Leveraging Smartphone Advances for Continuous Location Privacy

Wisam Eltarjaman, Prasad Annadata, Rinku Dewri and Ramakrishna Thurimella
Colorado Research Institute for Security and Privacy
Department of Computer Science, University of Denver, Denver, CO 80210, USA
Email: {wisam,prasad,rdewri,ramki}@cs.du.edu

***Abstract*—Location privacy preservation algorithms for nearby points-of-interest (POI) search have evolved in the recent years. However, a majority of the proposals assume that points of interests 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 for use in POI search that achieves much of the desired location privacy by restricting the usage of precise location data to the client device.**

***Keywords*—location privacy, POI search, mobile processing**

I. Introduction

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 attacker or a semi-trusted service provider can use this information to deduce user preferences and future locations, in addition to the current location.

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. 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 trusted third-party (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 [1]. 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.

We build upon our earlier work, and other published literature, to provide an algorithm that preserves users privacy in the multiple query scenario, while keeping the algorithm practical enough to be implemented on the mobile device itself. We first show 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 propose a new client side privacy algorithm to retrieve POIs, and analyze the inference risk of the algorithm under a Bayesian adversary model. Finally, we assess the privacy offered by the algorithm using a real world POI distribution, and report on the practicability of the process.

II. Related Work

Initial research in providing location privacy took solutions from other related domains such as statistical databases, and applied them to the location privacy problem. Gruteser et al. propose spatio-temporal cloaking [2] and Gedik et al. extend it using the k-anonymity model to make the algorithm configurable, and arguably introduced the concept of configurability of privacy level to the field of location privacy [3]. These algorithms propose a model that requires a TTP. There are a few that deviated from this TTP requirement. Mokbel et al. [4] propose anonymizing the service provider's database itself. Kalnis et al. [5] concentrated more on protecting the identity of the querying user rather than hiding her location. Olumofin et al. used Hilbert curves to anonymize, and to an extent, optimize K-nearest-neighbor queries [6]. Although the claim in most proposals is that K-nearest-neighbor queries form a bulk of LBS applications, we find that the location is not the only factor that actually goes into the ranking of objects.

Most algorithms use perturbation techniques to hide or generalize the location of the user. The differentiator of these algorithms from one another is the technique used to achieve perturbation. Gedik et al. propose an algorithm called *CliqueCloak* [3] that uses a quad-tree data structure to find a region that has at least a specified number of users. Similarly, Bamba et al. use *PrivacyGrid* [7] which incrementally adds or subtracts cells from a grid till a given minimum number of users are inside it.

Chaum propose an algorithm that uses anonymity sets [8], inspired by the dining cryptographers problem, and feel that the size of the anonymity set is a measure of privacy. Gruteser

et al. [2], who designed the k-anonmity based algorithm, treat the value k as a measure of privacy, while Xue et al. [9] import the concept of ℓ-diversity from the statistical database realm, and propose it as a companion measure. Reiter et al. [10] take a slightly different approach and propose the level of privacy in layman terms such as suspicion, probable suspicion and possible innocence. One of the measures we use for our study, *entropy*, is first proposed in the location privacy literature by Serjantov et al. [11]. Entropy as a measure of privacy is further extended by Diaz et al. [12] as normalized entropy, and Deng et al. [13] as relative entropy. Shokri et al. [14] start by modeling a more realistic attacker: an attacker with some prior knowledge (background knowledge) about the user's mobility. They model this knowledge as a probability distribution. Based on these models, they propose privacy measures that essentially quantify the failure of the attacker's estimation of users location as the measure of success of the algorithm.

The novelty of our approach is in the usage of local computations to determine the matching top results, using query platforms readily available from current search providers. In addition, we do not constrain ourselves to nearest neighbor results, which has been the major focus in earlier proposals.

III. Background

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. Most proposals treat this list to be sorted by distance and reduce the problem to a special case of the nearest-neighbor problem; however, most popular providers, e.g. Google, 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. In an earlier work, we proposed a TTP-free LPPM that leverages the processing power of modern mobile devices to perform most of the operations within the mobile device itself. For the sake of completeness, description of the model in the single query case is presented briefly, and then extended with elements that suit the multiple query scenario. Readers are requested to refer to [1] for details.

A. Single query scenario

A large geographical area is modeled as a $Z \times Z$ square grid of cells. A *cell* is defined as 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. The user configures her level of privacy by setting the value of a parameter b indicating that she does not care if the attacker narrows her location to an area larger than or equal to a $b \times b$ box (b cells by b cells). This is often specified in real life approximations such as the size of a mall, block or subdivision, that is translated by the application into the parameter b [1], [15]. The $Z \times Z$ grid is pre-partitioned into fixed non-overlapping boxes of $b \times b$ cells.

We use an architecture in which 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. 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, which a provider may not be willing to do anyways.

The client utilizes 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 POIs) for the user, and request details on those POIs from the server. 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 prevent such attacks, we need to augment the request set to include more than just the top-K POIs corresponding to the user's current cell.

In the pre-partitioned grid, if the box B happens to be the $b \times b$ box where the user is located, then an *interest set* I is computed as the union of the top-K POIs of the cells inside B, notationally the cells in a set C. Therefore, the interest set contains the POIs of interest to the user, as well as a few others. The client then queries the server for detailed information about the POIs in the interest set. In this case, an inversion attack will produce an area at least as large as $b \times b$ cells. The server responds with details of the POIs in the interest set. To determine the interest set, an algorithm using kd-tree data structures and heuristic pruning can significantly reduce the number of cells for which the top-K set has to be computed, effectively reducing the computation time to a few milliseconds [1].

B. Attacker model

Following the standard practice in the literature, we assume that the attacker knows the LPPM that is being utilized by the user. The attacker's goal is to determine the cell the user is in, based on the observed interest set: *localization attack*. To model an attacker in realistic terms, we assume that he has the following capabilities. The attacker has the ability to eavesdrop and observe the contents of the query and interest set; is not only aware that a LPPM is being used, but also knows the details of the algorithm, and its associated parameters; is also assumed to have some information (background knowledge) on the initial location of the user; knows the map of the geographical area and has access to the POI database; knows (or can accurately guess) the user's selection of privacy parameter b.

The attacker's knowledge of the user's location is modeled as a prior probability distribution Φ, where $\Phi(c_i)$ is the

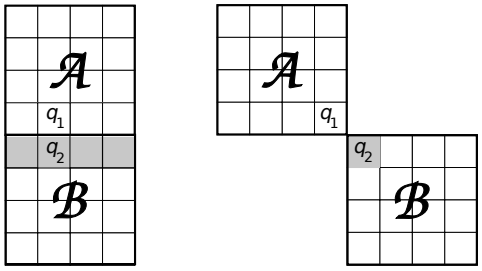


Fig. 1: Privacy breach during multiple queries.

attacker's prior estimate of the probability that the user is in cell c_i. Once the user sends the query, the algorithm obtains the POIs, determines the interest set and sends it to the service provider. During this time, the interest set I is observed by the attacker and he uses it to enhance his knowledge about the user's location. Given the set I, the attacker is able to find all boxes of size $b \times b$ that correspond to this interest set. One scenario is where the $b \times b$ box used for interest set generation, B, happens to be the only box that has I as its interest set. We assume this scenario for the discussion below. The attacker can obtain a refinement of the prior probability distribution using Bayesian inference.

$$Pr\left(c_i|B\right) = \frac{Pr\left(B|c_i\right)\Phi\left(c_i\right)}{Pr\left(B\right)} = \frac{Pr\left(B|c_i\right)\Phi\left(c_i\right)}{\sum_{c_j \in C} Pr\left(B|c_j\right)\Phi\left(c_j\right)}. \quad (1)$$

This posterior estimate is a constant multiple of the prior knowledge for any cell belonging to the box B. As long as the multiple queries by the user happen when the user is in the same box, the attacker's knowledge of the user's location will not be enhanced. The coarseness of the attacker's estimation remains $b \times b$.

Consider the 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 $Z \times Z$ grid. In this case, the attacker may be able to narrow down the user to an area less than $b \times b$. This is illustrated in Fig. 1. It shows two adjacent boxes $\mathcal{A}$ and $\mathcal{B}$, where $b = 4$. Let the time needed by the user to move by one cell be a constant T. Assume that the time interval between two queries is equal to the time required to move by one cell. Let us further assume that the user was in one of the cells in box $\mathcal{A}$ that borders with box $\mathcal{B}$. The user moves by one cell into box $\mathcal{B}$ and then issues her second query. The attacker sees that the first query's interest set matches with box $\mathcal{A}$ and second query's interest set matches with box $\mathcal{B}$. This clearly allows the attacker to narrow down the user's location to the boundary cells (shaded) of the box $\mathcal{B}$ since the user had only enough time to move by one cell. The attacker is able to narrow down the user's location to an area much smaller than $b \times b$, thus clearly breaching the user's privacy requirement. This shows that obfuscation is not guaranteed in the case of multiple queries happening across boxes: this is the main problem that we address in this work.

IV. Proposed Solution

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 and the first query, but not for the subsequent queries. In order to prevent the privacy loss during subsequent queries, we first create a new area, hereafter called the *selection area* S, by expanding the box generated at the previous query to its *neighboring cell*s. If n is the number of T time periods elapsed between two subsequent queries, then a neighboring cell is any cell that is no more than n rows or columns away from the current box. Formally, $n = \left\lceil \frac{t_k - t_{k-1}}{T} \right\rceil$ where t_k and t_{k-1} are the time of issuing the k^{th} and $(k-1)^{th}$ queries respectively. The box for the new query is determined by selecting a random $b \times b$ box from within the selection area such that it contains the user.

A. Attacker model – multiple query scenario

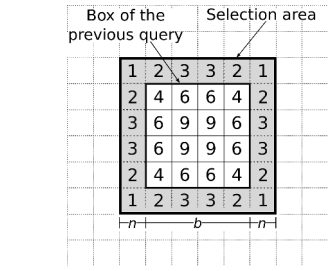
For the multiple query scenario, in addition to the capabilities mentioned in Section III-B, we assume that the attacker knows the unit time period T needed by the user to move by one cell. If the attacker observes two consecutive queries, based on the time elapsed between them, the attacker can estimate the maximum number of cells, n, that the user can move during that time. For the multiple query scenario, we continue utilizing the notations introduced so far, but extend it by using a subscript. The subscript represents the query number, e.g. B_k represents the box selected for interest set generation during the k^{th}query q_k. By definition, B_k contains the user's location at that time instance. Correspondingly, I_k is the generated interest set, and Φ_k represents the attackers estimated probability distribution at the end of the k^{th} query. The distribution Φ_1 is the attacker's estimation after the first query is made, as given by Equation 1.

The attacker's goal is to narrow down the user's location to a specific cell. Short of that, the attacker 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. Once the attacker observes the interest set I_k, he determines the set of boxes within the selection area whose interest set match I_k. In the specific scenario we are considering, this set has exactly one $b \times b$ box, the one that was actually used, namely B_k.

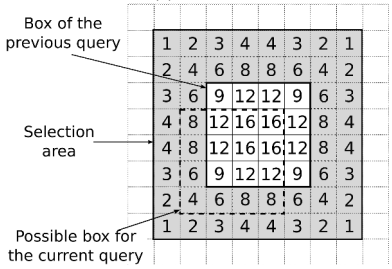
B. Selection area

If the user does not hit the border of the $Z \times Z$ grid while moving n steps from any cell in box B_{k-1}, then the selection area S_k for the query q_k forms a square of size $(b + 2n) \times (b + 2n)$. The selection area S_k represents all possible cells the user could be in for query q_k. Fig. 2a and 2b show S_k where $n = 1$ and $n = 2$ respectively. Let the cells of S_k be numbered $c_1, c_2, c_3 ... c_{(b+2n)^2}$ starting at the top left corner, continuing row by row.

Looking at selection area S_k, the number of possible $b \times b$ boxes that contain the cell c_1 is only one. We call this number the *weight* w_i of a cell, and is defined as the number of possible $b \times b$ boxes from which the algorithm could choose for the next query if the user is located in c_i. As we advance by one cell to the right along the horizontal direction, the number of possible boxes that include the cell c_2 increase by one as well. If $2n \geq b$, this increment continues until the cell c_b is reached. This is illustrated in Fig. 2b, where $b = 4$ and $n = 2$. When $2n < b$, the increment stops at the cell c_{2n+1}, as shown in Fig.



(a) $n = 1$.



(b) $n = 2$.

Fig. 2: Selection area with weight of a cell.

2a with $b = 4$ and $n = 1$. The process is symmetric if started from the right side, vertically down from the top corner, or vertically up from the bottom corner.

One could see that the selection area is a square in most instances. But it can be a rectangle if a border of the $Z \times Z$ grid is reached before the n steps. In later discussions, it will become clear that a square shape for the selection area has useful properties to our algorithm, and is therefore enforced by a transformation.

C. Modeling the user's movement

One of the differences between the single query and the multiple query scenarios is the user's movement between the queries. We model this user movement as follows. If the user was in cell $c_j \in C_{k-1}$ at the time of the previous query q_{k-1}, then by the time of the current query q_k, she can move to any neighboring cell c_i or continue to stay in c_j. The probability of her moving to a neighboring cell c_i from cell c_j, called the *transition probability*, is denoted as ρ_{ji}. We work with a uniform movement model, where for any neighboring cell c_i, the transition probability $\rho_{ji}, \forall c_j$ is a constant value equal to $\frac{1}{(1+2n)^2}$; for all other cells, it is zero.

Refer to Fig. 2. It shows the selection area S_k that is created based on the box B_{k-1} used in the previous query. Observe that the weight of each cell c_i in S_k also represents the number of cells in B_{k-1} that the user could be in and reach c_i in at most n steps. Note that for cells within B_{k-1}, the weight w_i includes the possibility that the user decides to stay in the same cell c_i.

Before the user performs query q_k, the attacker combines his knowledge of the user's movement model and B_{k-1} to compute a new distribution for the user's location as follows.

$$\lambda_k(c_i) = \sum_{j=1}^{Z^2} \left(\Phi_{k-1}(c_j) \times \rho_{ji}\right), \tag{2}$$

where λ_k represents the probability distribution of the user's location based on the movement model. When query q_k is made, the attacker observes the new interest set, and subsequently determines the box B_k used in the query. Next, the attacker enhances his knowledge using Bayesian inference,

$$\Phi_k(c_i) = Pr(c_i|B_k) = \frac{Pr(B_k|c_i) \times \lambda_k(c_i)}{\sum\limits_{c_j} \left(Pr(B_k|c_j) \times \lambda_k(c_j)\right)}. \tag{3}$$

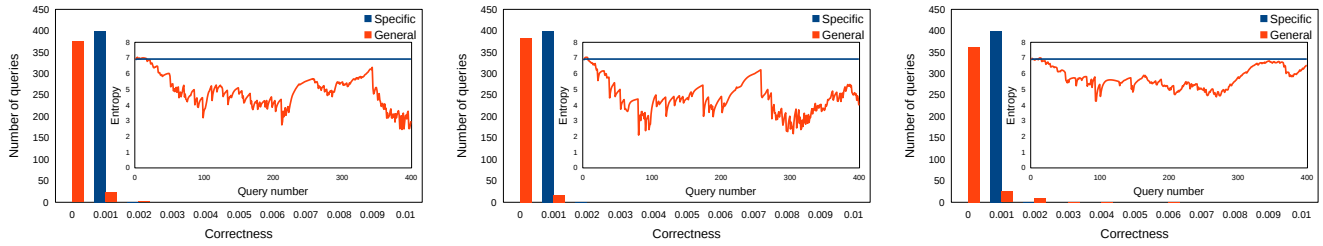
D. Algorithm

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. For the first query, as described in Section III-A, 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. We briefly mentioned the method used for subsequent queries in the beginning of this section; the detailed steps are discussed below.

Let us assume that the algorithm is trying to generate the interest set for query q_k and n be the maximum number of cells that the user could have moved after the previous query. Before making the $b \times b$ box selection for query q_k, the algorithm has to ensure that the selection area S_k does not touch the borders of the $Z \times Z$ grid. It does this by creating a new $Z \times Z$ area for each query q_k, such that it contains the box used for the previous query, i.e. B_{k-1}, at its center. Because of this realignment of the $Z \times Z$ grid, the client device may need to download a new set of matching POIs. This can add to the communication overhead between the client and the server. To reduce this additional overhead, the client caches the POI set of the previous query. Then, for the new $Z \times Z$ grid, it only needs to retrieve location and prominence data on POIs that may appear in the non-overlapping areas.

For the second query ($k \geq 2$), and subsequent ones, the algorithm first calculates n based on query timestamps and determines the selection area S_k. The box for the current query is selected by picking a $b \times b$ box uniformly at random from S_k such that it contains the user. The algorithm uses the same techniques used for the single query scenario to efficiently generate the interest set and sends it to the LBS to obtain the details. Finally, the client will prepare for the next query by realigning the $Z \times Z$ grid. 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.

The chances of the user being in a cell outside the selection area is zero, because the user cannot move any farther in the given number of time units (n). Refer to Fig. 2 for the cases where $n = 1$ and $n = 2$. A cell in the selection area has zero probability only if the prior estimate of the attacker for that cell, and its neighbors, are zero. As a result, a box chosen by our algorithm will have all non-zero probability cells, unless the attacker's background knowledge can help eliminate cells. Therefore, the algorithm preserves the required $b \times b$ cell obfuscation if the attacker's prior knowledge is not already stronger. It can also be shown that the probabilities of



(a) Path P_1. (b) Path P_2. (c) Path P_3.

Fig. 3: Entropy and correctness of location estimation.

the cells in the box selected for the current query are equal if the cells in the box selected for the previous query had uniform probabilities.

V. Empirical Evidence

In order to evaluate our algorithm, we first select a quantitative measure to evaluate privacy. To measure the attacker's uncertainty, we select *entropy*, $H(\Phi_k) = -\sum_{c_i} \Phi_k(c_i) \ln \Phi_k(c_i)$, as the measure. Intuitively, higher entropy values reflect higher uncertainty for the attacker about the user's exact cell. If the attacker's location estimation for all $c_i \in C_k$ is equal, then the entropy is maximum and is equal to $2 \ln b$ for a $b \times b$ box.

From the user's perspective, measuring location privacy boils down to the chance of the attacker locating her in the exact cell, also known as *correctness* [14]. The lower the probability that the attacker assigns to the actual cell of the user, the higher the privacy. Without any other background knowledge, if the attacker narrows down the user to a $b \times b$ box, he would assign equal probabilities ($= \frac{1}{b^2}$) to each cell in this box, which indicates the highest uncertainty for the attacker.

In this section, we discuss the experiments performed to support the conclusions from several aspects of the proposed algorithm. The first set of experiments are designed to show that the algorithm provides reasonable privacy protection, as measured using entropy and correctness. The second set of experiments are performed to show that the requesting of additional POIs in an incremental fashion has minimal performance impact. For the evaluation, we use an area of Los Angeles, CA, USA, divided into 320×320 cells, and POIs are obtained from the SimpleGeo database. A cell in this setting is $100m \times 100m$. We use a relatively high density POI category (608 cafes) in the evaluations, unless stated otherwise.

A. Entropy and correctness

To demonstrate how entropy and correctness change as more queries are made, we ran three movement simulations: two along pre-defined paths (denoted by P_1 and P_2), and one along a random path (denoted by P_3). The number of queries in a simulation (a session) is fixed at 400. The pre-defined paths are generated such that the user is exposed to varying local distributions of the POI. The privacy parameter b is set to 32, which implies an obfuscation expectation of roughly $10km^2$. We simulate the attacker's initial knowledge Φ as a two-dimensional Gaussian distribution centered at the user's actual location.

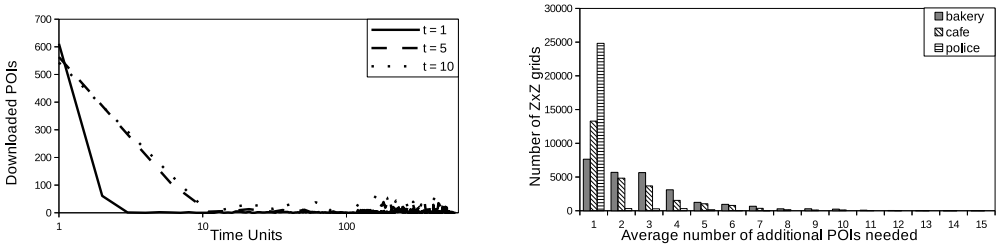
Fig. 3a, 3b and 3c summarize the entropy and correctness values for the three movement simulations in the context of the gaussian initial knowledge. The x-axis labels on the correctness plots signify the lower ends of the intervals used for generating the histogram. The general scenario signifies the case when the attacker finds multiple $b \times b$ boxes corresponding to the observed interest set, and accordingly accounts for it in the inference process (the details of the process are withheld here for space considerations). The quick convergence to a uniform posterior distribution (entropy saturation) is evident in all three simulations for the specific scenario when the exact box used by the algorithm is inferable by the attacker. It is also evident that once the entropy attains its highest value, changes do not occur. The entropy values do not demonstrate any monotonic behavior for the general (more realistic) scenario. However, there is a significant shift in the interval where majority of the correctness values (probability associated with the user's location) lie when considering the general scenario. This is partially because more cells could be associated with the user and the probabilities get distributed across these larger number of cells. We do not stress much on the numeric value of correctness since it is a function of the number of cells in the box(es) inferred by the attacker.

These observations suggest that, as time passes, the attacker's initial knowledge, no matter how strong, gets dispersed due to the movement of the user. Therefore, the chances of correctly locating the user reduces. We also observe that there is little or no correspondence between the entropy measure and the correctness measure of privacy, which conforms to conclusions made by Shokri et al. in a separate study [14].

B. Quality of service

Next we assess the overhead introduced by the algorithm in the grid realignment step. To maintain the square shape of selection areas, the algorithm proposes realignment of the $Z \times Z$ grid, and obtaining location and prominence data on additional POIs for each query. Using a Z value of 160, we count the number of new POIs that appear when the grid is realigned. We start the user in a random cell and simulate the movement in a pre-determined path while issuing queries.

The plot in Fig. 4a shows the number of previously unseen POIs appearing in each query. It shows the results for the



(a) Bakery.

(b) All possible 160×160 grids with $n = 1$.

Fig. 4: Number of POIs for which location and prominence is downloaded as multiple queries are made.

POI string "bakery", which represents a dense distribution of POI in the available data. Each simulation is executed with $t = 1T, 5T$, and $10T$ as the time interval between queries. The plots show the time on the x-axis (log scale) and the number of new POIs obtained for the query issued at that time on the y-axis. As expected, the initial set of POIs obtained for the first and second queries are large, but for all of the subsequent queries, the number of new POIs obtained is much less. This translates to a small and acceptable impact on the QoS. For Fig. 4b, we consider all possible 160×160 grids that can be embedded into the 320×320 map, and obtain the average number of additional POIs appearing when a grid is shifted to its neighbors. We see that, for different POI categories, the average number of additional POIs obtained is less than 5 in majority of the grids.

VI. Conclusion

In this work, we have taken a first attempt at leveraging the computational capabilities of modern mobile devices to design practical location privacy algorithms for points-of-interest (POI) search. We have presented a technique to repeatedly retrieve local POI data from a service provider, without requiring the user to reveal precise location information. The information exchanged in the process is subjected to inference analysis under a Bayesian adversary model. Through theoretical arguments and empirical evidence, we have shown that the proposed technique leaks little or no advantageous information to the attacker, can efficiently operate in a mobile device, and has minimal impact on the communication bandwidth. We are optimistic that future work in this area will continue in this direction of practicability, and generate innovative usages of the newly available computation node for location privacy preservation.

Acknowledgment

This work is funded in part by the National Science Foundation under Grant No. DUE-0911991. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the National Science Foundation.

References

[1] 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*, 2013, pp. 1–8.

[2] 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*, 2003, pp. 31–42.

[3] 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*, 2005, pp. 620–629.

[4] C.-Y. Chow, M. F. Mokbel, and X. Liu, "A peer-to-peer spatial cloaking algorithm for anonymous location-based service," in *Proceedings of the 14th Annual ACM International Symposium on Advances in Geographic Information Systems*, 2006, pp. 171–178.

[5] 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, pp. 1719–1733, 2007.

[6] F. Olumofin, P. K. Tysowski, I. Goldberg, and U. Hengartner, "Achieving efficient query privacy for location based services," in *Proceedings of the 10th International Conference on Privacy Enhancing Technologies*, 2010, pp. 93–110.

[7] B. Bamba, L. Liu, P. Pesti, and T. Wang, "Supporting anonymous location queries in mobile environments with privacygrid," in *Proceedings of the 17th International Conference on World Wide Web*, 2008, pp. 237–246.

[8] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 24, pp. 84–90, 1981.

[9] M. Xue, P. Kalnis, and H. Pung, "Location diversity: Enhanced privacy protection in location based services," in *Proceedings of the 4th International Symposium on Location and Context Awareness*, 2009, pp. 70–87.

[10] M. K. Reiter and A. D. Rubin, "Crowds: Anonymity for web transactions," *ACM Transactions on Information Systems Security*, vol. 1, pp. 66–92, 1998.

[11] A. Serjantov and G. Danezis, "Towards an information theoretic metric for anonymity," in *Proceedings of the 2nd International Conference on Privacy Enhancing Technologies*, 2002, pp. 41–53.

[12] C. Díaz, S. Seys, J. Claessens, and B. Preneel, "Towards measuring anonymity," in *Proceedings of the 2nd International Conference on Privacy Enhancing Technologies*, 2002, pp. 54–68.

[13] Y. Deng, J. Pang, and P. Wu, "Measuring anonymity with relative entropy," in *Proceedings of the 4th International Conference in Formal Aspects in Security and Trust*, 2006, pp. 65–79.

[14] R. Shokri, G. Theodorakopoulos, J.-Y. Le Boudec, and J.-P. Hubaux, "Quantifying location privacy," in *Proceedings of the 2011 IEEE Symposium on Security and Privacy*, 2011, pp. 247–262.

[15] 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*, 2009, pp. 348–357.