilistic k-skyline sets on uncertain data. Paper presented at: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management; 2015:1511-1520.
28. Tao Y, Xiao X, Pei J. Efficient skyline and top-k retrieval in subspaces. *IEEE Trans Knowl Data Eng*. 2007;19(8):1072-1088.
29. Zhang S, Mao X, Choo K-KR, Peng T, Wang G. A trajectory privacy-preserving scheme based on a dual-K mechanism for continuous location-based services. *Inf Sci*. 2019. https://doi.org/10.1016/j.ins.2019.05.054.
30. Liu Q, Hou P, Wang G, Peng T, Zhang S. Intelligent route planning on large road networks with efficiency and privacy. *J Parall Distrib Comput*. 2019;133:93-106.
31. Liu Q, Wang G, Li F, Yang S, Wu J. Preserving privacy with probabilistic indistinguishability in weighted social networks. *IEEE Trans Parall Distrib Syst*. 2017;28(5):1417-1429.
32. Chang W, Wu J. Progressive or conservative: rationally allocate cooperative work in mobile social networks. *IEEE Trans Parall Distrib Syst*. 2015;26(7):2020-2035.
33. Xu Y, Ren J, Wang G, Zhang C, Yang J, Zhang Y. A blockchain-based nonrepudiation network computing service scheme for industrial IoT. *IEEE Trans Ind Inform*. 2019;15(6):3632-3641.
34. Xu Y, Ren J, Zhang Y, Zhang C, Shen B, Zhang Y. Blockchain empowered arbitrable data auditing scheme for network storage as a service. *IEEE Trans Serv Comput*. 2019. https://doi.org/10.1109/TSC.2019.2953033.
35. Liu J, Yang J, Xiong L, Pei J. Secure skyline queries on cloud platform. Paper presented at: Proceedings of the 2017 IEEE 33rd International Conference on Data Engineering (ICDE); 2017:633-644; IEEE.
36. Yang Y, Papadopoulos S, Papadias D, Kollios G. Spatial outsourcing for location-based services. Paper presented at: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering; 2008:1082-1091.

37. Lin X, Xu J, Hu H. Authentication of locationbased skyline queries. Paper presented at: Proceedings of the 20th ACM International Conference on Information and Knowledge Management; 2011:1583-1588; ACM.
38. Lin X, Xu J, Gu J. Continuous skyline queries with integrity assurance in outsourced spatial databases. Paper presented at: Proceedings of the International Conference on Web-Age Information Management; 2012:114-126; Springer.
39. Lo H, Ghinita G. Authenticating spatial skyline queries with low communication overhead. Paper presented at: Proceedings of the 3rd ACM Conference on Data and Application Security and Privacy; 2013:177-180; ACM.
40. Lin X, Xu J, Hu H, Lee W-C. Authenticating location-based skyline queries in arbitrary subspaces. *IEEE Trans Knowl Data Eng*. 2013;26(6):1479-1493.
41. Bothe S, Karras P, Vlachou A. Eskyline: processing skyline queries over encrypted data. *Proc VLDB Endow*. 2013;6(12):1338-1341.
42. Liu J, Yang J, Xiong L, Pei J. Secure and efficient skyline queries on encrypted data. *IEEE Trans Knowl Data Eng*. 2019;31(7):1397-1411.
43. Qaosar M, Alam KMR, Zaman A, et al. A framework for privacy-preserving multi-party skyline query based on homomorphic encryption. *IEEE Access*. 2019;7:167481-167496.
44. Hua J, Zhu H, Wang F, et al. CINEMA: efficient and privacy-preserving online medical primary diagnosis with skyline query. *IEEE Internet Things J*. 2019;6(2):1450-1461.
45. Zhu X, Wu J, Chang W, Wang G, Liu Q. Authentication of skyline query over road networks. Paper presented at: Proceedings of the International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage; 2018:72-83; Springer.

How to cite this article: Zhu X, Wu J, Chang W, et al. On authenticated skyline query processing over road networks. *Concurrency Computat Pract Exper*. 2020;e5747. https://doi.org/10.1002/cpe.5747