Secure Spatial Top-k Query Processing via Untrusted Location-based Service Providers

Rui Zhang, *Member, IEEE,* Jinchao Sun, *Student Member, IEEE,* Yanchao Zhang, *Senior Member, IEEE,* and Chi Zhang, *Member, IEEE,*

Abstract—This paper considers a novel distributed system for collaborative location-based information generation and sharing which become increasingly popular due to the explosive growth of Internet-capable and location-aware mobile devices. The system consists of a data collector, data contributors, location-based service providers (LBSPs), and system users. The data collector gathers reviews about points-of-interest (POIs) from data contributors, while LBSPs purchase POI data sets from the data collector and allow users to perform spatial top-k queries which ask for the POIs in a certain region and with the highest k ratings for an interested POI attribute. In practice, LBSPs are untrusted and may return fake query results for various bad motives, e.g., in favor of POIs willing to pay. This paper presents three novel schemes for users to detect fake spatial snapshot and moving top-k query results as an effort to foster the practical deployment and use of the proposed system. The efficacy and efficiency of our schemes are thoroughly analyzed and evaluated.

Index Terms—Spatial top-k query, location-based service, security.

1 Introduction

The explosive growth of Internet-capable and location-aware mobile devices and the surge in social network usage are fostering collaborative information generation and sharing on an unprecedented scale. In particular, IDC believes that total worldwide smartphone shipments will reach 659.8 million units in 2012 and will grow at a CAGR of 18.6% until 2016.[1] Almost all smartphones have cellular / WiFi Internet access and can always acquire their precise locations via pre-installed positioning software. Also owing to the growing popularity of social networks, it is more and more convenient and motivating for mobile users to share with others their experience with all kinds of *points of interests* (POIs) such as bars, restaurants, grocery stores, coffee shops, and hotels. Meanwhile, it becomes commonplace for people to perform various spatial POI queries at online location-based service providers (LBSPs) such as Google and Yelp. As probably the most familiar type of spatial queries, a spatial (or location-based) top-k query asks for the POIs in a certain region and with the highest k ratings for a given POI attribute. For example, one may search for the best 10 Italian restaurants with the highest food ratings within five miles of his current location. This paper focuses on spatial top-k queries, and the term "spatial" will be omitted hereafter for brevity.

We observe two essential drawbacks with current top-k query services. First, individual LBSPs often have very small data sets comprising POI reviews. This would largely affect the usefulness and eventually hinder the more prevalent use of spatial top-k query services. Continue with the restaurant example. The data sets at individual LBSPs may not cover all the Italian restaurants within a search radius. Additionally, the same restaurant may receive diverse ratings at different LBSPs, so users may get confused by very different query results from different LBSPs for the same query. A leading reason for limited data sets at individual LBSPs is that people tend to leave reviews for the same POI at one or at most only a few LBSPs's websites which they often visit. Second, LBSPs may modify their data sets by deleting some reviews or adding fake reviews and return tailored query results in favor of the restaurants that are willing to pay or against those that refuse to pay. [2] Even if LBSPs are not malicious, they may return unfaithful query results under the influence of various attacks such as the Sybil attack [2], [3] whereby the same attacker can submit many fake reviews for the same POI. In either case, top-k query users may be misled by the query results to make unwise decisions.

A promising solution to the above two issues is to introduce some trusted data collectors as the central hubs for collecting POI reviews. In particular, data collectors can offer various incentives, such as free coffee coupons, for stimulating review submissions and then profit by selling the review data to individual LBSPs. Instead of submitting POI reviews to individual LBSPs, people (called *data contributors*) can now submit them

The preliminary version of this paper appeared in IEEE INFOCOM'12 [1].

R. Zhang is with the Department of Electrical Engineering, University of Hawaii, Honolulu, HI 96822 (email: ruizhang@hawaii.edu).
J. Sun and Y. Zhang are with the School of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe, AZ 85287 (email: {jcsun, yczhang}@asu.edu).
C. Zhang is with the School of Information Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China (email: chizhang@ustc.edu.cn).

1. http: / / www.idc.com / getdoc.jsp?containerId=233553

2. Similar misbehavior has been widely reported for the web-search industry.

to a few data collectors to earn rewards. The data sets maintained by data collectors can thus be considered the union of the small data sets currently at individual LBSPs. Such centralized data collection also makes it much easier and feasible for data collectors to employ sophisticated defenses, such as [2], [3], to filter out fake reviews from malicious entities like Sybil attackers. Data collectors can be either new service providers or more preferably existing ones with a large user base, such as Google, Yahoo, Facebook, Twitter, and MSN. Many of these service providers (e.g., Google) have already been collecting reviews from their users and offered open APIs for exporting selected data from their systems. We postulate that they may act as location-based data collectors and sellers if sound techniques and business models are in place.

The above system model is also highly beneficial for LBSPs. In particular, they no longer need struggle to solicit faithful user reviews, which is often a daunting task especially for small/medium-scale LBSPs. Instead, they can focus their limited resources on developing appealing functionalities (such as driving directions and aerial photos) combined with the high-quality review data purchased from data collectors. The query results they can provide will be much more trustworthy, which would in turn help them attract more and more users. This system model thus can greatly help lower the entrance bar for new LBSPs without sufficient funding and thus foster the prosperity of location-based services and applications.

A main challenge for realizing the appealing system above is how to deal with untrusted and possibly malicious LBSPs. Specifically, malicious LBSPs may still modify the data sets from data collectors and provide biased top-k query results in favor of POIs willing to pay. Even worse, they may falsely claim generating query results based on the review data from trusted data collectors which they actually did not purchase. Moreover, non-malicious LBSPs may be compromised to return fake top-k query results.

In this paper, we propose three novel schemes to tackle the above challenge for fostering the practical deployment and wide use of the envisioned system. The key idea of our schemes is that the data collector precomputes and authenticates some auxiliary information (called *authenticated hints*) about its data set, which will be sold along with its data set to LBSPs. To faithfully answer a top-k query, a LBSP need return the correct top-k POI data records as well as proper *authenticity* and *correctness* proofs constructed from authenticated hints. The authenticity proof allows the query user to confirm that the query result only consists of authentic data records from the trusted data collector's data set, and the correctness proof enables the user to verify that the returned top-k POIs are the true ones satisfying the query. The first two schemes both target snapshot top-k queries but differ in how authenticated hints are precomputed and how authenticity and correctness proofs are constructed and verified as well as the related communication and computation overhead. The third scheme, built upon the first scheme, realizes efficient and verifiable moving top-k queries. The efficacy and efficiency of our schemes are thoroughly analyzed and evaluated through detailed simulation studies.

The rest of this paper is organized as follows. Section 2 discusses the related work, and Section 3 gives the problem formulation. Section 4 presents two schemes for secure snapshot top-k query processing, which are extended for secure moving top-k query processing in Section 5. All the schemes are then thoroughly analyzed in Section 6 and evaluated via detailed simulations in Section 7. This paper is finally concluded in Section 8.

2 Related Work

Our work is most related to data outsourcing [4], for which we can only review representative schemes due to space constraints. The framework of data outsourcing was first introduced in [4], in which a data owner outsources its data to a third-party service provider who is responsible for answering the data queries from either the data owner or other users. In general, there are two security concerns in data outsourcing: data privacy and query integrity [5].

Ensuring data privacy requires the data owner to outsource encrypted data to the service provider, and efficient techniques are needed to support querying encrypted data. A bucketization approach was proposed in [6], [7] to enable efficient range queries over encrypted data, which was recently improved in [8]. Shi *et al.* presented novel methods for multi-dimensional range queries over encrypted data [9]. Some most recent proposals aim at secure ranked keyword search [10], [11] or fine-grained access control [12] over encrypted data. This line of work is orthogonal to our work, as we focus on publicly accessible location-based data without need for privacy protection.

Another line of research has been devoted to ensure query integrity, i.e., that a query result was indeed generated from the outsourced data (the authenticity requirement) and contains all the data satisfying the query (the correctness requirement). In these schemes, the data owner outsources both its data and also its signatures over the data 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. Many techniques were proposed for signature and VO generations, such as those [13]–[15] based on signature chaining and those [5], [16]–[18] based on the Merkle hash tree [19] or its variants. None of these schemes consider spatial top-k queries and are directly applicable to our intended scenario, as spatial top-k queries exhibit unique feature in that whether a POI is among the top-k is jointly determined by all the other POIs in the query region and that the query region cannot be predicted in practice.

Secure remote query processing in tiered sensor networks [20]–[24] is also loosely related to our work here. These schemes assume that some master nodes are in charge of storing data from regular sensor nodes and answering the queries from the remote network owner. Various techniques were proposed in [20]–[23] to ensure data privacy against master nodes and also enable the network owner to verify range-query integrity. Moreover, Zhang *et al.* [24] proposed efficient techniques for the network owner to validate the integrity of top-k queries. These schemes cannot be adapted to address our problem in this paper.

3 Problem Formulation

3.1 System Model

We assume a distributed system comprising a data collector, data contributors, LBSPs, and top-k query users. Data contributors are common people who submit POI reviews to the data collector's website. The data collector normally need offer some incentives, such as FourSquare's badges, to stimulate review submissions and also employ necessary countermeasures such as [2], [3] to filter out fake reviews from malicious data contributors. The data collector sells aggregated POI reviews in the form of a location-based data set to individual LBSPs. Every LBSP operates a website for users to perform top-k queries over the purchased data set and may add some appealing functionalities to the query result such as street maps and photos. In addition, although there might be multiple data collectors with each selling data to a number of LBSPs, we hereafter focus on one pair of data collector and LBSP for the purpose of this paper.

The data set is classified according to POI categories, such as restaurants, bars, and coffee shops, and it contains a unique record for every POI in every category. As a result, POIs falling into multiple categories (e.g., both a restaurant and bar) have one record for every affiliated category. This paper focuses on top-k queries involving a single category, which are most commonly used in practice, and the extension of our schemes to involve multiple categories is part of our future work. In particular, our discussion will focus on one POI category whose total data records form a set $\mathcal{D}$. For simplicity, we assume that the category has one numerical attribute taking values from a given range. For instance, if restaurant is the category under consideration, there may be $\lambda = 4$ attributes including *food*, *price*, *service*, and *hygiene*, with each rated on a scale of 1 to 10.

The geographic area covered by the data collector is partitioned into $M \geq 1$ equally-sized non-overlapping zones. For every zone i, let n_i denote the number of POIs, and $\mathsf{POI}_{i,j}$ and $D_{i,j}$ denote the jth POI and its corresponding data record, respectively. It follows that $\mathcal{D} = \bigcup_{i=1}^{M} \mathcal{D}_i$, $\mathcal{D}_i = \bigcup_{j=1}^{n_i} D_{i,j}$, and $\mathcal{D}_i \bigcap \mathcal{D}_j = \phi$ for all $i \neq j$. Also note that $\mathcal{D}_i$ can be empty for some $i \in [1, M]$, meaning that there is no POI in zone i that has been reviewed.

To illustrate the content of a data record, assume that the data collector got reviews about $\mathsf{POI}_{i,j}$ from $n_{i,j}$ data contributors. Every review includes a rating on every attribute and possibly text comments. We also let $A_{i,j,q}$ denote the rating for attribute q averaged over $n_{i,j}$ individual ratings. The data record $d_{i,j}$ for $\mathsf{POI}_{i,j}$ includes its name, location $l_{i,j}$, $\{A_{i,j,q}\}_{q=1}^{\lambda}$, $n_{i,j}$ reviews, and possibly other information.

3.2 Problem Statement

We consider two types of top-k queries in this paper. A *snapshot top-k query* includes the interested POI *category*, a *query region* $\mathbf{R}$, and an integer $k \geq 1$. As an example, the POI category and attribute can be *restaurant* and *food*, respectively. The query region can be in multiple formats. For instance, the user can specify a GPS location or street address along with a search radius, and he may also select multiple zones on a map provided by the LBSP. An authentic and correct query result should include the records for k POIs in the specified category of the data collector's true data set, all of which are in the query region $\mathbf{R}$, have the attribute-q rating among the highest k, and are ordered with respect to the attribute-q rating in the descending order. For brevity, we will refer to the POIs that are both authentic and correct as top-k POIs hereforth. In contrast, a *moving top-k query* can be viewed as the continuous version of snapshot top-k queries, whereby the user is interested in the top-k POIs in a moving region $\mathbf{R}$ defined with respect to the user's current location.

We assume that the data collector is trusted, while the LBSP is untrusted. In particular, the LBSP may alter the query result in favor of the POIs willing to pay, to which similar misbehavior has been widely reported in web-search industry. For example, the LBSP may replace some true top-k POIs with others not among the top k or even not in the data collector's data set, and it may also modify some data records by adding more good reviews and deleting bad ones. In addition, a LBSP good in nature may be compromised by attackers to forge query results as well.

Our design objective is to enable the user to verify the authenticity and correctness of the query result returned by the LBSP. The query result is considered authentic if all its k POI records exist in the data collector's data set and have not been tampered with, and it is called correct if it contains the true top-k POI records in the query region.

We summarize the major notations used throughout the paper in Table 1.

4 Secure Snapshot Top-k Query Processing

In this section, we illustrate our two schemes which both comprises three phases and differ in operation details. In the *data-preprocessing* phase, the data collector uses

TABLE 1: Default Simulation Settings

Notation	Meaning
M	The number of zones.
$\mathcal{D}_i$	All POI records in zone i.
$D_{i,j}$	The jth POI record in zone i.
$n_{i,j}$	The number of POI records in zone i.
$l_{i,j}$	The location of POI $D_{i,j}$.
$A_{i,j,q}$	The attribute-q rating of $D_{i,j}$.
$\phi_{i,j}$	The index of $D_{i,j}$.
$h_{i,j}$	The hash value corresponding to $D_{i,j}$.
$\mathcal{I}$	The set of candidate zones.
$\mathcal{I}_{i,j}$	The indexes of all the other zones in M_e i with their largest attribute-q ratings in $(A'_{i,j-1,q}, A'_{i,j,q})$.
$\mathcal{T}_i$	The auxiliary set of zone i.
$\mathbf{R}_a$	The query region of the ath snapshot query.
$\mathbf{P}_a$	The progression region or the area in $\mathbf{R}_{a+1} \setminus \mathbf{R}_a$.
$\mathbf{V}_{a\to b}$	The verification region $\bigcup_{x=a}^{b-1} \mathbf{P}_x$.
$\mathbf{S}_{a\to b}$	The suspicion region $\bigcup_{x=a}^{b-2} \mathbf{P}_x$.
kPOI$_a$	The top-k POIs for the ath snapshot query.
$\triangle t$	The delay between two consecutive snapshot queries.

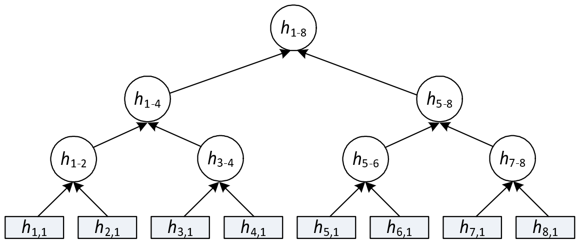


Fig. 1: An example of constructing the Merkle hash tree over $\{h_{i,1}\}_{i=1}^{8}$.

cryptographic methods to create authenticated hints over its data set. In the subsequent *query-processing* phase, the LBSP answers a top-k query by returning the query result as well as the authenticity and correctness proofs to the query user. In the final *verification* phase, the user verifies authenticity and correctness proofs. For ease of presentation, we shall temporarily assume that no two POIs have the same rating for any attribute $q \in [1, \lambda]$, which implies that there is one and only one correct result for any top-k query. We will also temporarily assume that there are always at least k POIs in the query region so that the query result contains exactly k POI records for arbitrary k. We discuss how to relax these two assumptions in the supplemental file.

4.1 Scheme 1

In Scheme 1, authenticated hints are created by chaining ordered POIs in every zone via cryptographic hash functions and then tieing the POIs in different zones via a Merkle hash tree [25]. The details about constructing and using authenticated hints are as follows.

4.1.1 Data Preprocessing

The data collector preprocesses its data set $\mathcal{D} = \bigcup_{i=1}^{M} \mathcal{D}_i$ before selling it to LBSPs, where M denotes the total number of zones. Recall that $\mathcal{D}_i = \bigcup_{j=1}^{n_i} D_{i,j}$, where $D_{i,j}$ denotes the record of POI$_{i,j}$ and includes its name, location $l_{i,j}$, received ratings $\{A_{i,j,q}\}_{q=1}^{\lambda}$ for q attributes, individual reviews, and some other information. The data collector performs the following operations for every attribute $q \in [1, \lambda]$.

First, for each $i \in [1, M]$, the data collector sorts $\mathcal{D}_i$ according to the attribute-q rating to generate an orderer list $\mathcal{D}'_i = \langle D'_{i,1}, D'_{i,2}, \ldots, D'_{i,n_i}\rangle$ such that $A'_{i,1,q} > A'_{i,2,q} > \cdots > A'_{i,n_i,q}$. It then computes an *index* for every $D'_{i,j} \in \mathcal{D}'_i$ as

$$\phi_{i,j} = \langle l'_{i,j}, A'_{i,j,q}, H(D'_{i,j})\rangle \ , \tag{1}$$

where $l'_{i,j}$ denotes the location of $D'_{i,j}$, and $H(\cdot)$ denotes a cryptographic hash function. Note that $\phi_{i,j}$ contains sufficient information for a user to determine whether $D'_{i,j}$ satisfies a top-k query, which will be further illustrated shortly.

Second, the data collector chains $\{\phi_{i,j}\}_{j=1}^{n_i}$ using a cryptographic hash function to ensure the correct order among them. In particular, recall that every attribute rating is on a given range $[A_{\min}, A_{\max}]$, say [1, 10]. Let χ denote a publicly known number smaller than $A_{\min}$. The data collector recursively computes a sequence of hash values as follows,

$$h_{i,j} = \begin{cases} H(\chi) & j = n_i + 1, \\ H(h_{i,j+1}||\phi_{i,j}) & 1 \le j \le n_i, \end{cases} \tag{2}$$

where $||$ denotes concatenation and $n_i \ge 0$. If $n_i = 0$, we let $\phi_{i,1} = h_{i,1} = H(\chi)$. Note that the data collector can also ensure the correct order among $\{\phi_{i,j}\}_{j=1}^{n_i}$ by building a Merkle hash tree on top of them, but doing so will incur higher communication and computation overhead during query processing and query-result verification.

Finally, the data collector builds a Merkle hash tree over $\{h_{i,1}\}_{i=1}^{M}$ to enable efficient authentication of query results. More specifically, assuming that $M = 2^d$ for some integer d, the data collector builds a binary tree of depth d, in which every leaf node corresponds to one of $\{h_{i,1}\}_{i=1}^{M}$, and every non-leaf node is computed as the hash of the concatenation of its immediate children nodes. We also define an *auxiliary set* $\mathcal{T}_i$ as the set of non-leaf nodes required along with any leaf node $h_{i,1}$ to compute the Merkle root hash. An example for $M = 8$ is shown in Fig. 1, in which $h_{1-2} = H(h_{1,1}||h_{2,1})$, $h_{3-4} = H(h_{3,1}||h_{4,1})$, $h_{5-6} = H(h_{5,1}||h_{6,1})$, $h_{7-8} = H(h_{7,1}||h_{8,1})$, $h_{1-4} = H(h_{1-2}||h_{3-4})$, $h_{5-8} = H(h_{5-6}||h_{7-8})$, and $h_{1-8} = H(h_{1-4}||h_{5-8})$. If $h_{3,1}$ is the given leaf node, we have $\mathcal{T}_3 = \{h_{4,1}, h_{1-2}, h_{5-8}\}$, as the root $h_{1-8} = H(H(h_{1-2}||H(h_{3,1}||h_{4,1}))||h_{5-8})$. Note that if M is not a power of two, some dummy leaf nodes need be introduced for constructing the Merkle hash tree.

Since there are totally λ attributes, every POI$_{i,j}$ has λ indexes, based on which the data collector builds a separate Merkle hash tree for every attribute and signs every root using its private key. In addition, the data collector need perform the above operations separately for the data set of every POI category.

4.1.2 Query Processing

The LBSP purchases the data sets of interested POI categories from the data collector. For every POI category selected by the LBSP, the data collector returns the original data set $\mathcal{D}$, the signatures on λ Merkle root hashes, and all the intermediate results for constructing the Merkle hash tree. Alternatively, the data collector can just return the first two pieces of information and let the LBSP itself perform a one-time process to derive the third piece in the same way as the date collector.

Now we illustrate the processing of a snapshot top-k query, including the desired POI category, the interested attribute $q \in [1, \lambda]$ for ranking POIs, the query region $\mathbf{R}$, and k. We denote the k POIs in $\mathbf{R}$ with the highest k attribute-q ratings by kPOI, among which the lowest attribute-q rating is denoted by γ. In addition, we call each zone either completely or partially covered by the query region a *candidate zone*. A correct and authentic query result needs to satisfy two conditions. The correctness condition requires the query result to contain at least the following information: (1) the complete data records for kPOI; (2) the data indexes (much shorter than data records) for all the POIs in each candidate zone but not in $\mathbf{R}$ whose attribute-q rating is larger than γ; and (3) some additional information to prove that the query result includes either the data record or index of every POI in every candidate zone with attribute-q rating not smaller than γ. In addition, the authenticity condition requires that the query result include the auxiliary set for every candidate zone for the calculation and verification of the qth Merkle root hash.

To satisfy the correctness condition, the LBSP first searches $\{\mathcal{D}'_i\}_{i=1}^{M}$ to locate kPOI and then determine the lowest attribute-q rating γ. Next, the LBSP determines the set of candidate zones, denoted by $\mathcal{I} \subseteq \{1, \ldots, M\}$. Let τ_i the number of POIs in zone i with attribute-q ratings higher than γ. Apparently, we have $n_i \geq \tau_i, \forall i \in \mathcal{I}$. It follows that $\sum_{i \in \mathcal{I}} \tau_i \geq k$, which holds because any candidate zone that partially overlaps with $\mathbf{R}$ may have some POIs outside $\mathbf{R}$ but with attribute-q ratings higher than γ. We further define

$$X_{i,j} = \begin{cases} D'_{i,j} & \text{if } l'_{i,j} \in \mathbf{R}, \\ \phi_{i,j} & \text{o.w.}, \end{cases} \tag{3}$$

for all $i \in \mathcal{I}, j \in [1, n_i]$. In other words, $X_{i,j}$ equals $D'_{i,j}$ if the POI is in $\mathbf{R}$ and a shorter index otherwise. The LBSP returns the following information $\mathcal{S}_i$ for each candidate zone $i \in \mathcal{I}$ in the query result to enable correctness verification.

- Case 1: if $n_i = 0$, $\mathcal{S}_i = \langle i \rangle$.
- Case 2: if $n_i = 1$, $\mathcal{S}_i = \langle i, X_{i,1} \rangle$.
- Case 3: if $n_i \geq 2$ and $\tau_i = 0$, $\mathcal{S}_i = \langle i, \phi_{i,1}, h_{i,2} \rangle$.
- Case 4: if $n_i \geq 2$ and $n_i > \tau_i \geq 1$,

$$\mathcal{S}_i = \langle i, X_{i,1}, \ldots, X_{i,\tau_i}, \phi_{i,\tau_i+1}, h_{i,\tau_i+2} \rangle \ .$$

- Case 5: if $n_i = \tau_i \geq 2$, $\mathcal{S}_i = \langle i, X_{i,1}, \ldots, X_{i,\tau_i} \rangle$.

Note that the last two fields in both Case 3 and Case 4 correspond to the POI in zone i with the largest attribute-q rating smaller than γ. Since the POIs in zone i have been ranked and chained together under cryptographic hash functions during data preprocessing, the inclusion of such fields is necessary for proving that every POI in every candidate zone whose attribute-q rating not smaller than γ has been covered in the query result in the form of either a data record or index. Such information has been implicitly covered in the other three cases as well. In addition, the LBSP returns $\mathcal{T} = \bigcup_{i \in \mathcal{I}} \mathcal{T}_i$ and the data collector's signature on the qth Merkle root hash to enable authenticity verification.

4.1.3 Query-result Verification

Now we discuss how the user verifies the authenticity and correctness of the query result, which can be done via a small plug-in developed by the data collector and installed on his web browser. The security analysis of Scheme 1 is postponed to Section 6.

For authenticity verification, the user checks if every piece of information in the query result can lead to the same Merkle root hash matching the data collector's signature. Specifically, the user first determines which of the above five cases $\mathcal{S}_i$ ($\forall i \in \mathcal{I}$) belongs to based on its message format. He then derives the indexes for all related POIs in $\{\mathcal{S}_i\}_{i \in \mathcal{I}}$. Note that the indexes of the POIs outside $\mathbf{R}$ are explicitly included in $\{\mathcal{S}_i\}_{i \in \mathcal{I}}$, while those of the POIs in $\mathbf{R}$ can be computed from their corresponding data records in $\{\mathcal{S}_i\}_{i \in \mathcal{I}}$. Subsequently, the user computes $h_{i,1}$ for each $i \in \mathcal{I}$ according to Eq. (2). Since the auxiliary information $\mathcal{T}_i$ for $h_{i,1}$ is also in the query result, the user further uses $h_{i,1}$ and $\mathcal{T}_i$ to compute the Merkle root hash. If the query result is authentic, the user can derive the same root hash for each $i \in \mathcal{I}$, in which case he further verifies whether the data collector's signature in the query result is a valid signature on the derived root hash. If so, he considers the query result authentic.

To perform correctness verification, the user first checks if zones $\mathcal{I}$ encloses the query region $\mathbf{R}$. If so, he proceeds with the following verifications in accordance with the aforementioned correctness condition used in query processing.

1. There are exactly k data records in the query result with POI locations all in $\mathbf{R}$, which correspond to the top-k POIs (i.e., kPOI) in $\mathbf{R}$. If so, the user locates the lowest attribute-k rating γ.
2. None of the POIs for which the data indexes (instead of data records) are returned satisfy the query. In particular, for each index $\phi_{i,j}$ ($i \in \mathcal{I}$), at least one of the following conditions does not hold.
   - $\phi_{i,j}$ contains a location $l'_{i,j} \in \mathbf{R}$.
   - $\phi_{i,j}$ contains an attribute$-q$ rating $A'_{i,j,q} > \gamma$.

In addition, since the query result is authentic, it must include either the data record or index for every POI in every candidate zone whose attribute-q rating is not smaller than γ. Therefore, the user considers the query result correct if the above two verifications succeed.

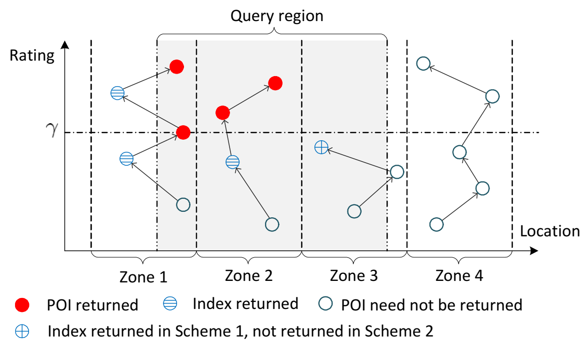


Fig. 2: An example for Scheme 1, where $M = 4$, $k = 4$, and the dots in zone i correspond to POI records $D'_{i,1}$ to $D'_{i,4}$ from top to bottom.

4.1.4 An Example

To better illustrate Scheme 1, we show an example in Fig. 2 with $M = 4$ zones, where we assume one-dimensional POI locations for simplicity, i.e., that all POIs are distributed on a straight line, and all the shown POIs have been ordered according to the attribute-q rating (q is omitted from subscripts for brevity). Suppose that the user queries the top-4 POIs in the query region that completely covers zone 2 and partially overlaps with zones 1 and 3. It follows that $\mathcal{I} = \{1, 2, 3\}$, and τ_1, τ_2, τ_3 are 3, 2, 0, respectively. For zone 1, there is one POI outside the query region with a rating higher than γ, so we have $\mathcal{S}_1 = \langle 1, D'_{1,1}, \phi_{1,2}, D'_{1,3}, \phi_{1,4}, h_{1,5}\rangle$. Similarly, we have $\mathcal{S}_2 = \langle 2, D'_{2,1}, D'_{2,2}, \phi_{2,3}, h_{2,4}\rangle$ for zone 2 and $\mathcal{S}_3 = \langle 3, \phi_{3,1}, h_{3,2}\rangle$ for zone 3. The query result includes $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$, the auxiliary indexes $\{\mathcal{T}_i\}_{i=1}^{3}$, and the data collector's signature on h_{1-4} which is the root of the Merkle hash tree with depth $d = 2$. Based on $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$, the user can derive $h_{1,1}, h_{2,1}$, and $h_{3,1}$, respectively. He can further compute three Merkle root hashes using $h_{1,1}$ and $\mathcal{T}_1$, $h_{2,1}$ and $\mathcal{T}_2$, and $h_{3,1}$ and $\mathcal{T}_3$, respectively. If the three root hashes are equal and match the data collector's signature, the user considers the query result authentic. If the query result can also pass the aforementioned three correctness verifications, the user considers the query result both authentic and correct.

4.2 Scheme 2

Scheme 1 requires the LBSP to return some information for every candidate zone even if it has no top-k POI satisfying the query. This may incur significant communication overhead for a large query region. Given this observation, we propose Scheme 2 which works by embedding some information among nearby zones to dramatically reduce the amount of information returned to the user.

The basic idea of Scheme 2 can be better illustrated using a simple example. Assume that zones i and j are two candidate zones. But neither contains a top-k POI. Under Scheme 1, the LBSP need return both $\langle i, \phi_{i,1}, h_{i,2}, \mathcal{T}_i\rangle$ and $\langle j, \phi_{j,1}, h_{j,2}, \mathcal{T}_j\rangle$ to prove that no POI in zones i or j satisfies the query. In contrast, if we could consider zones i and j as one virtual zone, the LBSP only need return $\langle x, \phi_{x,1}, h_{x,2}, \mathcal{T}_x\rangle$, where $x = i$ if the largest attribute-q rating in zone j is smaller than that in zone i, and $x = j$ otherwise. The amount of information returned to the user can thus be reduced.

4.2.1 Data Preprocessing

To implement the basic idea exemplified above, the data collector binds to every POI data index some additional information about the POIs in adjacent zones. In particular, the data collector partitions the original M zones into non-overlapping *macro zones*, each consisting of m nearby zones, where m is a public system parameter. Assuming that M is divisible by m, we let $\mathcal{M}_e$ denote the set of zones composing the macro zone $e \in [1, M/m]$.

Consider a macro zone e as an example. As in Scheme 1, the data collector first sorts $\mathcal{D}_i$ for every zone $i \in \mathcal{M}_e$ according to the descending order of the attribute-q rating to generate an orderer list $\mathcal{D}'_i = \langle D'_{i,1}, D'_{i,2}, \ldots, D'_{i,n_i}\rangle$. Let $A'_{j,0,q} = \overline{\chi}$ and $A'_{j,n_i+1,q} = \chi$ denote two public values larger than the largest possible attribute rating and smaller than the smallest possible attribute rating, respectively. The data collector further generates $\{\mathcal{I}_{i,j}\}_{j=1}^{n_i+1}$, where $\mathcal{I}_{i,j} = \{\langle s, A'_{s,1,q}\rangle | s \in \mathcal{M}_e \setminus \{i\}\}$ with $A'_{s,1,q} \in (A'_{i,j-1,q}, A'_{i,j,q})$. In other words, $\mathcal{I}_{i,j}$ comprises all the other zones in $\mathcal{M}_e \setminus \{i\}$ and their largest attribute-q ratings in $(A'_{i,j-1,q}, A'_{i,j,q})$. Apparently, we have $|\bigcup_{j=1}^{n_i+1} \mathcal{I}_{i,j}| = |\mathcal{M}_e \setminus \{i\}| = m - 1$. The data collector then computes an index as

$$\phi_{i,j} = \langle l_{i,j}, \mathcal{I}_{i,j}, A'_{i,j,q}, H(\mathcal{I}_{i,j}||D'_{i,j})\rangle \tag{4}$$

for all $j \in [1, n_i]$ and chains $\{\phi_{i,j}\}_{j=1}^{n_i}$ according to Eq. (2). Finally, it builds a Merkle hash tree over $\{h_{i,1}\}_{i=1}^{M}$ and signs the root as in Scheme 1. The essential difference in data preprocessing between Schemes 1 and 2 thus lies in the construction of POI indexes.

As in Scheme 1, the data collector builds a separate Merkle hash tree for every attribute $q \in [1, \lambda]$ in every POI category and signs every Merkle root hash using its private key.

4.2.2 Query Processing

The LBSP purchases the original data set $\mathcal{D}$, the signatures on λ Merkle root hashes, and all the intermediate results for constructing the Merkle hash tree of every interested POI category from the data collector.

After receiving a top-k query, the LBSP first determines the top-k POIs (i.e., kPOI) in the query region $\mathbf{R}$ and also the set of candidate zones $\mathcal{I} \subseteq \{1, \ldots, M\}$. The LBSP then determines the lowest attribute-q rating γ in kPOI and τ_i as the number of POIs in zone $i \in \mathcal{I}$ with attribute-q ratings not smaller than γ. Next, the LBSP defines

$$Y_{i,j} = \begin{cases} D'_{i,j}||\mathcal{I}_{i,j} & \text{if } l'_{i,j} \in \mathbf{R}, \\ \phi_{i,j} & \text{o.w.,} \end{cases} \tag{5}$$

for all $i \in \mathcal{I}, j \in [1, n_i]$, where $\mathcal{I}_{i,j}$ is as defined in the data-preprocessing phase. The query result includes the following information $\mathcal{S}_i$ for each zone $i \in \mathcal{I}$ with $\tau_i > 0$.

- Case 1: if $n_i = \tau_i \geq 1$, $\mathcal{S}_i = \langle i, Y_{i,1}, \ldots, Y_{i,\tau_i} \rangle$.
- Case 2: if $n_i \geq 2$ and $n_i > \tau_i > 0$,

$$\mathcal{S}_i = \langle i, Y_{i,1}, \ldots, Y_{i,\tau_i}, \phi_{i,\tau_i+1}, h_{i,\tau_i+2} \rangle \ .$$

Moreover, let $\mathcal{M}'_e = \{i | i \in \mathcal{M}_e \bigcap \mathcal{I}, \tau_i < n_i, n_i \neq 0\}$ denote the zones with at least one attribute-q rating smaller than γ in every macro zone $e \in [1, M/m]$. There are two cases.

- Case 3: if there is zone $i \in \mathcal{M}'_e, \tau_i > 0$, nothing need be done because this case has been covered by Case 2.
- Case 4: otherwise, we have $\tau_i = 0, \forall i \in \mathcal{M}'_e$. Assuming that $A'_{j,1,q}$ is the highest attribute-q rating in $\mathcal{M}'_e$, the LBSP also adds $\mathcal{S}_j = \langle j, \phi_{j,1}, h_{j,2} \rangle$ to the query result.

Furthermore, for any candidate macro zone e, if there is no POI in zones $\mathcal{M}_e \bigcap \mathcal{I}$ with attribute-q rating not smaller than γ, we must have $n_i = \tau_i$ for all $i \in \mathcal{M}_e \bigcap \mathcal{I}$, in which case the LBSP is required to return $\mathcal{S}_i = \langle i \rangle$ for each $i \in \mathcal{M}_e \bigcap \mathcal{I}$ if $n_i = \tau_i = 0$ (Case 5). Note that the case for $n_i = \tau_i > 0$ has been covered by Case 1 above.

As in Scheme 1, the LBSP additionally returns the data collector's signature on the qth Merkle root hash and $\mathcal{T} = \bigcup_{i \in \mathcal{I}'} \mathcal{T}_i$, where $I' \subseteq \mathcal{I}$ is the set of zones in which there at least one POI data record or index has been included in the query result. In contrast to Scheme 1, $\langle i, \phi_{i,1}, h_{i,2}, \mathcal{T}_i \rangle$ need not be returned for any zone $i \in \mathcal{I}$ when $\tau_i = 0$ in most cases due to the macro-zone idea, which can lead to much lower computation and communication overhead in practice.

4.2.3 Query-result Verification

After receiving the query result, the user first verifies its authenticity as in Scheme 1. If the authentication succeeds, he proceeds with correctness verification by checking whether the query result contains some information for every candidate macro zone $e \in [1, M/m]$ that overlaps with the query region $\mathcal{R}$. This verification should succeed for a correct query result according to the query-processing process. If so, the user further checks that the query result satisfies the same two conditions as in Scheme 1 (see Section 4.2.3) and then determines the lowest attribute-q rating γ among kPOI. Subsequently, based on the information format $\mathcal{S}_i$ for every zone i in the query result, the user determines τ_i (i.e., the number of POIs in zone i with attribute-q ratings $\geq \gamma$) and also the relationship between τ_i and n_i (the total number of POIs in zone i).

Unlike Scheme 1, Scheme 2 does not require some information to be returned for every candidate zone $i \in \mathcal{I}$ overlapping with the query region $\mathbf{R}$ if $\tau_i = 0$. The LBSP may exploit this situation and return no information for zone i even if $\tau_i > 0$. To detect this possible attack, the user conducts the following verifications for every candidate macro zone e in accordance with the five cases in query processing.

- If there is any zone $i \in \mathcal{I} \bigcap \mathcal{M}_e$ with $0 < \tau_i < n_i$ (i.e., Case 2 in query processing), the user checks whether the query result contains a valid $\mathcal{S}_x$ field corresponding to Case 1 or 2 in query processing for every zone $x \in \mathcal{I} \bigcap \mathcal{M}_e \bigcap (\bigcup_{j=1}^{\tau_i+1} \mathcal{I}_{i,j})$ that satisfies $A'_{x,1,q} \geq \gamma > A'_{i,\tau_i+1,q}$. If not, the user considers the query result incorrect. The reason is that the pair $\langle x, A'_{x,1,q} \rangle$ should have been inserted by the data collector in one of $\{\mathcal{I}_{i,j}\}_{j=1}^{\tau_i+1}$ if $x \in \mathcal{M}_e$ and $A'_{x,1,q} > A'_{i,\tau_i+1,q}$. If x is also in $\mathcal{I}$ and $A'_{x,1,q} \geq \gamma$, we have $\tau_x \geq 1$, so the LBSP should have returned a valid $\mathcal{S}_x$ for zone x corresponding to Case 1 or 2.
- If such zone i does not exist, the user checks if the query result contains $\mathcal{S}_j = \langle j, \phi_{j,1}, h_{j,2} \rangle = \langle j, l_{j,1}, \mathcal{I}_{j,1}, A'_{j,1,q}, H(\mathcal{I}_{j,1} || D'_{j,1}) \rangle$ with $A'_{j,1,q} < \gamma$ for $j \in \mathcal{I} \bigcap \mathcal{M}_e$, which corresponds to the case of $\tau_i = 0$ for all $i \in \mathcal{M}'_e = \{i | i \in \mathcal{M}_e \bigcap \mathcal{I}, \tau_i < n_i, n_i \neq 0\}$. If so, for every zone $x \in \mathcal{I} \bigcap \mathcal{M}_e \bigcap \mathcal{I}_{j,1}$ with $A'_{x,1,q} \geq \gamma > A'_{j,1,q}$, the user checks whether the query result contains a valid $\mathcal{S}_x$ corresponding to Case 1 or 2 in query processing. If not, the query result is considered incorrect. Note that this verification implicitly ensures the compliance with Case 4 in query processing, i.e., that the LBSP only returns the information for the highest attribute-q rating in $\mathcal{M}'_e$.
- If such zone j does not exist either, it must be true that $n_i = \tau_i$ for all $i \in \mathcal{I} \bigcap \mathcal{M}_e$ and that there is no attribute-q rating in zones $\mathcal{I} \bigcap \mathcal{M}_e$ smaller than γ. The user verifies this by checking if $n_i = 0$ or $n_i = \tau_i > 0$ for each zone $i \in \mathcal{I} \bigcap \mathcal{M}_e$ according to the corresponding field $\mathcal{S}_x$. If not, the user considers the query result incorrect.

If the query result pass all the above verifications, the user considers it both authentic and correct.

4.2.4 An Example

We continue with the example in Fig. 2, where we assume that zones 1 to 3 compose a macro zone. Unlike in Scheme 1, the LBSP need not return any information for zone 3, which has been embedded into the query result along with the information from zones 1 and 2. More specifically, we can see that the highest POI rating $A'_{3,1}$ in zone 3 satisfies $A'_{1,3} > A'_{3,1} > A'_{1,4}$ and $A'_{2,2} > A'_{3,1} > A'_{2,3}$. Therefore, $\langle 3, A'_{3,1} \rangle$ must have been embedded into $\mathcal{I}_{1,4}$ and also $\mathcal{I}_{2,3}$, so there is no need to include $\langle 3, A'_{3,1}, \mathcal{T}_3 \rangle$ in the query result. After verifying the query result, the user can find that no POI in zone 3 has a rating higher than γ.

5 Secure Moving Top-k Query Processing

In this section, we propose a novel scheme to realize secure moving top-k query processing.

5.1 Basics of Moving Top-k Queries

Similar to moving kNN queries [17], a moving top-k query asks for the top-k POIs in a moving query region $\mathbf{R}$. For example, a user may want to be kept updated about the top-10 restaurants within 5 miles radius when driving north in Manhattan. In this example, $\mathbf{R}$ is a changing circle of radius 5 miles centered at the user's current location.

One may think about two possible solutions for secure moving top-k queries. First, the user may query $k' > k$ POIs in a larger area than needed whereby to deduce the top-k POIs in a moving smaller query region. Unfortunately, this solution works only if POI density is relatively uniform across the larger region, otherwise it is difficult to choose suitable k' to ensure that the top-k' POIs in the larger region contains the top-k' POIs in each smaller region of interest. For example, there are many more good restaurants in the middle and lower Manhattan area than in the upper Manhattan area. If the user wants to deduce the top-10 restaurants in the upper Manhattan area from the top-k' restaurants in New York city, he has to choose k' large enough to ensure the query result contains the top-10 restaurants in the upper Manhattan area, which is not only difficult but may also incur unnecessarily high communication and computation overhead. Another possible solution is to securely process a moving top-k query as a sequence of snapshot top-k queries. In particular, the mobile user submits a snapshot top-k query at a sufficiently high frequency which can be processed by the LBSP using Scheme 1 or 2. Since the query results for consecutive snapshot top-k queries may largely overlap, this solution may also incur unnecessarily high communication and computation overhead. This observation motivates us to develop a more efficient solution to moving top-k queries.

5.2 Scheme 3

Our basic idea is to let the LBSP process consecutive snapshot top-k queries involved in a moving top-k query as a whole and only return a query result if there is any update in the top-k POIs satisfying the query. An update in the top-k POIs may occur when a current top-k POI is no longer in the moving query region or when a new POI appears in the moving query region, which has an attribute-q rating higher than the lowest among the current top-k POIs. The user can directly tell when the first situation occurs based on the current top-k POIs he knows, in which case he can issue a new snapshot top-k query for the current query region. The user, however, cannot tell when the second situation will occur. Without a sound defense in place, the LBSP can choose not to inform the user about updated top-k POIs in the second situation.

Scheme 3 aims at the second situation discussed above and can be built upon either Scheme 1 or Scheme 2. Due to space constraints, we focus on Scheme 1 and assume

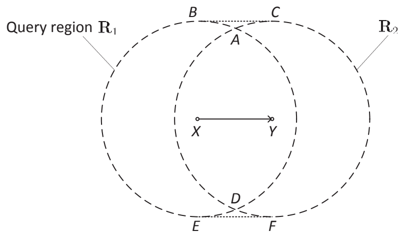


Fig. 3: An example of two consecutive snapshot top-k queries.

that the data set has been preprocessed by the data collector accordingly, and the same design principles apply when Scheme 2 is chosen instead. Without loss of generality, we assume that a user issues a moving top-k query for attribute q during time period $[0, T]$, where T may be unknown in advance. Since a moving top-k query involves a sequence of snapshot top-k queries, we denote the ath snapshot top-k query by Q_a and the corresponding query region $\mathbf{R}_a$. We also let kPOI_a be the top-k POIs in $\mathbf{R}_a$ and γ_a the lowest attribute-q rating among kPOI_a. In what follows, we detail the additional operations in Scheme 3 in contrast to Scheme 1, including *query scheduling*, *query processing*, and *query-result verification*.

5.2.1 Query Scheduling

To realize a moving top-k query, the user issues a sequence of snapshot top-k queries according to a query schedule. In particular, the user issues the ath snapshot top-k query (i.e., Q_a) at time

$$t_a = \begin{cases} 0 & \text{if } a = 1, \\ \min(t_{a-1} + \triangle t, t_u, T) & \text{o.w.}\ , \end{cases} \tag{6}$$

where $\triangle t$ is his personal parameter determining the lowest frequency at which snapshot queries are issued, and t_u denotes the time when the first POI in the current top-k POIs moves out of the query region. To be more clear, after receiving kPOI_a from the LBSP in response to Q_a, the user sets a timer of length $\triangle t$. Then he issues Q_{a+1} when the timer fires or when the first POI in kPOI_a is no longer in his moving query region, whichever comes first.

As before, Q_a includes the interested POI category, the interested attribute q, the current query region $\mathbf{R}_a$, and an integer $k \geq 1$. To facilitate query processing at the LBSP, it also includes both an integer id uniquely identifying this moving top-k query and a one-bit flag indicating whether Q_a is the last snapshot query for this moving top-k query.

5.2.2 Query Processing

Assume that the LBSP has purchased the data set from the data collector as in under Scheme 1. It processes the sequence of snapshot top-k queries of the same moving top-k query as follows.

We first define a special region to ease our subsequent illustration. Consider two consecutive snapshot top-k

queries Q_a and Q_{a+1} with query regions $\mathbf{R}_a$ and $\mathbf{R}_{a+1}$, respectively. Since the user's query region $\mathbf{R}$ is always defined with regard to his current location, we have $\mathbf{R} = \mathbf{R}_a$ at time t_a and $\mathbf{R} = \mathbf{R}_{a+1}$ at time t_{a+1}. We define the *progression region*, denoted by $\mathbf{P}_a$, as the area in $\mathbf{R}_{a+1}$ but not in $\mathbf{R}_a$. Consider Fig. 3 as an example where the user issues two consecutive snapshot top-k queries at locations X and Y with query regions $\mathbf{R}_1$ and $\mathbf{R}_2$, respectively. The progression region $\mathbf{P}_1$ is the area formed by arcs AD and $ACFD$.

On receiving query Q_1, the LBSP locates kPOI_1 in the query region $\mathbf{R}_1$ and then returns a complete query result constructed as in Scheme 1. In addition, the LBSP records id, $\mathbf{R}_1$, and kPOI_1 to facilitate the processing of subsequent snapshot top-k queries with the same moving top-k identifier id. Then it processes any subsequent query Q_b ($b > a$) as follows. Without loss of generality, assume that the last complete query result the LBSP returned is in response to Q_a ($a \geq 1$), which contains kPOI_a in $\mathbf{R}_a$. In other words, we assume that the top-k POIs $\{\mathsf{kPOI}_a\}_{i=a}^{b-1}$ in the query regions $\{\mathbf{R}_i\}_{i=a}^{b-1}$ are all equal to kPOI_a.

First, the LBSP checks if a complete query result containing the top-k POIs (i.e., kPOI_b) in the current query region $\mathbf{R}_b$ need be returned by checking the following two conditions.

- kPOI_b have different POIs from kPOI_a.
- The one-bit flag in Q_b is set, meaning that it is the last snapshot query for the current moving top-k query identified by the same id.

If neither condition holds, the LBSP returns a short ACK containing a predefined flag to the user, which means that the previously returned top-k POIs in kPOI_a remain valid in the current query region $\mathbf{R}_b$. Otherwise, the LBSP constructs a complete query result as follows.

First, the LBSP locates the top-k POIs (i.e., kPOI_b) in the query region $\mathbf{R}_b$ whose attribute-q ratings are among the highest k. Second, the LBSP retrieves the recorded query regions $\{\mathbf{R}_x\}_{x=a}^{b}$ based on their same moving top-k identifier id, based on which to compute the progressive regions $\{\mathbf{P}_x\}_{x=a}^{b-1}$. Next, the LBSP computes a *verification region* as $\mathbf{V}_{a\to b} = \bigcup_{x=a}^{b-1} \mathbf{P}_x$ whereby to find the set of zones either completely or partially covered by $\mathbf{R}_b \bigcup \mathbf{V}_{a\to b}$, denoted by $\mathcal{I}_{a\to b}$.

Let γ_a and γ_b be the lowest attribute-q rating among kPOI_a and kPOI_b, respectively. For each zone $i \in \mathcal{I}_{a\to b}$, we define

$$\tau_i = \begin{cases} \tau_{b,i} & \text{if zone } i \text{ only overlaps with } \mathbf{R}_b, \\ \tau_{a,i} & \text{if zone } i \text{ only overlaps with } \mathbf{V}_{a\to b}, \\ \max(\tau_{a,i}, \tau_{b,i}) & \text{if zone } i \text{ overlaps with both } \mathbf{R}_b \\ & \text{and } \mathbf{V}_{a\to b}\ , \end{cases}$$

where $\tau_{a,i}$ and $\tau_{b,i}$ are the number of POIs in zone i with the attribute-q rating $\geq \gamma_a$ or γ_b, respectively. We further define

$$Z_{i,j} = \begin{cases} D'_{i,j} & \text{if } l'_{i,j} \in \mathbf{R}_b \text{ and } A'_{i,j,q} \geq \gamma_b, \\ \phi_{i,j} & \text{otherwise}\ , \end{cases}$$

which means that the LBSP only needs to return a much shorter index instead of the complete record for any POI not in the query region $\mathbf{R}_b$ or not among the top-k. Similar as in Scheme 1, the LBSP finally returns the following information $\mathcal{S}_i$ for each zone $i \in \mathcal{I}_{a\to b}$ as part of the query result.

- Case 1: if $n_i = 0$, $\mathcal{S}_i = \langle i \rangle$.
- Case 2: if $n_i = 1$, $\mathcal{S}_i = \langle i, Z_{i,1} \rangle$.
- Case 3: if $n_i \geq 2$ and $\tau_i = 0$, $\mathcal{S}_i = \langle i, \phi_{i,1}, h_{i,2} \rangle$.
- Case 4: if $n_i \geq 2$ and $n_i > \tau_i \geq 1$,

$$\mathcal{S}_i = \langle i, Z_{i,1}, \ldots, Z_{i,\tau_i}, \phi_{i,\tau_i+1}, h_{i,\tau_i+2} \rangle\ .$$

- Case 5: if $n_i = \tau_i \geq 2$, $\mathcal{S}_i = \langle i, Z_{i,1}, \ldots, Z_{i,\tau_i} \rangle$.

In addition, the LBSP returns $\mathcal{T} = \bigcup_{i\in\mathcal{I}_{a\to b}} \mathcal{T}_i$ and the data collector's signature on the qth Merkle root hash.

5.2.3 Query-result Verification

For every snapshot top-k query Q_b of the same moving top-k query, the LBSP (if benign) should return a complete query result if $b = 1$ or there has been any change in the top-k POIs, or return an ACK if $b > 1$ and the previously returned top-k POIs are still valid. Accordingly, there are three cases for the user to verify the query result in response to Q_b. First, if the user receives an ACK when Q_b is the final snapshot query, he can immediately tell that the result is incorrect. Second, if receiving an ACK when Q_b is not the final snapshot query, he marked this query result unverified and waits for the next complete query result. Third, if receiving a complete query result for Q_b (no matter whether Q_b is the final query), he verifies it as follows.

First, the user checks if the query result is authentic as in Scheme 1. If so, the user derives the set of zones $\mathcal{I}_{a\to b}$ from the POI information returned in the query result and checks if zones $\mathcal{I}_{a\to b}$ encloses the region $\mathbf{R}_b \bigcup \mathbf{V}_{a\to b}$. If so, he locates the lowest attribute-q rating γ_b in the query result whereby to check whether all the following conditions hold.

1. There are exactly k POI records in the query result.
2. Every returned POI record is in $\mathbf{R}_b$.
3. None of the POIs for which the indexes are returned satisfy the query. In particular, for each index $\phi_{i,j}, \forall i \in \mathcal{I}_{a\to b}$, at least one following condition does not hold.
    - $\phi_{i,j}$ contains a location $l'_{i,j} \in \mathbf{R}_b$.
    - $\phi_{i,j}$ contains an attribute rating $A'_{i,j,q} \geq \gamma_b$.

If so, the top-k POI records are correct.

Assume that the last complete query result the user verified is for Q_a and contains kPOI_a in the region $\mathbf{R}_a$ and that $b > a + 1$. The user should have accumulated $b - a - 1$ unverified query results for queries $\{Q_x\}_{x=a+1}^{b-1}$ and can verify their correctness by checking whether the LBSP should have returned a complete query result instead of an ACK for each of them instead. Let γ_a again denote the lowest attribute-q rating in kPOI_a and $\mathbf{S}_{a\to b} = \bigcup_{j=a}^{b-2} \mathbf{P}_j$ denote the *suspicion region*. If all the unverified query results are correct, there should not be any POI in

$\mathbf{S}_{a\to b}$ with attribute-q rating higher than γ_a. According to the query-processing process, the LBSP should have returned one or multiple data indices for every zone i that overlaps with $\mathbf{S}_{a\to b}$; otherwise, the query result for Q_b would not have passed the verification. The user thus proceeds to check whether at least one following condition does not hold for any such index, say $\phi_{i,j}$.

- $\phi_{i,j}$ contains a location $l'_{i,j} \in \mathbf{S}_{a\to b}$.
- $\phi_{i,j}$ contains an attribute rating $A'_{i,j,q} > \gamma_a$.

If so, all the unverified query results are marked verified; otherwise, the LBSP has misbehaved.

6 Performance Analysis

In this section, we analyze Schemes 1∼3 with regard to their correctness in detecting inauthentic and/or incorrect query results and the related communication/computation overhead. To make the quantitative analysis tractable, we make the following assumptions.

- There are $n > k$ POIs uniformly distributed in each zone, i.e., $n_i = n, \forall i \in [1, M]$, where $M = 2^d$ for an integer $d > 1$.
- All attribute ratings are i.i.d. random variables uniformly distributed in the range $[0, 1]$ after proper normalization.
- The query-region size is δ times of the zone size.

6.1 Analysis of Scheme 1

The following proposition is for the correctness of Scheme 1.

***Proposition** 1: Scheme 1 can detect any incorrect and/or inauthentic query result from a misbehaving LBSP.*

We give the proof of the above proposition in the supplemental file.

The main extra computation overhead incurred by Scheme 1 on top-k query processing involves hash computations and signature generations/verifications. Consider the data collector first. For every zone $i \in [1, M]$ and every attribute, the data collector performs n hash computations to generate the indexes $\{\phi_{i,j}\}_{j=1}^{n}$ and n hash computations to derive $h_{i,1}$, which leads to totally $2Mn$ hash computations. In addition, the data collector needs $M-1$ hash computations to construct the Merkle hash tree of every attribute and one signature generation for the root hash. Since there are q POI attributes, the total computation overhead per POI category at the data collector is $\lambda(2Mn + M - 1)$ hash computations and λ signatures. Moreover, the computation overhead at the LBSP is negligible because the LBSP need not perform any hash or signature operations for query processing. Finally, we consider the computation overhead at the user. For every query result, the user needs one signature verification for the Merkle root hash and also a certain number of hash computations given below.

***Proposition** 2: The expected number of hash computations the user performs to verify the query result under Scheme 1 is given by*

$$\mathsf{E}[N_{\text{hash},1}] = k + |\mathcal{I}| \cdot \frac{(k+\delta)n+1}{\delta n+1} + \sum_{j=1}^{d-1} 2^{j-1}(1-(1-2^{-(j-1)})^{|\mathcal{I}|}) \ . \tag{7}$$

We give the proof of the above proposition in the supplemental file.

Now we analyze the communication overhead associated with transmitting the necessary information for authenticity and correctness proofs from the data collector to the LBSP. Let L_h, L_loc, L_r, and L_sig denote the bit-lengths of a hash value $H(\cdot)$, a POI location, an attribute rating, and the data collector's signature, respectively. For each of λ POI attributes, the data collector sends n indexes of $L_\text{loc} + L_\text{r} + L_\text{h}$ bits for each of M zones as well as a Merkle hash tree of $(M-1)L_\text{h}$ bits. The extra communication overhead in bits per POI category Scheme 1 incurs between the data collector and LBSP is thus

$$\mathsf{S}_1 = \lambda(Mn(L_\text{loc} + L_\text{r} + L_\text{h}) + (M-1)L_\text{h} + L_\text{sig}). \tag{8}$$

We also have the following proposition about the extra communication overhead associated with sending authenticity and correctness proofs of a top-k query result from the LBSP to the user.

***Proposition** 3: The additional communication overhead between the LBSP and the user incurred by Scheme 1 is given by*

$$\mathsf{E}[\mathsf{T}_1] = (|\mathcal{I}| \cdot \frac{(k+\delta)n+1}{\delta n+1} - k)(L_\text{loc} + L_\text{r} + L_\text{h}) + |\mathcal{I}| \cdot d + \sum_{j=1}^{d-1} 2^{j}(1-(1-2^{-j})^{|\mathcal{I}|})L_\text{h} + L_\text{sig} \ , \tag{9}$$

We give the proof of the above proposition in the supplemental file.

6.2 Analysis of Scheme 2

The following proposition is for the efficacy of Scheme 2.

***Proposition** 4: Scheme 2 can detect any incorrect and/or inauthentic query result from a misbehaving LBSP.*

We give the proof of the above proposition in the supplemental file.

Scheme 2 incurs the same computation overhead to the data collector and LBSP as Scheme 1, which has been analyzed before. To verify the authenticity and correctness of a top-k query result, the user performs one signature verification on the Merkle root hash and also a certain number of hash computations given in the following theorem.

***Proposition** 5: The expected number of hash computations the user performs to verify the query result under Scheme 2* *is given by*

$$\mathsf{E}[N_{\text{hash},2}] = |\mathcal{I}|\mu_1 + \sum_{j=1}^{d-1} 2^{j-1}(1-(1-2^{-(j-1)})^{|\mathcal{I}|(1-\mu_2^n)}) \ , \tag{10}$$

where $\mu_1 = (n - n\mu_2 + 1 - \mu_2^n)$ *and* $\mu_2 = \frac{\delta n-k+1}{\delta n+1}$.

We give the proof of the above theorem in the supplemental file.

Now we analyze the communication overhead incurred by Scheme 2. In Scheme 2, every zone belongs to a macro zone of m zones. For every zone i in a macro zone $\mathcal{M}_e$, the set $\{j, A'_{j,1,q}\}_{j\in\mathcal{M}_e\setminus\{i\}}$ need be transmitted along with both POI records and indexes. Since a zone ID is of $\log_2 M = d$ bits, Scheme 2 requires the data collector to additionally transmit $2(m-1)(d+L_\text{r})$ bits for attribute q in contrast to Scheme 1. The extra communication overhead per POI category Scheme 2 incurs between the data collector and LBSP is thus

$$\mathsf{S}_2 = \mathsf{S}_1 + 2(m-1)\lambda(d+L_\text{r}) \ , \tag{11}$$

where S_1 is given in Eq. (8). We also have the following proposition about the communication overhead for sending authenticity and correctness proofs of a query result from the LBSP to the user.

Proposition *6: Assuming that the query region comprises $\check{m}$ zones $\mathcal{I}$ fully contained in a macro zone $\mathcal{M}_e$ with m zones. The expected additional communication overhead Scheme 2 incurs between the LBSP and user is bounded as follows,*

$$\begin{aligned}\mathsf{T}_2 \le\ & \check{m}(1-\mu^n)d + \check{m}(n - n\mu + 1 - \mu^n)(L_\text{loc} + L_\text{r} + L_\text{h}) \\ & + (\check{m}(1-\mu^n) + \sum_{j=1}^{d-1} 2^j(1-(1-2^{-j})^{\check{m}(1-\mu^n)}))L_\text{h} \\ & + \check{m}(1-\mu^n)(m-\check{m})(1-(\frac{n-\nu}{n+1})^n)(d+L_\text{r}) \\ & + g(g-1)(d+L_\text{r}) + L_\text{sig} \ ,\end{aligned} \tag{12}$$

where $\mu = (\check{m}n - k + 1)/(\check{m}n+1), \nu = n(1-\mu)/(1-\mu^n)$, *and* $g = \min(k, \check{m})$.

We give the proof of the above proposition in the supplemental file. We have not been able to obtain a close-form solution for the more general case, which we will evaluate using simulation in the next section.

6.3 Analysis of Scheme 3

The following proposition is for the correctness of the Scheme 3.

Proposition *7: Any misbehavior of the LBSP, including returning incorrect/inauthentic query result and omitting complete query results, will be eventually detected under Scheme 3.*

We give the proof of the above proposition in the supplemental file. We will use simulation to evaluate the communication and computation overhead incurred by Scheme 3 in the next section.

TABLE 2: Default Simulation Settings

Para.	**Val.**	**Para.**	**Val.**	**Para.**	**Val.**	**Para.**	**Val.**
M	10000	m	100	n	100	δ	10
k	5	d	14	d	20	L_h	160
L_loc	20	L_sig	160	L_r	10		

7 Simulation Results

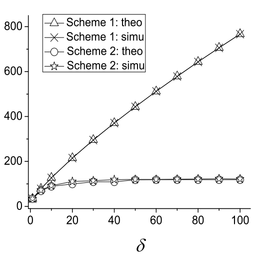
In this section, we evaluate our schemes and validate the theoretical results we obtained in Section 6 using simulations on a synthetic dataset. We assume that the data set covers 100×100 unit square zones of $1000\times 1000\text{m}^2$, each containing 100 POIs distributed uniformly at random. The simulation code is written in C++, and each data point represents an average of 50 simulation runs with different random seeds. In addition, our simulations use the default parameters in Table 2, unless stated otherwise.

7.1 Snapshot Top-k Queries

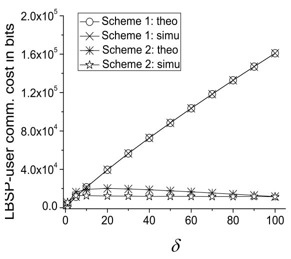
We first report the simulation results for Schemes 1 and 2. Recall that δ denote the ratio of the query-region size to the zone size and that $\mathcal{I}$ represent the set of candidate zones that completely or partially overlap with the query region $\mathbf{R}$. We assume that $\mathbf{R}$ exactly covers an integer number of zones, which means that $\mathcal{I} = \mathbf{R}$ and $|\mathcal{I}| = \delta$. We have also simulated the top-k queries with query region $\mathbf{R}$ being a circle of radius r centered at a random location and give the simulation results in the supplemental file.

Fig. 4a shows the impact of δ on the user's computation overhead for $k = 5$, where the single signature verification is not included for brevity. Clearly, our analytical and simulation results closely match under both schemes. In addition, the user's computation overhead increases with δ under Scheme 1, while it initially increases as δ goes from 1 to 10 and then is relatively stable under Scheme 2. The reason is that Scheme 1 requires the LBSP to return information for every zone in $\mathbf{R}$ for the user to verify. Therefore, the larger δ, the higher the user's computation overhead in Scheme 1. In contrast, Scheme 2 requires the LBSP to return information only for the zones that have at least one POI among the top-k POIs under our simulation settings, and there are at most k such zones in $\mathbf{R}$. Therefore, Scheme 2 incurs lower computation overhead on the user for small k and large δ.

Fig. 4b shows the impact of δ on the LBSP-user communication overhead for $k = 5$. It is clear that the simulation results are always below the corresponding theoretical upper bounds. As in Fig. 4a, we can also observe that the LBSP-user communication overhead in Scheme 1 always increases with δ and is higher than that in Scheme 2. In contrast, the LBSP-user communication overhead under Scheme 2 is relatively stable and even slightly decreases when δ grows. The reason is that the kth largest attribute rating becomes large as δ increases, which means that the query result contains less infor-

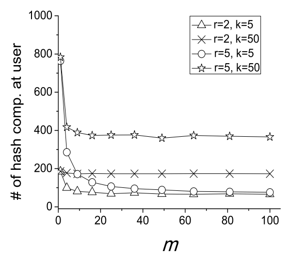


(a) computation cost

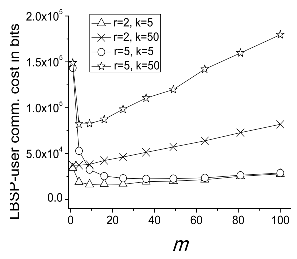


(b) communication cost

Fig. 4: The impact of δ, where $k = 5$.

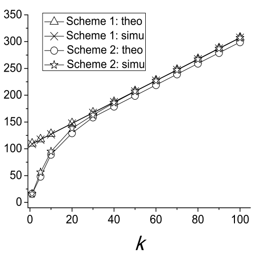


(a) computation cost

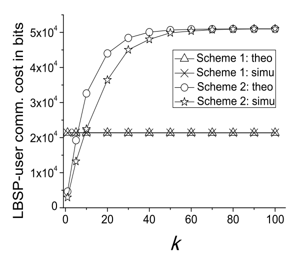


(b) communication cost

Fig. 6: The impact of m on Scheme 2.

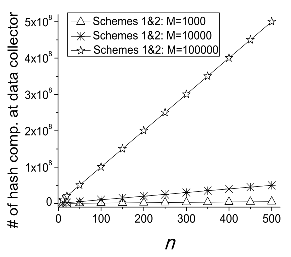


(a) computation cost

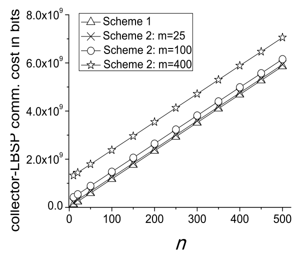


(b) communication cost

Fig. 5: The impact of k, where $\delta = 10$.



(a) computation cost



(b) communication cost

Fig. 7: The impact of n.

mation for other zones in the same macro zone with attribute ratings higher than any top-k rating.

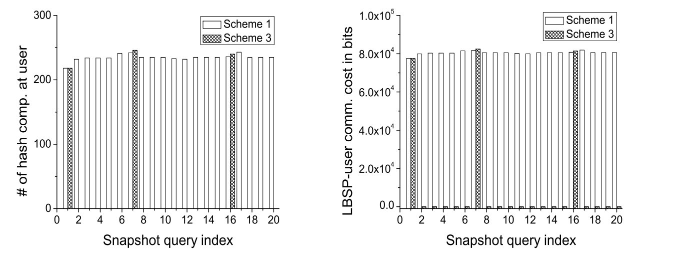
Fig. 5a shows the impact of k on the user's computation overhead for $\delta = 10$. We can see that our simulation and analytical results closely match and increase with k under both schemes. The reason is that the number of hash computations increases with the number of zones with information in the query result, which itself increases with k. In addition, since Scheme 2 does not require the LBSP to return any information for zones without a top-k POI, it requires the user to perform fewer hash computations and thus incurs smaller computation overhead than Scheme 1. The difference between the two schemes gradually diminishes when k goes beyond 20, as the number of zones in $\mathbf{R}$ without a top-k POI quickly decreases for sufficiently large k.

Fig. 5b shows the impact of k on the LBSP-user communication overhead for $\delta = 10$. Again, our simulation and analytical results closely match. In addition, the LBSP-user communication overhead of Scheme 1 is not affected by k because it only involves transmitting $|\mathcal{I}| = \delta$ POI indexes. In contrast, the LBSP-user communication overhead of Scheme 2 always increases with k, as the number of POI records or indexes increases with k, and accordingly the information about other zones in the same macro zone returned along with every POI record or index also increases.

Fig. 6a shows that the user's computation overhead decreases rapidly as m increases from 1 to 10 and slowly as m further increases. The reason is that the LBSP returns only one index and the corresponding auxiliary set for each candidate macro zone that has no top-k POI. When k is small and $\mathbf{R}$ is large, most zones in $\mathbf{R}$ do not have any top-k POI, so the number of indexes and auxiliary sets returned is approximately proportional to the number of macro zones and thus inversely proportional to m when m is not too large. Otherwise, the number of macro zones overlapping with $\mathbf{R}$ approaches a constant, leading to relatively stable computation overhead.

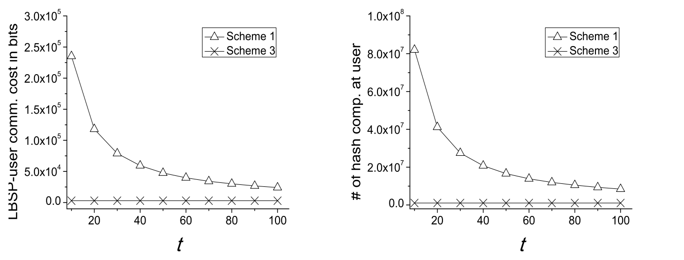
Fig. 6b shows that the LBSP-user communication overhead quickly decreases as m increases from 1 to 10. The reason is that the larger m, the fewer POIs and corresponding auxiliary sets returned to the user. As m further increases, the communication overhead slowly increases, as a larger m requires the LBSP to return more information about other zones in the same macro zone along with every POI record or index in the query result.

Figs. 7a and 7b show the impact of n, the number of POIs per zone, on the data collector's computation overhead and the collector-LBSP communication overhead. For brevity, we only show the simulation results which apply to both Type-1 or Type-2 queries. Fig. 7a shows that the data collector's computation overhead increases linearly with n under both schemes. The reason is that the data collector performs one hash computation to generate the index and chain it with adjacent indexes for each POI record in both schemes. Moreover, as anticipated, the larger M, the more POIs, and the higher the computation overhead. In addition, Fig. 7b shows that the collector-LBSP communication overhead under both schemes increases with n, and Scheme 2 incurs larger overhead because it requires additional information for other zones in the same macro zone to be transmitted for each POI record.



(a) computation cost (b) communication cost

Fig. 8: Comparison of the first 20 snapshot queries in Schemes 1 and 3.



(a) computation cost (b) communication cost

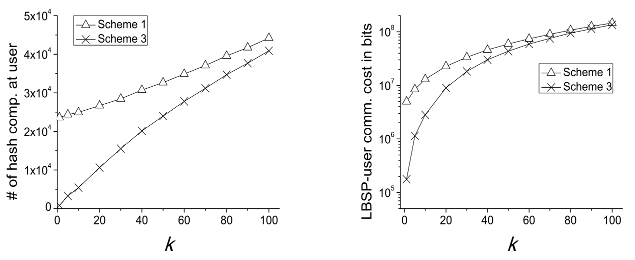
Fig. 9: The impact of $\triangle t$ on Scheme 3.

7.2 Moving Top-k Queries

In this subsection, we report the simulation results for Scheme 3. In particular, we compare Scheme 3 with realizing moving top-k query via independent snapshot queries under Scheme 1. We simulate a moving top-k query in which the query region is a circular area of radius $r = 5000$m centered at the user's location. The user starts at the random location along a random direction, moves at a speed of 5m/s for a total distance of 5000m.

Figs. 8a and 8b show the user's computation overhead and LBSP-user communication overhead incurred by the first 20 snapshot top-k queries under Schemes 1 and 3, respectively, where $\triangle t = 20$s. We can see that both schemes incur the same user-side computation overhead and LBSP-user communication overhead for the first snapshot top-k query, as the LBSP need return a complete query result in both cases. Under Scheme 1, each snapshot query incurs similar computation and communication costs, while under Scheme 3, all the snapshot queries (except the 1st, 7th, and 16th) incur negligible user-side computation overhead and LBSP-user communication overhead. This is anticipated, as the LBSP always need return a complete query result for any snapshot query under Scheme 1 but does so only when there is an update in the top-k POIs from the previous ones under Scheme 3.

Figs. 9a and 9b come Schemes 1 and 3 when $\triangle t$, the delay between two consecutive snapshot queries, varies. We can see that the total computation and communication cost incurred by Scheme 3 are relatively insensitive to the change in $\triangle t$, as no matter how frequently the user issues snapshot top-k queries, the LBSP only need return a complete query result when there is an update in the top-k POIs. In contrast, the total computation and communication costs incurred by Scheme 3 are inversely proportional to $\triangle t$, since the LBSP treats each snapshot query independently by always returning a complete query result. These results demonstrate the significant advantage of Scheme 3 over Scheme 1.



(a) computation cost (b) communication cost

Fig. 10: The impact of k on Scheme 3.

Figs. 10a and 10b compare Schemes 1 and 3 when k varies. We can see that the user-side computation overhead and LBSP-user communication overhead both increase as k increases under both schemes. This is because that the larger k, the more updates in the top-k POIs for the same distance that the user travels, and vice versa. Under both schemes, the LBSP need return more complete query results, which lead to higher user-side computation overhead and LBSP-user communication overhead. When k is small, Scheme 3 incurs significantly lower user-side computation overhead and LBSP-user communication overhead than Scheme 1 does.

8 Conclusion

This paper considers a novel distributed system for collaborative location-based information generation and sharing. We have proposed three novel schemes to enable secure top-k query processing via untrusted LBSPs for fostering the practical deployment and wide use of the envisioned system. Our schemes support both snapshot and moving top-k queries, which enable users to verify the authenticity and correctness of any top-k query result. The efficacy and efficiency of our schemes are thoroughly analyzed and evaluated through detailed simulation studies.

Acknowledgment

This work was supported in part by the US National Science Foundation under grants CNS-0844972 (CAREER) and CNS-1117462. The work of C. Zhang was partially supported by the National Natural Science Foundation of China under Grant 61202140. We would also like to thank anonymous reviewers for their constructive comments and helpful advice.

References

[1] 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.

[2] H. Yu, M. Kaminsky, P. Gibbons, and A. Flaxman, "SybilGuard: Defending against sybil attacks via social networks," *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 576–589, June 2008.

[3] H. Yu, P. Gibbons, M. Kaminsky, and F. Xiao, "SybilLimit: a near-optimal social network defense against sybil attacks," *IEEE/ACM Trans. Netw.*, vol. 18, pp. 885–898, June 2010.

[4] H. Hacigümüs, S. Mehrotra, and B. Iyer, "Providing database as a service," in *IEEE ICDE*, Aalborg, Denmark, Feb. 2002.

[5] W.-S. Ku, L. Hu, C. Shahabi, and H. Wang, "Query integrity assurance of location-based services accessing outsourced spatial databases," in *Int. Sym. on Advances in Spatial and Temporal Databases*, Aalborg, Denmark, July 2009.

[6] 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.

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

[8] 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, 2012.

[9] 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.

[10] N. Cao, Z. Yang, C. Wang, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," in *IEEE ICDCS'11*, Minnesapolis, MN, June 2011.

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

[12] 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.

[13] H. Pang and K.-L. Tan, "Verifying completeness of relational query answers from online servers," *ACM Trans. Inf. Syst. Secur.*, vol. 11, no. 2, pp. 1–50, Mar. 2008.

[14] M. Narasimha and G. Tsudik, "Authentication of outsourced databases using signature aggregation and chaining," in *DASFAA'06*, Singapore, Apr. 2006, pp. 420–436.

[15] H. Pang, J. Zhang, and K. Mouratidis, "Scalable verification for outsourced dynamic databases," *PVLDB*, vol. 2, no. 1, pp. 802–813, 2009.

[16] Y. Yang, S. Papadopoulos, D. Papadias, and G. Kollios, "Spatial outsourcing for location-based services," in *IEEE ICDE*, Cancún, México, Apr. 2008, pp. 1082–1091.

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

[18] M. Yiu, E. Lo, and D. Yung, "Authentication of moving kNN queries," in *IEEE ICDE*, Hannover, Germany, Apr. 2011, pp. 565–576.

[19] R. Merkle, "A certified digital signature," in *CRYPTO*, Santa Barbara, CA, Aug. 1989, pp. 218–238.

[20] B. Sheng and Q. Li, "Verifiable privacy-preserving range query in sensor networks," in *IEEE INFOCOM'08*, Phoenix, AZ, Apr. 2008, pp. 46–50.

[21] J. Shi, R. Zhang, and Y. Zhang, "Secure range queries in tiered sensor networks," in *IEEE INFOCOM'09*, Rio de Janeiro, Brazil, Apr. 2009.

[22] R. Zhang, J. Shi, and Y. Zhang, "Secure multidimensional range queries in sensor networks," in *ACM MobiHoc'09*, New Orleans, LA, May 2009, pp. 197–206.

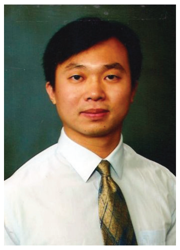
[23] F. Chen and A. Liu, "SafeQ: Secure and efficient query processing in sensor networks," in *INFOCOM'10*, San Diego, CA, Mar. 2010, pp. 1–9.

[24] R. Zhang, J. Shi, Y. Liu, and Y. Zhang, "Verifiable fine-grained top-k queries in tiered sensor networks," in *INFOCOM'10*, San Diego, CA, Mar. 2010.

[25] R. Merkle, "Protocols for public key cryptosystems," in *IEEE S&P'80*, Oakland, CA,USA, Apr. 1980, pp. 122–134.

Rui Zhang received the B.E. in Communication Engineering and the M.E. in Communication and Information System from Huazhong University of Science and Technology, China, in 2001 and 2005, respectively, and the PhD degree in electrical engineering from the Arizona State University, in 2013. He was a software engineer in UTStarcom Shenzhen R&D center from 2005 to 2007. He has been an assistant professor in the Department of Electrical Engineering at the University of Hawaii since July 2013. His primary research interests are network and distributed system security, wireless networking, and mobile computing.

Jingchao Sun received the B.E. in Electronics and Information Engineering and the M.E. in Communication and Information System from Huazhong University of Science and Technology, China, in 2008 and 2011, respectively. He is currently a Ph.D. student in School of Electrical, Computer, and Energy Engineering at Arizona State University. His primary research interests are network and distributed system security and privacy, wireless networking, and mobile computing.



Yanchao Zhang received the B.E. in Computer Science and Technology from Nanjing University of Posts and Telecommunications, China, in 1999, the M.E. in Computer Science and Technology from Beijing University of Posts and Telecommunications, China, in 2002, and the Ph.D. in Electrical and Computer Engineering from the University of Florida in 2006. He is currently as an Associate Professor in School of Electrical, Computer, and Energy Engineering at Arizona State University. Before ASU, he was an Assistant Professor of Electrical and Computer Engineering at New Jersey Institute of Technology from 2006 to 2010. His primary research interests are network and distributed system security, wireless networking, and mobile computing. He is (was) an Associate Editor of IEEE Transactions on Vehicular Technology, a Feature Editor of IEEE Wireless Communications, a Guest Editor of IEEE Wireless Communications Special Issue on Security and Privacy in Emerging Wireless Networks in 2010, and a TPC Co-Chair of Communication and Information System Security Symposium, IEEE GLOBECOM 2010. He received the NSF CAREER Award in 2009.

Chi Zhang Chi Zhang received the B.E. and M.E. degrees in Electrical and Information Engineering from Huazhong University of Science and Technology, China, in 1999 and 2002, respectively, and the Ph.D. degree in Electrical and Computer Engineering from the University of Florida in 2011. He joined the University of Science and Technology of China in September 2011 as an Associate Professor of the School of Information Science and Technology. His research interests are in the areas of network protocol design, network performance analysis, and network security guarantee, particularly for wireless networks and social networks. He is the recipient of the 7th IEEE ComSoc Asia-Pacific Outstanding Young Researcher Award.