Authenticating Outsourced Location-Based Skyline Queries under Shortest Path Distance

Yidan Hu*, Yukun Dong†, Wenxin Chen‡, Yingfei Dong‡, Rui Zhang†
*Department of Cybersecurity, Rochester Institute of Technology, Rochester, NY 14623 USA
†Department of Computer and Information Sciences, University of Delaware, Newark DE, 19716 USA
‡Department of Electrical and Computer Engineering, University of Hawaii at Manoa, Honolulu, HI 96822, USA
yidan.hu@rit.edu, yukun@udel.edu, wenxinc@hawaii.edu, yingfei@hawaii.edu, ruizhang@udel.edu

***Abstract*—An increasing number of location-based service providers are taking the advantage of cloud computing by outsourcing their Point of Interest (POI) datasets and query services to third-party cloud service providers (CSPs), which answer various location-based queries from users on their behalf. A critical security challenge is to ensure the integrity and completeness of any query result returned by CSPs. As an important type of queries, a location-based skyline query (LBSQ) asks for the POIs not dominated by any other POI with respect to a given query position, i.e., no POI is both closer to the query position and more preferable with respect to a given numeric attribute. While there have been several recent attempts on authenticating outsourced LBSQ, none of them support the shortest path distance that is preferable to the Euclidian distance in metropolitan areas. In this paper, we tackle this open challenge by introducing AuthSkySP, a novel scheme for authenticating outsourced LBSQ under the shortest path distance, which allows the user to verify the integrity and completeness of any LBSQ result returned by an untrusted CSP. We confirm the effectiveness and efficiency of our proposed solution via detailed experimental studies using both real and synthetic datasets.**

I. Introduction

The widespread use of Internet-capable and location-aware mobile devices is driving the rapid growth in location-based services (LBSes). Mobile users are increasingly accustomed to quering nearby points of interests (POIs) such as restaurants from various location-based service providers (LBSPs). As an important type of queries, location-based skyline queries (LBSQs) [1] allow users to retrieve the most "interesting" POIs based on both location proximity and user's preferences among a large collection of POIs while filtering out those that are clearly inferior. Specifically, an LBSQ asks for the POIs that are not *dominated* by any other POI with respect to the query position. One POI dominates another with respect to a user's query position if and only if the former is both closer to the query position and more preferable in terms of the numeric attributes of interest such as price. For example, a budget-sensitive user may issue an LBSQ to find restaurants, for each of which there is no other restaurant that is simultaneously cheaper and closer to his current location.

Recent years have witnessed a growing number of LBSPs have outsourced their POI datasets and query services to third-party cloud service providers (CSPs), which in turn answer various queries from mobile users on their behalf. For example, Yelp, a popular LBSP that offers POI searching and crowdsourced review sharing, hosts its dataset and services on Amazon Web Services. Data outsourcing offers several advantages over LBSPs operating their own dedicated private servers, including flexible access, elasticity, and reduced storage and operation costs [2]. Meanwhile, a well-known security challenge is that CSPs cannot be fully trusted, which may return forged or incomplete query results in favor of POIs willing to pay. This situation requires sound mechanisms for authenticating any query result returned by an untrusted CSP. In particular, a query result is considered authentic if it does not include forged POI information and complete if it contains all the POIs that satisfy the query condition.

Despite the significant efforts on authenticating outsourced query processing, there are only a few attempts [3]–[7] on authenticating outsourced LBSQ. Common to these efforts is the assumption that the distance between any POI and query position is measured by the Euclidean distance. While Euclidean distance is a widely used distance metric, it cannot accurately capture a user's true travel distance between two locations in metropolitan areas. In particular, two locations with a small Euclidean distance may be far apart due to buildings and obstacles in a metropolitan area. As a result, the shortest path distance is a much better metric for LBSQ. Unfortunately, since the shortest path distance between any two positions depends on the underlying road network, a small change in the query position may result in drastically different LBSQ results. This makes authenticating LBSQ a much more challenging problem and renders existing solutions [3]–[7] inapplicable. To the best of our knowledge, how to authenticate outsourced LBSQ under the shortest path distance remains unknown.

In this paper, we tackle this open challenge by AuthSkySP, a novel scheme for authenticating outsourced LBSQ processing under the shortest path distance metric by exploiting two unique properties of LBSQs. First, any LBSQ over a large region can be decomposed into multiple LBSQs with each over a subregion. Second, the skyline POIs of a road subnetwork that does not contain the query position must be a subset of a special POI set that can be precomputed without knowing the query position. Based on these two properties, AuthSkySP divides the road network into multiple subnetworks and authenticates the local skyline POI set in each subnetwork to allow the user to verify both the integrity and completeness of any LBSQ result.

979-8-3503-3945-1/23/$31.00 ©2023 IEEE

Our contributions can be summarized as follows.

- To the best of our knowledge, we are the first to study authenticating outsourced LBSQ processing under the shortest path distance.
- We introduce AuthSkySP, a novel scheme that allows a user to verify the integrity and completeness of any LSBQ result returned by an untrusted CSP.
- We implement AuthSkySP and confirm its efficacy and efficiency via detailed experimental studies using both real and synthetic datasets.

II. Related Work

As mentioned in Section I, authenticating outsourced LBSQ has been studied in [3]–[7]. In [3], Lin *et al.* considered LBSQs over a general 2D area and introduced a solution based on MR-Sky-tree and pre-computed skyline scope. This work was subsequently improved in [4] and [5] to support continuous LBSQ processing and LBSQs involving multiple numeric attributes, respectively. Authenticating outsourced LBSQ over a road network were studied in [6], [7]. Common to these efforts is that the distance between query position and POI is measured by the Euclidean distance, while the shortest path distance provides better indication for the user's traveling time in metropolitan areas. Since the shortest path between two positions depends on the underlying road network, none of these solutions [3]–[7] is applicable to our target problem.

Authenticating query processing in data outsourcing has received much attention in recent years. Various types of queries have been studied, including range queries [8]–[10], top-k queries [11]–[14], kNN queries [15], SQL queries [16], centerpoint query [17], social graph query [18], and so on. We tackle a totally different problem from them in this paper.

Skyline queries have been studied extensively in the data management community. Since the seminar work on skyline operator by Borzsony *et al.* [19], significant efforts have been made on efficient skyline query processing. For example, Chomicki *et al.* [20] introduced sort-filter-skyline algorithm to improve the efficiency of basic algorithm in [19] by pre-sorting tuples according to a particular dimension and no tuples can be dominated by the subsequent tuples. Zhang *et al.* [21] improved skyline computation efficiency by maintaining a much shorter skyline candidate list. Tang *et al.* [22] partitioned input datasets into disjoint subsets and compute skyline candidates in parallel for better efficiency. LShape [23] further improved the processing efficiency using a grid-based partitioning strategy. However, none of these work considers the integrity and completeness of the query result, and they are orthogonal to our work.

III. Problem Formulation

A. System Model

We consider a data outsourcing system consisting of an LBSP, a CSP, and many mobile users. The LBSP outsources its POI dataset to the CSP, which in turn answers LBSQs from mobile users on the LBSP's behalf. Every mobile user carries a smartphone and may issue LBSQs at any location through the LBSP's mobile app.

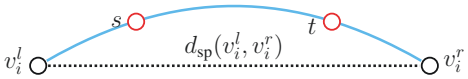


Fig. 1: Shortest path between s and t in the same road seg..

We assume that all the POIs reside over a road network. We model the road network as an undirected and weighted planar graph $G = (V, E)$ on the 2D plane, where V is the set of vertices each corresponding to one road junction, and $E = \{e_1, \ldots, e_m\}$ is the set of road segments, where m is the number of road segments. We leave the extension of our work to directed graph as our future work. Each road segment e_i connects its two vertices (i.e., road conjunctions), denoted by v_i^l and v_i^r, respectively, and has a weight w_i. In this paper, we assume that road segments may be of arbitrary shape and that the weight of a road segment is the arc length of e_i, i.e., the distance between v_i^l and v_i^r along the road segment e_i.

The POIs are commonly organized into different categories, such as gas stations, restaurants, and bars. For simplicity, this paper considers a set of POIs O in a single category. Let $O_i = \{o_{i,j} | 1 \leq j \leq n_i\}$ be the set of POIs that reside along road segment e_i and $o_{i,j}$ denotes the jth POI in e_i. It follows that $O = \bigcup_{i=1}^{m} O_i$, and $O_i \bigcap O_j = \emptyset$ for all $i \neq j$.

Each POI $o_{i,j}$ corresponds to one POI record $D_{i,j}$ in the LBSP's dataset represented as

$$D_{i,j} = \langle \mathsf{id}_{i,j}, x_{i,j}, \gamma_{i,j}, \mathsf{aux}_{i,j} \rangle, \tag{1}$$

where $\mathsf{id}_{i,j}$ is an ID assigned by the LBSP uniquely identifying $o_{i,j}$, $x_{i,j}$ is the arc length between v_i^l and $o_{i,j}$ along road segment e_i, $\gamma_{i,j} \in [\gamma_{\text{min}}, \gamma_{\text{max}}]$ is the numeric attribute of interest such as price, and $\mathsf{aux}_{i,j}$ denotes any auxiliary information such as its name, text reviews, and photos that does not affect whether $o_{i,j}$ satisfies a given LBSQ but is desirable to the user if it does. It follows that the arc length between $o_{i,j}$ and v_i^r is $w_i - x_{i,j}$. We assume that each POI has only one numeric attribute and leave the extension of LBSQ involving multiple numeric attributes as our future work. Moreover, we also assume that every POI is associated with one unique position, i.e., no two POIs share the same address. Our solutions can be easily adapted to relax the last assumption.

B. Query Model

We first provide the definition of the shortest path distance between any two vertices.

Definition 1. ***(Shortest path distance between two vertices)*** *Given two vertices $v_s, v_t \in V$, the shortest path between v_s and v_t is a sequence of road segments $(e_1, e_2, \ldots, e_z)$ that connects v_s to v_t, such that the total weight $\sum_{j=1}^{z} w_j$ is minimized, and the minimized weight, denoted by $d_{sp}(v_s, v_t)$, is the shortest path distance between v_s and v_t.*

We now extend the above definition into the shortest path distance between any two positions in the road network. Consider Fig. 1 as an example in which two positions s and t are in the same road segment e_i. The shortest path between s and t is

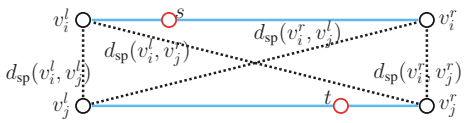


Fig. 2: Shortest path between s and t in different road seg..

either a partial segment of e_i or traverses two endpoints v_i^l and v_i^r and the shortest path between v_i^l and v_i^r. Let $d(s, v_i^l)$ and $d(t, v_i^l)$ be the arc distance between s and v_i^l and the arc distance between t and v_i^l along road segment e_i, respectively. Without loss of generality, we also assume that $d(s, v_i^l) < d(t, v_i^l)$, i.e., s is closer to v_i^l than t along e_i. The shortest path distance between s and t is then given by

$$\begin{aligned} d_{\text{sp}}(s,t) = &\min(d(t, v_i^l) - d(s, v_i^l), \\ &d(s, v_i^l) + d_{\text{sp}}(v_i^l, v_i^r) + w_i - d(t, v_i^l)), \end{aligned} \tag{2}$$

where w_i is the length of road segment e_i. For two positions in different road segments, let us consider Fig. 2 as an example. Since the shortest path from $s \in e_i$ must pass either v_i^l or v_i^r and that from $t \in e_j$ must pass either v_j^l or v_j^r, we have

$$\begin{aligned} d_{\text{sp}}(s,t) = \min(&d(s, v_i^l) + d_{\text{sp}}(v_i^l, v_j^l) + d(t, v_j^l), \\ &d(s, v_i^l) + d_{\text{sp}}(v_i^l, v_j^r) + d(t, v_j^r), \\ &d(s, v_i^r) + d_{\text{sp}}(v_i^r, v_j^l) + d(t, v_j^l), \\ &d(s, v_i^r) + d_{\text{sp}}(v_i^r, v_j^r) + d(t, v_j^r)). \end{aligned} \tag{3}$$

In practice, the shortest path distance between every pair of vertices can be precomputed and stored in a table so that the shortest path distance between any two positions can be efficiently computed by table lookup and Equation (3).

We now provide the definitions of dominance and location-based skyline query.

Definition 2. ***(Dominance)*** *For two POIs $o_{i,j}$ and $o_{i',j'}$, we say $o_{i,j}$* dominates *$o_{i',j'}$ with respect to query position q if and only if $d_{\text{sp}}(q, o_{i,j}) \leq d_{\text{sp}}(q, o_{i',j'})$ and $\lambda_{i,j} \leq \lambda_{i',j'}$ but the two equalities do not both hold.*

Definition 3. ***(Location-based skyline query(LBSQ))*** *An LBSQ $\mathsf{skl}(O|q)$ asks for the set of POIs in O that are not dominated by any other POI with respect to query position q.*

C. Adversary Model and Design Goals

We assume that when using the CSP's service for the first time, a user downloads an authentic copy of the road network (V, E) with no POI information. Assume that the user issues an LBSQ at query position q to the CSP to retrieve the skyline POI set $\mathsf{skl}(O|q)$. The query position can be any position within any road segment that cannot be predicted in advance.

We assume that the LBSP is trusted to faithfully follow system operations. In contrast, the CSP is not trusted and may return LBSQ results that contain forged or tampered POI records or POI records that are not among the skyline POIs. The CSP may also purposefully omit some skyline POI records.

We seek to enable verification of the integrity and completeness of any LBSQ result returned by the CSP. A query result is considered authentic if it does not include any forged or tampered POI record and complete if it contains all the true skyline POI records.

IV. AuthSkySP

In this section, we introduce AuthSkySP, a novel scheme for authenticating outsourced LBSQ under the shortest path distance. We first give an overview and then detail its design.

A. Overview

AuthSkySP is designed by exploring the decomposability of LBSQ characterized by the following theorem.

Theorem 1. ***(Decomposability of LBSQ)*** *Let O be a set of POIs and $O_1, \ldots, O_k$ a family of subsets of O such that $O = \bigcup_{j=1}^{k} O_j$. For any query position q, we have*

$$\mathsf{skl}(O|q) = \mathsf{skl}(O'|q) \ ,$$

where $O' = \bigcup_{j=1}^{k} \mathsf{skl}(O_j|q)$.

Proof. We first prove that $\mathsf{skl}(O|q) \subseteq \mathsf{skl}(O'|q)$. For any POI $o \in \mathsf{skl}(O|q)$, since $O = \bigcup_{j=1}^{k} O_j$, there must exist O_x, where $1 \leq x \leq k$ and $o \in O_x$. Since no other POI in O dominates o and $O_x \subseteq O$, no other POI in O_x dominates o either. It follows that $o \in \mathsf{skl}(O_x|q)$ and that $o \in O'$. Similarly, since $O' \subseteq O$, no other POI in O' dominates o, and therefore $o \in \mathsf{skl}(O'|q)$. We thus have $\mathsf{skl}(O|q) \subseteq \mathsf{skl}(O'|q)$.

We now prove that $\mathsf{skl}(O'|q) \subseteq \mathsf{skl}(O|q)$ by contradiction. Assume that there exists POI $o \in \mathsf{skl}(O'|q)$ such that $o \notin \mathsf{skl}(O|q)$. Since $o \notin \mathsf{skl}(O|q)$, there must exist POI $o' \in \mathsf{skl}(O|q)$ that dominates o with respect to query position q. Without loss of generality, suppose that $o' \in O_y$. There are two cases. First, if $o' \in \mathsf{skl}(O_y|q)$, then $o' \in O'$. Since o' dominates o with respect to query position q, we have $o \notin \mathsf{skl}(O'|q)$, leading to a contradiction. Second, if $o' \notin \mathsf{skl}(O_y|q)$, then there must exist $o'' \in \mathsf{skl}(O_y|q)$ that dominates o'. Since o' dominates o, it follows that o'' dominates o. Therefore, $o \notin \mathsf{skl}(O'|q)$, which also leads to a contradiction. We can thus conclude that $\mathsf{skl}(O'|q) \subseteq \mathsf{skl}(O|q)$.

Finally, since $\mathsf{skl}(O|q) \subseteq \mathsf{skl}(O'|q)$ and $\mathsf{skl}(O'|q) \subseteq \mathsf{skl}(O|q)$, we have $\mathsf{skl}(O|q) = \mathsf{skl}(O'|q)$, and the theorem is proved. □

A weaker version of Theorem 1 can be found in [6].

The decomposability of LBSQ makes it possible to authenticate any LBSQ over a large POI set into authenticating multiple LBSQs with each over a subset of POIs. We hereafter refer to $\mathsf{skl}(O|q)$ as the *global skyline set* and $\mathsf{skl}(\bigcup_{e_j \in E'} O_j|q)$ as a *local skyline set* with respect to the subset of road segments E'. Since the query position q must be in one of the road segments, we consider the two cases: (1) the road segments that do not contain query position q and (2) the road segment containing q.

1) Road Segments Not Containing q: We observe that for any subgraph $G' = (V', E')$ of the road network that does not contain query position q, the local skyline POI set, $\mathsf{skl}(O_{E'}|q)$, must be a subset of the union of multiple special skyline

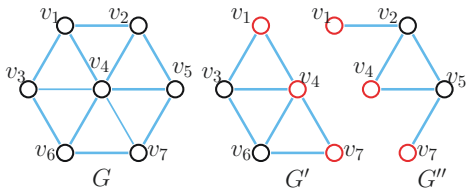


Fig. 3: An example of edge partition.

sets that can be precomputed without the knowledge of q. Specifically, let E' and $E \setminus E'$ be an *edge partition* of the road network $G = (V, E)$. Also let $G' = (V', E')$ and $G'' = (V'', E \setminus E')$ be the two subgraphs induced by E' and $E \setminus E'$, respectively, where V' and V'' each consist of every end vertex of the edges in E' and $E \setminus E'$, respectively. Note that some vertexes may be replicated in both V' and V'', as they are incident to road segments in both E' and $E \setminus E'$. We subsequently call the $\mathsf{entry}(E') = V' \bigcap V''$ the *entry vertex set* of the subgraph induced by E'. Fig. 3 shows an exemplary edge partition of a road network G, where G' and G'' are the two subgraphs induced by two road segment subsets, and v_1, v_4 and v_7 are the vertices replicated in both G' and G'', i.e., the entry vertexes of E'. We then have the following theorem.

Theorem 2. *Let $G = (V, E)$ be a road network. For any $E' \subseteq E$, let $G' = (V', E')$ and $G'' = (V'', E \setminus E')$ be the two subgraphs induced by E' and $E \setminus E'$, respectively. For any query position $q \notin E'$, we have*

$$\mathsf{skl}(O_{E'}|q) \subseteq \bigcup_{v \in \mathsf{entry}(E')} \mathsf{skl}(O_{E'}|v), \tag{4}$$

where $O_{E'} = \bigcup_{e_i \in E'} O_i$ and $\mathsf{entry}(E') = V' \bigcap V''$.

Proof. We prove this theorem by contradiction. For any query position $q \notin E'$, assume that there exists a POI $o_{i,j}$ such that $o_{i,j} \in \mathsf{skl}(O_{E'}|q)$ and $o_{i,j} \notin \bigcup_{v \in \mathsf{entry}(E')} \mathsf{skl}(O_{E'}|v)$.

Since $o_{i,j} \notin \bigcup_{v \in \mathsf{entry}(E')} \mathsf{skl}(O_{E'}|v)$, for every $v \in \mathsf{entry}(E')$, there must exist $o_{i_v,j_v} \in O_{E'}$ such that $\gamma_{i_v,j_v} \leq \gamma_{i,j}$ and $d_{\mathrm{sp}}(o_{i_v,j_v}, v) \leq d_{\mathrm{sp}}(o_{i,j}, v)$, but the two equalities do not both hold.

In addition, since the shortest path from $o_{i,j}$ to q must pass one of entry vertexes in $\mathsf{entry}(E')$, we have

$$\begin{aligned} d_{\mathrm{sp}}(o_{i,j}, q) &= \min\{(d_{\mathrm{sp}}(o_{i,j}, v) + d_{\mathrm{sp}}(v, q))|v \in \mathsf{entry}(E')\} \\ &\geq \min\{(d_{\mathrm{sp}}(o_{i_v,j_v}, v) + d_{\mathrm{sp}}(v, q))|v \in \mathsf{entry}(E')\} \\ &\geq \min\{d_{\mathrm{sp}}(o_{i_v,j_v}, v)|v \in \mathsf{entry}(E')\}. \end{aligned}$$

It follows that $o_{i,j}$ cannot be closer to q than all POIs in $\{o_{i_v,j_v}|v \in \mathsf{entry}(E')\}$. Since $\gamma_{i_v,j_v} \leq \gamma_{i,j}$ for all $v \in \mathsf{entry}(E')$, there exist at least one POI in $\{o_{i_v,j_v}|v \in \mathsf{entry}(E')\}$ that dominates $o_{i,j}$ with respect to query position q. Therefore, $o_{i,j}$ cannot be in $\mathsf{skl}(O_{E'}|q)$, leading to a contradiction. The theorem is therefore proved. □

We refer to $\bigcup_{v \in \mathsf{entry}(E')} \mathsf{skl}(O_{E'}|v)$ as the *local skyline union* of subgraph G' hereafter.

The decomposability of LBSQ along with Theorem 2 provides us with a general way to authenticate local skyline sets $\mathsf{skl}(\bigcup_{j=1,j\neq i}^{m} O_j|q)$ for any query position $q \in e_i$. Let $G_1, \ldots, G_k$ be a family of subgraphs of $G = (V, E)$ such that $\bigcup_{i=1}^{k} E_i = E \setminus \{e_i\}$. If we can require the CSP to return $\mathsf{skl}(\bigcup_{j=1,j\neq i}^{m} O_j|q)$ as well as the local skyline union of each subgraph $G_1, \ldots, G_k$, i.e., $\bigcup_{v \in \mathsf{entry}(E_1)} \mathsf{skl}(O_{E'}|v), \ldots, \bigcup_{v \in \mathsf{entry}(E_k)} \mathsf{skl}(O_{E'}|v)$, then the user would be able to verify if $\mathsf{skl}(\bigcup_{j=1,j\neq i}^{m} O_j|q)$ is a subset of $\bigcup_{j=1,j\neq i}^{m}(\bigcup_{v \in \mathsf{entry}(E_i)} \mathsf{skl}(O_{E'}|v))$.

2) Road Segment Containing q: For road segment e_i that contains query position q, we extend the 1D-SKY scheme in [6] to enable authenticating $\mathsf{skl}(O_i|q)$ under the shortest path distance. 1D-SKY [6] was designed to support authenticating $\mathsf{skl}(O_i|q)$ over a single *straight* road segment under the Euclidian distance. However, the shortest path distance between two positions may not be the partial road segment connecting them as shown in Fig. 1. Fortunately, we find a novel transformation of POI set O_i that makes it possible for us to apply 1D-SKY.

The key idea behind the transformation is a novel *skyline preserving mapping* that maps every position in road segment e_i with an arc length w_i into a virtual straight road segment such that skyline queries on the original road segment e_i under the shortest path distance is equivalent to skyline queries on the virtual road segment under the Euclidian distance. Since any position in e_i can be uniquely identified by its arc distance from v_i^l, we can represent the road segment e_i by the range $R_i = [0, w_i]$. The mapping takes the range $R_i = [0, w_i]$, the shortest path distance $d_{\mathrm{sp}}(v_i^l, v_i^r)$ between v_i^l and v_i^r, and a reference position $p \in R_i$ as input and assigns every position $x \in R_i$ a new coordinate $f_p(x)$ in the virtual road segment in two steps.

First, we divide the range R_i into two subranges R_i^{p-} and R_i^{p+}, where R_i^{p-} (or R_i^{p+}) consists of all positions $x \in R_i$ such that the shortest path from x to p reaches p from the left (or right). For any position $x \in R_i$, we can easily determine whether $x \in R_i^{p-}$ or R_i^{p+}. Denote by $d_{\mathrm{sp}}^{\rightarrow}(x, p)$ and $d_{\mathrm{sp}}^{\leftarrow}(x, p)$ the distance of the shortest path from x to p that reaches p from the left and right, respectively. We have

$$d_{\mathrm{sp}}^{\rightarrow}(x, p) = \begin{cases} p - x & \text{if } x \leq p, \\ w_i - x + d_{\mathrm{sp}}(v_i^l, v_i^r) + p & \text{if } x > p\,, \end{cases} \tag{5}$$

and

$$d_{\mathrm{sp}}^{\leftarrow}(x, p) = \begin{cases} x + d_{\mathrm{sp}}(v_i^l, v_i^r) + w_i - p & \text{if } x \leq p, \\ x - p & \text{if } x > p\,. \end{cases} \tag{6}$$

It follows that $x \in R_i^{p-}$ if $d_{\mathrm{sp}}^{\rightarrow}(x, p) \leq d_{\mathrm{sp}}^{\leftarrow}(x, p)$ and $x \in R_i^{p+}$ if $d_{\mathrm{sp}}^{\rightarrow}(x, p) > d_{\mathrm{sp}}^{\leftarrow}(x, p)$.

The second step of the mapping is to assign every position $x \in R_i$ a new coordinate in the virtual straight road segment. Specifically, for any position $x \in R_i$, its new coordinate in the virtual road segment is given by

$$f_p(x) = \begin{cases} -d_{\mathrm{sp}}(x, p) & \text{if } x \in R_i^{p-}, \\ d_{\mathrm{sp}}(x, p) & \text{if } x \in R_i^{p+}. \end{cases} \tag{7}$$

where $d_{\mathrm{sp}}(x, p) = \min(d_{\mathrm{sp}}^{\rightarrow}(x, p), d_{\mathrm{sp}}^{\leftarrow}(x, p))$, which is equivalent to the one given in Eq. (2). It is easy to see that under this

transformation $f_p(p) = 0$ for reference position p.

The above mapping has a number of important properties. First, the mapping is invertible. Since no two positions in the original road segment will be mapped to the same position in the virtual road segment, the inverse mapping must exist for any valid $f_p(x)$. In fact, given any coordinate y in the virtual road segment, the arc length w_i of e_i, the shortest path distance $d_{\text{sp}}(v_i^l, v_i^r)$, and reference position $p \in R_i$, the position y's coordinate in the original road segment can be computed according to the following three cases.

- Case 1: if $d_{\text{sp}}(v_i^l, v_i^r) < w_i \ p > \frac{d_{\text{sp}}(v_i^l, v_i^r)+w_i}{2}$, and $w_i - p + d_{\text{sp}}(v_i^l, v_i^r) \le y < \frac{d_{\text{sp}}(v_i^l, v_i^r)+w}{2}$, then
$$f_p^{-1}(y) = y + p - w_i - d_{\text{sp}}(v_i^l, v_i^r) \ . \tag{8}$$
- Case 2: if $d_{\text{sp}}(v_i^l, v_i^r) < w_i, \ p \le \frac{w - d_{\text{sp}}(v_i^l, v_i^r)}{2}$, and $\frac{-d_{\text{sp}}(v_i^l, v_i^r) - w}{2} \le y \le -p - d_{\text{sp}}(v_i^l, v_i^r)$, then
$$f_p^{-1}(y) = y + p + w_i + d_{\text{sp}}(v_i^l, v_i^r) \ . \tag{9}$$
- Case 3: if other than Cases 1 or 2, then
$$f_p^{-1}(y) = y + p \ . \tag{10}$$

We apologize for omitting the detailed derivation of Eqs. (8) to (10) due to space limitations. For convenience, we also postulate that $f_p(-\infty) = -\infty$ and $f_p^{-1}(-\infty) = -\infty$ for all $p \in R_i$.

Second, for any reference position $p \in R_i$, we can divide POI set O_i into O_i^{p-} and O_i^{p+} depending on whether $x_{i,j} \in R_i^{p-}$ or R_i^{p+}. Third, the shortest path distance between any POI and reference location p remain the same after the mapping. It follows that $\mathsf{skl}(O_i^{p-}|p)$ and $\mathsf{skl}(O_i^{p+}|p)$ in the original road segment do not change after mapping to the virtual road segment. Moreover, the decomposability of LBSQ indicates that $\mathsf{skl}(O_i|p) = \mathsf{skl}(\mathsf{skl}(O_i^{p-}|p) \bigcup \mathsf{skl}(O_i^{p+}|p))$.

We further introduce a few key concepts based on the above mapping which are similar to those in [6]. For any two POIs $o_{i,j}, o_{i,k} \in O_i$, we say $o_{i,k}$ is the *right skyline neighbor* [1] of $o_{i,j}$ with respect to query position $q \in e_i$ if and only if (1) both $o_{i,j}, o_{i,k} \in \mathsf{skl}(O_i^{q-}|q) \bigcup \mathsf{skl}(O_i^{q+}|q)$ where O_i^{q-} and O_i^{q+} are defined based on the skyline preserving mapping with q being the reference position, (2) $f_q(x_{i,j}) < f_q(x_{i,k})$, i.e., $o_{i,k}$ is on the right of $o_{i,j}$ in the virtual road segment, and (3) no other POI in $\mathsf{skl}(O_i^{q-}|q) \bigcup \mathsf{skl}(O_i^{q+}|q)$ reside between them. Furthermore, for any two POIs $o_{i,j}$ and $o_{i,k}$, we call $o_{i,k}$ a *possible right skyline neighbor* of $o_{i,j}$ if there exists at least one query position $q \in e_i$ such that $o_{i,k}$ is the right skyline neighbor of $o_{i,j}$ with respect to q. Let $N(o_{i,j}) \subset O_i$ be the set of possible right skyline neighbors of POI $o_{i,j}$. For each $o_{i,k} \in N(o_{i,j})$, there exists a *neighbor query range*, denoted by $\mathsf{range}(o_{i,k}|o_{i,j}) \subseteq e_i$, such that $o_{i,k}$ is the right skyline neighbor of $o_{i,j}$ with respect to q if and only if $q \in \mathsf{range}(o_{i,k}|o_{i,j})$.

A key observation is that for every POI $o_{i,j} \in O_i$, the set of possible right skyline neighbors $N(o_{i,j})$ and the neighbor query range $\mathsf{range}(o_{i,k}|o_{i,j})$ of each $o_{i,k} \in N(o_{i,j})$ can be efficiently computed in a similar way as 1D-SKY [6].

Consider POI $o_{i,j}$ at position $x_{i,j}$ as an example. We first perform a skyline preserving mapping for O_i with $x_{i,j}$ being the reference position such that every POI $o_{i,k} \in O_i$ obtains a new coordinate $f_{x_{i,j}}(x_{i,k})$ according to Eq. (7). After the mapping, the POI set O_i can be viewed as residing on a straight road segment where the shortest path distance between any two POIs is equivalent to their Euclidean distance. We can then use Algorithm 1 in 1D-SKY [6] to compute $N(o_{i,j})$ and the neighbor query range for every $o_{i,k} \in N(o_{i,j})$. More specifically, Algorithm 1 in [6] takes $\{(f_{x_{i,j}}(x_{i,k}), \lambda_{i,k}) | 1 \le k \le n_i\}$ as input and returns $o_{i,j}$'s possible skyline neighbor set $N(o_{i,j})$ and a neighbor query range $(l_{j,k}, r_{j,k})$ for every $o_{i,k} \in N(o_{i,j})$. Since the range $(l_{j,k}, r_{j,k})$ is defined over the virtual road segment, we need to take an extra step to convert it back to the neighbor query range in the original road segment via the inverse mapping as $\mathsf{range}(o_{i,k}|o_{i,j}) = (f_{x_{i,j}}^{-1}(l_{j,k}), f_{x_{i,j}}^{-1}(r_{j,k}))$, where $f_{x_{i,j}}^{-1}(\cdot)$ is given in Eqs. (8) to (10). We refer readers to [6] for details of Algorithm 1 due to space limitations.

In summary, for every POI $o_{i,j} \in O_i$, we can compute its set of possible skyline neighbors $N(o_{i,j})$, and the query range $\mathsf{range}(o_{i,k}|o_{i,j})$ for each $o_{i,k} \in N(o_{i,j})$ such that $o_{i,k}$ is the right skyline neighbor of $o_{i,j}$ with respect to query position q if and only if $q \in \mathsf{range}(o_{i,k}|o_{i,j})$. As we will see shortly, the precomputed skyline neighbor relationship allows us to chaining adjacent skyline neighbors via cryptographic primitives to allow the user to verify the completeness of $\mathsf{skl}(O_i^{q-}|q) \bigcup \mathsf{skl}(O_i^{q+}|q)$ for any $q \in e_i$.

In what follows, we first introduce how to construct a graph partition tree from road network $G = (V, E)$ and then detail the AuthSkySP operations, which consist of data preprocessing at the LBSP, query processing at the CSP, and query-result verification at the user.

B. Graph Partition Tree Construction

To facilitate efficient authentication of $\mathsf{skl}(\bigcup_{j=1, j\neq i}^{m} O_j|q)$, we construct a *graph partition tree* over the set of road segments E. A graph partition tree is a binary tree, in which the root corresponds to the entire road network G, every leaf node corresponds to the subgraph induced by a single road segment, and every non-leaf node corresponds to the subgraph induced by the union of its two childrens road segments.

Given road network $G = (V, E)$, we construct a graph partition tree T in a top-down fashion. Starting from the root that corresponds to the whole road network G, for every non-leaf node v that corresponds to subgraph $G_{\mathsf{v}} = (V_{\mathsf{v}}, E_{\mathsf{v}})$, we construct two children nodes by finding a *balanced 2-way vertex-cut* [24] of G_{v}. Specifically, we divide its edge set E_{v} into two subsets $E_{\mathsf{v},0}$ and $E_{\mathsf{v},1}$ of approximately equal size while minimizing the number of vertices replicated in the two subgraphs $G_{\mathsf{v},0}$ and $G_{\mathsf{v},1}$ induced by $E_{\mathsf{v},0}$ and $E_{\mathsf{v},1}$, respectively. We then construct the two children of node v as $\mathsf{v}.0 = G_{\mathsf{v},0}$ and $\mathsf{v}.1 = G_{\mathsf{v},1}$. Since the set of replicated vertices is always a subset of the entry vertex set of each subgraph, doing so can keep the size of each subgraph's entry vertex

[1] Left skyline neighbor can be defined accordingly but is not used.

set small. While finding the a balanced 2-way vertex cut of a graph is NP-hard [25], we adopt the algorithm in [26] that runs in polynomial-time with a guaranteed approximation ratio. The process continues until every node only has one road segment.

C. Data Preprocessing

The LBSP preprocesses its POI dataset $\{D_{i,j}|1 \leq i \leq m, 1 \leq j \leq n_i\}$ before outsourcing it to the CSP.

First, for each road segment $e_i, 1 \leq i \leq m$, the LBSP inserts a special virtual POI $o_{i,0}$ and POI record $D_{i,0} = \langle \mathsf{id}^*, x^*, \gamma^* \rangle$, where id^* is a publicly known special string, $x^* = -\infty$, and γ^* is a special public value smaller than $\gamma_{\min}$. Since $\gamma^* < \gamma_{\min}$ and $d_{\mathrm{s,p}}(x^*, q) = \infty$ for any query position $q \in e_i$, POI $o_{i,0} \in \mathsf{skl}(O_i|q)$ for any $q \in e_i$ and is always the leftmost skyline POI in $\mathsf{skl}(O_i^{q-}|q) \bigcup \mathsf{skl}(O_i^{q+}|q)$. $o_{i,0}$ will serve as an anchor POI for e_i and can prevent the CSP from claiming that $O_i = \emptyset$.

Second, for every POI record $D_{i,j}, 1 \leq i \leq m, 0 \leq j \leq n_i$, the LBSP constructs a compressed POI record as $C_{i,j} = \langle \mathsf{id}_{i,j}, d_{i,j}, \lambda_{i,j}, h(\mathsf{aux}_{i,j}) \rangle$, where $h(\cdot)$ denotes a good cryptographic hash function such as SHA-256.

Third, for every non-root node v of the graph partition tree T, the LBSP finds its entry vertex set $\mathsf{entry}(E_\mathsf{v})$, computes the local skyline union of subgraph E_v as $U_\mathsf{v} = \bigcup_{v \in \mathsf{entry}(E_\mathsf{v})} \mathsf{skl}(O_{E_\mathsf{v}}|v)$ and constructs a local skyline union record as $C_\mathsf{v} = ||_{o_{i,j} \in U_\mathsf{v}} C_{i,j}$.

Fourth, for every road segment e_i with POI set $O_i, 1 \leq i \leq m$, the LBSP creates a chaining relationship among skyline neighboring POIs. Next, for every POI $o_{i,j}, 0 \leq j \leq n_i$, the LBSP computes $o_{i,j}$'s possible right skyline neighbor set and corresponding query ranges as

$$\mathsf{nb}_{i,j} = \{\langle \mathsf{id}_{i,k}, \mathsf{range}(o_{i,k}|o_{i,j}) \rangle | o_{i,k} \in N(o_{i,j})\} \tag{11}$$

using Algorithm 1 in [6] in combination with mapping $f_{x_{i,j}}(\cdot)$ given in Eq. (7) and inverse mapping $f^{-1}_{x_{i,j}}(\cdot)$ given in Eqs. (8) to (10) as discussed in Section IV-A2. Moreover, the LBSP computes a *neighbor-embedded POI record* as $D^+_{i,j} = \langle \mathsf{id}_{i,j}, d_{i,j}, \gamma_{i,j}, \mathsf{nb}_{i,j}, \mathsf{aux}_{i,j}$ and a *compressed neighbor-embedded POI record* as $C^+_{i,j} = \langle \mathsf{id}_{i,j}, d_{i,j}, \gamma_{i,j}, \mathsf{nb}_{i,j}, h(\mathsf{aux}_{i,j}) \rangle$.

Fifth, the LBSP builds one Merkle hash tree T^c over all compressed neighbor-embedded POI records $\{C^+_{i,j}|1 \leq i \leq m, 0 \leq j \leq n_i\}$ and another Merkle hash tree T^u over all local skyline record unions $\{C_\mathsf{v}\}_{\mathsf{v} \in \mathsf{T}}$ with each leaf node corresponding to one non-root node in graph partition tree T and signs the roots of both T^c and T^u.

Finally, the LBSP sends road network G, graph partition tree T, all neighbor-embedded POI records $\{D^+_{i,j}|1 \leq i \leq m, 0 \leq j \leq n_i\}$ and its signatures on both Merkle root hashes to the CSP. Anyone can computes $D_{i,j}$, $C^+_{i,j}$, and $C_{i,j}$ given $D^+_{i,j}$.

D. Query Processing

Assume that the user issues an LBSQ at position q in road segment e_i. The CSP computes the global skyline POI set $\mathsf{skl}(O|q)$ and constructs the query result as follows.

First, the CSP constructs a partial query result for road segments other than e_i according to the graph partition tree T. Specifically, the CSP first finds a minimal-size subset S of internal nodes of the graph partition tree T such that the union of their edge sets covers $E \setminus \{e_i\}$. This can be done easily by traversing T from the root to the leaf node corresponding to road segment e_i. For each non-root node we visit, we add its sibling node to S. For each node $\mathsf{v} \in \mathsf{S}$, the CSP includes the following information as partial query result

$$\mathsf{R}_\mathsf{v} = \{X_{j,k}|o_{j,k} \in U_\mathsf{v}\}, \tag{12}$$

where $X_{j,k} = D_{j,k}$ if $o_{j,k} \in \mathsf{skl}(O|q)$ and $C_{j,k}$ otherwise.

Second, the CSP constructs a partial query result for road segment e_i. Specifically, the CSP divides O_i into O_i^{q-} and O_i^{q+} with q being the reference position and computes $\mathsf{skl}(O_i^{q-}|q)$ and $\mathsf{skl}(O_i^{q+}|q)$. It then constructs the partial query result as

$$\mathsf{R}_i = \{X_{i,k}|o_{i,k} \in \mathsf{skl}(O_i^-|q) \bigcup \mathsf{skl}(O_i^+|q)\}, \tag{13}$$

where $X_{i,k} = D^+_{i,k}$ if $o_{i,k} \in \mathsf{skl}(O|q)$ and $C^+_{i,k}$ otherwise.

Finally, the CSP returns the information needed for the user to verify the integrity of the query result, which include all the internal nodes in T^u's needed for computing the root from every returned local skyline record union C_v, all the internal nodes in T^c needed for computing the root every returned neighbor-embedded POI record $C^+_{i,k}$, and the LSBP's signatures on the root hashes of Merkle trees T^c and T^u.

E. Query-Result Verification

Every user downloads an authenticated copy of the road network G and the graph partition tree T upon registration. On receiving the query result, the user verifies its integrity and completeness as follows.

1) Integrity Verification: The user first verifies the integrity of the query result using two Merkle hash trees and the LBSP's signatures. Assume that the user has received $\{\mathsf{R}_\mathsf{v}|\mathsf{v} \in \mathsf{S}'\}$ and R_i as the query result, where S' is the subset of nodes in T with local skyline unions returned.

For every $\mathsf{R}_\mathsf{v}, \mathsf{v} \in \mathsf{S}'$, the user first converts every original POI record therein into corresponding compressed POI record whereby to reconstruct local skyline record union C_v. The user then computes the Merkle hash root using corresponding internal nodes of T^u. If all the local skyline record unions lead to the same Merkle hash root, the user further verifies the LBSP's signature on the root. Similarly, the user verifies the integrity of every POI record in R_i using corresponding internal nodes of T^c and the LBSP's signature. If all verifications succeed, the user considers the query result authentic.

2) Completeness Verification: The user then verifies the completeness of the query result in three steps. First, the user checks if $\bigcup_{\mathsf{v} \in \mathsf{S}'} E_\mathsf{v} = E \setminus \{e_j\}$. If so, he extracts the local skyline union $U_\mathsf{v} = \bigcup_{v \in \mathsf{entry}(E_\mathsf{v})} \mathsf{skl}(O_{E_\mathsf{v}}|v)$ for every $\mathsf{v} \in \mathsf{S}'$, where $U_\mathsf{v} = \bigcup_{v \in \mathsf{entry}(E_\mathsf{v})} \mathsf{skl}(O_{E_\mathsf{v}}|v)$

Second, the user verifies whether R_i includes all POIs in $\mathsf{skl}(O_i^{q-}|q) \bigcup \mathsf{skl}(O_i^{q+}|q)$. Specifically, the user first converts every neighbor-embedded POI record into the corresponding compressed neighbor-embedded POI record. For every $C^+_{i,j}$ in R_i, the user performs a skyline preserving mapping with

TABLE I: Summary of Real Datasets

Dataset	# of POIs	# of Road Segments
MAN	4,840	8,035
CHI	1,427	2,899
LA	1,589	11,070
SF	3,168	5,030

reference position q to compute a new coordinate $f_q(x_{i,j})$ according to Eqs. (5) to (7). The user then sorts all the $C^+_{i,j}$s according to $f_q(x_{i,j})$ in an ascending order. Without loss of generality, assume that R_i consists of compressed neighbor-embedded POI records $C^+_{i,j_1}, \ldots, C^+_{i,j_k}$ where $f_q(x_{i,j_1}) < \cdots < f_q(x_{i,j_k})$. The user first checks if the leftmost record $C^+_{i,j_1} = \langle \mathsf{id}^*, x^*, \gamma^*, \mathsf{nb}_{i,0} \rangle$, which should correspond to the special virtual POI inserted by the LBSP. Then for every pair of adjacent POI records C^+_{i,j_y} and $C^+_{i,j_{y+1}}$, $0 \le y \le k-1$, the user checks if $o_{i,j_{y+1}} \in N(o_{i,j_y})$ and $q \in \mathsf{range}(o_{i,j_{y+1}} | o_{i,j_y})$ according to nb_{i,j_y} in C^+_{i,j_y}. Last, the user checks if nb_{i,j_k} in C^+_{i,j_k} shows that POI o_{i,j_k} has no right skyline neighbor with respect to query position q. If all the verifications succeed, the user considers R_i includes all POIs in $\mathsf{skl}(O_i^{q-}|q) \bigcup \mathsf{skl}(O_i^{q+}|q)$.

Finally, the user verifies the completeness of returned global skyline POIs based on the decomposability of LBSQ. Let O_q be the set of POIs for which either a neighbor-embedded POI record or POI record is returned. The user checks whether $O_q = \mathsf{skl}(O')$ where $O' = \mathsf{skl}(O_i^{q-}|q) \bigcup \mathsf{skl}(O_i^{q+}|q) \bigcup_{\mathsf{v} \in \mathsf{S}'} U_{\mathsf{v}}$. If so, the user considers the query result complete.

V. Experiment Results

A. Datasets

We collect four real restaurant datasets from Yelp via Yelp Fusion APIs [27] using a customized Python program, which consists of the records of restaurants in (1) the Manhattan area, New York (MAN), (2) Chicago, Illinois (CHI), (3) Los Angeles, California (LA), and (4) San Francisco, California (SF), respectively. Each restaurant record consists of its ID, name, rating, price, street address, coordinates, and other information such as category, phone number, user reviews, and images, and we use the rating as the numeric attribute.

For each dataset, we construct the corresponding road network using the OpenStreetMap Overpass API [28]. Specifically, we first collect all road segments and construct a connected graph as the road network. We then map every restaurant to one road segment based on its latitude and longitude. Table I summarizes the four datasets.

We also generate two synthetic datasets from the MAN dataset to evaluate the impacts of the number of POIs and the number of road segments, respectively. The first synthetic dataset is generated by fixing the number of road segments to 8,035, while randomly generating different numbers of POIs distributed uniformly in the road network with numeric attribute drawn from (0, 5] uniform at random. The second synthetic dataset is generated by randomly selecting different numbers of connected road segments from the MAN dataset while keeping the POIs in the selected road segments.

TABLE II: Default Experiment Settings

Para.	Value	Description
$\lvert \mathrm{id}_{i,j} \rvert$	16	The length of id in bit
$\lvert d_{i,j} \rvert$	32	The length of distance in bits
$\lvert \lambda_{i,j} \rvert$	32	The length of numeric attribute in bits
$\lvert \mathrm{nb}^r_{i,j} \rvert$	80	The length of neighbor information in bits
$\lvert h(\cdot) \rvert$	256	The length of hash function $h(\cdot)$ in bits
	1024	The length of LBSP's signature in bits

B. Experimental Settings

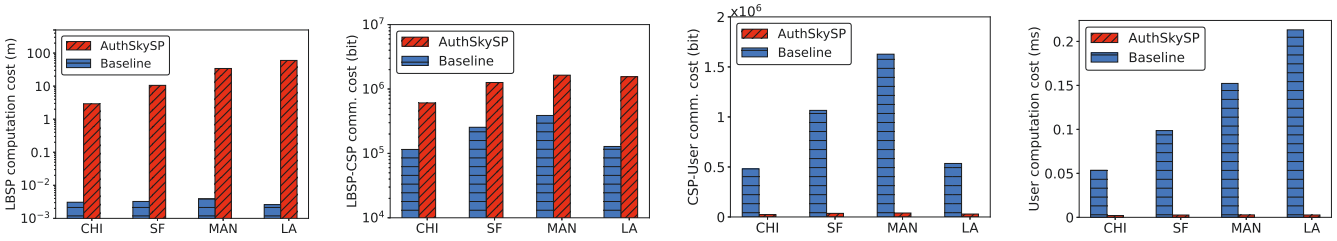
We implement AuthSkySP in Python using packages *networkx*, *cryptography*, and *hashlib* for efficient shortest path distance calculation, signature and verification, and hash computation, respectively. We deploy AuthSkySP on three desktops, which act as the LBSP, the CSP, and the user, respectively. Each desktop has a i7-6700 CPU, 16GB RAM, and 64-bit Win10 operating system, and they communicate with each other using the socket API [29]. We choose SHA-256 as the cryptographic hash function and 1024-bit RSA for digital signature. Table II summarizes other default parameters.

Since there is no prior work on authenticating outsourced LBSQ under the shortest path distance, we compare AuthSkySP with an intuitive *Baseline* which lets the user retrieve all the POI records and calculate the LBSQ result locally. Specifically, the LBSP first generates a barebone POI record for each POI that includes only the ID, location, and numeric attribute. It then signs the hash of their concatenation. On receiving an LBSQ, the CSP returns the compressed POI record for every global skyline POI, the barebone POI record for every other POI, and the LBSP's signature. The user verifies the integrity of the query result using the LBSP's signature and verifies its completeness by computing the global skyline set.

We use four metrics for performance comparison, including (1) *LBSP computation cost*, which is the time needed for preprocessing the dataset, (2) *LBSP-CSP communication cost*, which is the amount of extra information in bits transmitted from the LBSP to the CSP, (3) *CSP-user communication cost*, which is the amount of extra information in bits incurred by a single LBSQ, and (4) *user computation cost*, which is the time needed for verifying an LBSQ result by the user.

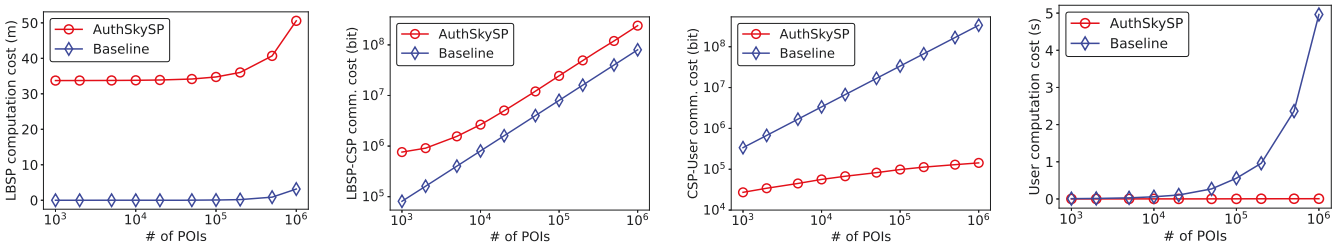
C. Results from Real Datasets

Figs. 4(a) to 4(d) compares the LBSP computation cost, LBSP-CSP communication cost, CSP-user communication cost, and user computation cost, respectively, under AuthSkySP and *Baseline* on the four datasets. As we can see in Fig. 4(a), AuthSkySP incurs much higher LBSP's computation cost than *Baseline*, which is anticipated because *Baseline* does not involve any preprocessing other than signing the concatenation of the POI records. In contrast, AuthSkySP also needs to construct graph partition tree T and Merkle hash tree T^u where all the non-root nodes of the graph partition tree T are its leaf nodes. Fig. 4(b) shows that *Baseline* also incurs lower LBSP-CSP communication cost and shares similar trend with Fig. 4(a). The reason is that, the LBSP only needs to transmit POI records with no skyline neighbor information to the CSP



(a) LBSP computation cost. (b) LBSP-CSP comm. cost. (c) CSP-user comm. cost. (d) User computation cost.

Fig. 4: The performance of AuthSkySP from real datasets.



(a) LBSP computation cost (b) LBSP-CSP comm. cost. (c) CSP-user comm. cost (d) User computation cost

Fig. 5: The impact of the total number of POIs on AuthSkySP from synthetic datasets.

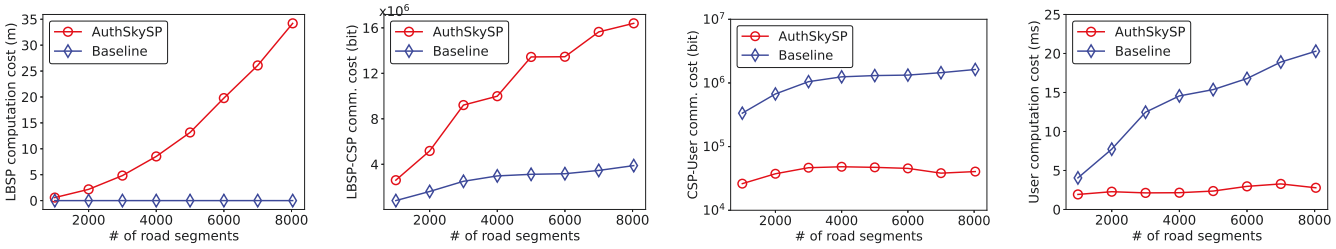
under *Baseline*. In contrast, under AuthSkySP, LBSP needs to send one local skyline union for every non-root node of T. From Fig. 4(c), we can see that AuthSkySP outperforms *Baseline* with a much reduced CSP-user communication cost. The reason is that under *Baseline*, the CSP needs to return all POI information in the form of either compressed POI record or POI record to the users. In contrast, under AuthSkySP, the CSP only needs to return one local skyline union for every node along the path from the root of the graph partition tree T to the leaf node containing the query position. We can also see that the user computation cost under AuthSkySP is approximately one order of magnitude lower than that under *Baseline* in Fig. 4(d). The main reason is that the user needs to verify all compressed POI records and compute the global skyline set under *Baseline*. In contrast, the number of POI records returned under AuthSkySP is much fewer than that under *Baseline* so computing the global skyline set is much faster.

Despite the fact that AuthSkySP incurs higher LBSP computation and LBSP-CSP communication cost than *Baseline*, data preprocessing is one-time process, while the CSP needs to process many LBSQs from possibly many users. The above results from the four real datasets consistently demonstrate the significant advantages of AuthSkySP over *Baseline*.

D. Results from Synthetic Datasets

Impact of Number of POIs. Figs. 5(a) to 5(d) compare the performance of AuthSkySP and *Baseline* with the number of POIs varying from 1,000 to 1,000,000. As we can see from Fig. 5(a), AuthSkySP has a higher LBSP computation cost than *Baseline*, which coincides with the results from the real datasets. In addition, the LBSP computation cost grows slowly for *Baseline* as the number of POIs increases with AuthSkySP being faster. The reason is that when the number of POIs further increases, each subgraph in graph partition tree T contains more POIs and so it takes more time to compute the local skyline unions. Fig. 5(b) shows that the LBSP-CSP communication cost increases nearly linearly as the number of POIs increases under the two schemes, which is resulted from the increase in the number of compressed POI records that need be transmitted. We can also see that AuthSkySP incurs higher LBSP-CSP communication cost than *Baseline*, which is anticipated as the LBSP needs to transmit the local skyline unions of T's internal nodes under AuthSkySP. Fig. 5(c) shows that CSP-user communication cost increases as the number of POIs increases under the two schemes. This is anticipated, as the number of POI records that need be returned increase as the number of POIs increases. Moreover, the CSP-user communication cost under *Baseline* increases linearly as the number of POIs increases. In contrast, the CSP-user communication cost of the AuthSkySP increases much slower than *Baseline* as there is no need to return any records for non-local skyline POI under AuthSkySP. As a result, the gap of the CSP-user communication cost between AuthSkySP and *Baseline* continuously grows as the number of POIs further increases. Similar to Fig. 5(c), the user computation cost for verifying an LBSQ result exponentially increases as the number of POIs increases under *Baseline*. In contrast, user computation cost under AuthSkySP is significantly lower than *Baseline* and relatively insensitive to the increase in the number of POIs.

Impact of Number of Road Segments. Figs. 6(a) to 6(d) compare the performance of AuthSkySP and *Baseline* with the number of road segments varying from 1,000 to 8,035. Similar to what we have seen from Figs. 5(a) to 5(d), AuthSkySP incurs the higher LBSP's computation cost and LBSP-CSP communication cost but the lower CSP-user communication cost and user computation cost. As we mentioned earlier, data



(a) LBSP computation cost (b) LBSP-CSP comm. cost (c) CSP-user comm. cost (d) User computation cost

Fig. 6: The impact of number of road segments on AuthSkySP from synthetic datasets.

preprocessing is a one-time process while the CSP needs to process many LBSQs. It is thus favorable to significantly reduce CSP-user communication cost and user computation cost at a moderate sacrifice of LBSP computation cost and LBSP-CSP communication cost. These results further confirm the significant advantages of AuthSkySP.

VI. Conclusion

This paper has presented the design and evaluation of AuthSkySP, a novel scheme for authenticating outsourced LBSQ under the shortest path distance. AuthSkySP explores several unique properties of LBSQs to enable a user to verify both the integrity and completeness of any LBSQ result returned by an untrusted CSP. We have confirmed the efficiency of AuthSkySP via detailed experimental studies using both real and synthetic datasets.

Acknowledgment

This work was supported in part by the US National Science Foundation under grants CNS-2245689 (CRII), CNS-1933047, CNS-1651954 (CAREER), and a 2022 Meta Research Award for Privacy-Enhancing Technologies.

References

[1] M. Goncalves, D. Torres, and G. Perera, "Making recommendations using location-based skyline queries," in *IEEE DEXA'12*, Vienna, Austria, Sep. 2012, pp. 111–115.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, April 2010.

[3] X. Lin, J. Xu, and H. Hu, "Authentication of location-based skyline queries," in *ACM CIKM*, Glasgow, Scotland, UK, Oct. 2011.

[4] X. Lin, J. Xu, and J. Gu, "Continuous skyline queries with integrity assurance in outsourced spatial databases," in *Web-Age Information Management*, Harbin, China, Aug. 2012, pp. 114–126.

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

[6] W. Chen, M. Liu, R. Zhang, Y. Zhang, and S. Liu, "Secure outsourced skyline query processing via untrusted cloud service providers," in *IEEE INFOCOM*, San Francisco, CA, April 2016, pp. 1–9.

[7] X. Zhu, J. Wu, W. Chang, G. Wang, and Q. Liu, "Authentication of skyline query over road networks," in *SpaCCS*, Melbourne, NSW, Australia, Dec. 2018, pp. 72–83.

[8] 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, June 2009.

[9] D. Yung, E. Lo, and M. L. Yiu, "Authentication of moving range queries," in *ACM CIKM*, Maui, Hawaii, USA, Oct 2012, pp. 1372–1381.

[10] Y. Hu, X. Yao, R. Zhang, and Y. Zhang, "Freshness authentication for outsourced multi-version key-value stores," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 3, pp. 2071–2084, 2023.

[11] Q. Chen, H. Hu, and J. Xu, "Authenticating top-k queries in location-based services with confidentiality," *Proceedings of the VLDB Endowment*, vol. 7, no. 1, pp. 49–60, Sep. 2013.

[12] R. Zhang, Y. Zhang, and C. Zhang, "Secure top-k query processing via untrusted location-based service providers," in *IEEE INFOCOM*, 2012, pp. 1170–1178.

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

[14] X. Yu, Y. Hu, R. Zhang, Z. Yan, and Y. Zhang, "Secure outsourced top-k selection queries against untrusted cloud service providers," in *IEEE/ACM IWQOS*, Aug. 2021, pp. 1–10.

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

[16] Y. Zhang, D. Genkin, J. Katz, D. Papadopoulos, and C. Papamanthou, "vsql: Verifying arbitrary sql queries over dynamic outsourced databases," in *IEEE Symposium on Security and Privacy*, San Jose, CA, May 2017.

[17] M. Haxen, M. Raeburn, P. Afshani, and P. Karras, "Centerpoint query authentication," in *ACM CIKM*, 2021, pp. 3083–3087.

[18] X. Yao, R. Zhang, D. Huang, and Y. Zhang, "Verifiable query processing over outsourced social graph," *IEEE/ACM Transactions on Networking*, vol. 29, no. 5, pp. 2313–2326, 2021.

[19] S. Borzsony, D. Kossmann, and K. Stocker, "The skyline operator," in *IEEE ICDE*, Heidelberg, Germany, April 2001, pp. 421–430.

[20] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with presorting," in *IEEE ICDE*, Bangalore, India, March 2003, pp. 717–719.

[21] J. Zhang, W. Wang, X. Jiang, W. Ku, and H. Lu, "An mbr-oriented approach for efficient skyline query processing," in *IEEE ICDE*, Macao, China, April 2019, pp. 806–817.

[22] M. Tang, Y. Yu, W. G. Aref, Q. M. Malluhi, and M. Ouzzani, "Efficient parallel skyline query processing for high-dimensional data," *IEEE TKDE*, vol. 30, no. 10, pp. 1838–1851, Oct. 2018.

[23] H. Wijayanto, W. Wang, W.-S. Ku, and A. L. Chen, "Lshape partitioning: Parallel skyline query processing using mapreduce," *IEEE TKDE*, vol. 34, no. 7, pp. 3363–3376, 2022.

[24] J. E. Gonzalez, Y. Low, H. Gu, D. Bickson, and C. Guestrin, "Powergraph: Distributed graph-parallel computation on natural graphs," in *USENIX OSDI'12*, Hollywood, CA, Oct. 2012, pp. 17–30.

[25] F. Bourse, M. Lelarge, and M. Vojnovic, "Balanced graph edge partition," in *ACM KDD*, New York, New York, Aug. 2014, pp. 1456–1465.

[26] L. Li, R. Geda, A. B. Hayes, Y. Chen, P. Chaudhari, E. Z. Zhang, and M. Szegedy, "A simple yet effective balanced edge partition model for parallel computing," *SIGMETRICS Perform. Eval. Rev.*, vol. 45, no. 1, pp. 6–6, June 2017.

[27] "Yelp fusion api," https://www.yelp.com/developers/documentation/v3/business_search.

[28] "Overpass api," https://wiki.openstreetmap.org/wiki/Overpass_API.

[29] "Socket," https://docs.python.org/3/library/socket.html#module-socket.