Private Retrieval of POI Details in Top-K Queries

Wisam Eltarjaman, Rinku Dewri and Ramakrishna Thurimella
Colorado Research Institute for Security and Privacy
Department of Computer Science, University of Denver, Denver, CO 80208,USA

***Abstract*—Location privacy preservation algorithms in the context of location-based services have evolved in the recent years. However, a majority of the proposals assume that points of interests (POI) are ranked only by distance, and demand extensive architectural changes. As a result, a significant gap remains between academic proposals and the industry standard of implementing location based services. Recent advances in mobile device capabilities, more specifically in their computational power and energy efficiency, have opened the possibility of engaging the client hardware more actively in the execution of a privacy algorithm, thereby relaxing strong dependencies on trusted third parties or the service provider. With this motivation, we propose a novel privacy algorithm that determines the most prominent result set through operations restricted to the client device, thereby limiting the communication of precise location information to the service provider. The service provider only acts as a data source, and is required to perform operations that are within existing industry norms. By measuring the privacy offered by the algorithm under a formal threat model, we demonstrate its robustness and practicability, and supplement our conclusions with empirical evidence.**

***Index Terms*—Location privacy, top-K queries, mobile search**

I. Introduction

Location-based services (LBSs) have become a part of everyday life in most societies, and are used extensively by anyone with a smart phone. Along with these very useful services comes the ability of service providers to track user locations accurately, and use this information for commercial purposes. Location privacy loss has been a recognized problem for several years [1], and hence, several attempts have been made by the research community to design algorithms that preserve location privacy, without degrading the utility of LBSs. These location privacy preserving mechanisms (LPPMs) come in various flavors. Initially, LPPMs were one-size-fit-all solutions that offered enhanced privacy to all users of the system at an equal level [2], [3], [4]. Soon, researchers realized that privacy needs are different for different users, and even for the same user, privacy requirements are different in different situations [5]. The most recent LPPMs provide to their users configurable levels of privacy that is balanced against quality of service (QoS) [6], [7].

Out of the several types of LBSs in use today, direct or indirect search for points-of-interest (POIs), and navigating to the most suitable of them, is arguably the most widely utilized application. This particular application warrants deeper study from the privacy research community as it reveals several facets of a user's lifestyle beyond just the location. A sophisticated adversary or a semi-trusted service provider can use this information to deduce user preferences and future locations, in addition to the current location [8].

Most earlier algorithms in private POI search not only required a trusted third party (TTP), but also required that this TTP has the knowledge of the locations of all users that subscribe to the anonymization service, thus creating a single server bottleneck. This requirement of a TTP has a significant impact on the QoS as users often have to wait till other users accumulate. Some algorithms even propose discarding the user's query if it cannot be answered by the TTP within a specified time period [9]. The research community soon realized this bottleneck and, since then, has extensively proposed novel methods to eliminate the need for a TTP.

Despite the tremendous effort put forward by the community, mass adoption of location privacy controls are yet to be seen. A potential reason for this could be the implicit requirement for architectural changes that a service provider has to undergo, and the possible adverse impact it can have on the quality of service. For instance, methods utilizing cloaking regions may require transmission of massive amounts of POI data, while formal methods such as private information retrieval puts a burden on the service provider to install trusted/incorruptible hardware and perform expensive computations. It must be said that researchers are very much aware of these limitations, and the proposals are very much driven by the need to perform sophisticated computations and the lack of an efficient platform to do so, except for a TTP or the LBS servers. Fortunately, the landscape has changed significantly in the past few years. Mobile devices (the point of origin of a request) are no longer a simple piece of radio hardware, but full-fledged computing platforms, often faster than desktop servers from a decade ago. It is therefore reasonable to attempt novel location privacy protection mechanisms that incorporate this new computation node.

Our earlier work demonstrated that, with a minor change in the control flow of existing LBS server software, POI search can be performed without requiring the user to transmit precise location data outside the device [10]. The work assumed a single snapshot query model; therefore, the privacy guarantees do not hold when users make multiple queries in a short period of time. Motivated by the potential applicability of modern mobile hardware for location privacy preservation, we continued our work to facilitate a privacy preserving multi-query system, thereby providing a complete private POI search paradigm [11]. We showed how the privacy algorithm executed in the mobile device for single query systems is susceptible to localization attacks (determining where the user is at a specific point in time) when used in the context of multiple queries. To eliminate this concern, we proposed a new client side privacy algorithm to retrieve POIs, and analyzed the inference risk of the algorithm under a Bayesian adversary model.

The work in this study supplements our earlier works, with the following differences: this manuscript (i) includes a more detailed discussion on the adversarial model, and the privacy algorithm for the multi-query scenario, (ii) uses an enhanced adversarial model with transition matrices, information on areas that cannot be occupied by a person, and trajectories between frequent source/destination locations; the resulting empirical results are therefore different here, (iii) uses two different privacy evaluation metrics, and (iv) provides a comparative evaluation of the approach with the state-of-the-art geo-indistinguishability approach [12].

In the next section, we provide a brief overview of the work done in this field and show the evolution of LPPMs over time. In Section III, we elaborate on the local search architecture assumed in this work. We also provide a brief background on the geo-indistinguishability approach in this section. Section IV presents the adversary model, and discusses the inference mechanism assumed to be in use by the adversary. Section V presents the experimental setup and the metrics used to evaluate the privacy mechanisms. Section VI elaborates on the single query scenario and provides runtime evaluations of performing search operations on a mobile device. Section VII explores the multiple query scenario. We finally conclude in Section VIII.

II. Prior Work

A. Anonymity sets

Location privacy has earlier been achieved through the use of obfuscation and dummy queries. A user can hide her actual query in a set of dummy queries and achieve location privacy [2]. Gruteser and Grunwald [3] proposed the use of spatial and temporal cloaking to obfuscate user locations. The cloaking is performed at a trusted third party site. Individual preferences in terms of temporal and spatial tolerances can also be incorporated during such cloaking [5]. Enforcing properties such as k-anonymity ensures that users will not be uniquely located inside a region in a given period of time. Multiple other suggestions are available on how the cloaking region should be formed. Bamba et al. enforced a location l-diversity requirement where the number of still-object counts must also be above a user-specified threshold [13]; Liu et al. proposed that a minimum level of entropy should also be maintained in the queries originating from the cloaking region [4]; Mokbel et al. presented how different query formats can be supported using cloaking regions [14]; Dewri et al. proposed enforcing query m-invariance in cloaking regions [15]; Shin et al. anonymized the profile of the user using k-anonymity [16]; Riboni et al. proposed smoothing out differences in the distribution of query parameters [17]. Ghinita et al. proposed a decentralized architecture where mobile nodes utilize a distributed protocol to self-organize into a fault-tolerant overlay network, from which a k-anonymous cloaking set of users can be determined [18]. Kalnis et al. proposed that all obfuscation methods should satisfy the reciprocity property [19] in order to prevent inversion attacks where knowledge of the underlying anonymizing algorithm can be used to identify the actual user [20].

B. Beyond anonymity sets

Moving beyond anonymity sets, Khoshgozaran et al. proposed a protocol where K-nearest neighbor queries are reduced to a set of private block retrieval operations on a database [21], [22]. These retrievals can be performed using a tamper-resistant co-processor located at the server so that the content provider is oblivious of the retrieved blocks. In addition to the trusted hardware requirement, shuffling operations involved in these protocols often have to be performed on the database, requiring preprocessing or the availability of special APIs at the server side. The absence of these required components in existing services makes these approaches currently unsuitable for mainstream adoption. While recent proposals have attempted to address this challenge for K-nearest neighbor queries [23], private retrieval protocols for top-K queries are still a challenge. The approach presented in this work is ready for deployment using available APIs in most major location based service providers.

Most anonymity set based proposals suffer from the requirement to specify privacy parameters that are not intuitive or difficult to gauge. Xu and Cai explored this problem by treating privacy as a feeling-based property and proposed using the popularity of a public region as the privacy level [24]. Soriano et al. showed that the privacy assurances of this model do not hold when the adversary possesses footprint knowledge on the spatial regions over time [25]. Niu et al. recently revisited the use of dummy queries with the objective of addressing side information that the adversary may have on query probabilities from different locations [26]. In a subsequent work, the authors demonstrated that caching of query results can help improve the privacy in dummy query models [27]. These works are driven by an entropy-based privacy metric. Much like the result reuse approach in caching, Shokri et al. proposed a collaborative model where users can retrieve search results from their mobile peers, whenever possible, thereby not requiring location disclosures to the service provider [28]. Infrastructure-dependent architectures are also being currently explored in some domains, where centrally operated access points generate digitally signed location and distance proofs for the user [29].

Shokri et al. argued that location privacy should be quantified based on the expected estimation error of an adversary [8]. They provided a method to arrive at different types of inferences regarding a user's location based on a known mobility profile of the user. Using methods of likelihood estimations, the authors showed that above measures such as the anonymity set size or entropy do not correctly quantify the privacy enforced by the method [30]. Moreover, all these approaches treat privacy as an immutable property that must be strictly enforced. In reality, most of us like to weigh location privacy with the prospective gains in service before skewing our preferences for one. In this context, this work provides an intuitive privacy parameter (a region) and assesses the privacy level under the assumption of a strong adversarial model. The evaluation is performed in terms of metrics operating directly on the probabilistic inferences that such an adversary can draw from a Bayesian analysis.

C. Differential privacy

Dewri introduced the idea of merging a well-known form of privacy in databases, namely differential privacy, and k-anonymity [31]. Under this model, an anonymity set of size k is first formed and then an obfuscated location is generated such that the probabilities of reporting this location from any of the k locations are close to each other. Andrés et al. improved this approach by proposing a new model called geo-indistinguishability, where the dependence on the anonymity set is removed, and a privacy radius is introduced as a parameter [12]. This integration of location privacy and differential privacy remains the state-of-the-art in privacy models for location privacy protection. The primary drawback of these models is that the choice of the obfuscated location is driven only by privacy requirements, and no attempt is made to accommodate its impact on the query results. We differentiate our work in this dimension by offering an approach that does not impact the accuracy of results, and adapts the privacy level when changes in the result set are limited across arbitrary distances.

D. Privacy-accuracy trade-off

Examination of the privacy/accuracy trade-off in location-based applications is rare. Shokri et al. explored an optimal location obfuscation method that can hinder privacy attacks and provide the best service quality, essentially targeting an equilibrium solution in a Stackelberg Bayesian game [32]. They compute quality loss as the average dissimilarity in service quality between the user's true location and a pseudo-location. Privacy is computed as the expected error of the adversary in an inference attack. Along similar lines, Bordenabe et al. provided a mechanism to minimize the service quality loss for a given degree of geo-indistinguishability [7]. Similar to most of the earlier works, both of these works assume that service quality in an application is directly proportional to the distance between the pseudo-location and the true location; however, this assumption hardly holds in a local search application (and others as well) where the quality of search results depends on multiple factors other than distance.

To the best of our knowledge, our prior work is the only known attempt to consider arbitrary ranking functions for local search results, instead of the commonly assumed K-nearest neighbors [6]. We proposed the use of multi-dimensional scaling to do a lossy compression of the result set similarity information into a RGB image. A client then uses the image to decide an obfuscated location with a tolerable loss in service quality. This work is our attempt to eliminate the requirement for the server to produce metadata for privacy decisions, and instead employ the computational power available in a mobile device to make informative decisions.

III. Local Search Architecture

A typical POI search transaction starts with a user searching for a POI by using some keyword. The LBS provider receives this query and returns a list of POIs that match the query. When this list is generated based only on distance, the problem can be reduced to finding K nearest neighbors, K being the number of POIs in the list. But, if one were to observe most popular POI search providers these days, the list of POIs is not necessarily sorted only on the distance. In fact, most popular providers, e.g. Google [33], use a combination of distance and other criteria.

The service provider returns the list already sorted, and short-listed (typically 10-20 items), to the requesting application. One has to note that privacy in this scenario is based entirely on the privacy policy of the service provider. If the service provider is assumed to be *semi-honest,* i.e. honest but curious, then location privacy of the user is completely lost. In this case, one of the LPPMs discussed in Section II needs to be implemented to enhance the location privacy of the user. If feasibility demands ruling out algorithms that either require a TTP, or large changes at the LBS server, or ones that rank the POIs only based on the distance from the user location, then there is a shortage of appropriate techniques. Therefore, we seek alternative architectures where users can continue to retrieve information on POIs of importance, albeit without requiring the disclosure of accurate location information.

In the following, we assume that a large geographical area is modeled as a $Z \times Z$ square grid of cells. The user is interested in POIs contained in this large area. A *cell* defines the smallest distance that a user has to move to be recognized as existing in a different location, i.e. as long as the user moves within the boundaries of a cell, she will be considered as staying in the same location. For example, we use a cell side length of 100 meters in this study.

A. Geo-indistinguishability

Andrés et al. introduced the geo-indistinguishability principle to generate perturbed locations for location-based POI search [12]. The principle provides probabilistic limits on the inferential advantage that an adversary can gain with knowledge of the perturbed location and the perturbation mechanism. Given a user cell c and a privacy parameter ϵ, the mechanism adds random noise drawn from a planar Laplace (extension of the Laplace distribution to two dimensions) distribution to the user's location and generates the perturbed location z. Doing so provides the guarantee that

$$\frac{Pr(z|c_1)}{Pr(z|c_2)} \leq e^{\epsilon d(c_1,c_2)}, \tag{1}$$

where c_1 and c_2 are any two cells, and d is a distance function. To retrieve POI details, the mechanism then issues a query using z and an area of retrieval (AOR) around z. All POIs inside the AOR are retrieved from the service provider. Andrés et al. provide confidence bounds on the size of the AOR that also includes a specific area around the actual location of the user (called the area of interest, or AOI). The user can then choose a POI from the retrieved set depending on how far it is from her location.

B. A privacy supportive architecture

We use a different request-response architecture to support the retrieval of POIs based on distance, as well as their importance. Under this architecture, the client first determines

a large geographical area (say $500km^2$) that includes the user's location, and sends the coordinates of this area along with the search keywords to the server. An example of this query format will be a query such as "asian gift shop in Los Angeles, CA," as is supported by the Radar Search method of the Google Places API. Effectively, the user initiates a generic query for, say, a large section of a city. The server finds the list of matching POIs within the area from its database, and sends back only the locations and *prominence* values for the obtained POIs. The prominence value here is determined by the service provider based on criteria such as user reviews, reference counts, and financial gain, among others. POI search is not a typical K-nearest-neighbor search when prominence values are involved. We emphasize that the transmission of location and prominence values is not equivalent to downloading the entire POI database to the client, since no details about the POIs are available to the client at this time. These details include features such as business name, street address, reputation, user reviews, services offered, and others. The transmission of only location and prominence data (and no feature data) at this stage also reduces the bandwidth requirement of the technique by preventing the download of large fractions of the POI database.

The client application can utilize the location and the prominence data to locally rank the received list of POIs. By doing so, the client can determine the most highly ranked K POIs (*top-K POI*s) for the user, and request details on those POIs from the server. Note that a POI present in the top-K set for the user may be outside the AOR in the geo-indistinguishability approach.

However, this straightforward method is susceptible to inversion attacks, where an adversary can compute the top-K POIs of every cell, match them to the set requested by the client, and thereby infer likely cells where the user could be located. In order to ensure that the number of inferred cells under such an attack is lower bounded, we need to ensure that the sets requested by the user contain the top-K POIs corresponding to multiple cells. The user can cache POI details and generate requests for POIs only if it has not been done as part of an earlier query. In this case, we refer to the set of POIs for which details are requested in a query as the *interest set I* of the query. One approach to compute an interest set is to compute the union of the top-K POIs of multiple cells. For example, we consider a region of $b \times b$ cells containing the user's location, and form the interest set from the top-K sets of all cells in the region (minus any that has been retrieved earlier). This method takes advantage of the fact that nearby cells will often have very similar top-K POIs, in which case, the size of the interest set will not grow linearly to the number of cells in the region. An inversion attack will also produce an area at least as large as $b \times b$ cells.

C. Readiness

Much of the client-server programming interfaces necessary to implement the proposed method are readily available today. Here we discuss the interfaces available in the Google Places API that can be adopted to implement the protocol. The Google Places API Radar Search Service allows an application developer to search and retrieve information for up to 200 places at once, but with less detail than is typically returned in other forms of search. A request is made using a HTTP URL, and can include the query keyword, a location, and a radius. For the method proposed in this work, the location used may be a popular landmark, or simply the center of a randomly generated large box that includes the user. The maximum allowed radius is $50km$. The result of the request is returned either as a JSON object or an XML document, which includes the matching POIs' *geometry* (latitude and longitude)*, place_id* (a unique identifier of the POI) and *rating*, among other metadata. Although the developer can use the place rating as a prominence value, the exact value used by Google is not yet contained in this result. Details of POIs in the interest set can be obtained using a Place Details request. Such a request returns detailed information about a place identified using a *place_id*. The returned JSON object or XML document includes data such as the address of the POI, current events happening there, phone number, opening hours, photos, price level of services offered, user reviews, the POI's rating, and the website of the business, among other things. The availability of programming interfaces such as these makes the proposed privacy preserving POI search architecture feasible in the current market.

We next discuss the assumed adversary model in this study, and then present our methodology to compute an interest set.

IV. Adversary Model

Following the standard practice in the literature, we assume that the adversary knows the LPPM that is being utilized by the user. The adversary's goal is to determine the cell the user is in, based on the output produced by the LPPM, also known as a *localization attack*. To model an adversary in realistic terms, we assume that he has the following capabilities. The adversary:

- has the ability to eavesdrop and observe the contents of all queries;
- is aware of which LPPM is being used, and knows the details of the algorithm and its associated parameters;
- has coarse information on the movement patterns of the user;
- can accurately estimate the maximum number of cells that a user can move in a given time period;
- knows the map of the geographical area and has access to the POI database.

The adversary's goal is to narrow down the user's location to a specific cell. Short of that, the adversary tries to determine the likelihood of existence of the user in a particular cell by calculating a probability distribution of the user's existence in each cell. Let us consider time in unit increments, $t = 0, 1, 2, ...$; each increment is equal to the time required to move by one cell. The timestamp of a query always aligns with one of these time steps. Let Φ_t denote the distribution computed by the adversary at the end of time step t. If a query is not made at a particular time step t, then the adversary only knows the distribution at the end of the previous time step (Φ_{t-1}) and the

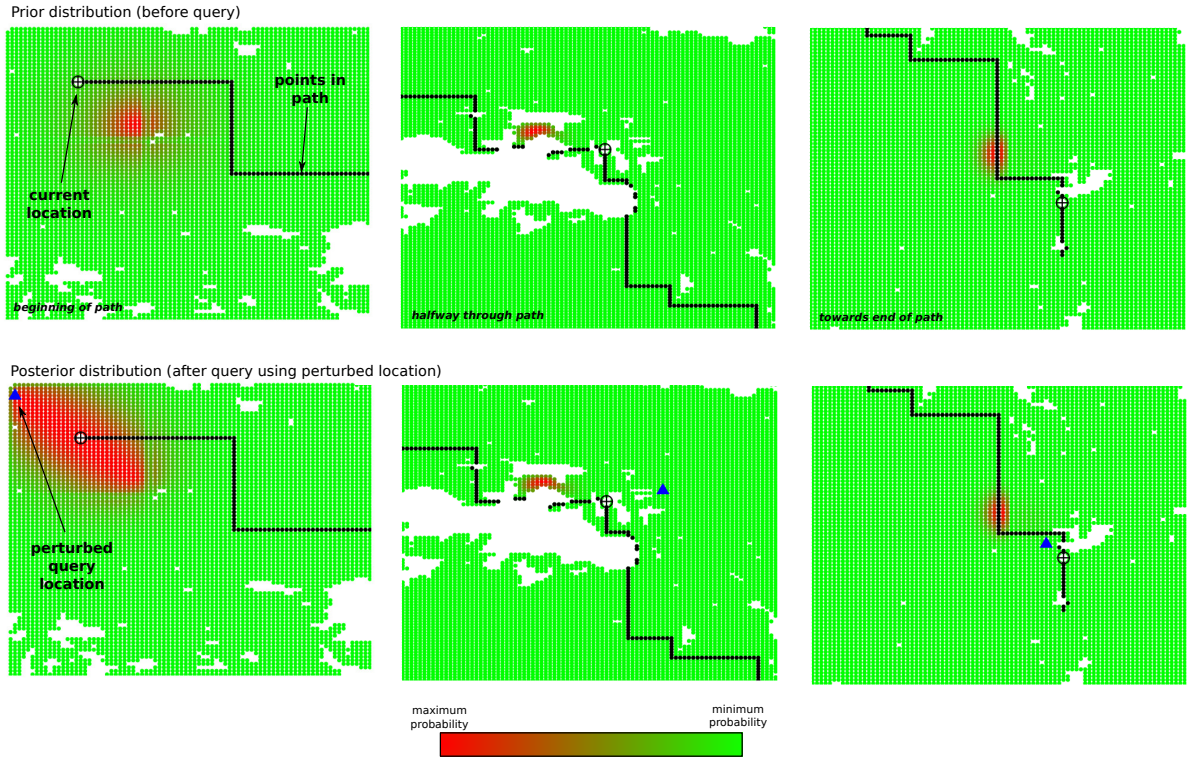


Fig. 1: Prior and posterior distribution after queries along a path. Darker (red) regions signify higher probability mass. White regions are unreachable cells.

movement pattern of the user. Otherwise, the adversary also knows the output from the LPPM (such as a perturbed location or an interest set). In both cases, the adversary can refine the prior knowledge to arrive at the distribution Φ_t. Note that Φ_0 models the knowledge of the adversary before any queries are issued to the LBS. A common form of this knowledge is a uniform distribution spread over a subset of cells in the grid, or the stationary distribution corresponding to a Markov model of the user's movement patterns. It is also reasonable to assume that Φ_o spreads over an area larger than the smallest area inferable from an LPPM (e.g. the AOR or the region used for interest set generation); otherwise, the background knowledge of the adversary is already stronger than the guarantees of the LPPM. We next formalize this inferencing process.

A. First query

Consider a user that first uses the LPPM at time $t = 1$. We denote by O_t the output generated by the LPPM during its use at time t. For geo-indistinguishability, the output signifies the perturbed location z, and for our method, this output signifies the interest set. Under a Bayesian adversarial model, an adversary can refine the prior distribution Φ_o and obtain the posterior distribution Φ_1 using Bayes rule.

$$\Phi_1(c) = Pr(c \mid O_1) = \frac{Pr(O_1|c)\Phi_0(c)}{\sum_{c'} Pr(O_1|c')\Phi_0(c')}, \qquad (2)$$

where c is any cell in the $Z \times Z$ grid. For geo-indistinguishability, $Pr(O_1|c)$ can be computed from the planar Laplace density function [12]. In our approach, this probability depends on the exact mechanism used to generate the region from which the interest set is derived. We revisit this computation in Sections VI-D and VII-C.

B. User movement

After the first query, the user starts moving along a path and uses the LPPM to retrieve POIs at time $t > 1$. A summary of the user's movement patterns is available to the adversary in the form of a *transition matrix*. A transition matrix $\mathcal{T}(A_i, A_j)$ provides the probability of the user moving from one area A_i in the grid to another area A_j. For example, the set of cells can be divided into non-overlapping areas of $b \times b$ cells, and the adversary's knowledge of the user transitioning between these areas is encoded in the transition matrix. A transition matrix can be extracted from available traces of a user's movement [8]. We consider the transition probability between individual cells to be directly proportional to the transition probability between the areas to which they belong. If c_s and c_d are two cells in areas A_i and A_j respectively, we compute the cell-wise transition probability matrix as

$$Pr(c_s \rightarrow c_d) = \frac{Pr(A_i \rightarrow A_j)}{b^2 \times b^2} = \frac{\mathcal{T}(A_i, A_j)}{b^4}. \qquad (3)$$

This computation assumes that all cells in an area are habitable; it is easy to accommodate the case of some cells being inhabitable by modifying the proportionality constant $\frac{1}{b^4}$.

C. Queries on the path

Let t_l denote the time step when the user last used the LPPM, and the current time be $t = t_l + \Delta$. The LPPM

outputs O_t at this time step. Before using this new observation, the adversary refines his location distribution using the known movement model of the user. Starting with the notation $\lambda_0 = \Phi_{t_l}$, the adversary first obtains λ_Δ by iteratively applying the following expression.

$$\lambda_k(c) = \sum_{c'} \left(\lambda_{k-1}(c') \times Pr(c' \rightarrow c)\right). \tag{4}$$

This computation updates the adversary's last known distribution with information based only on the movement patterns of the user. This is same as the *forward variable* in a typical forward-backward algorithm. In the right hand side of the equation, $\lambda_{k-1}(c')$ is the probability of the user being at cell c', and $Pr(c' \rightarrow c)$ is the probability of the user transitioning into cell c from c'. By considering all cells c', the equation computes the probability of the user moving into cell c in one time step. In the absence of any output from the LPPM, $\Phi_t = \lambda_\Delta$; otherwise, a final update can be performed using the output O_t, similar to as in the first query.

$$\Phi_t(c) = Pr(c|O_t) = \frac{Pr(O_t|c)\lambda_\Delta(c)}{\sum_{c'} Pr(O_t|c')\lambda_\Delta(c')}. \tag{5}$$

Fig. 1 shows snapshots of the inference process of the adversary at different points in time. The user in this case uses an LPPM to generate a perturbed location by adding planar Laplace distributed noise, and makes queries using this location. The figure depicts how the location of the perturbed query point (in addition to knowledge of the perturbation mechanism and the user's transition matrix) helps the adversary refine the prior distribution. As seen in the beginning of the path, the refinement resulted in good approximation of the user's actual location. As the user moves, the distribution becomes more concentrated, and refinements depend on where the perturbed point is generated.

Given all outputs generated by a LPPM along an entire path, the posterior distribution at a time step can be better refined by using knowledge of what output was produced after that point in time (by using a backward variable). We do not perform this refinement in the computation of Φ_t. As such, the accuracy of the adversary's method in estimating intermediate locations of the user (the path) is assumed to be not critical; it is the final destination that we seek to keep private. The presented method aligns with the forward-backward algorithm when executed at the last query point; no observations exist after the last query to compute a backward variable.

V. Experimental Setup

We consider a 320×320 grid over a 32×32 km^2 broad area (each cell is 100×100 m^2) centered at Los Angeles, CA downtown (34.0522^o N, 118.2428^o W). We further process the grid to identify cells that are not habitable (potentially because of a natural or artificial blockage). We perform this step by collecting the latitude and longitude of the center of each cell, and then using the snapToRoads function in Google's Maps Roads API to determine the cells that have a road within 100 meters. This gives us a bitmap signifying if a cell is habitable or not.

We use multiple search keywords to obtain different POI distributions in terms of size and density. The business listings are obtained from the SimpleGeo Places database. Object prominence values are assigned to the POIs from $\{0.95, 0.90, ..., 0.2, 0.25\}$ using a Zipf distribution with exponent 0.8. Lower scores are more frequent under this distribution.

Experiments are performed on a 2.8 GHz quad-core Intel Xeon system running Mac OS X 10.8.2 with 8GB memory. Runtimes of the algorithms to be executed on mobile devices are obtained on an Android emulator running a virtual device with a ARM Cortex-A8 processor ($\sim$800 MHz) and 512MB memory. We also run the algorithms on a virtual device using the Intel Atom system image (with 1GB memory). All implementations are single-threaded.

1) Paths and transition matrix: To generate paths along which queries will be made, we consider five regions surrounding Los Angeles–El Segundo, Pasadena, Hollywood, Montebello and the Los Angeles downtown–and use them as sources/destinations that the user mostly travels between. We randomly choose a pair of cells from these five regions as source and destination locations of the user, and then generate a path originating at the source cell and ending at the destination cell. We generate a set of 100 paths using this method. A path is always generated such that it contains habitable cells only.

We encode the 100 paths into multiple transition matrices for use in adversarial inference. Assuming region sizes of 16×16 cells ($2.56km^2$), 32×32 cells ($10.24km^2$), and 64×64 cells ($40.96km^2$), we create three transition matrices. Each transition matrix is obtained by dividing the 320×320 grid into regions of the corresponding size, and then counting the frequency of transitions happening between regions in the 100 paths. The three different region sizes are used in parametric evaluation of our method. For all other experiments, the 32×32 size is used as the default. A transition matrix is known to the adversary, while the exact paths are unknown. Note that a transition matrix created in this manner implies strong background knowledge since it captures all (and only those) paths on which we will apply a LPPM. It also implies that the adversary's background knowledge is always considered correct, i.e. the user can never be in a region (or cell) where the transition probability is zero.

2) POI retrieval and local cache: We consider that query results must be up-to-date at all points along a path; therefore, the LPPM is invoked at every point along a path. However, we implement the LPPMs with local caching functionality, i.e. results retrieved earlier will not be downloaded again. The geo-indistinguishability approach, as described in the original work, cannot directly make use of the local cache. We assume a modification where the server only returns identifiers of POIs inside the area of retrieval (AOR); details are then retrieved only for POIs not in the cache. The geo-indistinguishability AOR is also not guaranteed to contain the top-K POIs for the user. For fair comparison, we always set the radius of the AOR to the smallest value such that it contains the user location, as well as all POIs in the top-K set of the user. We use a value of $\epsilon = \frac{\ln(6)}{16 \text{ cells}}$ in the routine that generates perturbed locations.

3) Evaluation metrics: We consider two metrics to evaluate location privacy based on the posterior distribution inferred by an adversary. The first of these metrics is based on the adversary's best guess according to the distribution. We call this the *nearness privacy metric.* Given a probability distribution Φ over the cells in the grid, and the user's current location c_{user}, nearness privacy is computed as

$$nearness(\Phi, c_{user}) = distance(\text{argmax}_c \Phi(c), c_{user}). \quad (6)$$

We use Euclidean distance in our evaluation. If multiple cells have the most probable value, then we pick the cell closest to c_{user} as the adversary's guess. Note that when Φ is a uniform distribution, nearness will be zero. The second metric eliminates this ambiguity by using the concentration of the probabilities in the entire distribution. The adversary can sample one cell at a time (without replacement) based on the distribution Φ. The second metric is the expected number of cells that the adversary has to sample before being guaranteed that the user's cell is contained in the sampling. In other words, the number of sampled cells create an obfuscation area for the user. We call this the *areal privacy metric.* It can be computed using the following closed form expression.

$$areal(\Phi, c_{user}) = \sum_{c \neq c_{user}} \frac{\Phi(c)}{\Phi(c) + \Phi(c_{user})}. \quad (7)$$

When Φ is a uniform distribution over a subset C of cells, areal privacy is equal to $\frac{|C|-1}{2}$ cells. For example, for a uniform distribution over an area of 32×32 cells, the areal privacy is 511.5 cells. We will often present it in terms of area, which is obtained by multiplying the metric's value with the area of one cell ($0.01km^2$). Areal privacy can be viewed as the smallest obfuscation area one can expect if the attacker is *successful* in learning an approximate presence area using the sampling method. Compared to the expected estimation error metric proposed by Shokri et al. [8], the nearness and areal privacy metrics consider specific methods by which an adversary may use the inferred posterior distribution.

VI. Box Set Computation

For the first query, we cluster the cells in the $Z \times Z$ grid into non-overlapping *boxes* (formally sets) of $b \times b$ cells. For ease, b is chosen such that Z is a multiple of b; this results in a total of $(\frac{Z}{b})^2$ boxes. Let B_1 be the box containing the location of the user. We compute a *box set* $\mathbf{BS}$ for B_1 as the union of the top-K POIs of each cell in B_1.

$$\mathbf{BS}(B_1) = \cup_{c_i \in B} \text{top-}K(c_i). \quad (8)$$

This box set is also used as the interest set for the first query.

Let $P = \{p_1, ..., p_n\}$ denote the set of POIs inside the large area that matches the search keyword of the user. Each POI p_i has an associated location c_i (the cell where it is located) and a prominence value $0 < \beta_i \leq 1$. Given cell c_{user} where the user is currently located, the rank of object p_i is computed by a weighted combination of its normalized distance from c_i and the prominence value, i.e.

$$rank'(p_i, c_{user}) = \alpha d_{norm}(c_{user}, c_i) + (1 - \alpha)(1 - \beta_i), \quad (9)$$

where d_{norm} is a normalized distance function. We use the length of the diagonal of the large area as the normalization factor for distance. α is the weighing co-efficient such that $0 < \alpha \leq 1$. The ranks of objects are then between 0 and 1, with lower values implying better choices. In most situations, values of α and β_i may not be revealed to the user. Hence, we redefine the ranking function as

$$rank(p_i, c_{user}) = \frac{rank'(p_i, c_{user})}{\alpha} = d_{norm}(c_{user}, c_i) + \gamma_i, \quad (10)$$

where $\gamma_i = \frac{1-\alpha}{\alpha}(1 - \beta_i)$. Since α is a constant, both functions result in the same ranked ordering of the objects. Therefore, as a result of the generic query in the privacy-supportive architecture, the server sends the tuples $\Omega = \{\langle c_i, \gamma_i \rangle | i = 1...n\}$ to the client. Using Ω and the current location of the user, the client can compute a box set $\mathbf{BS}(\cdot)$. However, this step is performed in a resource constrained mobile device, and hence computation time is critical to prevent noticeable delays in user experience. We next present our method and optimizations for this computation, and then report on observed run times.

A. Top-K of a box

Given the set of POIs P, and a box B of cells, we need a method to calculate the set of top-K POIs for each cell in the box, and return the union of those sets. A brute force technique to achieve this is trivial, although it results in a sluggish user experience. We propose a faster algorithm consisting of the following steps. We observed this algorithm to be 76 times faster than a standard QuickSort based approach.

1) *Computation for border cells:* Using the full set P of POIs, we execute a kd-tree based branch-and-bound search algorithm (detailed in the next section) to compute the top-K POI set of each cell on the border of B. We denote the union of these sets as $\tilde{P}_{border}$.
2) *Reducing the search space:* Determine the subset of POIs that are inside the box B, denoted as $P_{internal} \subseteq P$; consider the set $P_{reduced} = \tilde{P}_{border} \cup P_{internal}$, which are POIs either inside the box or part of the top-K set of some border cell of the box.
3) *Computation for internal cells:* Compute the top-K POI set for each non-border cell in B; the union of all these sets is denoted by $\tilde{P}_{non\text{-}border}$. We use the kd-tree algorithm for this step too; however, the tree is created over the reduced set of POIs $P_{reduced}$. We generate $\tilde{P}_{non\text{-}border}$ in an incremental manner (one cell at a time) and stop when $P_{internal} \subseteq \tilde{P}_{non\text{-}border}$, or when the top-$K$ set for all non-border cells have been evaluated.
4) *Final result:* The box set $\mathbf{BS}(B)$ is returned as $\tilde{P}_{border} \cup \tilde{P}_{non\text{-}border}$.

The general idea in this algorithm is to first compute the top-K sets of border cells only using the entire set of POIs. Thereafter, the top-K sets for internal cells are computed from a smaller set of POIs obtained by combining the results in the first step and the POIs inside the box. This reduction of the search space provides for a faster search.

Step 3 of this process requires a correctness argument since $P_{reduced} = \tilde{P}_{border} \cup P_{internal}$ should be guaranteed to contain

Algorithm 1 Compute the top-K set for reference cell.

Input: Root node $root$ of kd-tree; Query cell c_{ref}; Parameter K
Output: Ordered list of top-K POIs w.r.t cell c_{ref}

```
1: function CellTopPOIs(root, c_ref, K)
2:     T ← empty list
3:     L ← empty list
4:     L.Append(root)
5:     while L is not empty do
6:         n_test ← L.Remove-And-Return-Head()
7:         if T.size = K and ∄i such that n_test.lbound ≤ T[i].rank then exit while
8:         n_test.CalculateRank(c_ref)
9:         T.PriorityInsert(n_test)                          ▷ rank based.
10:        if T.size > K then T.RemoveLast()
11:        if ∃n_test.left node then L.PriorityInsert(n_test.left)     ▷ lbound based.
12:        if ∃n_test.right node then L.PriorityInsert(n_test.right)   ▷ lbound based.
13:    end while
14:    return T
15: end function
```

the top-K POIs of all internal cells of the box. One can easily verify that in the plane $\mathbb{R}^2$, the top-K POIs of a point inside any given box is either inside the box, or is the same as the top-K POIs of some point on the boundary of the box. The same argument also applies to our discretized grid, provided the discretization is fine enough that, for any real-valued query point between two cells on a border of the box, the top-K set matches the top-K set of either one of the neighboring cells. The physical distance between two cells in our empirical evaluation is 100 meters, which is reasonably small to maintain the accuracy of this heuristic.

B. kd-Tree branch-and-bound search

We construct a kd-tree using the POIs in P as nodes. The tree in this case is binary, with the x and y coordinates of the POIs as the two dimensions. The construction uses standard mechanisms for determining the root, and left and right subtrees—nodes are split into left and right subtrees alternating between the x and y values as the splitting dimension. We implement a construction algorithm that pre-sorts all objects and operates in $O(kn \log n)$ time and $O(n)$ storage [34]. Additionally, each node in the tree is augmented with the minimum possible γ (scaled prominence) value of POIs included in the subtree rooted at that node (including the node itself). We denote this value by $\gamma_{min}(\cdot)$. The constructed tree is reused in all searches where the searched POI set is the same.

To calculate the top-K POIs for a given query cell c_{ref}, two lists are maintained: (a) a list T representing the top-K nodes (POIs), ordered according to the rank of the nodes with respect to c_{ref}, and (b) a list L of nodes to be explored, sorted by a lower bound value. The lower bound value for a node represents the minimum possible rank achievable in the subtree rooted at that node.

During the search, Algorithm 1 explores the nodes in L in their order of appearance, and terminates when L becomes empty, or it is determined that no node in the subtree can potentially change the existing T list (lines $[6-7]$). The latter case can happen when the T list is already at full capacity (K), and the lower bound value of the first node in L is greater than the rank value of all nodes in T. Exploring a node involves the steps of (i) checking if the node can be inserted in the T list based on its rank (lines $[8-10]$), (ii) computing the lower bounds for the left and right children, and (iii) inserting them in L (lines $[11-12]$). We provide an example illustrating the execution of this algorithm in an earlier work [10].

C. Improving the search time

When top-K POIs are being determined for consecutive cells (e.g. a row or column of cells), it may be possible to skip the top-K search for certain cells. Assume that T is the vector of top-$(K+1)$ POIs obtained for a cell c_s using the kd-tree search. Let c_t be a subsequent cell in the same row or column. Given the structure of the ranking function, the rank of any POI with respect to c_s can at best reduce by $d_{norm}(c_s, c_t)$ (the γ values are constant) when computed with respect to c_t. Consider the $(K+1)^{th}$ top POI for c_s, i.e. $T[K+1]$. The rank of this POI, and any other POI not in T, can at best be $r = rank(T[K+1], c_s) - d_{norm}(c_s, c_t)$ when computed with respect to c_t. Therefore, if we reorder T based on the ranks of the POIs with respect to c_t, and observe that the K^{th} POI rank is less than or equal to r, then no other POI can replace the first K POIs in the reordered T. In that case, the kd-tree search for c_t can be skipped, and the first K POIs in the reordered T are the top-K POIs for c_t. Note that the $(K+1)^{th}$ POI is only known for cell c_s, the last cell where a full search was performed. Hence, r should always be calculated using the last fully searched cell (i.e. c_s).

D. Computing $Pr(O|c)$

The output generated by our LPPM is an interest set. Let I_1 denote the interest set observed by the adversary in the first query. Assuming that any existing cache is flushed in the first query, this interest set is same as the box set. For adversarial

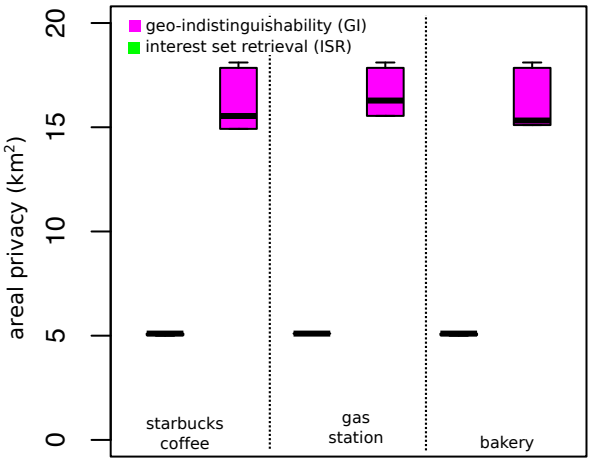


Fig. 2: Areal privacy in the first query; $b = 32$.

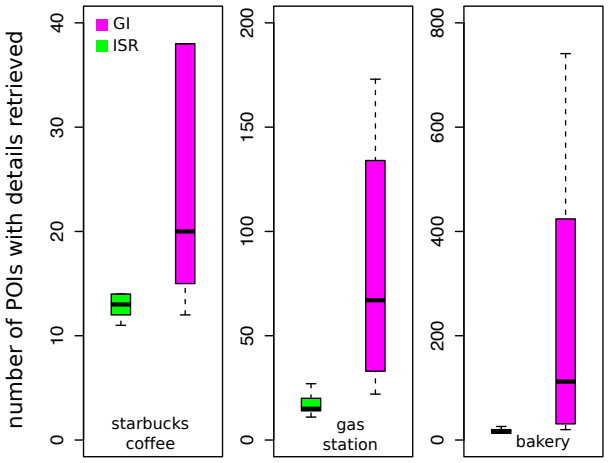


Fig. 3: Number of POIs for which details are retrieved in the first query.

inference (Eq. 2), the proposed method of generating the interest set in the first query results in $Pr(I_1|c)$ being either zero or one. It is zero if $\mathbf{BS}(B) \neq I_1$ for $c \in B$; otherwise one. Therefore, we can say that $\Phi_1 \propto \Phi_0$ within the subset of cells where the box set is I_1. As a direct result of the method of generating I_1, this subset will have all cells in the box B_1 used during the computation.

E. Evaluation

1) Box set computation time: Table I lists the average time to compute the box sets for a given size (the parameter b). The average is taken over all the boxes induced as a result of the pre-partitioning. Recall that in a $Z \times Z$ grid, there will be $\frac{Z}{b} \times \frac{Z}{b}$ boxes. Values for α (weight on distance) and K are set at 0.8 and 10 respectively. The execution time for each parameter value is taken as the average of 10 identical runs. Each cell in the table shows the time for the two different devices (ARM and Atom). Except for when large boxes (64×64 cells $\approx 40.9km^2$) are created for some high density POIs, the execution time using the ARM processor is within one second; in fact, less than 500 milliseconds for a majority of the cases. We get almost a five fold improvement in the computation time by using the Atom processor, with most computations in the 20 to 100 ms range. Although the Atom processors are currently more suitable for tablet computers, efforts have already been successful in porting them to smartphones. We also tested our algorithm on a physical Samsung Galaxy Note smartphone with a dual-core 1.5GHz Snapdragon S3 processor. The observed run times for the 32×32 box size are a three fold improvement over the emulated values on the ARM device.

The client incurs as additional network overhead while retrieving the locations and γ values of the matching POIs inside the large area. However, the overhead is negligible—if the latitude, longitude and γ values of a POI are encoded as 32-bit numbers, and 1000 matching POIs exist inside the large area, then a total of 12KB of data needs to be downloaded in order to compute the box set. Assuming a 3G connection with 320 KB/s speed [35], this download will incur an additional 37.5 milliseconds to the process (ignoring connection latency).

2) Privacy: With Φ_0 as a uniform distribution over a subset of cells, the posterior distribution Φ_1 will also be a uniform distribution over the subset of cells that has a non-zero probability in Φ_0 and has the same box set as the one appearing in the user's query. As such, the nearness metric is not applicable here. Fig. 2 shows a box plot of the areal metric on the first query issued on the 100 paths. The plot depicts the case for three POI search keywords—"starbucks coffee" (92 POIs: low density), "gas station" (347 POIs: medium density), and "bakery" (834 POIs: high density).

As can be observed, our approach results in a low variance areal privacy values due to the deterministic nature of the process. The minimum value always corresponds to 511.5 cells ($5.1km^2$), which is the same as the theoretical value for $32 \times 32 = 1024$ cells with uniform distribution. The values generated by geo-indistinguishability have a comparatively higher variance, but does generate areal privacy values larger by a factor of three. However, this comes at the expense of a comparatively larger amount of bandwidth usage. When the AOR in geo-indistinguishability is made large enough to encompass the top-K POIs corresponding to the user's location, the resulting retrieval is as high as a factor of ten in the case of the high density POI (bakery) (Fig. 3). The variance is also comparatively higher as a result of the randomness in the perturbed location.

Note that, by design, the sought top-K set is always available to the user in the interest set approach. As such, accuracy of results do not suffer in the approach.

VII. Queries on a Path

As long as the multiple queries by the user happen when the user is in the same box, the adversary's knowledge of the user's location will not be enhanced. Consider the case where the user moves from one box to another between two consecutive queries and the time interval between these two queries is larger than the time needed by the user to move across the large geographical area. In this case, the adversary is still not able to enhance his knowledge about the user's

TABLE I: Average time (milliseconds) to compute box set for different sizes of a box with $b \times b$ cells. Top and bottom values correspond to the ARM processor and the Atom processor virtual devices respectively.

search query	no. of POIs	box size ($b \times b$)		
		16×16	32×32	64×64
bus station	32	24.41 4.71	43.47 8.62	76.96 15.18
farmers market	50	45.69 8.12	74.04 13.62	140.88 26.78
police	84	71.25 12.41	128.58 22.93	396.82 70.38
starbucks coffee	92	63.26 10.79	117.24 20.57	478.98 84.84
grocery	95	64.43 12.02	119.55 23.03	386.67 75.34
restaurant italian	124	81.74 13.77	155.23 27.02	539.14 94.71
liquor store	125	92.82 14.8	180.66 29.42	831.41 135.69
bookstore	126	80.66 14.83	152.26 29.55	650.29 128.76
library	141	102.61 16.19	194.44 31.38	783.86 128.72
night club	149	96.27 15.35	172 29.59	644.26 115.82
clothing store	169	120.06 20.53	235.17 42.34	920.34 171.79
car rental	196	142.95 23.97	252.14 44.91	889.84 165.54
parking	281	202.45 33.26	363.43 62.34	1248.92 220.6
atm	297	190.53 29.72	339.13 57.46	1382.03 250.42
gas station	347	210.34 34.82	383.49 69.25	1580.96 308.41
pharmacy	369	247.32 36.97	446.25 70.33	1846.01 303.26
cafe	608	385.3 61.73	613.32 107.47	2097.42 407.17
bakery	834	525.32 80.89	803.75 137.3	2776.04 539.39

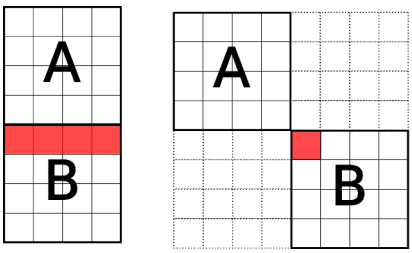


Fig. 4: Location inference during multiple queries.

location. This is because, there is enough time for the user to move to any cell in the $Z \times Z$ grid.

Now we consider the converse scenario, where the time interval between the two queries is less than the time needed by the user to go from the current cell to the farthest possible cell in the grid. In this case, if we continue to use the deterministic algorithm in Section VI, the adversary has an advantage. This is illustrated in Fig 4. It shows two adjacent boxes in the pre-partitioning: **A** and **B**, where $b = 4$. The user was in one of the cells in box **A** and requests the box set of **A**. The user then moves for one time step (into box **B)** and then issues another query. If the box set of **B** is unique, then the adversary can narrow down the user's location to the "shaded" boundary cell(s) of **B**. He sees that the first query's interest set matches box **A** and second query's interest set matches box **B**, and the user had only enough time to move by one cell. Therefore, the techniques proposed for first query need to be enhanced for subsequent queries.

A. Selection area

The reason for privacy loss during multiple queries with time interval constraints is due to the fixed pre-partitioning. Fixed pre-partitioning is suitable for the single query scenario, but not for the subsequent queries. In order to deter the kind of attacks mentioned above, we first create a new area, hereafter called the *selection area* S, by expanding the box used at the previous query to its *neighboring cell*s. If Δ time steps have elapsed between two subsequent queries, then a neighboring cell is any cell that is no more than Δ rows or columns away from the current box. The selection area S represents all possible cells the user could be in for query q_t, given as

$$\begin{aligned} S(B_{t-\Delta}) &= b \times b \text{ boxes in } \cup_{c \in B_{t-\Delta}} nbr(c, \Delta) \\ nbr(c, \Delta) &= \{c' | c' \text{ can be reached} \\ &\qquad \text{from } c \text{ in } \Delta \text{ steps}\} \end{aligned} \tag{11}$$

Figs. 5a and 5b show S where $\Delta = 1$ and $\Delta = 2$ respectively. The numbers inside each cell signify the number of $b \times b$ boxes inside the selection area that contain the cell.

B. Choosing a box

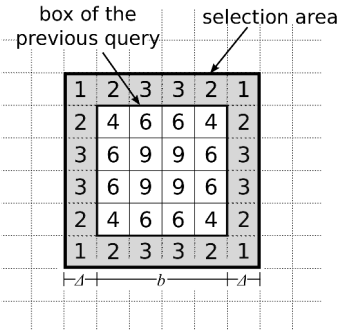
Our algorithm makes the selection of the $b \times b$ box (for interest set generation) based on whether the issued query is the first one, or one of the subsequent ones. Algorithm 2 uses pseudo functions whose objectives are discussed next. For the first query, as described in Section VI, the $Z \times Z$ grid is pre-partitioned into fixed non-overlapping boxes of size $b \times b$. The algorithm simply chooses the box that contains the user's cell.

Let us assume that the algorithm is trying to generate the interest set for query q_t and Δ be the maximum number of cells that the user could have moved since the previous query. For the second query ($t > 1$), and subsequent ones, the algorithm first calculates Δ based on query timestamps and determines the selection area S. The box for the current query is selected by picking a $b \times b$ box B_t uniformly at random from $S(B_{t-\Delta})$ such that it contains the user. The algorithm uses the same techniques used for the single query scenario to efficiently generate the box set. Since the client caches earlier results, it only requests details for POIs that are new to this box, i.e. $I_t = \mathbf{BS}(B_t) - \mathbf{I}$, where $\mathbf{I}$ is the cache (set) of all POIs retrieved earlier. This will continue till the current user session ends. When the time interval is large enough for the user to reach the farthest cell in the $Z \times Z$ grid, the algorithm starts a new session with the fixed pre-partitioning step.

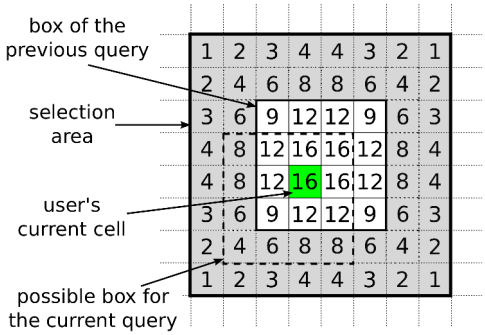
Algorithm 2 Box selection for interest set generation.

Global Initialization: Time of previous query $t_l = 0$; Previous box $B_p = \phi$
Input: Current time t; Box size b; User cell c_{user}
Output: Box B

function BOXFORCURRENTQUERY(t, b, c_{user})
  $G \leftarrow Z \times Z$ grid ▷ Initial grid.
  if $t = 1$ **then**
    $B \leftarrow G$.FixedBox(b, c_{user}) ▷ $(b \times b)$ box from pre-partitoned grid.
    $t_l \leftarrow 1$
  else
    $\Delta \leftarrow t - t_l$
    $S \leftarrow G$.GetSelectionAreaBoxes(B_p, Δ) ▷ $(b \times b)$ boxes in selection area.
    $B \leftarrow$ RandomSampling(S, c_{user}) ▷ Random $(b \times b)$ box from the selection area containing user cell.
    $t_l \leftarrow t$
  end if
  $B_p \leftarrow B$
  return B
end function



(a) $\Delta = 1$.



(b) $\Delta = 2$.

Fig. 5: Selection area for a given box (4×4 size) and Δ.

C. Computing $Pr(O|c)$

Let I_t denote the interest set observed by the adversary at time step $t > 1$. Recall that the POIs of interest to the user may only partially be in this set, since some of them may be available in the cache $\mathbf{I}$ of earlier queries. Consider the following two sets.

$$\nu_t(c) = \{B | B \in S(B_{t-\Delta}) \text{ and } c \in B\}$$
$$\eta_t(c) = \{B | B \in \nu_t(c) \text{ and } I_t \subseteq \mathbf{BS}(B) \subseteq \mathbf{I} \cup I_t\} \quad (12)$$

$\nu_t(c)$ denotes the set of $b \times b$ boxes in the selection area S at time t that contain cell c; $\eta_t(c)$ are boxes in $\nu_t(c)$ such that all newly requested POIs (I_t) are part of the box set, which itself is fully contained in the union of the cache and the newly requested set. $Pr(I_t|c)$ is then equal to $\frac{|\eta_t(c)|}{|\nu_t(c)|}$.

Computation of $\eta_t(c)$ is straightforward for the adversary once $\nu_t(c)$ is known. However, the adversary does not know S since it depends on $B_{t-\Delta}$. Nonetheless, the adversary can perform approximations, denoted by $\hat{\nu_t}(c)$ and $\hat{\eta_t}(c)$. As base cases, we have $\hat{\nu_1}(c) = \{B | c \in B\}$ and $\hat{\eta_1}(c) = \{B | c \in B \text{ and } \mathbf{BS}(B) = I_1\}$—the same formulation as we had for the first query. To approximate $\hat{\nu_t}(c)$, the adversary can consider the union of all selection areas that can be generated from boxes in $\hat{\eta}_{t-\Delta}(c)$.

$$\hat{\nu_t}(c) = \{B | B' \in \hat{\eta}_{t-\Delta}(c) \text{ and } B \in S(B') \text{ and } c \in B\}$$
$$\hat{\eta_t}(c) = \{B | B \in \hat{\nu_t}(c) \text{ and } I_t \subseteq \mathbf{BS}(B) \subseteq \mathbf{I} \cup I_t\} \quad (13)$$

We use in $\frac{|\hat{\eta_t}(c)|}{|\hat{\nu_t}(c)|}$ as an estimate of $Pr(I_t|c)$ in Eq. 2.

D. Evaluation

1) Privacy: When multiple queries are performed, the distribution of the POIs along the path has a direct impact on the interest set requested by the user. As such, the nearness and areal privacy metric can fluctuate (both increase or decrease) over time. Fig. 6 depicts the values of these two metrics for one of the paths, with "gas station" as the search keyword, and box length $b = 32$ cells. Since our interest set retrieval mechanism (ISR) issues requests only if the POIs are not in the cache, the number of queries are comparatively lower than in the geo-indistinguishability (GI) approach. The GI approach also makes use of a cache; however, it must issue a query at every time step to determine which POIs in the cache correspond to the result set. When retrievals are made infrequently, the nearness metric can be sustained comparatively higher. More queries enable better approximations for the adversary. It is difficult to make a similar observation on the areal metric from the figure.

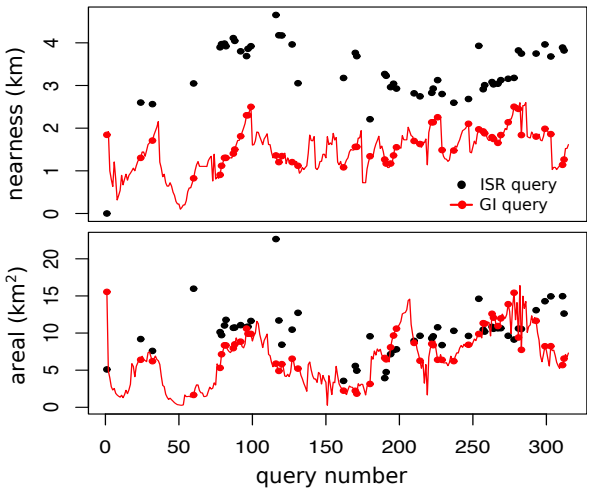


Fig. 6: Nearness and areal privacy variation along a path. Search keyword = *gas station* and $b = 32$ cells.

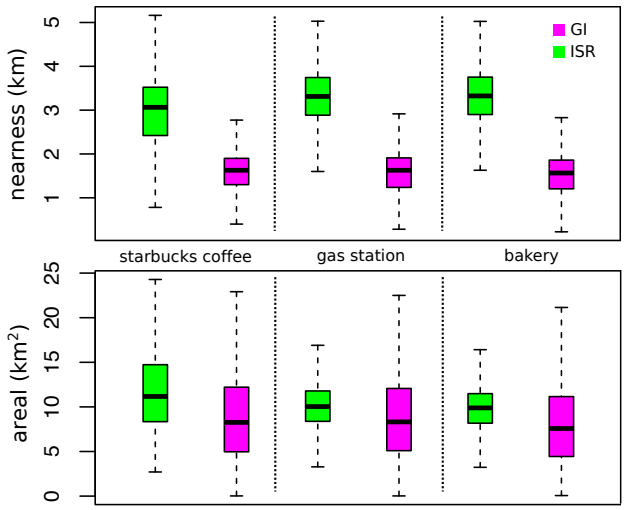


Fig. 7: Nearness and areal privacy quartiles. $b = 32$ cells.

In order to understand the overall behavior in multiple paths, we look at the quartiles of the two metrics across all the paths and three POIs with varying density (Fig. 7). Similar to the observation stated above, nearness in the ISR approach is maintained at a higher level than that in the GI approach. The maximum value of nearness in the lower quartile (25%) of the ISR data is indeed higher than the upper quartile (75%) of the GI data. The interquartile ranges show no overlap, demonstrating strong evidence that the median nearness value in the two approaches are significantly different.

For areal privacy, we observe that ISR has a higher median value in the three POI types; although, there is no evidence of significant differences. We do observe that for medium and high density POIs ("gas station" and "bakery" respectively), the ISR approach maintains comparatively lower variance in the areal privacy than the GI approach. This is indicative of a stable approach, irrespective of the path taken by the user during the queries. For most parts (upper 75%), the metric is also higher ($\geq 8km^2$) than the theoretical minimum ($5.1km^2$).

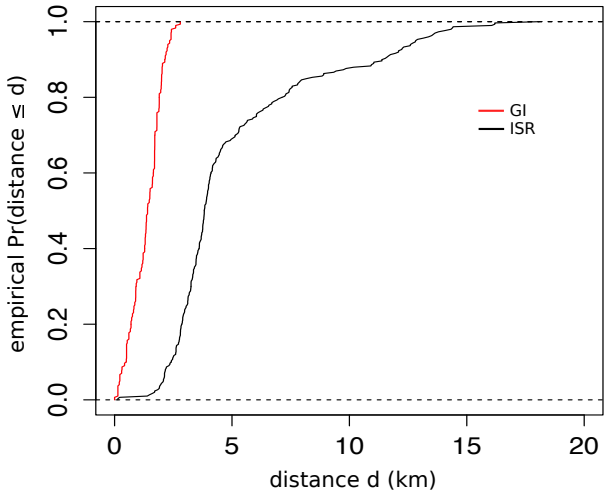


Fig. 8: Empirical CDF of distance between last cell in a path and the most probable cell in the inferred distribution. $b = 32$ cells.

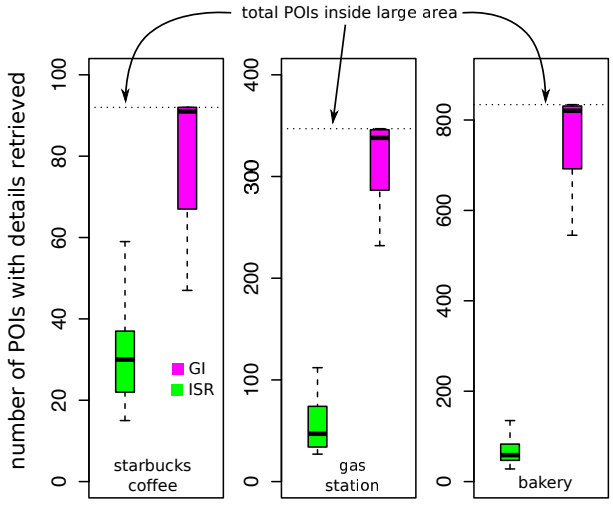


Fig. 9: Total number of POIs for which details are retrieved during queries along the paths. $b = 32$ cells.

We can assess how much information a privacy mechanism has revealed by also focussing strictly on the final destination of a path. Destinations of travel can be argued to be more private than the exact path taken by the user to the destination. Fig. 8 shows the empirical cumulative distribution function (CDF) of the distance between the last cell of a path and the cell with the highest probability in the adversary's final inferred distribution, i.e. the nearness value. The CDF is generated by collating observations in all the 100 paths and the three POI keywords. Comparatively, about 10% of the cases in the ISR approach has a final nearness value as low ($\approx 1.8km$) as the GI approach; in general, the values are always better. The adversary's inferred location in the ISR approach is significantly distant ($> 5km$) in about 30% of the cases, and more than $2km$ in 95% of the cases.

2) Bandwidth impact : The interest set retrieval mechanism aims to exploit the fact that top-K sets do not frequently (and

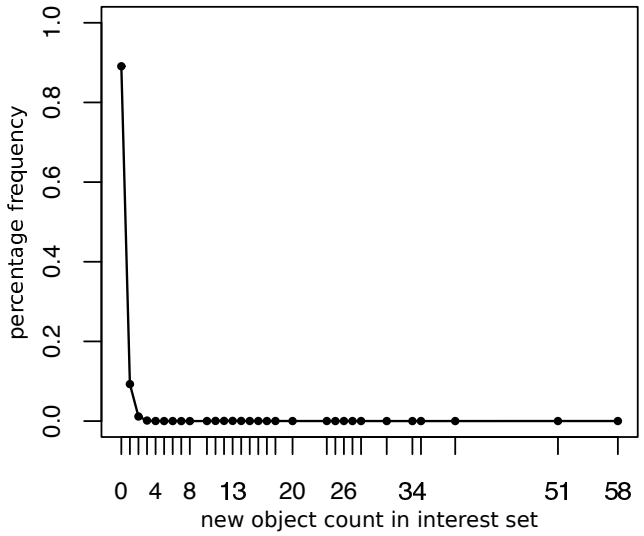


Fig. 10: Frequency distribution of interest set (POIs for which details are retrieved) sizes. Total potential queries = 264,627.

significantly) change for nearby query points. We exemplified this in Fig. 6, which shows gaps between query points. Further, since the approach only retrieves details on pertinent objects, i.e. objects that are part of a top-K set along the path, we also expect that download bandwidth is preserved. Fig. 9 shows the quartiles for the total number of POIs for which details are retrieved by the ISR and GI approaches. The figure also shows the maximum number of POIs inside the large geographical area for each of the three keywords. It can be seen that the GI approach could potentially result in the download of details for all the POIs as a result of using an AOR. To contrast this, the ISR approach downloads details on a median of 12%–30% of the POIs. Although the ISR approach uses an union of various top-K sets to generate the box set, it does not result in too many redundant downloads.

Fig. 10 displays the frequency distribution of the cardinality of the interest set across all queries (three search keywords and all paths). A total of 264,627 data points are used to generate this distribution. Recall that the interest set is the set for which details are retrieved from the server at a query point. It is empty if the required details are already in the cache (retrieved earlier). We observe that in approximately 90% of the query points, no communication was necessary with the server (empty interest set). This confirms our statement that top-K sets seldom undergo changes between query points. Further, the distribution is heavy tailed, with the frequency dropping significantly as size increases. In other words, smaller interest sets are abundant. This is evident of the fact that whenever top-K sets change between successive query points, the changes are often very small (one or two POIs). The top-K set landscape in local search is recurrently plateaued and is slow rising; Micinski et al. have independently validated this observation by studying the changes in search results of six Android applications instrumented to use a truncated location [36]. To the best of our knowledge, this characteristic is rarely exploited by a privacy mechanism.

3) Impact of box size: Fig. 11 depicts the nearness and areal privacy for three different choices of the parameter b. Recall that smaller box sizes imply lower expectations of privacy. The trends we observed in the case of 32×32 box size is repeated for other box sizes too, albeit at varying degrees. For example, using a smaller box size leads to lower privacy values, and they increase as larger box sizes are used. A larger box size does imply a larger box set, and can lead to the retrieval of more number of POI details.

VIII. Conclusion

Through this work, we have demonstrated how private local search mechanisms can leverage the computational improvements in modern mobile devices. We have considered a result ranking procedure that includes the prominence of POIs in addition to their distance from the query point. Our approach implicitly exploits the continuity in the top-K result sets and retrieves details on POIs as needed. The empirical evidence suggests that our approach does not require any retrieval in most cases, and when needed, the number of POIs retrieved are significantly small. This aids in preventing an adversary from obtaining significant refinements in the probability distribution associated with the user's current location.

Our empirical analysis demonstrates that the interest set approach can limit adversarial inferences in a setting with real world POI distributions. However, the approach currently does not provide a theoretical guarantee on this limit. We desire to be able to bound the probabilistic advantage that the adversary gains after observing the interest set. We will direct our future work along this line.

References

[1] J. Freudiger, R. Shokri, and J.-P. Hubaux, "Evaluating the Privacy Risk of Location-Based Services," in *Proceedings of the 15th International Conference on Financial Cryptography and Data Security*, pp. 31–46, 2011.

[2] H. Kido, Y. Yanagisawa, and T. Satoh, "An Anonymous Communication Technique Using Dummies for Location-Based Services," in *Proceedings of the IEEE International Conference on Pervasive Services*, pp. 88–97, 2005.

[3] M. Gruteser and D. Grunwald, "Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking," in *Proceedings of the 1st International Conference on Mobile Systems, Applications, and Services*, pp. 31–42, 2003.

[4] F. Liu, K. A. Hua, and Y. Cai, "Query l-Diversity in Location-Based Services," in *Proceedings of the 10th International Conference on Mobile Data Management: Systems, Services and Middleware*, pp. 436–442, 2009.

[5] B. Gedik and L. Liu, "Protecting Location Privacy with Personalized k-Anonymity: Architecture and Algorithms," *IEEE Transactions on Mobile Computing*, vol. 7, no. 1, pp. 1–18, 2008.

[6] R. Dewri and R. Thurimella, "Exploiting Service Similarity for Privacy in Location-Based Search Queries," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 2, pp. 374–383, 2014.

[7] N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Optimal Geo-Indistinguishable Mechanisms for Location Privacy," in *Proceedings of the 21th ACM Conference on Computer and Communications Security*, pp. 251–262, 2014.

[8] R. Shokri, G. Theodorakopoulos, J.-Y. L. Boudec, and J.-P. Hubaux, "Quantifying Location Privacy," in *Proceedings of the 32nd IEEE Symposium on Security and Privacy*, pp. 247–262, 2011.

[9] B. Gedik and L. Liu, "Location Privacy in Mobile Systems: A Personalized Anonymization Model," in *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*, pp. 620–629, 2005.

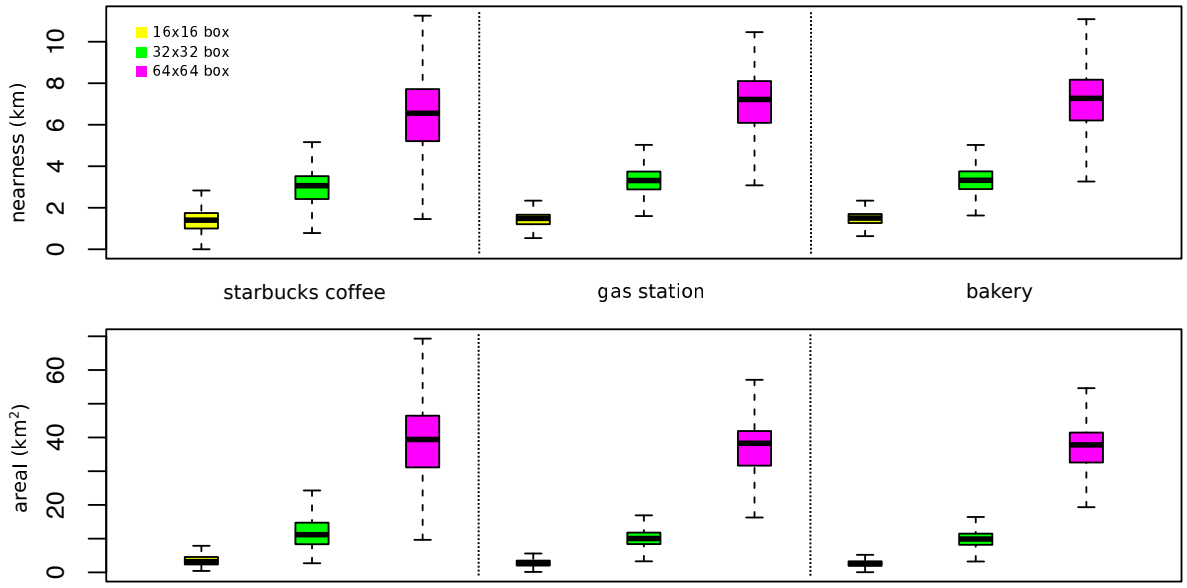


Fig. 11: Nearness and areal privacy quartiles for different box size b and search keywords.

[10] R. Dewri, W. Eltarjaman, P. Annadata, and R. Thurimella, "Beyond the Thin Client Model for Location Privacy," in *Proceedings of the 2013 International Conference on Privacy and Security in Mobile Systems*, pp. 1–8, 2013.

[11] W. Eltarjaman, P. Annadata, R. Dewri, and R. Thurimella, "Leveraging Smartphone Advances for Continuous Location Privacy," in *Proceedings of the 16th IEEE International Conference on Mobile Data Management*, pp. 197–202, 2015.

[12] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, " Geo-indistinguishability: Differential Privacy for Location-Based Systems," in *Proceedings of the 20th ACM Conference on Computer and Communications Security*, pp. 901–914, 2013.

[13] B. Bamba, L. Liu, P. Pesti, and T. Wang, "Supporting Anonymous Location Queries in Mobile Environments with Privacy Grid," in *Proceedings of the 17th International World Wide Web Conference*, pp. 237–246, 2008.

[14] M. F. Mokbel, C. Chow, and W. G. Aref, "The New Casper: Query Processing for Location Services Without Compromising Privacy," in *Proceedings of the 32nd International Conference on Very Large Data Bases*, pp. 763–774, 2006.

[15] R. Dewri, I. Ray, I. Ray, and D. Whitley, "Query m-Invariance: Preventing Query Disclosures in Continuous Location-Based Services," in *Proceedings of the 11th International Conference on Mobile Data Management*, pp. 95–104, 2010.

[16] H. Shin, J. Vaidya, and V. Atluri, "A Profile Anonymization Model for Location Based Services," *Journal of Computer Security*, vol. 19, no. 5, pp. 795–833, 2011.

[17] C. B. D. Riboni, L. Pareschi and S. Jajodia, "Preserving Anonymity of Recurrent Location-Based Queries," in *Proceedings of the 16th International Symposium on Temporal Representation and Reasoning*, 2009.

[18] G. Ghinita, P. Kalnis, and S. Skiadopoulos, "PRIVE: Anonymous Location-Based Queries in Distributed Mobile Systems," in *Proceedings of the 16th International Conference on World Wide Web*, pp. 371–380, 2007.

[19] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias, "Preventing Location-Based Identity Inference in Anonymous Spatial Queries," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 12, pp. 1719–1733, 2007.

[20] G. Ghinita, K. Zhao, D. Papadias, and P. Kalnis, "A Reciprocal Framework for Spatial k-Anonymity," *Journal of Information Systems*, vol. 35, no. 3, pp. 299–314, 2010.

[21] A. Khoshgozaran, C. Shahabi, and H. Shirani-Mehr, "Location Privacy: Going beyond k-Anonymity, Cloaking and Anonymizers," *Journal of Knowledge and Information Systems*, vol. 26, no. 3, pp. 435–465, 2011.

[22] S. Papadopoulos, S. Bakiras, and D. Papadias, "Nearest Neighbor Search with Strong Location Privacy," *VLDB Endowment*, vol. 3, no. 1-2, pp. 619–629, 2010.

[23] X. Yi, R. Paulet, E. Bertino, and V. Varadharajan, "Practical Approximate k Nearest Neighbor Queries with Location and Query Privacy," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 6, pp. 1546–1559, 2016.

[24] T. Xu and Y. Cai, "Feeling-Based Location Privacy Protection for Location-Based Services," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, pp. 348–357, 2009.

[25] M. Soriano, S. Qing, and J. Lopez, "Time Warp: How Time Affects Privacy in LBSs," in *Proceedings of the 12th International Conference on Information and Communications Security*, pp. 325–339, 2010.

[26] B. Niu, Q. Li, X. Zhu, G. Cao, and H. Li, "Achieving K-anonymity in Privacy-aware Location-based Services," in *Proceedings of the 33rd Annual IEEE International Conference on Computer Communications*, pp. 754–762, 2014.

[27] B. Niu, Q. Li, X. Zhu, G. Cao, and H. Li, "Enhancing Privacy through Caching in Location-based Services," in *Proceedings of the 34th Annual IEEE International Conference on Computer Communications*, pp. 1017–1025, 2015.

[28] R. Shokri, G. Theodorakopoulos, P. Papadimitratos, E. Kazemi, and J.-P. Hubaux, "Hiding in the Mobile Crowd: LocationPrivacy through Collaboration," *IEEE Transactions on Dependable and Secure Computing*, vol. 11, no. 3, pp. 266–279, 2014.

[29] A. Pham, K. Huguenin, I. Bilogrevic, I. Dacosta, and J.-P. Hubaux, "SecureRun: Cheat-Proof and Private Summaries for Location-Based Activities," *IEEE Transactions on Mobile Computing*, vol. 15, no. 8, pp. 2109–2123, 2016.

[30] R. Shokri, C. Troncoso, C. Diaz, J. Freudiger, and J.-P. Hubaux, "Unraveling an Old Cloak: k-Anonymity for Location Privacy," in *Proceedings of the 9th Annual ACM Workshop on Privacy in the Electronic Society*, pp. 115–118, 2010.

[31] R. Dewri, "Local Differential Perturbations: Location Privacy Under Approximate Knowledge Attackers," *IEEE Transactions on Mobile Computing*, vol. 12, no. 12, pp. 2360–2372, 2013.

[32] R. Shokri, G. Theodorakopoulos, C. Troncoso, J.-P. Hubaux, and J.-Y. L. Boudec, "Protecting Location Privacy: Optimal Strategy Against Localization Attacks," in *Proceedings of the 19th ACM Conference on Computer and Communications Security*, pp. 617–627, 2012.

[33] B. O'Clair, D. Egnor, and L. E. Greenfield, "Scoring local search results based on location prominence," 2011. US Patent 8,046,371.

[34] I. Wald and V. Havran, "On building fast kd-Trees for Ray Tracing, and on doing that in O(N log N)," in *IEEE Symposium on Interactive Ray Tracing*, pp. 61–70, 2006.

[35] M. Sullivan, "3G and 4G Wireless Speed Showdown: Which Networks Are Fastest?," *PC World*, April 2012.

[36] K. Micinski, P. Phelps, and J. S. Foster, "An Empirical Study of Location Truncation on Android," in *Proceedings of Mobile Security Technologies 2013*, 2013.