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

Rui Zhang and Yanchao Zhang
School of Electrical, Computer, and Energy Engineering
Arizona State University
Tempe, Arizona, 85287-5706
Email: {ruizhang,yczhang}@asu.edu

Chi Zhang
School of Information Science and Technology
University of Science of Technology of China
Hefei, Anhui 230027, China
Email: chizhang@ustc.edu.cn

***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 location-based 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 two novel schemes for users to detect fake 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.**

I. 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, eMarketer projected that US and worldwide smartphone users will reach 73.3 and 571.1 million in 2011, respectively, and 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 interest*s (POIs) such as bars, restaurants, grocery stores, coffee shops, and hotels. Meanwhile, it becomes commonplace for people to perform various location-based POI queries at online location-based service providers (LBSPs) such as Google and Yelp. As probably the most familiar type of location-based queries, a **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 20 miles of his current location.

We observe two essential drawbacks with current location-based 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 location-based 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 which would like to pay. Even if LBSPs are not malicious, they may return unfaithful query results under the influence of various attacks such as the Sybil attack [1], [2] 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 make profit by selling the data sets to individual LBSPs. Instead of submitting POI reviews to individual LBSPs, people (called *data contributor*s) can now submit them 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 [1], [2] 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, and Twitter. Many of these service providers have offered open APIs for exporting selected data from their systems. We postulate that they may act as location-based data collectors and sellers in the future 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 to spend tremendous efforts in soliciting 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 data sets purchased from data collectors. The query results they can provide will be much more trustworthy, which would in turn help them attract more users. This model can also 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 data sets 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 two novel schemes to tackle the above challenge for fostering the practical deployment and wide use of the envisioned system. The key idea of both schemes is that the data collector precomputes and authenticates some auxiliary information (called *authenticated hint*s) 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 POIs are the true ones satisfying the top-k query. Our two schemes 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 efficacy and efficiency of our schemes are thoroughly analyzed and evaluated through detailed simulation studies.

II. Related Work

Our work is most related to data outsourcing [3], for which we can only review a few schemes due to space limitations. The framework of data outsourcing was first introduced in [3], 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 [4].

Ensuring data privacy requires the data owner to outsource encrypted data to the service provider and correspondingly efficient techniques for querying encrypted data. A bucketization approach was proposed in [5], [6] to enable efficient range queries over encrypted data. Shi *et al.* presented novel methods for multi-dimensional range queries over encrypted data [7]. Some most recent proposals aim at secure ranked keyword search [8], [9] or fine-grained access control [10] 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 [11]–[13] based on signature chaining and those [4], [14]–[16] based on the Merkle hash tree [17] or its variants. None of these schemes consider location-based top-k queries and thus are not directly applicable to our intended scenario.

Secure remote query processing in tiered sensor networks [18]–[22] 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 [18]–[21] to ensure data privacy against master nodes and also enable the network owner to verify range-query integrity. Moreover, Zhang *et al.* [22] proposed efficient techniques for the network owner to validate the integrity of top-k queries.

III. Problem Formulation

A. 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 [1], [2] to filter out fake reviews from malicious data contributors. The actual design of incentive and review-filtering mechanisms can be based on many existing results and is orthogonal to our work in this paper. 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 location-based 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 hotels, 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 $\mathcal{D}$. We also assume that the category has $\lambda \geq 1$ numerical attributes taking values from a given range, where λ is a category-specific parameter. For instance, if restaurant is the category under consideration, there may be $\lambda = 4$ attributes including *food*, *cost*, *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, $\mathcal{D}_i$ denote the data records of all POIs, and $\text{POI}_{i,j}$ 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, 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 $\text{POI}_{i,j}$ 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 $\text{POI}_{i,j}$ includes its name and location, $n_{i,j}$ reviews, $\{A_{i,j,q}\}_{q=1}^{\lambda}$, and possibly other information.

B. Problem Statement

We consider the following problem. Assume that a user issues a top-k query through the user-friendly web interface of a LBSP. A top-k query includes the interested POI *category* and *attribute* $q \in [1, \lambda]$, a *query region* $\mathcal{R}$, 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, have the attribute-q rating among the highest k, and are ordered with respect to the attribute-q rating in the descending order.

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. 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 good reviews and deleting bad ones. In addition, a LBSP good in nature may also be compromised by attackers to forge query results.

Given the above problem setting, 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.

IV. Secure Top-k Query Processing

In this section, we propose two novel schemes for secure location-based top-k query processing via untrusted LBSPs.

Both schemes comprise three phases but differ in operation details. In the *data-preprocessing* phase, the data collector uses

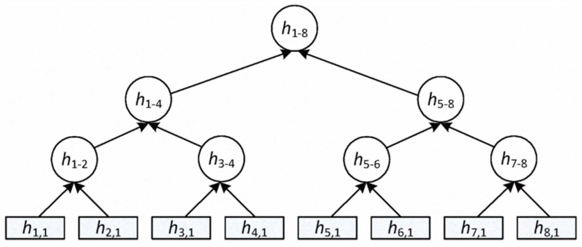


Fig. 1. An example of constructing the Merkle hash tree over $\{h_{i,1}\}_{i=1}^{8}$.

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 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 query region so that the query result contains exactly k POI records for arbitrary k. These two assumptions are relaxed in Section IV-C.

A. 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 [23]. The details about constructing and using authenticated hints are as follows.

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 zones. Recall that $\mathcal{D}_i = \bigcup_{j=1}^{n_i} D_{i,j}$, where $D_{i,j}$ denotes the record of $\text{POI}_{i,j}$ 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 $\phi_{i,j} = \langle l'_{i,j}, A'_{i,j,q}, H(D'_{i,j}) \rangle$, 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 necessary information for a user to determine whether $D'_{i,j}$ satisfies his query, which will be further illustrated shortly.

Second, the data collector chains $\{\phi_{i,j}\}_{j=1}^{n_i}$ using cryptographic hash functions to enable correctness verifications of query results. 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,

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

where $||$ denotes concatenation and $n_i \geq 0$. Note that if $n_i = 0$, we let $\phi_{i,1} = h_{i,1} = H(\chi)$.

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 $\mathrm{POI}_{i,j}$ 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 for the data set of every POI category.

2) Query Processing: The LBSP purchases the data sets from the data collector. For every POI category, 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.

Now we illustrate how the LBSP processes a top-k query, which includes the desired POI category, the interested attribute $q \in [1, \lambda]$ for ranking POIs, the query region $\mathcal{R}$, and k. It first searches $\{\mathcal{D}'_i\}_{i=1}^{M}$ to locate k POIs in $\mathcal{R}$ whose attribute-q ratings are among the highest k. Let kPOI denote these top-k data records ordered according to the attribute-q rating in the descending order. Next, the LBSP determines the zones either completely or partially covered by $\mathcal{R}$, denoted by $\mathcal{I} \subseteq \{1, \ldots, M\}$. In addition, let γ be the lowest attribute-q rating in kPOI and τ_i the number of POIs in zone i with attribute-q ratings $\geq \gamma$. 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 zone partially overlaping with $\mathcal{R}$ may have some POIs outside $\mathcal{R}$ but with attribute-q ratings $\geq \gamma$. We further define

$$X_{i,j} = \begin{cases} D'_{i,j} & \text{if } l'_{i,j} \in \mathcal{R}, \\ \phi_{i,j} & \text{otherwise,} \end{cases} \tag{2}$$

for all $i \in \mathcal{I}, j \in [1, n_i]$. In other words, $X_{i,j}$ equals the data record of $\mathrm{POI}_{i,j}$ if it is in $\mathcal{R}$ and its index otherwise.

The LBSP returns the following information $\mathcal{S}_i$ for each zone $i \in \mathcal{I}$ as part of the query response.

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

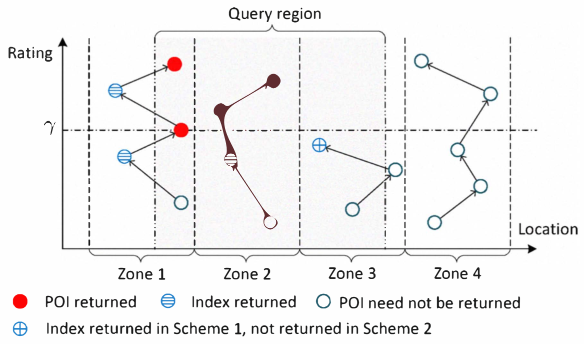


Fig. 2. An example for Scheme 1, where $M = 4$, $k = 4$, and the dots in zone i correspond to POI records $D'_{i,1}$ to $D'_{i,4}$ from top to bottom.

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.

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 V.

For authenticity verification, 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 $\mathcal{R}$ are explicitly included in $\{\mathcal{S}_i\}_{i\in\mathcal{I}}$, while those of the POIs in $\mathcal{R}$ 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. (1). 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.

For correctness verification, the user first checks if zones $\mathcal{I}$ encloses the query region $\mathcal{R}$. If so, he determines the lowest attribute-q rating γ among received data records whereby to check whether all the following conditions hold.

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

If so, the user considers the query result correct.

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 here from subscripts for brevity). Suppose that the user queries the POIs with the highest $k = 4$ attribute-q ratings 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 $\mathcal{S}_2 = \langle 2, D'_{2,1}, D'_{2,2}, \phi_{1,3}, h_{1,4}\rangle$ 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 correct.

B. Scheme 2

Scheme 1 requires the LBSP to return some information for every zone i that completely or partially overlaps with the query region even if zone i 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.

1) Data Preprocessing: In Scheme 2, the data collector partitions the original M zones into non-overlapping *macro zone*s, 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 $e \in [1, M/m]$.

Different from in Scheme 1, the data collector processes the data records in each zone along with the highest rating for the same attribute of other zones in the same macro zone. Without loss of generality, we take a macro zone e as an example to illustrate the preprocessing process. As in Scheme 1, the data collector first sorts $\mathcal{D}_i$ ($\forall i \in [1, M]$) 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. For every zone $i \in \mathcal{M}_e$, the data collector further generates $\{\mathcal{I}_{i,j}\}_{j=1}^{n_i+1}$, where $\mathcal{I}_{i,j} = \{s, A'_{s,1,q} | s \in \mathcal{M}_e \setminus \{i\}, A'_{i,j-1,q} < A'_{s,1,q} < A'_{i,j,q}\}$. 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 $|\bigcup_{j=1}^{n_i+1} \mathcal{I}_{i,j}| = |\mathcal{M}_e \setminus \{i\}|$. The data collector then computes an index $\phi_{i,j} = \langle l_{i,j}, \mathcal{I}_{i,j}, A'_{i,j,q}, H(\mathcal{I}_{i,j} || D'_{i,j})\rangle$ for all $j \in [1, n_i]$ and chains $\{\phi_{i,j}\}_{j=1}^{n_i}$ according to Eq. (1). Finally, it builds a Merkle hash tree over $\{h_{i,1}\}_{i=1}^{M}$ and signs the root as in Scheme 1. 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.

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 constructs a set kPOI containing the top-k data records in the query region $\mathcal{R}$ and also $\mathcal{I} \subseteq \{1, \ldots, M\}$ as the set of zones completely or partially covered by $\mathcal{R}$. The LBSP then determines γ as the lowest attribute-q rating in kPOI and τ_i as the number of POIs in zone $i \in \mathcal{I}$ with attribute-q ratings $\geq \gamma$. We redefine

$$X_{i,j} = \begin{cases} D'_{i,j} || \mathcal{I}_{i,j} & \text{if } l'_{i,j} \in \mathcal{R}, \\ \phi_{i,j} & \text{otherwise}, \end{cases} \tag{3}$$

for all $i \in \mathcal{I}, j \in [1, n_i]$. The final query result contains the following information $\mathcal{S}_i$ for each zone $i \in \mathcal{I}$.

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

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

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 each macro zone $e \in [1, M/m]$. We further require the LBSP to return one POI index from $\mathcal{M}'_e$ if necessary. There are two cases.

- If there exists zone $i \in \mathcal{M}'_e, \tau_i > 0$, nothing need be done because this case has been covered by Case 2 above.
- Otherwise, it must be true that $\tau_i = 0$ for all $i \in \mathcal{M}'_e$. Assuming that $A'_{j,1,q}$ is the highest attribute-q rating in zones $\mathcal{M}'_e$, the LBSP also adds $\mathcal{S}_j = \langle j, \phi_{j,1}, h_{j,2}\rangle$ to the query result.

Finally, if no POI in zones $\mathcal{I} \bigcap \mathcal{M}_e$ has attribute-q rating lower than γ, it must be true that $\mathcal{M}'_e$ is empty. It follows that $n_i = \tau_i$, for all $i \in \mathcal{M}_e \bigcap \mathcal{I}$. Then the LBSP need return $\mathcal{S}_i = \langle i \rangle$ for each $i \in \mathcal{M}_e \bigcap \mathcal{I}, n_i = \tau_i = 0$. 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 $\mathcal{T} = \bigcup_{i \in \mathcal{I}} \mathcal{T}_i$ and the data collector's signature on the qth Merkle root hash. 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, which can lead to much lower computation and communication overhead.

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.

The user first checks whether the query result contains some information for every macro zone $e \in [1, M/m]$ that overlaps with the query region $\mathcal{R}$. If so, he then determines the lowest attribute-q rating γ in the received POI records and verifies whether the three correctness conditions in Scheme 1 (see Section IV-B3) all hold. If so, he further determines τ_i and the relationship between n_i and τ_i based on the information format for zone i in the query result. Finally, he performs the following verifications for every macro zone e overlapping with $\mathcal{R}$.

- If there is any zone $i \in \mathcal{I} \bigcap \mathcal{M}_e$ with $\tau_i \in (0, n_i)$ (i.e., Case 2 in query processing), the user checks whether the query result contains a valid $\mathcal{S}_x$ corresponding to Case 1 or 2 in query processing for every zone $x \in$

$\mathcal{I}\bigcap\mathcal{M}_e\bigcap(\bigcup_{j=1}^{\tau_i+1}\mathcal{I}_{i,j})$ with $A'_{x,1,q} \geq \gamma > A'_{i,\tau_i+1,q}$. 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 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.
- If 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 $\mathcal{I}\bigcap\mathcal{M}_e$ lower than γ. The user verifies this by checking if $n_i = 0$ or $n_i = \tau_i > 0$ for each zone $i \in \mathcal{I}\bigcap\mathcal{M}_e$. If not, the query result is considered incorrect.

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.

C. Discussion

So far we have assumed that there are at least k POIs in the query region and that no POIs have the same rating for any attribute. This section discusses the impact on our schemes if these assumptions do not hold.

1) Insufficient POIs in the Query Region: If there are less than k POIs in the query region $\mathcal{R}$, any POI there satisfies the top-k query. Therefore, the LBSP need prove to the user that the query result contains every POI record in $\mathcal{R}$ by returning all the POIs in zones $\mathcal{I}$ which completely or partially overlap with $\mathcal{R}$. Take Scheme 1 as an example. On receiving a top-k query, the LBSP includes $\mathcal{S}_i = \langle i, X_{i,1}, \ldots, X_{i,n_i}\rangle$ for each zone $i \in \mathcal{I}$ in the query result, which the user can verify in the same way. Similar modifications can be made to Scheme 2 and are omitted here.

2) Multiple POIs with Equal Attribute Ratings: Due to the limited rating range, multiple POIs may have an equal rating for the same attribute. The tie can be easily broken by considering additional information for comparing POIs. For example, we can add the time of the last review into the index $\phi_{i,j}$ of any POI$_{i,j}$ in Schemes 1 and 2. In case there are multiple POIs with equal ratings, the one with the most recent review is preferred. The impact of such scenarios on our schemes is thus negligible.

V. Performance Analysis

In this section, we analyze Schemes 1 and 2 with regard to their efficacy 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.

A. Analysis of Scheme 1

The following theorem is for the efficacy of Scheme 1.

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

Proof: We first show that the user can detect any inauthentic query result containing fake POI records or indexes. Recall that the hash of every record is embedded in its index, and adjacent indexes are chained together. Therefore, any fake record or index will make the user compute an invalid leaf node for the Merkle hash tree, which can be immediately detected by the user after verifying the data collector's non-forgeable signature on the Merkle root hash.

Now we show that incorrect query results can also be detected. The key rationale is that if the LBSP returns $X_{i,j} = D'_{i,j}$ or $\phi_{i,j}$, he must also return $X_{i,1}, \cdots, X_{i,j-1}$ for the query result to pass the authenticity check. Let kPOI denote the correct top-k records with the lowest attribute rating γ and $\widetilde{k\text{POI}}$ the incorrect top-k records with the lowest attribute rating $\tilde{\gamma} \neq \gamma$. If $\tilde{\gamma} < \gamma$, there must be at least k POI records with attribute ratings higher than $\tilde{\gamma}$ in the query region, which should all be returned for $\widetilde{k\text{POI}}$ to pass the authenticity check. This apparently contradicts with the fact that $\tilde{\gamma}$ is the lowest rating in $\widetilde{k\text{POI}}$. If $\tilde{\gamma} > \gamma$ instead, $\widetilde{k\text{POI}}$ must contain at least one POI record outside the query region with a rating higher than γ, which can be directly detected. ■

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 and also a certain number of hash computations given below.

Theorem 2. *The expected number of hash computations needed to verify a query result under Scheme 1 is given by*

$$\mathsf{E}[N_{hash,1}] = k+|\mathcal{I}|\cdot(\mathsf{E}[\psi]+1)+\sum_{j=1}^{d-1} 2^{j-1}(1-(1-2^{-(j-1)})^{|\mathcal{I}|}) \,, \tag{4}$$

where $\mathsf{E}[\psi] = \frac{kn}{\delta n+1}$.

Due to space constraints, we refer readers to our online technical report [24] for the proofs of Theorem 2 and subsequent Theorems 3, 5, and 6.

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 $L_{\mathrm{loc}}+L_{\mathrm{r}}+L_{\mathrm{h}}$ bits for each of M zones and a Merkle hash tree of $(M-1)L_{\mathrm{h}}$ bits. The extra communication overhead in bits per POI category Scheme 1 incurs between the data collector and LBSP is thus

$$\mathsf{S}_1 = \lambda(Mn(L_{\mathrm{loc}}+L_{\mathrm{r}}+L_{\mathrm{h}})+(M-1)L_{\mathrm{h}}+L_{\mathrm{sig}}). \tag{5}$$

The following theorem is about the extra communication overhead associated with sending authenticity and correctness proofs of a top-k query result from the LBSP to the user.

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

$$\begin{aligned}\mathsf{E}[\mathsf{T}_1] = (|\mathcal{I}|\cdot\frac{(k+\delta)n+1}{\delta n+1}-k)(L_{\mathrm{loc}}+L_{\mathrm{r}}+L_{\mathrm{h}})+|\mathcal{I}|\cdot d \\ +\sum_{j=1}^{d-1}2^{j}(1-(1-2^{-j})^{|\mathcal{I}|})L_{\mathrm{h}}+L_{\mathrm{sig}} \,.\end{aligned} \tag{6}$$

B. Analysis of Scheme 2

The following theorem is about the efficacy of Scheme 2.

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

Proof: As in the proof of Theorem 1, the user can easily detect any inauthentic query result containing fake POI records or indexes. The proof is omitted here for brevity.

Now assume that the LBSP returns an authentic but incorrect query result, from which the user derives incorrect top-k POI records $\widetilde{k\mathrm{POI}}$ with the lowest attribute rating $\tilde{\gamma}$. Again, let γ denote the correct lowest top-k attribute rating. If $\tilde{\gamma}>\gamma$, we can apply the same argument in proving Theorem 1 to show that the user can discover that at least one POI in $\widetilde{k\mathrm{POI}}$ is outside the query region. If $\tilde{\gamma}<\gamma$, the LBSP should have deleted at least one POI record in the query region with an attribute rating higher than $\tilde{\gamma}$. Suppose that the LBSP did not return $D'_{i,j}$ with $A'_{i,j,q}>\tilde{\gamma}$ in the macro zone e. There are two cases.

- If the LBSP returned nothing from zone i, it must have returned at least one index with a rating $<\tilde{\gamma}$ in the macro zone e, say ϕ_{i_1,j_1}. It follows that $A'_{i,1,q}>A'_{i,j,q}>\tilde{\gamma}>A'_{i_1,j_1,q}$ and $\langle j, A'_{i,1,q}\rangle \in \bigcup_{x=1}^{j_1}\mathcal{I}_{i_1,x}$, from which the user knows the LBSP omitted valid information from zone i.
- If the LBSP returned some POI records or indexes for zone i, it must have returned $X_{i,1},\ldots,X_{i,\tilde{\tau}_i+1}$ to pass the authenticity check. Since $A'_{i,j,q}>\tilde{\gamma}$, we have $j<\tilde{\tau}_i$, and $D'_{i,j}$ or $\phi_{i,j}$ must have been returned, leading to a contradiction.

Therefore, the user can detect any incorrect query result. ■

Scheme 2 incurs the same computation overhead to the data collector and LBSP as Scheme 1, which has been analyzed before. The following theorem is about the number of hash computations needed to verify a query result.

Theorem 5. *The expected number of hash computations needed to verify a query result under Scheme 2 is given by*

$$\mathsf{E}[N_{hash,2}] = |\mathcal{I}|\mu_1+\sum_{j=1}^{d-1}2^{j-1}(1-(1-2^{-(j-1)})^{|\mathcal{I}|(1-\mu_2^n)}) \,, \tag{7}$$

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

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 $2(m-1)(d+L_{\mathrm{r}})$ bits for attribute q in contrast to Scheme 1. The communication overhead per POI category Scheme 2 incurs between the data collector and LBSP is thus

$$\mathsf{S}_2=\mathsf{S}_1+2(m-1)\lambda(d+L_{\mathrm{r}}) \,, \tag{8}$$

where S_1 is given in Eq. (5). We also have the following theorem about the extra communication overhead between the LBSP and user.

Theorem 6. *Assuming that the query region comprises $\breve{m}$ 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,*

$$\begin{aligned}\mathsf{T}_2 \le{}& \breve{m}(1-\mu^n)d+\breve{m}(n-n\mu+1-\mu^n)(L_{\mathrm{loc}}+L_{\mathrm{r}}+L_{\mathrm{h}}) \\ &+(\breve{m}(1-\mu^n)+\sum_{j=1}^{d-1}2^j(1-(1-2^{-j})^{\breve{m}(1-\mu^n)}))L_{\mathrm{h}} \\ &+\breve{m}(1-\mu^n)(m-\breve{m})(1-(\frac{n-\nu}{n+1})^n)(d+L_{\mathrm{r}}) \\ &+t(t-1)(d+L_{\mathrm{r}})+L_{\mathrm{sig}} \,.\end{aligned} \tag{9}$$

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

TABLE I
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		

VI. SIMULATION RESULTS

In this section, we evaluate our schemes using simulations. We assume that the data set covers 100×100 unit square zones of equal size and that there are 100 POIs uniformly distributed in each zone. We simulate the following two types of queries.

- **Type-1 queries**: $\mathcal{R}$ exactly covers an integer number of zones, which means that $\mathcal{I} = \mathcal{R}$ and $|\mathcal{I}| = \delta$.
- **Type-2 queries**: $\mathcal{R}$ is a circle of radius r centered at a random location, which means that $\mathcal{I} > \mathcal{R}$ and $|\mathcal{I}| > \delta$.

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 I, unless stated otherwise.

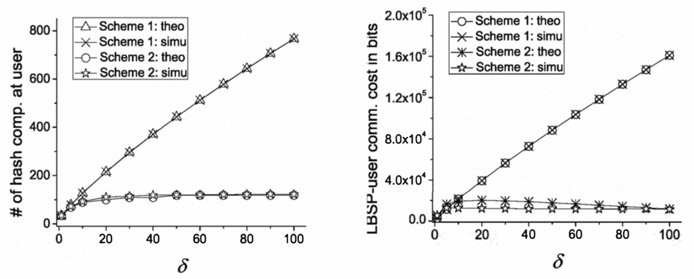
A. Type-1 Queries: $|\mathcal{I}| = \delta$

For this set of simulations, we let the query region $\mathcal{R}$ formed by δ zones randomly chosen from the same macro zone.

Fig. 3(a) 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 $\mathcal{R}$ 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 $\mathcal{R}$. Therefore, Scheme 2 incurs lower computation overhead on the user for small k and large δ.

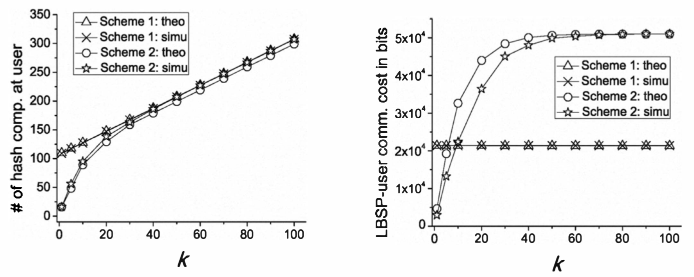
Fig. 3(b) shows the impact of δ on the LBSP-user communication overhead for $k = 5$. It is clear that the simulation results are always below the corresponding theoretical upper bounds. As in Fig. 3(a), we can also observe that the LBSP-user communication overhead in Scheme 1 always increases with δ 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 information for other zones in the same macro zone with attribute ratings higher than any top-k rating.

Fig. 4(a) 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 $\mathcal{R}$ without a top-k POI quickly decreases for sufficiently large k.



(a) computation cost (b) communication cost

Fig. 3. The impact of δ for Type-1 queries, where $k = 5$.



(a) computation cost (b) communication cost

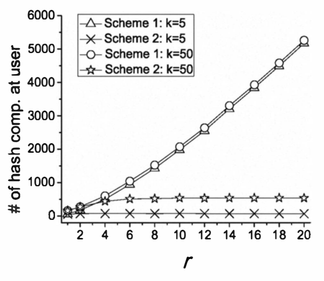
Fig. 4. The impact of k for Type-1 queries, where $\delta = 10$.

Fig. 4(b) 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.

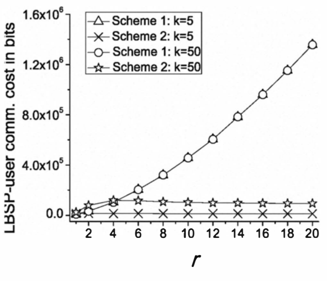
B. Type-2 Queries: $|\mathcal{I}| > \delta$

For this set of simulations, we simulate a circular query region with radius r centered at a random location and only report the simulation results for simplicity.

Figs. 5(a) and 5(b) show the impact of query radius r on the user's computation overhead and the LBSP-user communication overhead, respectively, for $k = 5$ or 50. Note that $\delta = \pi r^2$ increases quadratically with r, so does the number of zones partially or completely overlapping with the query region $\mathcal{R}$. It is thus not surprising to see that the user's computation overhead and the LBSP-user communication overhead both increase as r increases under Scheme 1. In contrast, both

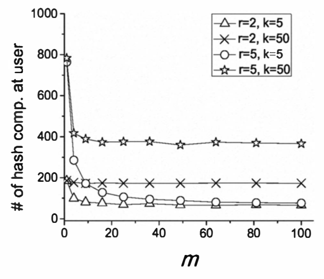


(a) computation cost

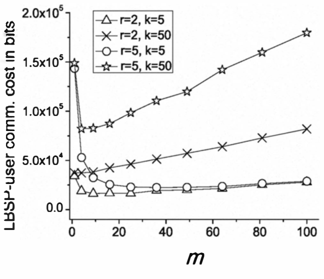


(b) communication cost

Fig. 5. The impact of query radius r for Type-2 queries.



(a) computation cost



(b) communication cost

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

metrics are relatively insensitive to r under Scheme 2 as the number of zones having at least one top-k POI is at most k.

C. Impact of m on Scheme 2

Now we illustrate the impact of m, the number of zones in each macro zone, on Scheme 2. For simplicity, we show the simulation results for Type-2 queries only.

Fig. 6(a) 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 macro zone overlapping with the query region $\mathcal{R}$ that has no top-k POI. When k is small and $\mathcal{R}$ is large, most zones in $\mathcal{R}$ 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 $\mathcal{R}$ approaches a constant, leading to relatively stable computation overhead.

Fig. 6(b) 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.

VII. Conclusion

In this paper, we have proposed two 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 can enable users to verify the authenticity and correctness of any location-based top-k query results. The efficacy and efficiency of our schemes are thoroughly analyzed and evaluated through detailed simulation.

Acknowledgement

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

References

[1] 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.

[2] H. Yu, P. Gibbons, M. Kaminsky, and F. Xiao, "SybilLimit: A near-optimal social network defense against sybil attacks," *IEEE/ACM Transactions on Networking*, vol. 18, no. 3, pp. 885–898, June 2010.

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

[4] W.-S. Ku, L. Hu, C. Shahabi, and H. Wang, "Query integrity assurance of location-based services accessing outsourced spatial databases," in *SSTD'09*, Aalborg, Denmark, July 2009.

[5] 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.

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

[7] 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.

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

[9] 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.

[10] 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.

[11] 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.

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

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

[14] Y. Yang *et al.*, "Spatial outsourcing for location-based services," in *IEEE ICDE*, Cancún, México, Apr. 2008, pp. 1082–1091.

[15] M. Yiu *et al.*, "Efficient verification of shortest path search via authenticated hints," in *IEEE ICDE*, Long Beach, CA, Mar. 2010, pp. 237–248.

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

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

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

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

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

[21] 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.

[22] R. Zhang *et al.*, "Verifiable fine-grained top-k queries in tiered sensor networks," in *INFOCOM'10*, San Diego, CA, Mar. 2010.

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

[24] R. Zhang, Y. Zhang, and C. Zhang, "Secure top-k query processing via untrusted location-based service providers," Tech. Rep., July 2011, http://wins.lab.asu.edu/files/topk-tr.pdf.