Secure Outsourced Skyline Query Processing via Untrusted Cloud Service Providers

Wenxin Chen*[1], Mengjun Liu†[1][2], Rui Zhang*, Yanchao Zhang‡, and Shubo Liu†
*University of Hawaii, Honolulu, HI, USA
† Wuhan University, Wuhan, Hubei, China
‡Arizona State University, Tempe, AZ, USA
*{wenxinc, ruizhang}@hawaii.edu, †{mjliu, liu.shubo}@whu.edu.cn, ‡yczhang@asu.edu

Abstract—**Recent years have witnessed a growing number of location-based service providers (LBSPs) outsourcing their points of interest (POI) datasets to third-party cloud service providers (CSPs), which in turn answer various data queries from mobile users on their behalf. A main challenge in such systems is that the CSPs cannot be fully trusted, which may return fake query results for various bad motives, e.g., in favor of POIs willing to pay. As an important type of queries, location-based skyline queries (LBSQ) ask for the POIs that are not spatially dominated by any other POI with respect to some query position. In this paper, we propose three novel schemes that enable efficient verification of any LBSQ result returned by an untrusted CSP by embedding and exploring a novel neighboring relationship among POIs. The efficacy and efficiency of our schemes are thoroughly analyzed and evaluated.**

I. Introduction

Recent years have witnessed the growing popularity of location-based services (LBSs) driven by the explosive growth of location-aware and Internet-capable mobile devices. eMarketer projected that global smartphone users will surpass two billion in 2016 and reach 2.56 billion in 2018. Almost all concurrent mobile devices have Internet access via the WiFi or cellular interface and are capable of acquiring their whereabout via various localization techniques. It has become increasingly common for mobile users to perform various queries at location-based service providers (LBSPs) to learn about all kinds of points of interests (POIs) such as restaurants and parks near any interested location at any time.

As an important type of queries, location-based skyline queries (LBSQs) [1]–[3] ask for the POIs that are not *spatially dominated* by any other POI with respect to a given query position. Specifically, a POI is often characterized by its location as a spatial attribute and one or more numeric attributes such as quality, price, and average rating, and we say one POI spatially dominates another if the former is both closer to the query position and preferable in the numeric attribute of interest. For example, a driver may issue an LBSQ to find all the car washes, for each of which there exists no other car wash that is both closer and cheaper.

Owing to the wide adoption of cloud computing, a growing number of LBSPs have outsourced their POI datasets to third-party cloud service providers (CSPs), which in turn answer various data queries from mobile users on their behalf. For example, Yelp, a popular LBSP that offers POI searching and crowdsourced review sharing, has outsourced its entire dataset and services to Amazon Web Services (AWS). Providing query services via third-party CSPs cannot only reduce LBSP's storage and operation costs, but also greatly enhance the elasticity and scalability of the service [4].

A major challenge for LBSPs to outsource their POI datasets and query processing is to ensure query-result integrity against possibly dishonest CSPs. In particular, CSPs cannot be fully trusted to faithfully return correct query results for various reasons. For example, a dishonest CSP may modify the LBSP's POI dataset or provide biased query results in favor of POIs willing to pay. More specifically, the CSP may return a POI record that is not in the LBSP's dataset, modify the POI record, and/or return a POI record that is not among the skyline POIs. We thus need to develop sound mechanisms to ensure the *authenticity* and *completeness* of any query result returned by a CSP. An LBSQ result is authentic if all the POIs returned are indeed authentic records in the LBSP's dataset and is complete if it contains all the skyline POIs.

To the best of our knowledge, references [5], [6] are the only two pieces of work targeting verifiable outsourced LBSQ processing via untrusted CSPs, in which Lin *et al.* presented several schemes based on a novel data structure called Merkle Skyline R-tree. Both assume that the POIs are distributed in a general 2D plane, while in practice the POIs could be better modeled as distributed over a road network. In addition, the schemes in [5], [6] only support LBSQs over the entire POI dataset, while in practice users are usually interested in the skyline POIs in selected areas that cannot be predicted in advance. These situations call for more practical solution that supports verifiable LBSQ processing over user-selected areas.

In this paper, we fill this gap by proposing three novel schemes for verifiable LBSQ processing via untrusted CSPs by embedding and exploring a novel skyline neighboring relationship among POIs. In particular, we observe that every POI can appear in the result for infinite number of LBSQs but can only have a limited number of possible neighboring POIs with each corresponding to a distinct query range. Based on this observation, we let the LBSP bind every POI record with all its skyline neighbors and corresponding query ranges using

[1] The first two authors contributed equally to this work.
[2] This work was done when M. Liu was a visiting student at the University of Hawaii.

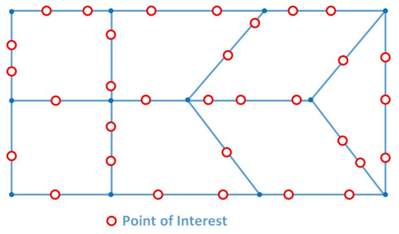


Fig. 1: An exemplary road network.

cryptographic techniques before outsourcing its dataset to the CSP. To answer an LBSQ, the CSP must return embedded skyline neighbor information for each returned POI record in order to pass the authenticity verification, based on which the user can further verify the completeness of the query result. Our first scheme supports verifiable LBSQ processing over POIs in a single road segment, and the second scheme supports multiple road segments. The third scheme further improves the second scheme by reducing its communication and computation overhead. The efficacy and efficiency of all three schemes are confirmed by detailed analysis and simulation studies.

II. Related Work

In this section, we discuss some work that is most germane to our work in addition to [5], [6] discussed in Section I.

Significant effort has been made to ensure query integrity against untrusted service providers such that a query result was indeed generated from the outsourced dataset and contains all the data satisfying the query. Various types of query have been studied, including range queries [7], [8], spatial top-k queries [9]–[11], kNN queries [12], [13], shortest-path queries [14], spatial skyline queries [15], etc. Common to existing proposals is to let the data owner outsource both its dataset and its signature on the dataset to the service provider which returns both the query result and a *verification object* (VO) computed from the signatures for the querying user to verify query integrity.

Another line of research is to ensure data privacy against untrusted service providers. A common approach is to encrypt the dataset before outsourcing it to the third-party service provider, and various techniques have been proposed to enable efficient query processing over encrypted data. Early research focuses on one-dimensional range queries [16]–[18] as well as multi-dimensional range queries [19]. More recent work targets secure ranked keyword search [20]–[24], fine-grained access control [25], and circular range query [26] over encrypted data. This line of work is orthogonal to our work, as we focus on publicly accessible data without need for privacy protection.

III. Models and Problem Formulation

A. System Model

We consider an LBSP outsourcing its POI dataset to a third-party CSP, which in turn answers LBSQs from mobile users on the LBSP's behalf. Mobile users are people carrying smartphones or tablets who may issue LBSQs by either directly visiting the LBSP's website or through installed mobile app offered by the LBSP.

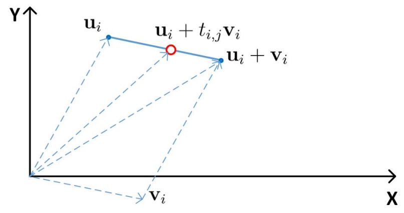

Y
$\mathbf{u}_i$
$\mathbf{u}_i + t_{i,j}\mathbf{v}_i$
$\mathbf{u}_i + \mathbf{v}_i$
$\mathbf{v}_i$
X

Fig. 2: Representation of a road segment e_i, where $t_{i,j}$ is POI $o_{i,j}$'s relative position.

We assume that the LBSP's dataset involves a set of POIs $\mathcal{O}$ of the same category, e.g., hotel, and each POI is characterized by its location and one numeric attribute (e.g., price) taking values from a known range. Our solution can be easily extended to support POIs involve multiple categories with multiple numeric attributes.

We assume that the area where the POIs reside can be modeled as a road network like lower town Manhattan and represented as a planar graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ in a 2D plane as shown in Fig. 1, where $\mathbb{V}$ is the set of vertices, i.e., road intersections, and $\mathbb{E} = \{e_1, \ldots, e_m\}$ is the set of road segments. As shown in Fig. 2, each road segment can be represented as $e_i = \{\mathbf{u}_i + t\mathbf{v}_i | t \in [0,1]\}$ for two *reference vectors* $\mathbf{u}_i, \mathbf{v}_i \in \mathbb{R}^2$, where $\mathbf{u}_i$ and $\mathbf{u}_i + \mathbf{v}_i$ are the two end points.

We now illustrate the content of the LBSP's dataset. Denote by $\mathcal{O}_i$ the set of POIs in road segment e_i. It follows that $\mathcal{O} = \bigcup_{i=1}^{m} \mathcal{O}_i$ and $\mathcal{O}_i \bigcap \mathcal{O}_j = \emptyset$ for all $i \neq j$. Assume that there are n_i POIs in road segment e_i for all $i \in [1, m]$. Also let $o_{i,j}$ and $D_{i,j}$ denote the jth POI in road segment e_i and its corresponding data record, respectively. Every POI record can be represented as

$$D_{i,j} = \langle \mathsf{id}_{i,j}, t_{i,j}, \lambda_{i,j}, \mathsf{info}_{i,j} \rangle,$$

where $\mathsf{id}_{i,j}$ is a unique ID assigned by the LBSP that identifies $o_{i,j}$, $t_{i,j} \in [0,1]$ is its relative position with respect to road segment e_i, $\lambda_{i,j} \in [\lambda_{\min}, \lambda_{\max}]$ is the numeric attribute of interest, and $\mathsf{info}_{i,j}$ includes all other information about the POI such as its name, user text review, and photo. Given the reference vectors of road segment e_i, the actual position of $o_{i,j}$ can be computed as $\mathbf{p}_{i,j} = \mathbf{u}_i + t_{i,j}\mathbf{v}_i$. For a POI at the intersection of two road segments, we assume that it belongs to only one road segment. Finally, we assume that no two POIs share the same position.

B. Location-Based Skyline Query

Assuming that lower numeric attribute (e.g., price) is preferable, we now give the definitions for spatial dominance and location-based skyline query.

Definition 1. (**Spatial dominance**) For any two POIs $o_{i,j}$ and $o_{i',j'}$, we say $o_{i,j}$ *spatially dominates* $o_{i',j'}$ with respect to query position $\mathbf{q}$ if and only if $d(\mathbf{q}, \mathbf{p}_{i,j}) \leq d(\mathbf{q}, \mathbf{p}_{i',j'})$ and $\lambda_{i,j} \leq \lambda_{i',j'}$ but the two equalities do not both hold, where $d(\cdot, \cdot)$ is some proper distance function.

Definition 2. (**Location-based skyline query**) A location-based skyline query $\mathsf{sky}(O|\mathbf{q})$ asks for the POIs that are not spatially dominated by any other POI in O with respect to $\mathbf{q}$.

We assume Euclidian distance for $d(\cdot,\cdot)$ in this paper and leave the extension to other distance metrics like shortest path distance as our future work.

C. Problem Formulation

We assume that when a user uses the CSP's service for the first time, he download a copy of $(\mathbb{V},\mathbb{E})$ and all the reference vectors and the LBSP's signature.

To issue an LBSQ, the user submits to the CSP $\langle \mathbf{q}, I\rangle$, where $\mathbf{q}\in\mathbb{R}^2$ is the query position, and $I\subseteq\{1,\dots,m\}$ is a set of indexes of road segments. The CSP is supposed to return the skyline POI set $\mathsf{sky}(O|\mathbf{q})$, where $O=\bigcup_{i\in I}\mathcal{O}_i$.

We assume that the LBSP is trusted, while the CSP is untrusted. In particular, the CSP may alter POI records, forge POI records that are not in LBSP's data set, replace some skyline POI records with fake ones that not skyline POIs, or purposely omit some true skyline POI records.

Given the above problem setting, our design objective is to enable user to verify the authenticity and completeness of the query result returned by the CSP. The query result is considered authentic if all the returned POI records exist in the LBSP's dataset and have not been tampered with, and it is called complete if it contains all the true skyline POI records.

IV. 1D-SKY: Verifiable 1D LBSQ Processing

In this section, we introduce 1D-SKY, a novel scheme for verifiable 1-dimensional LBSQ processing via a untrusted CSP, which forms the foundation of the solutions for more general 2D cases introduced in Sections V and VI.

For 1D LBSQ, we assume that the road network contains only one road segment $e=\{\mathbf{u}+t\mathbf{v}|t\in[0,1]\}$, along which n POIs are distributed. We then simplify the notation by denoting the set of POIs as $\mathcal{O}=\{o_j|1\le j\le n\}$, where o_j is the jth POI with corresponding data record $D_j=\langle \mathsf{id}_j,t_j,\lambda_j,\mathsf{info}_j\rangle$. Moreover, we assume that the user issues LBSQ $\mathsf{sky}(\mathcal{O}|\mathbf{q})$ with arbitrary query position $\mathbf{q}\in\mathbb{R}^2$.

In what follows, we first introduce two properties of LBSQ that our scheme depends on in Section IV-A. We then introduce the notions of and the procedures for computing skyline neighbor and neighbor range in Sections IV-B and IV-C, respectively. We finally detail 1D-SKY's three phases, *data preprocessing*, *query processing*, and *query-result verification* in Sections IV-D to IV-F, respectively.

A. Properties of LBSQ

We first have the following proposition regarding 1D LBSQ.

Proposition 1. Let $\mathcal{O}$ be the set of POIs distributed along road segment $e=\{\mathbf{u}+t\mathbf{v}|t\in[0,1]\}$. For any query position $\mathbf{q}\in\mathbb{R}^2$, let $p(\mathbf{q})$ denote its projection on the straight line $\{\mathbf{u}+t\mathbf{v}|t\in\mathbb{R}\}$ that contains e. We have

$$\mathsf{sky}(\mathcal{O}|\mathbf{q})=\mathsf{sky}(\mathcal{O}|p(\mathbf{q})). \tag{1}$$

We give the proof of Proposition 1 in our technical report [27]. Proposition 1 shows that $\mathsf{sky}(\mathcal{O}|\mathbf{q})$ is determined by $\mathbf{q}$'s projection $p(\mathbf{q})$. Assume that $p(\mathbf{q})=\mathbf{u}+t_q\mathbf{v}$, where t_q is $p(\mathbf{q})$'s relative position on straight line $\{\mathbf{u}+t\mathbf{v}|t\in\mathbb{R}\}$ and given by

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

Since $\mathsf{sky}(\mathcal{O}|\mathbf{q})$ is determined by $p(\mathbf{q})$, which in turn can be represented by t_q, our subsequent discussion will use $\mathsf{sky}(O|\mathbf{q})$, $\mathsf{sky}(O|p(\mathbf{q}))$ and $\mathsf{sky}(O|t_q)$ interchangeably when no confusion arises.

We also have the following proposition regarding the decomposability of LBSQ.

Proposition 2. Let $O_1,\dots,O_k$ be a family of subsets of O such that $O=\bigcup_{i=1}^k O_i$. For any query position $\mathbf{q}$, we have

$$\mathsf{sky}(O|\mathbf{q})\subseteq\bigcup_{i=1}^{k}\mathsf{sky}(O_i|\mathbf{q})\ .$$

We give the proof in our technical report [27]. Proposition 2 shows that we can decompose one LBSQ into multiple LBSQs each over a subset of POIs, and the union of the later skyline POI sets always contains the original skyline POI set.

B. Skyline Neighbor and Neighbor Range

We start by classifying LBSQs into two categories according to the relative position of the query position's projection. Specifically, let $t_{\text{min}}=\min(t_1,\dots,t_n)$ and $t_{\text{max}}=\max(t_1,\dots,t_n)$. We say an LBSQ $\mathsf{sky}(\mathcal{O}|t_q)$ is *single-sided* if $t_q\le t_{\text{min}}$ or $t_q>t_{\text{max}}$ and *double-sided* if $t_{\text{min}}<t_q\le t_{\text{max}}$.

We can always decompose a double-sided LBSQ into two single-sided ones. In particular, for any query position t_q, we can divide $\mathcal{O}$ into two subsets according to POIs' relative positions as

$$\begin{aligned}\mathcal{O}^-&=\{o_i|o_i\in\mathcal{O},t_i<t_q\},\\ \mathcal{O}^+&=\{o_i|o_i\in\mathcal{O},t_i\ge t_q\}.\end{aligned} \tag{3}$$

It is easy to see that both $\mathsf{sky}(\mathcal{O}^-|t_q)$ and $\mathsf{sky}(\mathcal{O}^+|t_q)$ are single-sided. Moreover, according to Proposition 2, we have

$$\mathsf{sky}(\mathcal{O}|t_q)\subseteq\mathsf{sky}(\mathcal{O}^-|t_q)\bigcup\mathsf{sky}(\mathcal{O}^+|t_q)\ .$$

We define skyline neighbors with respect to single-sided LBSQ.

Definition 3. (**Skyline neighbor**) Assume that $\mathsf{sky}(\mathcal{O}|t_q)$ is single-sided. For any $o_i\in\mathsf{sky}(\mathcal{O}|t_q)$, we define its *left* (or *right*) *skyline neighbor* with respect to query position t_q, denoted by $N_l(o_i|t_q)$ (or $N_r(o_i|t_q)$), as the closest POI $o_j\in\mathsf{sky}(\mathcal{O}|t_q)$ with $t_j<t_i$ (or $t_j>t_i$).

Note that o_i may have no left (or right) neighbor if $t_i=\min\{t_j|o_j\in\mathsf{sky}(\mathcal{O}|t_q)\}$ (or $t_i=\max\{t_j|o_j\in\mathsf{sky}(\mathcal{O}|t_q)\}$).

Definition 4. (**Skyline neighbor set**) We define the *left (or right) skyline neighbor set* of POI o_i as the set of its all

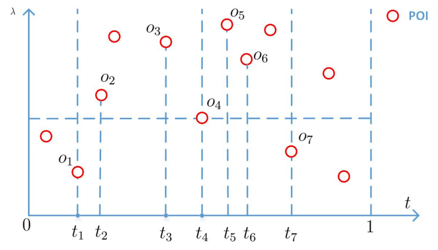


Fig. 3: An example of computing skyline neighbor set and neighbor ranges for o_4.

possible left (or right) skyline neighbors, which are given by

$$\mathcal{N}_l(o_i) = \{o_j | \exists t_q, \text{s.t. } o_i, o_j \in \mathsf{sky}(\mathcal{O}|t_q), o_j = N_l(o_i|t_q)\},$$
$$\mathcal{N}_r(o_i) = \{o_j | \exists t_q, \text{s.t. } o_i, o_j \in \mathsf{sky}(\mathcal{O}|t_q), o_j = N_r(o_i|t_q)\}.$$

Each skyline neighbor corresponds to a distinct query range, which we call a *neighbor range*.

Definition 5. (**Neighbor range**) For any $o_j \in \mathcal{N}_l(o_i)$ or $\mathcal{N}_l(o_i)$, we define the *neighbor range* of o_j with respect to o_i as $\mathsf{range}(o_i|o_j)$ such that $o_j = N_l(o_i|t_q)$ or $o_j = N_r(o_i|t_q)$ if and only if $t_q \in \mathsf{range}(o_i|o_j)$.

C. Computing Skyline Neighbor Set and Neighbor Ranges

1D-SKY requires the LBSP to compute the skyline neighbor set and the corresponding neighbor ranges for every POI.

Consider Fig. 3 as an example, where all the POIs are distributed along a road segment, and the x- and y-coordinates represent each POI's relative position and numeric attribute, respectively. Suppose that we want to compute the left and right skyline neighbor sets and corresponding neighbor ranges for POI o_4.

We first find the query range R such that $o_4 \in \mathsf{sky}(\mathcal{O}|t_q)$ (i.e., o_4 has some skyline neighbor) if and only if $t_q \in R$. For this purpose, we can see from Fig. 3 that o_1 and o_7 are the two closest POIs with numeric attributes lower than λ_4 on o_4's left and right sides, respectively. It follows that o_1 dominates o_4 if $t_q \le t_1$ and that o_7 dominates o if $t_q > t_7$. In addition, we have $t_q \in (t_1, t_4]$, $o_4 \in \mathsf{sky}(\mathcal{O}^+|t_q)$, and when $t_q \in (t_4, t_7)$, $o_4 \in \mathsf{sky}(\mathcal{O}^-|t_q)$. We therefore have $R = (t_1, t_7)$.

Now we consider the case where $t_1 < t_q < t_7$. We define two POI sets as

$$\mathcal{O}_l^+ = \{o_j | o_j \in \mathcal{O}, t_1 < t_j < t_4, \lambda_j > \lambda_4\},$$
$$\mathcal{O}_r^+ = \{o_j | o_j \in \mathcal{O}, t_4 < t_j < t_7, \lambda_j > \lambda_4\}.$$

If we issue two single-sided LBSQs over $\mathcal{O}_l^+$ and $\mathcal{O}_r^+$ with query position $t_q = t_4$, we get $\mathsf{sky}(\mathcal{O}_l^+|t_4) = \{o_2, o_3\}$ and $\mathsf{sky}(\mathcal{O}_r^+|t_4) = \{o_5, o_6\}$.

We claim that o_4's left and right skyline neighbor sets are $\mathcal{N}_l(o_4) = \{o_1\} \bigcup \mathsf{sky}(\mathcal{O}_l^+|t_4) = \{o_1, o_2, o_3\}$ and $\mathcal{N}_r(o_4) = \{o_7\} \bigcup \mathsf{sky}(\mathcal{O}_r^+|t_4) = \{o_5, o_6, o_7\}$, respectively. We detail each case as follow.

- **Case 1**: If $t_q \in (t_1, t_2]$, then $N_l(o_4|t_q) = o_2$ and $N_r(o_4|t_q) = o_7$.
- **Case 2**: If $t_q \in (t_2, t_3]$, then $N_l(o_4|t_q) = o_3$ and $N_r(o_4|t_q) = o_7$.
- **Case 3**: If $t_q \in (t_3, t_4]$, then o_3 and o_4 are the rightmost and the leftmost POIs in $\mathsf{sky}(\mathcal{O}^-|t_q)$ and $\mathsf{sky}(\mathcal{O}^+|t_q)$, respectively. We postulate $N_l(o_4|t_q) = o_3$ in this case. In addition, we have $N_r(o_4|t_q) = o_7$.
- **Case 4**: if $t_q \in (t_4, t_5]$, then o_4 and o_5 are the rightmost and the leftmost POIs in $\mathsf{sky}(\mathcal{O}^-|t_q)$ and $\mathsf{sky}(\mathcal{O}^+|t_q)$, respectively. We have $N_l(o_4|t_q) = o_1$ and postulate that $N_r(o_4|t_q) = o_5$ in this case.
- **Case 5**: If $t_q \in (t_5, t_6]$, then $N_l(o_4|t_q) = o_1$ and $N_r(o_4|t_q) = o_5$.
- **Case 6**: If $t_q \in (t_6, t_7]$, then $N_l(o_4|t_q) = o_1$ and $N_r(o_4|t_q) = o_6$.

Summarizing the above cases, we have $\mathcal{N}_l(o_4) = \{o_1, o_2, o_3\}$ with neighbor ranges $\mathsf{range}(o_1|o_4) = (t_4, t_7)$, $\mathsf{range}(o_2|o_4) = (t_1, t_2]$, and $\mathsf{range}(o_3|o_4) = (t_2, t_4)$, and $\mathcal{N}_r(o_4) = \{o_5, o_6, o_7\}$ with neighbor ranges $\mathsf{range}(o_5|o_4) = (t_4, t_6)$, $\mathsf{range}(o_6|o_4) = (t_6, t_7)$, and $\mathsf{range}(o_7|o_4) = (t_1, t_4)$.

In general, for any POI $o_i \in \mathcal{O}$, the query range within which $o_i \in \mathsf{sky}(\mathcal{O}^+|t_q) \bigcup \mathsf{sky}(\mathcal{O}^-|t_q)$ is the range between its two closest POIs with numeric attributes lower than λ_i, one on each side both, say $o_{i,l}$ and $o_{i,r}$. Let $\mathcal{O}_{i,l}^+$ and $\mathcal{O}_{i,r}^+$ be the sets of POIs between $t_{i,l}$ and t_i and between t_i and $t_{i,r}$ all with numeric attribute higher than λ_i, respectively. The $\mathcal{N}_l(o_i)$ comprises $o_{i,l}$ and $\mathsf{sky}(\mathcal{O}_{i,l}^-|t_i)$ and $\mathcal{N}_r(o_i)$ comprises $o_{i,r}$ and $\mathsf{sky}(\mathcal{O}_{i,r}^+|t_i)$.

We summarize the above procedure in Alg. 1, which simultaneously computes $\mathcal{P}_l = \{\langle \mathsf{id}_j, \mathsf{range}(o_j|o_i)\rangle | o_j \in \mathcal{N}_l(o_i)\}$ and $\mathcal{P}_r = \{\langle \mathsf{id}_j, \mathsf{range}(o_j|o_i)\rangle | o_j \in \mathcal{N}_r(o_i)\}$. Note that Alg. 1 assumes that there is at least one POI with numeric attribute lower than λ_i at each side of o_i, an assumption that will always hold in our cases.

D. Data Preprocessing

The LBSP preprocesses its POI dataset $\{D_i\}_{i=1}^n$ before outsourcing it to the CSP, where $D_i = \langle \mathsf{id}_i, t_i, \lambda_i, \mathsf{info}_i\rangle$. Without loss of generality, we assume that $t_1 < t_2 < \cdots < t_{n+1}$.

First, the LBSP inserts two special POI records $D_0 = \langle \mathsf{id}_0, t_0, \lambda^*\rangle$ and $D_{n+1} = \langle \mathsf{id}_{n+1}, t_{n+1}, \lambda^*\rangle$, where id_0 and id_{n+1} are two unique IDs assigned by LBSP to identify the two special records, $t_0 = -\infty$, $t_{n+1} = \infty$, and $\lambda^* < \lambda_{\min}$ is a special value publicly known to the user. Since the distances between any possible query position and D_0 and D_{n+1} are both infinity, neither of them will spatially dominate any real POI in LBSP's dataset. In addition, since $\lambda^* < \lambda_{\min}$, D_0 and D_{n+1} are not spatially dominated by any other POI and should always be returned. Therefore, inserting these two special POI records will not affect any other POI in the query result.

Second, for every POI record $D_i, i \in [0, n+1]$, the LBSP does the following.

- Compute $\{\langle \mathsf{id}_j, \mathsf{range}(o_j|o_i)\rangle | o_j \in \mathcal{N}_r(o_i)\}$ using Alg. 1.
- Compute an extended POI record as

$$E_i = \langle \mathsf{id}_i, t_i, \lambda_i, \mathsf{nbinfo}_i^{\mathsf{r}}, \mathsf{info}_i\rangle\ ,$$

where $\mathsf{nbinfo}_i^{\mathsf{r}} = \{\langle \mathsf{id}_j, \mathsf{range}(o_j|o_i)\rangle | o_j \in \mathcal{N}_r(o_i)\}$.

Algorithm 1: Computing skyline neighbor sets and neighbor ranges

```
input : POI set $\mathcal{O}$ and $o_i$
output: $\{\langle \mathsf{id}_j, \mathsf{range}(o_j|o_i)\rangle | o_j \in \mathcal{N}_l(o_i)\}$ and
        $\{\langle \mathsf{id}_j, \mathsf{range}(o_j|o_i)\rangle | o_j \in \mathcal{N}_r(o_i)\}$
$\mathcal{P}_l \longleftarrow \emptyset, \mathcal{P}_r \longleftarrow \emptyset, \mathcal{N}_l(o_i) \longleftarrow \emptyset, \mathcal{N}_r(o_i) \longleftarrow \emptyset$;
$\mathcal{O}^-_{i,l} \longleftarrow \{o_j | o_j \in \mathcal{O}, t_j < t_i, \lambda_j < \lambda_i\}$;
$\mathcal{O}^-_{i,r} \longleftarrow \{o_j | o_j \in \mathcal{O}, t_j > t_i, \lambda_j < \lambda_i\}$;
$o_{l_0} \longleftarrow$ the rightmost POI in $\mathcal{O}^-_{i,l}$, $o_{r_0} \longleftarrow$ the leftmost POI in $\mathcal{O}^-_{i,r}$;
$\mathcal{O}^+_{i,l} \longleftarrow \{o_j | o_j \in \mathcal{O}, t_{l_0} < t_j < t_i, \lambda_j > \lambda_i\}$;
$\mathcal{O}^+_{i,r} \longleftarrow \{o_j | o_j \in \mathcal{O}, t_i < t_j < t_{r_0}, \lambda_j > \lambda_i\}$;
Compute $\mathsf{sky}(\mathcal{O}^+_{i,l}|t_i) = \langle o_{l_1}, o_{l_2}, \ldots, o_{l_\alpha}\rangle$ where $t_{l_1} < \cdots < t_{l_\alpha}$;
Compute $\mathsf{sky}(\mathcal{O}^+_{i,r}|t_i) = \langle o_{r_1}, o_{r_2}, \ldots, o_{r_\beta}\rangle$ where $t_{r_1} > \cdots > t_{l_\beta}$;
$\mathsf{range}(o_{l_0}|o_i) \longleftarrow (t_i, t_{r_0})$;
$\mathcal{P}_l \longleftarrow \mathcal{P}_l \bigcup \langle \mathsf{id}_{l_0}, \mathsf{range}(o_{l_0}|o)\rangle$;
foreach $x \in \{1, \ldots, \alpha\}$ do
    if $x \neq \alpha$ then
        $\mathsf{range}(o_{l_x}|o_i) \longleftarrow (t_{l_{x-1}}, t_{l_x})$;
    else
        $\mathsf{range}(o_{l_x}|o_i) \longleftarrow (t_{l_{x-1}}, t_i)$;
    $\mathcal{P}_l \longleftarrow \mathcal{P}_l \bigcup \langle \mathsf{id}_{l_x}, \mathsf{range}(o_{l_x}|o)\rangle$;
$\mathsf{range}(o_{r_0}|o_i) \longleftarrow (t_{l_0}, t_i)$;
$\mathcal{P}_r \longleftarrow \mathcal{P}_r \bigcup \langle \mathsf{id}_{r_0}, \mathsf{range}(o_{r_0}|o_i)\rangle$;
foreach $x \in \{1, \ldots, \beta\}$ do
    if $x \neq \beta$ then
        $\mathsf{range}(o_{r_x}|o_i) \longleftarrow (t_{r_x}, t_{r_{x-1}})$;
    else
        $\mathsf{range}(o_{r_x}|o_i) \longleftarrow (t_i, t_{r_{x-1}})$;
    $\mathcal{P}_r \longleftarrow \mathcal{P}_r \bigcup \langle \mathsf{id}_{r_x}, \mathsf{range}(o_{r_x}|o_i)\rangle$;
return $\mathcal{P}_l$ and $\mathcal{P}_r$;
```

- Compute an condensed POI record as

$$C_i = \langle \mathsf{id}_i, t_i, \lambda_i, \mathsf{nbinfo}^{\mathsf{r}}_i, H(E_i)\rangle$$

where $H(\cdot)$ denotes a cryptographic hash function.

Third, the LBSP builds a Merkle hash tree over $\{C_i\}_{i=0}^{n+1}$ to enable efficient authentication of the query result. Specifically, assuming that $n+2 = 2^d$ for some integer d, the LBSP builds a binary tree of depth d, in which every leaf node corresponds to one of $\{C_i\}_{i=0}^{n+1}$, and every non-leaf node is computed as the hash of the concatenation of its immediate two children nodes. We also define an *auxiliary set* $\mathcal{T}_i$ as the set of non-leaf nodes required along with any leaf node C_i to compute the Merkle root hash. Note that if $n+2$ is not a power of two, some dummy leaf nodes need be introduced for constructing the Merkle hash tree.

Finally, the LBSP signs the root of the Merkle hash tree and sends the extended POI records $\{E_i\}_{i=0}^{n+1}$ and its signature on the Merkle root hash to the CSP, which in turn computes $\{C_i\}_{i=0}^{n+1}$ and all the intermediate results for constructing the Merkle hash tree.

E. Query Processing

Assume that the user issues an LBSQ $\mathsf{sky}(\mathcal{O}|\mathbf{q})$. The CSP constructs the query result as follow.

- Compute t_q from $\mathbf{q}$ as in Eq. (2).
- Divide $\mathcal{O}$ into two subsets $\mathcal{O}^-$ and $\mathcal{O}^+$ as Eq. (3).
- Compute $\mathsf{sky}(\mathcal{O}|t_q)$, $\mathsf{sky}(\mathcal{O}^-|t_q)$ and $\mathsf{sky}(\mathcal{O}^+|t_q)$ using existing LBSQ processing algorithm such as [3].
- For each $o_i \in \mathsf{sky}(\mathcal{O}^-|t_q) \bigcup \mathsf{sky}(\mathcal{O}^+|t_q)$, the CSP returns the following information as part of the query result.
  - If $o_i \in \mathsf{sky}(\mathcal{O}|t_q)$, then CSP returns E_i.
  - If $o_i \notin \mathsf{sky}(\mathcal{O}|t_q)$, then CSP returns C_i.

In addition, the CSP returns $\bigcup_{o_i \in \mathsf{sky}(\mathcal{O}^-|t_q) \bigcup \mathsf{sky}(\mathcal{O}^+|t_q)} \mathcal{T}_i$, and the LBSP's signature on the Merkle root hash.

F. Query-Result Verification

On receiving the query result from the CSP, the user verifies its authenticity and completeness using the embedded neighbor information. Without loss of generality, assume that the CSP has returned extended or condensed POI records for u POIs $o_{j_1}, \ldots, o_{j_u}$, where $t_{j_1} < t_{j_2} < \cdots < t_{j_u}$.

For authenticity verification, the user first computes h_{j_x} from either E_{j_x} or C_{j_x} for each $x \in [1, u]$. Since the auxiliary set $\mathcal{T}_{j_x}$ for h_{j_x} is also in the query result, the user further uses h_{j_x} and $\mathcal{T}_{j_x}$ to compute the Merkle root hash. If the query result is authentic, the user can derive the same root hash for every o_{j_x}. The user can further verify whether the LBSP's signature is a valid signature on the derived root hash. If both verifications succeed, the query result is considered authentic.

The user proceeds to check the completeness of the query result in the following three steps.

First, he checks whether o_{j_1} and o_{j_u} are two special POIs inserted by the LBSP by verifying whether $t_{j_0} = -\infty$ and $t_{j_u} = \infty$, as these two records should always be returned.

Second, he further checks whether every pair of adjacent POIs in $o_{j_1}, \ldots, o_{j_u}$ are indeed skyline neighbors of each other with respect to query position t_q using the embedded neighbor information. Specifically, for every $o_{j_x}, x \in [1, u-1]$, the user checks $\mathsf{nbinfo}^{\mathsf{r}}_{j_x}$ to see whether $\mathsf{id}_{j_{x+1}} \in \mathcal{N}_r(o_{j_x})$ and $t_q \in \mathsf{range}(o_{j_{x+1}}|o_{j_x})$. If any POI does not pass the verification, the query result is considered incomplete.

Third, he checks whether the POIs with extended records returned are indeed the skyline POIs.

- For every POI with extended record returned, check if it is dominated by some other returned POI. If so, the query result is considered incomplete.
- For every POI with condensed record returned, check if it is indeed dominated by some POI with extended record returned. If not, the query result is considered incomplete.

If all the verifications succeed, the user considers the query result as complete and incomplete otherwise.

V. 2D-SKY: Verifiable 2D LBSQ Processing

In this section, we present 2D-SKY for 2D LBSQ processing where the road network consists of $m > 1$ road segments.

2D-SKY is built upon 1D-SKY and Proposition 2. Under 2D-SKY, the LBSP preprocesses the POI dataset for each road segment independently as in 1D-SKY. On receiving an LBSQ $\langle I, \mathbf{q}\rangle$ asking for $\mathsf{sky}(\bigcup_{i \in I} \mathcal{O}_i|\mathbf{q})$ from the user, the CSP returns local skyline POIs for each road segment e_i ($i \in I$) independently as in 1D-SKY. The user can then verify the authenticity and the completeness of each local skyline POI set as in 1D-SKY. According to Proposition 2, the union of all local skyline POI sets must contain the global skyline POI set $\mathsf{sky}(\bigcup_{i\in I}\mathcal{O}_i|\mathbf{q})$.

A. Data Preprocessing

Assume that the LBSP has a dataset $\{\mathcal{D}_i\}_{i=1}^{m}$, where $\mathcal{D}_i = \{D_{i,j}\}_{i=1}^{n_i}$ and $D_{i,j} = \langle \mathsf{id}_{i,j}, t_{i,j}, \lambda_{i,j}, \mathsf{info}_{i,j}\rangle$. The LBSP preprocesses its POI dataset before outsourcing it to the CSP.

First, the LBSP processes each $\mathcal{D}_i (i \in [1,m])$ as in 1D-SKY. Specifically, the LBSP inserts two special POI records $D_{i,0}$ and D_{i,n_i+1} and for every POI record $D_{i,j}, j \in [0, n_i + 1]$, computes its right neighbor set and corresponding neighbor ranges $\{\langle \mathsf{id}_{i,x}, \mathsf{range}(o_{i,x}|o_{i,j})\rangle | o_{i,x} \in \mathcal{N}_r(o_{i,j})\}$ (using Alg. 1), an extended POI record $E_{i,j} = \langle \mathsf{id}_{i,j}, t_{i,j}, \lambda_{i,j}, \mathsf{nbinfo}^{\mathsf{r}}_{i,j}, \mathsf{info}_{i,j}\rangle$, and a condensed POI record $C_{i,j} = \langle \mathsf{id}_{i,j}, t_{i,j}, \lambda_{i,j}, \mathsf{nbinfo}^{\mathsf{r}}_{i,j}, H(E_{i,j})\rangle$.

Second, the LBSP then builds a Merkle hash tree over all the condensed POI records $\{C_{i,j} | 1 \le i \le m, 1 \le j \le n_i\}$ and signs the root.

Finally, the LBSP sends all the extended POI records $\{E_{i,j} | 1 \le i \le m, 0 \le j \le n_i + 1\}$ and its signature on Merkle hash tree root to the CSP, which in turn computes $\{C_{i,j} | 1 \le i \le m, 0 \le j \le n_i + 1\}$ as well as all the intermediate results for constructing the Merkle hash tree.

B. Query Processing

On receiving an LBSQ $\langle I, \mathbf{q}\rangle$ asking for $\mathsf{sky}(\bigcup_{i\in I}\mathcal{O}_i|\mathbf{q})$ from the user, the CSP constructs the query result as follow.

- Compute the global skyline POI set $\mathsf{sky}(O|\mathbf{q})$ using existing LBSQ processing algorithm such as [3].
- For every road segment e_i ($i \in I$), the CSP does the following.
  - Compute $t_{q,i}$, the relative position of $\mathbf{q}$'s projection on straight line $l_i = \{\mathbf{u}_i + t\mathbf{v}_i | t \in \mathbb{R}\}$ as in Eq. (2).
  - Divide $\mathcal{O}_i$ into $\mathcal{O}_i^- = \{o_{i,j} | o_{i,j} \in \mathcal{O}_i, t_{i,j} < t_{q,i}\}$ and $\mathcal{O}_i^+ = \{o_{i,j} | o_{i,j} \in \mathcal{O}_i, t_{i,j} \ge t_{q,i}\}$.
  - Compute $\mathsf{sky}(\mathcal{O}_i^-|t_{q,i})$ and $\mathsf{sky}(\mathcal{O}_i^+|t_{q,i})$ using existing LBSQ processing algorithm such as [3].
  - For each $o_{i,j} \in \mathsf{sky}(\mathcal{O}^-|t_q) \bigcup \mathsf{sky}(\mathcal{O}^+|t_q)$, the CSP returns $E_{i,j}$ if $o_{i,j} \in \mathsf{sky}(\mathcal{O}|\mathbf{q})$ and $C_{i,j}$ otherwise.

In addition, the CSP returns $\bigcup_{o_{i,j}\in\mathcal{U}} \mathcal{T}_{i,j}$, where $\mathcal{U} = \bigcup_{i\in I}(\mathsf{sky}(\mathcal{O}_i^-|t_{q,i}) \bigcup \mathsf{sky}(\mathcal{O}_i^+|t_{q,i}))$ and the LBSP's signature on the Merkle root hash.

C. Query-Result Verification

Assume that the CSP has returned the result in response to LBSQ $\langle I, \mathbf{q}\rangle$ asking for $\mathsf{sky}(\bigcup_{i\in I}\mathcal{O}_i|\mathbf{q})$. The user first verifies that all the returned extended or condensed POI records are authentic using returned auxiliary sets and the LBSP's signature as in 1D-SKY.

If the query result is authentic, the user proceeds to verify the completeness of the query result. Without loss of generality, assume that for each road segment $e_i, i \in I$, the CSP has returned extended or condensed POI record for $u[i]$ POIs $\mathcal{R}_i = \{o_{i,j_1}, \ldots, o_{i,j_{u[i]}}\}$. The user first verifies whether $\mathcal{R}_i = \mathsf{sky}(\mathcal{O}_i^+|\mathbf{q}) \bigcup \mathsf{sky}(\mathcal{O}_i^-|\mathbf{q})$ following the first two steps of completeness verification in 1D-SKY. If the POI records of every road segment pass the verification, the user knows that $\mathsf{sky}(\bigcup_{i\in I}\mathcal{O}_i|\mathbf{q}) \subseteq \bigcup_{i\in I}\mathcal{R}_i$ according to Proposition 2. The user further checks whether the POIs with extended records returned are indeed the skyline POIs as in the third step of completeness verification in 1D-SKY. If all the verifications succeed, the user considers the query result as complete and incomplete otherwise.

VI. 2D-SKY$^+$: An Advanced Scheme

2D-SKY allows the user to verify the authenticity and completeness of any LBSQ result returned by the CSP but incurs relatively high computation and communication overhead if the number of road segments queried is large, as most of the returned POIs are local instead of global skyline POIs. In this section, we further introduce 2D-SKY$^+$ to significantly reduce the number of condensed POI records that need be returned.

We observe that every local skyline POI that is not a global skyline POI must be spatially dominated by some global skyline POI, and these local skyline POIs could be potentially omitted to reduce the size of the query result. The challenge is how to verify that the LBSP has not omitted any global skyline POI. To tackle this challenge, we further observe that if a global skyline POI spatially dominates some POIs among $\mathsf{sky}(\mathcal{O}_i^+|\mathbf{q})$ or $\mathsf{sky}(\mathcal{O}_i^-|\mathbf{q})$ for some road segment e_i, then the spatially dominated POIs must be consecutive. Consider $\mathsf{sky}(\mathcal{O}_i^+|\mathbf{q})$ as an example. Without loss of generality, assume that $\mathsf{sky}(\mathcal{O}_i^+|\mathbf{q}) = \{o_{i,j_1}, \ldots, o_{i,j_u}\}$, where $t_{i,j_1} < \cdots < t_{i,j_u}$. It follows that $\lambda_{i,j_1} > \cdots > \lambda_{i,j_u}$. Now suppose that global skyline POI $o_{r,s}$ spatially dominates $\{o_{i,j_\gamma}, \ldots, o_{i,j_\delta}\}$, where γ and δ can be computed as

$$\begin{aligned}\gamma &= \min\{x | 1 \le x \le u, d(\mathbf{q}, \mathbf{p}_{r,s}) < d(\mathbf{q}, \mathbf{p}_{i,j_x})\},\\ \delta &= \max\{x | 1 \le x \le u, \lambda_{r,s} < \lambda_{i,j_x}\}\ .\end{aligned} \tag{4}$$

2D-SKY$^+$ lets the CSP to replace $\{o_{i,j_\gamma}, \ldots, o_{i,j_\delta}\}$ with $\mathsf{id}_{r,s}$ in the query result, whereby the user can verify whether the CSP has omitted any global skyline POI by checking whether $o_{r,s}$ spatially dominates o_{i,j_γ} and o_{i,j_δ} using embedded neighbor information.

A. Data Preprocessing

The data precessing phase of 2D-SKY$^+$ is very similar to that of of 2D-SKY, where the only difference is that the LBSP need bind both the left and right neighbor sets with corresponding neighbor ranges to every POI record. In other words, we have $E_{i,j} = \langle \mathsf{id}_{i,j}, t_{i,j}, \lambda_{i,j}, \mathsf{nbinfo}^{\mathsf{l}}_{i,j}, \mathsf{nbinfo}^{\mathsf{r}}_{i,j}, \mathsf{info}_{i,j}\rangle$ and $C_{i,j} = \langle \mathsf{id}_{i,j}, t_{i,j}, \lambda_{i,j}, \mathsf{nbinfo}^{\mathsf{l}}_{i,j}, \mathsf{nbinfo}^{\mathsf{r}}_{i,j}, H(E_{i,j})\rangle$ for all $1 \le i \le m$ and $1 \le j \le n_i$ under 2D-SKY$^+$.

B. Query Processing

On receiving an LBSQ $\langle I, \mathbf{q}\rangle$ asking for $\mathsf{sky}(O|\mathbf{q})$ from the user, where $O = \bigcup_{i\in I}\mathcal{O}_i$, the CSP constructs the query result as follow.

First, the CSP computes the global skyline POI set $\mathsf{sky}(O|\mathbf{q})$ as in 2D-SKY.

Second, the CSP constructs a partial query result for each road segment by replacing some condensed records of dominated local skyline POIs with the IDs of corresponding dominating global POIs.

Consider road segment e_i as an example. The CSP firsts computes $\mathsf{sky}(\mathcal{O}_i^-|t_{q,i})$ and $\mathsf{sky}(\mathcal{O}_i^+|t_{q,i})$, where $\mathcal{O}_i^- = \{o_{i,j}|o_{i,j} \in \mathcal{O}_i, t_{i,j} < t_{q,i}\}$ and $\mathcal{O}_i^+ = \{o_{i,j}|o_{i,j} \in \mathcal{O}_i, t_{i,j} \geq t_{q,i}\}$, and $t_{q,i}$ is the relative position of $\mathbf{q}$'s projection. Assume that $\mathsf{sky}(\mathcal{O}_i^-|t_{q,i}) = \{o_{i,j_1}, \dots, o_{i,j_{v[i]}}\}$ and that $\mathsf{sky}(\mathcal{O}_i^+|t_{q,i}) = \{o_{i,j_{v[i]+1}}, \dots, o_{i,j_{u[i]}}\}$, where $t_{i,j_1} < \cdots < t_{i,j_{u[i]}}$. The CSP then finds two POIs $o_{r,s}, o_{r',s'} \in \mathsf{sky}(O|\mathbf{q})$ that spatially dominate the most POIs among $o_{i,j_1}, \dots, o_{i,j_{v[i]}}$ and $o_{i,j_{v[i]+1}}, \dots, o_{i,j_{u[i]}}$, respectively. Without loss of generality, assume that $o_{r,s}$ spatially dominates $o_{i,j_{\alpha[i]}}, \dots, o_{i,j_{\beta[i]}}$ and that $o_{r',s'}$ spatially dominates $o_{i,j_{\gamma[i]}}, \dots, o_{i,j_{\delta[i]}}$. We can compute $\alpha[i], \beta[i], \gamma[i]$, and $\delta[i]$ as

$$\begin{aligned}
&\alpha[i] = \min\{x | 1 \leq x \leq v[i], \lambda_{i,j_x} > \lambda_{r,s}\},\\
&\beta[i] = \max\{x | 1 \leq x \leq v[i], d(\mathbf{q}, \mathbf{p}_{i,j_x}) > d(\mathbf{q}, \mathbf{p}_{r,s})\},\\
&\gamma[i] = \min\{x | v[i] < x \leq u[i], d(\mathbf{q}, \mathbf{p}_{i,j_x}) > d(\mathbf{q}, \mathbf{p}_{r',s'})\},\\
&\delta[i] = \max\{x | v[i] < x \leq u[i], \lambda_{i,j_x} > \lambda_{r',s'}\}.
\end{aligned} \tag{5}$$

The CSP then returns a partial query result for road segment e_i as $\mathcal{R}_i = \langle \mathcal{R}_i^-, \mathcal{R}_i^+ \rangle$, where

$$\begin{aligned}
\mathcal{R}_i^- &= \langle X_{i,j_1}, \dots, X_{i,j_{\alpha[i]-1}}, \mathsf{id}_{r,s}, X_{i,j_{\beta[i]+1}}, \dots, X_{i,j_{v[i]}} \rangle,\\
\mathcal{R}_i^+ &= \langle X_{i,j_{v[i]+1}}, \dots, X_{i,j_{\gamma[i]-1}}, \mathsf{id}_{r',s'}, X_{i,j_{\delta[i]+1}}, \dots, X_{i,j_{u[i]}} \rangle,
\end{aligned} \tag{6}$$

where

$$X_{i,j_x} = \begin{cases} E_{i,j_x} & \text{if } o_{i,j_x} \in \mathsf{sky}(O|\mathbf{q}), \\ C_{i,j_x} & \text{if } o_{i,j_x} \notin \mathsf{sky}(O|\mathbf{q}). \end{cases} \tag{7}$$

Special care is needed if $\beta[i] = v[i]$ and $\gamma[i] = v[i] + 1$. In this case, we require the CSP to additionally return $C_{i,j_{v[i]}}$ to facilitate subsequent completeness verification. The partial query result returned for road segment e_i is then

$$\begin{aligned}
\mathcal{R}_i = \langle &X_{i,j_1}, \dots, X_{i,j_{\alpha[i]-1}}, \mathsf{id}_{r,s}, C_{i,j_{v[i]}},\\
&\mathsf{id}_{r',s'}, X_{i,j_{\delta[i]+1}}, \dots, X_{i,j_{u[i]}} \rangle,
\end{aligned}$$

where X_{i,j_x} is given in Eq. (7). Without returning $C_{i,j_{v[i]}}$, user cannot verify whether $o_{i,j_{\alpha[i]}}, \dots, o_{i,j_{\delta[i]}}$ are indeed all spatially dominated by $\mathsf{id}_{r,s}$ or $\mathsf{id}_{r',s'}$.

Finally, the CSP returns $\bigcup_{o_{i,j} \in \mathcal{U}} \mathcal{T}_{i,j}$, where $\mathcal{U} = \bigcup_{i \in I} \mathsf{sky}(\mathcal{O}_i^-|t_{q,i}) \bigcup \mathsf{sky}(\mathcal{O}_i^+|t_{q,i})$ and the LBSP's signature on the Merkle root hash.

C. Query-Result Verification

Assume that the CSP has returned a query result in response to an LBSQ $\mathsf{sky}(O|\mathbf{q})$, where $O = \bigcup_{i \in I} \mathcal{O}_i$. The user first verifies that all the returned extended or condensed POI records are authentic as in 2D-SKY.

If the query result is authentic, the user proceeds to verify its completeness. For each road segment $e_i, i \in I$, the partial query result $\mathcal{R}_i$ returned by the CSP must follow one of the following two formats.

Case 1. The CSP has returned

$$\begin{aligned}
\mathcal{R}_i = \langle &X'_{i,j_1}, \dots, X'_{i,x[i]}, \mathsf{id}_{a,b}, X'_{i,x[i]+1}, \dots,\\
&X'_{i,y[i]}, \mathsf{id}_{a',b'}, X'_{i,y[i]+1}, \dots, X'_{i,j_{u[i]}} \rangle.
\end{aligned}$$

In this case, the user checks $\mathcal{R}_i$ as follow.

- Check whether the LBSP has returned both extended records $E_{a,b}$ and $E_{a',b'}$, i.e., $o_{a,b}$ and $o_{a',b'}$ are both global skyline POIs.
- Check whether $o'_{i,1}$ and $o'_{i,u[i]}$ are two special POIs inserted by LBSP by verifying whether $t'_{i,1} = -\infty$ and $t'_{i,u[i]} = \infty$.
- Check whether every pair of adjacent POIs in $\langle o'_{i,1}, \dots, o'_{i,x[i]} \rangle$, $\langle o'_{i,x[i]+1}, \dots, o'_{i,y[i]} \rangle$, and $\langle o'_{i,y[i]+1}, \dots, o'_{i,u[i]} \rangle$ are indeed skyline neighbors of each other with respect to query position $t_{q,i}$. Specifically, for every $o'_{i,w}, w \in \{1, \dots, x[i] - 1\} \bigcup \{x[i]+1, \dots, y[i]-1\} \bigcup \{y[i]+1, \dots, u[i]-1\}$, the user checks $\mathsf{nbinfo}'^{\mathsf{r}}_{i,w}$ to see whether $\mathsf{id}'_{i,w+1} \in \mathcal{N}_r(o'_{i,w})$ and $t_{q,i} \in \mathsf{range}(o'_{i,w+1}|o'_{i,w})$.
- Check whether $o_{a,b}$ spatially dominates $N_r(o'_{i,x[i]}|t_q)$ and $N_l(o'_{i,x[i]+1}|t_q)$ according to $\mathsf{nbinfo}'^{\mathsf{r}}_{i,x[i]}$ and $\mathsf{nbinfo}'^{\mathsf{l}}_{i,x[i]+1}$, respectively, and whether $o_{a',b'}$ spatially dominates $N_r(o'_{i,y[i]}|t_q)$ and $N_l(o'_{i,y[i]+1}|t_q)$ according to $\mathsf{nbinfo}'^{\mathsf{r}}_{i,z[i]}$ and $\mathsf{nbinfo}'^{\mathsf{l}}_{i,z[i]+1}$, respectively.

Case 2: the CSP has returned

$$\begin{aligned}
\mathcal{R}_i = \langle &X'_{i,1}, \dots, X'_{i,x[i]}, \mathsf{id}_{a,b}, C_{i,x[i]+1},\\
&\mathsf{id}_{a',b'}, X'_{i,x[i]+2}, \dots, X'_{i,u[i]} \rangle.
\end{aligned}$$

In this case, the user checks $\mathcal{R}_i$ as follow.

- Perform the first two checks as in Case 1.
- Check whether every pair of adjacent POIs in $\langle o'_{i,1}, \dots, o'_{i,x[i]} \rangle$ and $\langle o'_{i,x[i]+2}, \dots, o'_{i,u[i]} \rangle$ are indeed skyline neighbors of each other with respect to query position $t_{q,i}$ using the embedded neighbor information.
- Check whether $o_{a,b}$ spatially dominates $N_r(o'_{i,x[i]}|t_q)$ and $o'_{i,x[i]+1}$ and whether $o_{a',b'}$ spatially dominates $N_r(o'_{i,x[i]+1}|t_q)$ and $N_l(o'_{i,x[i]+2}|t_q)$.

The user further checks whether the POIs with extended records returned are the global skyline POIs as in 2D-SKY.

If all the verifications succeed, the user considers the query result as complete and incomplete otherwise.

VII. Proofs of Correctness

We have the following three theorems regarding to the correctness of 1D-SKY, 2D-SKY, and 2D-SKY$^+$, respectively.

Theorem 1. *1D-SKY can detect any inauthentic and/or incomplete query result.*

Proof: (Sketch) First, the user can detect any inauthentic query result due to the unforgeability of LBSP's signature and the collision-resilient property of the cryptographic hash function used to construct the Merkle hash tree.

TABLE I: Default Simulation Settings

Value	Description
20	The number of road segments.
1000	The number of POIs per road segment.
32	The length of $\mathsf{id}_{i,j}$.
32	The length of relative position $t_{i,j}$.
32	The length of numeric attribute $\lambda_{i,j}$.
96	The length of neighbor $\mathsf{neighbor}_{i,j}$.
160	The length of hash image.
160	The length of the LBSP's signature.
128	The length of each reference vector.

Now suppose that $\mathsf{sky}(\mathcal{O}^-|t_q) \bigcup \mathsf{sky}(\mathcal{O}^+|t_q) = \{o_{j_1}, \dots, o_{j_u}\}$, where $t_{j_1} < \cdots < t_{j_u}$. Since o_{j_1} and o_{j_u} must correspond to the two special records inserted by the LBSP that should always be returned. Moreover, since every POI record E_{j_x} or C_{j_x}, $x \in [1, u-1]$, contains o_{j_x}'s right skyline neighbor set and corresponding neighbor ranges, among which there is a unique right neighbor with neighbor range contains t_q, inserting any POI into or omitting any POI from $o_{j_1}, \dots, o_{j_u}$ can be detected by the user.

Finally, according to Proposition 2, we have $\mathsf{sky}(\mathcal{O}|t_q) \subseteq \mathsf{sky}(\mathcal{O}^-|t_q) \bigcup \mathsf{sky}(\mathcal{O}^+|t_q)$. Since every extended or condensed POI record contains the position and numeric attribute, the user can verify whether the POIs with extended records returned are indeed $\mathsf{sky}(\mathcal{O}|t_q)$.

■

Theorem 2. Built upon 1D-SKY, 2D-SKY can detect any inauthentic and/or incomplete query result.

Theorem 3. Built upon 2D-SKY, 2D-SKY$^+$ can detect any inauthentic and/or incomplete query result.

We give the proofs of Theorems 2 and 3 in [27].

VIII. Simulation Results

In this section, we evaluate the computation and communication overhead of all three schemes using simulations on a synthetic POI dataset.

A. Simulation Settings

We simulate $m = 2f$ road segments in a square region of 10×10 km^2, including f horizontal and f vertical equally-spaced road segments. We assume that there are $n_i = n$ POIs distributed uniformly at random along each road segment e_i for all $i \in [1, m]$. In addition, all numeric attributes and relative positions are i.i.d. random variables uniformly distributed in the range $[0, 1]$ after proper normalization. Finally, we assume that the user always queries the skyline POIs among all the POIs. The simulation code is written in Java, and each data point represents the average of 100 simulation runs each with a distinct random seed. In addition, our simulation uses the default parameters in Table I unless stated otherwise.

B. Impact of n

Fig. 4a shows the impact of n on the number of hash computations the LBSP need perform for data preprocessing in 1D-SKY, 2D-SKY, and 2D-SKY$^+$. We can see that 2D-SKY and 2D-SKY$^+$ require the number of hash computations at the LBSP because they share data preprocessing procedure. Moreover, the number of hash computations needed increases as n increases, which is expected. We can also see that the number of hash computations needed experiences sharp increases when n increases from 250 to 300 as well as from 500 to 550. The reason is that when n just exceeds the perfect power of two such as 256 and 512, the number of dummy nodes needed to construct the Merkle hash tree increases drastically. Note that the LBSP also need sign the root of the Merkle root hash.

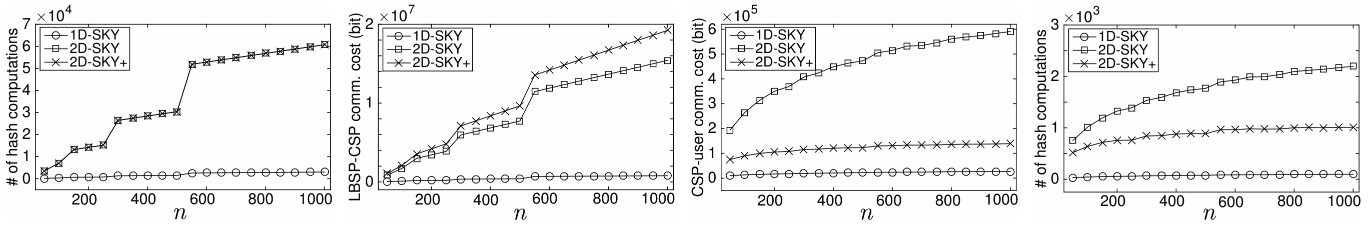
Fig. 4b shows the LBSP-CSP communication overhead of all three schems with n varying from 100 to 1000, where we only consider the extra communication overhead introduced by our schemes. We can see that the communication overhead of the 2D-SKY$^+$ is always higher than that of the 2D-SKY, because every extended POI record additionally contains the left neighbor range set and corresponding neighbor ranges of each POI under 2D-SKY$^+$. Similar to the trend in Fig.4a, the LBSP-CSP communication overhead increases sharply as n increases from 250 to 300 and from 500 to 550, as a large number of dummy hash values need be sent in these cases.

Fig. 4c shows the impact of n on the CSP-user communication overhead for all three schemes. We can see that the CSP-user communication overheads of all three schemes increase as n increases, which is expected. Moreover, the CSP-user communication overhead of 2D-SKY is always higher and increases faster than that of 2D-SKY$^+$. This result agrees with our expectation, as 2D-SKY$^+$ significantly reduces the number of POI records that need be returned.

Fig. 4d shows the impact of n on the number of hash computation needed for verifying a query result. We can see that the number of hash computations needed for verifying a query result in all three schemes increase as n increases. Moreover, the number of hash computations needed for verifying a query result under 2D-SKY is always larger and increases faster than that under 2D-SKY$^+$. Note that the user need verify the LBSP's signature on the root of the Merkle root hash, and the operation can be amortized over multiple queries.

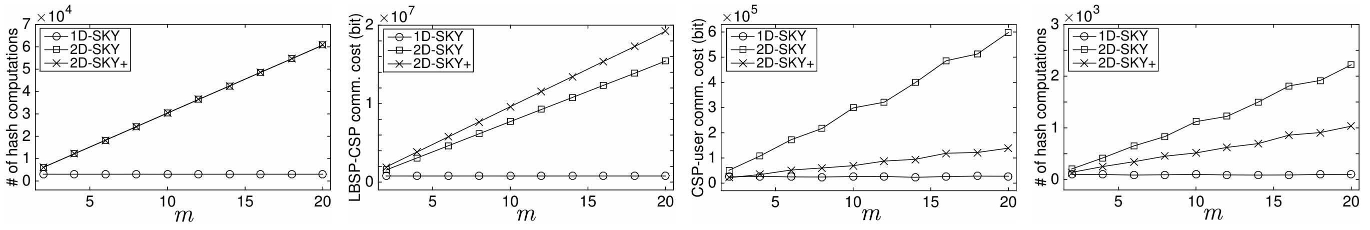
C. Impact of m

Figs. 5a to 5d show the LBSP's computation overhead, LBSP-CSP communication overhead, CSP-user communication overhead, and user's computation overhead of all three schemes with m varying from 2 to 20. In general, we can see that all four types of overheads increase linearly as m increases, which is expected. Similar to what we have observed in Figs. 4a to 4d, in comparison with 2D-SKY, 2D-SKY$^+$ incurs the same amount of computation overhead at the LBSP, higher LBSP-CSP communication overhead, much lower CSP-user communication overhead and computation overhead at the user. Since data preprocessing is a one-time process, the above results clearly demonstrate the significant advantages of 2D-SKY$^+$ over 2D-SKY.



(a) LBSP's computation overhead (b) LBSP-CSP comm. overhead (c) CSP-user comm. overhead (d) User's computation overhead

Fig. 4: The impact of n on 1D-SKY, 2D-SKY, and 2D-SKY$^+$



(a) LBSP's computation overhead (b) LBSP-CSP comm. overhead (c) CSP-user comm. overhead (d) User's computation overhead

Fig. 5: The impact of m on 1D-SKY, 2D-SKY, and 2D-SKY$^+$.

IX. Conclusion

In this paper, we have presented three novel schemes that allow efficient verification of any LBSQ result returned by an untrusted CSP by embedding and exploring a novel neighboring relationship among POIs. We have theoretically proved the correctness of our schemes and evaluated their performance via detailed simulation studies.

Acknowledgment

We would also like to thank the anonymous reviewers for insightful comments. This work was partially supported by the US National Science Foundation under grants CNS-1514381, CNS-1513141, CNS-1421999, CNS-1514014, and CNS-1422301.

References

[1] B. Zheng, K. C. K. Lee, and W.-C. Lee, "Location-dependent skyline query," in *MDM'08*, Beijing, China, April 2008.

[2] M. Goncalves, D. Torres, and G. Perera, "Making recommendations using location-based skyline queries," in *DEXA'12*, Sep. 2012.

[3] K. C. K. Lee, "Efficient evaluation of location-dependent skyline queries using non-dominance scopes," in *COM.Geo'11*, Washington, DC, 2011.

[4] M. Armbrust, *et al.*, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010.

[5] X. Lin, J. Xu, and H. Hu, "Authentication of location-based skyline queries," in *CIKM'11*, New York, NY, Oct. 2011, pp. 1583–1588.

[6] X. Lin, J. Xu, H. Hu, and W.-C. Lee, "Authenticating location-based skyline queries in arbitrary subspaces," *IEEE Trans. Knowl. Data Eng*, vol. 26, no. 6, pp. 1479–1493, June 2014.

[7] Y. Yang, S. Papadopoulos, D. Papadias, and G. Kollios, "Authenticated indexing for outsourced spatial databases," *The VLDB Journal*, vol. 18, no. 3, pp. 631–648, Jun. 2009.

[8] H. Hu, J. Xu, Q. Chen, and Z. Yang, "Authenticating location-based services without compromising location privacy," in *SIGMOD'12*, 2012.

[9] R. Zhang, Y. Zhang, and C. Zhang, "Secure top-k query processing via untrusted location-based service providers," in *INFOCOM'12*, Orlando, FL, Mar. 2012.

[10] Q. Chen, *et al.*, "Authenticating top-k queries in location-based services with confidentiality," *PVLDB*, vol. 7, no. 1, pp. 49–60, Sep. 2013.

[11] R. Zhang, J. Sun, Y. Zhang, and C. Zhang, "Secure spatial top-k query processing via untrusted location-based service providers," *IEEE Trans. Dependable Secure Comput.*, vol. 12, no. 1, pp. 111–124, Jan 2015.

[12] M. L. Yiu, E. Lo, and D. Yung, "Authentication of moving knn queries," in *ICDE'11*, Hannover, Germany, Apr. 2011, pp. 565–576.

[13] L. Hu, W.-S. Ku, S. Bakiras, and C. Shahabi, "Spatial query integrity with voronoi neighbors," *IEEE Trans. Knowl. Data Eng*, vol. 25, no. 4, pp. 863–876, Apr. 2013.

[14] M. L. Yiu, Y. Lin, and K. Mouratidis, "Efficient verification of shortest path search via authenticated hints," in *ICDE'10*, Long Beach, CA, Mar. 2010, pp. 237–248.

[15] H. Lo and G. Ghinita, "Authenticating spatial skyline queries with low communication overhead," in *CODASPY'13*, San Antonio, TX, 2013.

[16] H. Hacigümüş, B. Iyer, C. Li, and S. Mehrotra, "Executing SQL over encrypted data in the database-service-provider model," in *ACM SIGMOD'02*, Madison, Wisconsin, 6 2002, pp. 216–227.

[17] B. Hore, S. Mehrotra, and G. Tsudik, "A privacy-preserving index for range queries," in *VLDB'04*, Toronto, Canada, Aug. 2004, pp. 720–731.

[18] B. Hore, S. Mehrotra, M. Canim, and M. Kantarcioglu, "Secure multidimensional range queries over outsourced data," *The VLDB Journal*, vol. 21, no. 3, pp. 333–358, June 2012.

[19] E. Shi, J. Bethencourt, H. Chan, D. Song, and A. Perrig, "Multi-dimensional range query over encrypted data," in *IEEE S&P'07*, Oakland, CA, May 2007, pp. 350–364.

[20] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *INFOCOM'11*, Shanghai, China, Apr. 2011.

[21] W. Sun, S. Yu, W. Lou, Y. Hou, and H. Li, "Protecting your right: Attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud," in *INFOCOM'14*, Toronto, Canada, April 2014, pp. 226–234.

[22] B. Wang, S. Yu, W. Lou, and Y. Hou, "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud," in *INFOCOM'14*, Toronto, Canada, Apr. 2014, pp. 2112–2120.

[23] W. Sun, X. Liu, W. Lou, Y. Hou, and H. Li, "Catch you if you lie to me: Efficient verifiable conjunctive keyword search over large dynamic encrypted cloud data," in *INFOCOM'15*, Hong Kong, China, Apr. 2015.

[24] B. Wang, W. Song, W. Lou, and Y. Hou, "Inverted index based multi-keyword public-key searchable encryption with strong privacy guarantee," in *INFOCOM'15*, Hong Kong, China, Apr. 2015.

[25] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained access control in cloud computing," in *IEEE INFOCOM'10*, San Diego, CA, Mar. 2010.

[26] B. Wang, M. Li, H. Wang, and H. Li, "Circular range search on encrypted spatial data," in *IEEE CNS'15*, Florence, Italy, Sep. 2015.

[27] W. Chen, M. Liu, R. Zhang, Y. Zhang, and S. Liu "Secure outsourced skyline query processing via untrusted cloud service providers," Tech. Rep., Dec. 2015, available at http://www2.hawaii.edu/~ruizhang/paper/chen-skyline-TR.pdf.