Beyond the Thin Client Model for Location Privacy

Rinku Dewri, Wisam Eltarjaman, Prasad Annadata, Ramakrishna Thurimella
Department of Computer Science, University of Denver
{rdewri,wisam,prasad,ramki}@cs.du.edu

***Abstract*—Location privacy in emerging location-based applications has been the center of extensive research in the past decade. While a number of efficient models and algorithms have been proposed over the years, majority of them were designed assuming a thin client model of computing. As a result, the dependence on third party systems, or the requirement for significant upgrades to the application architecture, could not be eliminated. The state-of-the-art in current mobile devices is now comparable to traditional desktop systems five years ago, presenting us the opportunity to move beyond the thin client model. Motivated by this observation, we propose a novel architecture for performing location-based points-of-interest search, where the client device can locally determine the top results using a small amount of metadata from the server. Precise location data of the user is never transmitted outside the device. We demonstrate that the computational power required to efficiently execute the algorithms is within the capabilities of current mobile devices.**

***Index Terms*—Location privacy, smartphone, adversarial knowledge.**

I. Introduction

The ability to search for nearby points-of-interest (POI) has been greatly enhanced by the introduction of global positioning systems in mobile devices. This technological innovation now allows any application developer to provide recommendations based not only on the relevance or importance of results, but also on the geographic location of the user. Location-based search, or *local search*, is becoming a de facto standard to find objects of interest, and many third-party developers (e.g. Google, Yelp, AroundMe, etc.) are already involved in getting a share of this expanding market. Without surprise, there is also an alarming concern about the location data that gets collected at the service provider, and its subsequent sharing with unauthorized or untrusted parties. In order to mitigate the privacy risks, a large body of research in the past decade has proposed models and algorithms that prevent a service provider from getting direct access to the location data of the user.

A. Related Work

Gruteser and Grunwald [1] propose 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 [2]. 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—thresholding the number of still-objects [3], maintaining a minimum level of entropy in the queries originating from the cloaking region [4], enforcing reciprocity [5], [6] and using the popularity of a public region as the privacy level [7], [8], among others.

Shokri et al. provide a method to arrive at different types of inferences regarding a user's location based on a known mobility profile of the user [9]. Using methods of likelihood estimations, they show that measures such as the anonymity set size or entropy, do not correctly quantify the privacy enforced by the method [10]. Their recent work on optimal location obfuscation methods introduces user-centric service quality requirements in location-based applications [11], which relies on LP solvers that may not be practical for a mobile device.

The obfuscation based approach is the most extensively researched method for location privacy. A relatively fewer attempts are seen using other approaches, e.g. the mix zone model of Beresford and Stajano [12], the dummy queries method of Kido et al. [13], and the private information retrieval techniques of Khoshgozaran and Shahabi [14].

B. Motivation

It is not difficult to observe that a majority of these attempts try to preserve privacy by significantly augmenting the server-side functionalities, or engaging an intermediate party to perform computationally expensive operations. The motivation for doing so is justified, given that mobile devices have only recently become powerful enough to be considered full fledged computing platforms. The state-of-the-art in the processing capabilities of mobile devices was around 400 MHz in 2007 (e.g. the 32-bit RISC ARM in the first iPhone), which has grown exponentially to support multicore systems with clock speeds of up to 1.7GHz in 2012 (e.g. the quad-core NVIDIA Tegra 3 on the HTC One X+). To the best of our knowledge, prior to this work, such advances in the computing power of mobile devices have not been exploited by the privacy research community.

C. Contributions

Our first contribution is a novel architecture to perform local search, where a significant part of the processing involved in computing the top matches for a query is performed on the client side of the application (on the mobile device). Given a set of object locations and a reference location, the algorithms implement a basic branch-and-bound search technique to obtain the top results for the reference point. Using this capability, the client device can identify the top results for multiple geographic locations, and request details for a superset of her actual results. The computation happens

at the client device, without requiring the transmission of the exact user location to the service provider or any third party. Such offloading of computations to client devices also massively increases the scalability of the entire application system. Our results indicate that we can facilitate the process with sub-second delays in user experience, even on moderately old hardware.

As our second contribution, we propose a privacy evaluation model for assessing the privacy level of users in the proposed architecture. Our formalization uses a probabilistic model of attacker knowledge, and allows us to compute the geographic uncertainty that an adversary would have in accurately locating the user, given that the user requests details on certain points of interest only. During the formalization of the model, we also identify necessary properties that every privacy algorithm for local search should guarantee, and propose a general optimization problem whose solution satisfies those requirements. Part of the solution requirements of this problem is to minimize the communication overhead associated with obtaining result details. Our privacy algorithm is motivated by this problem, although the optimal solution is yet to be determined.

The remainder of the paper is organized as follows. Section II presents our architecture for local search. Section III presents the attacker model and the general problem statement. Section IV describes our algorithm, followed by an empirical study in section V. The paper is concluded in section VI.

II. Mobile Local Search

In a mobile local search application, the user specifies a search term signifying the type of objects to be retrieved, along with a location tag indicating the geographic position of the user. With proper permissions, the location tag can be directly obtained by the application from an on-board positioning device. Based on this geographic position, and associated features of the relevant objects, the service provider determines a top-K list of objects to return to the user. The top-K list is the first K objects in a ranked ordering of the objects that match the search term, for some constant integer K. Values of $K = 10$ or 20 are typical.

Result ranking in local search is based on multiple factors. For example, ranking in the Google Places [15] application is driven by two factors, namely the distance of the POIs from the user's location and their prominence. The prominence of a POI is derived from multiple sub-factors such as reference counts, highest score of objects that refer to this object, number of user reviews, and the extent of services offered, among others.

A. Request-response architecture

In a typical two step process of doing local search, a user sends the query to the service provider, along with the location tag, and receives a response to the query. In order to prevent unwarranted data collection, we adopt a privacy supportive architecture where the client and server ends of the application perform intermediate meta-data exchanges to generate the final result set. The following steps summarize the communication pattern in the proposed architecture.

Step 1 Client-end determines a broad geographic area that includes the user's current location. This can be done by generating a random rectangular region of a sufficiently large area (say 1000 km^2) such that the user's location is contained within the region.

Step 2 Client sends the broad area and search keyword to the server-end.

Step 3 Server obtains a list of POIs matching the search keyword.

Step 4 Server sends the location and "importance" information about the matched POIs to the client. The server minimizes the communication overhead by not transmitting any feature data (e.g. object names, phone numbers, addresses, etc.) at this point. The method of calculating the importance of a POI is discussed later. The location and "importance" data enable a client to locally compute the top-K POIs.

Step5 Client uses the obtained data to determine a subset of POIs. The subset contains the top-K objects with respect to the user's current location, and a few additional ones depending on the algorithm used to generate the subset.

Step 6 Client requests detailed feature data for the POI subset from the server.

Step7 Server responds with the feature data of the requested POI subset.

Compared to the conventional process, steps 4, 5 and 6 incur slight additional communication and computation overhead. However, as demonstrated later, the overhead can be maintained within levels that does not have a noticeable impact on user experience.

B. Formal model

Consider a broad rectangular region that has been discretized into a $z \times z$ grid of cells. The set of all cells is denoted by C. A geographic location is then signified by a point c in C. A user located in a cell issues a search query as specified in step 2 of the architecture. We assume that queries are issued in a sporadic manner, i.e. the time gap between successive queries cannot be used to infer the distance traveled by the user in that period (e.g. in continuous query systems). Let $P = \{p_1, ..., p_n\}$ denote the set of POIs inside the broad 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. $rank'(p_i, c_{user}) = \alpha \times \text{dist}_{norm}(c_{user}, c_i) + (1 - \alpha)(1 - \beta_i)$. We use the length of the diagonal of the broad 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 ranks implying better choices. In most situations, values of α and β_i are not revealed to the user. Hence, we redefine the ranking function as

$$rank(p_i, c_{user}) = \frac{rank'(p_i, c_{user})}{\alpha} = \text{dist}_{norm}(c_{user}, c_i) + \gamma_i, \quad (1)$$

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, in step 4 of the process, the server sends the tuples $T = \{\langle c_i, \gamma_i \rangle | i = 1...n\}$ to the client. Using T and the current location, the client uses a privacy-preserving algorithm to derive a subset I of P. We refer to the set I as the *interest set* of the user for the specific query. This set is used in steps 6 and 7 of the architecture.

III. Measuring Location Privacy

The adversarial model in our work is similar to what is typically assumed in the literature—honest but curious. Therefore, the client and server end of the application follow the protocol outlined in section II-A, yet the service provider (or an eavesdropping adversary) may attempt to infer the cell of the user from the exchanged information. We model the adversary's background knowledge as a probability distribution Φ, such that $\Phi(c)$ is the adversary's prior estimate of the probability that the user is at cell c.

A. Designing a privacy algorithm

Once the adversary obtains the interest set I from the user, the prior probabilities for the cells can be refined. This is because the interest set itself reveals some information about the likely locations of the user. Therefore, the adversary estimates the posterior distribution of the cells, i.e. the probability of the user being at cell c given that the set I was generated by the privacy algorithm, given as

$$Pr(c \mid I) = \frac{Pr(I|c)Pr(c)}{Pr(I)}. \tag{2}$$

Here $Pr(I|c)$ is the probability of generating the set I if the user is at cell c, and $Pr(c)$ is the probability of the user being at c, which is equal to the prior probability $\Phi(c)$.

The privacy-preserving algorithm in this context can be abstracted as the composition of two functions $\mathcal{A}(\mathcal{R}(\cdot))$, such that $\mathcal{R}$ maps a cell to a *region* (set of cells, i.e. an element of the power set of C) and $\mathcal{A}$ maps a region to a (sub)set of the matching POIs, i.e. $\mathcal{R} : C \rightarrow 2^C$ and $\mathcal{A} : 2^C \rightarrow 2^P$. The domain of the function $\mathcal{A}$ is determined by the range of $\mathcal{R}$, denoted as $C_1, ..., C_m \in 2^C$. Correspondingly, let $I_1, ..., I_m$ be the POI (sub)sets that are mapped to these regions by $\mathcal{A}$. Without loss of generality, assume an arbitrary set $I_u \in \{I_1, ..., I_m\}$ that is generated as the interest set for a user located in region C_u.

Consider the case when $I_u \neq I_t, t = 1, ..., u-1, u+1, ...m$. Then, for all cells $c_i \in C_u$, we have the posterior probability as

$$Pr(c_i \mid I_u) = \frac{Pr(I_u \mid c_i)\Phi(c_i)}{Pr(I_u)} = \frac{Pr(I_u \mid c_i)\Phi(c_i)}{\sum_{cell\ c_j} Pr(I_u|c_j)\Phi(c_j)}, \tag{3}$$

and for all cells $c_j \notin C_u$, we have $Pr(c_j \mid I_u) = 0$. In order to achieve *obfuscation*, we should have a "large" number of cells (including the cell of the user) in C_u with positive probabilities. This number may be specified as part of the privacy policy of the user. However, user specified requirements cannot be met if the adversary already has background knowledge that probabilistically ties the user to a smaller area. Hence, a precise statement (such as the Φ function) is necessary on the adversary's knowledge.

Revealing I_u does reveal some location information about the user, the region C_u in this case. This disclosure is an artifact of using the service in the first place. The posterior probability distribution may further indicate that some cells in C_u are more likely than others. However, the same inference may simply have been derived based on the prior knowledge, implying that no additional information about the user's location within C_u was revealed by the posterior distribution. This can be guaranteed if the posterior probability distribution in the region C_u is directly proportional to the prior probability distribution in the region. We refer to this as the *convergence* requirement of the algorithm. In Eq. 3, convergence is achieved if $Pr(I_u \mid c_i)$, for any $c_i \in C_u$, is a constant. Therefore, the probability of producing the output I_u should be equal for all cells in the region C_u.

In addition, the need for a high *quality of service* (low communication overheads in this context) warrants the creation of smaller POI (sub)sets. Ideally, the sets $I_1, ..., I_m$ should each have only K elements. The three requirements should also hold for the case when more than one region in the domain of $\mathcal{A}$ maps to I_u, i.e. $I_u = I_t$ for some t. Specifically, the convergence requirement should collectively hold in the union of all those regions that are mapped to I_u by $\mathcal{A}$.

B. Problem statement

It is difficult to state what all forms can be taken by a privacy algorithm that provides reasonable levels of obfuscation, convergence and high service quality. We present here a problem formulation, the solution to which preserves the desired properties and may be solvable in an efficient manner.

Consider a partitioning of the set C into regions $R_1, R_2, ..., R_q$, such that the top-K POI sets for every cell in a given region R_i are the same. Let $P_1, ..., P_q$ represent the top-K POI sets of these regions. We consider a function $\mathcal{R}$ that creates disjoint groups $C_1, ..., C_m$ of the q regions such that the following two conditions hold.

1) *(Obfuscation requirement)* For all groups C_i, $| \{c \in C_i | \Phi(c) > 0\} | \geq \delta$.
   $\delta \leq |\{c \in C | \Phi(c) > 0\}|$ is a parameter signifying the level of obfuscation desired by the user. A trivial method to meet this requirement is to create a single group of the union of all regions. However, this solution may negatively impact the service quality of the application, motivating us to include the second constraint.
2) *(QoS requirement)* A service quality function $g(C_1, ..., C_m)$ is minimized. The function g specifies the service level impact of grouping the different regions. For example, in the context of minimizing communication overhead, g may be defined as $\max_{C_i} |P_{i_1} \cup ... \cup P_{i_t}|$, where C_i is created by grouping regions $R_{i_1}, R_{i_2,...}, R_{i_t}$.

Note that $\mathcal{R}$ is a many-to-one function, i.e. a cell can belong to only one of the $C_1, C_2, ..., C_m$ regions. We next consider a function $\mathcal{A}$ such that $I_i = \mathcal{A}(C_i) = P_{i_1} \cup ... \cup P_{i_t}$, where $C_i = R_{i_1} \cup ... \cup R_{i_t}$. Therefore, given a cell c_{user}, a privacy algorithm using $\mathcal{R}$ and $\mathcal{A}$ returns the set I_u as the interest set if $c_{user} \in C_u$. Note that $\mathcal{A}$ is not necessarily a one-to-one mapping, but is many-to-one. The obfuscation and convergence requirements hold for this algorithm. Assuming I_u is unique ($\mathcal{A}(C_i) = I_u \iff i = u$), for all cells $c_i \in C_u$, we have $Pr(I_u|c_i) = 1$.

$$\therefore \quad \forall c_i \in C_u, Pr(c_i \mid I_u) = \frac{Pr(I_u \mid c_i)\Phi(c_i)}{Pr(I_u)} = \frac{\Phi(c_i)}{Pr(I_u)} = \text{a constant} \times \Phi(c_i) \tag{4}$$

For all cells $c_j \notin C_u$, we have $Pr(c_j \mid I_u) = 0$. By virtue of the first problem constraint, we have $|\{c \in C_u | \Phi(c) > 0\}| \geq \delta$; therefore, at least δ cells (including the cell of the user) in C_u have positive posterior probability (obfuscation). Also, the posterior probabilities associated with this reduced set of cells is directly proportional to the background knowledge (convergence). The observations trivially hold when I_u is not unique.

To summarize, an efficient privacy algorithm with the obfuscation, convergence and QoS requirements can be created if the function $\mathcal{R}$ can be constructed efficiently.

C. Estimating privacy and overhead

The obfuscation requirement for a privacy algorithm gives us the freedom to generate different groupings of the cells with minimal impact on service quality. This is particularly useful in cases when the δ value is significantly lower than the number of cells with positive prior probabilities. However, the δ value itself is not an accurate measure of the privacy level generated by the algorithm.

Assume that the adversary wants to find the user in real time. We consider *localization* to be the process of using the posterior probability distribution to determine where the user is present. We call it *exact localization* if the adversary has one attempt to determine the user's cell. The probability of exact localization is $Pr(c_u|I_{user_u})$, where c_u is the true cell of the user and I_{user_u} is the interest set generated for that cell. Therefore, $1 - Pr(c_u|I_{user_u})$ can be treated as the privacy of the user in exact localization. In *inexact localization*, the adversary has multiple attempts to determine the user's cell. The attempts in this case are directed by the posterior probability distribution. The adversary samples a cell based on the posterior distribution, followed by another, and another, until the user's cell is chosen. This is similar to the adversary visiting different locations before finding the user. Therefore, the number of cells chosen by the adversary before choosing the user's cell can be viewed as the uncertainty area for the adversary; consequently, the privacy of the user. The expected value of this random variable is

$$\eta(c_u, I_{user_u}) = \sum_{i \neq u} \frac{Pr(c_i|I_{user_u})}{Pr(c_i|I_{user_u}) + Pr(c_u|I_{user_u})}. \tag{5}$$

Considering for the case that the adversary has prior estimates on the user's location, the expected privacy in the exact case is given as

$$E[Privacy_{exact}] = \sum_{cell\ c_i} (1 - Pr(c_i|I_{user_i}))\,\Phi(c_i), \tag{6}$$

where I_{user_i} is the interest set for cell c_i. Similarly, the expected privacy in the inexact case is given as

$$E[Privacy_{inexact}] = \sum_{cell\ c_j} \eta(c_j, I_{user_j})\Phi(c_j). \tag{7}$$

Eqs. 6 and 7 are correct under the assumption that the algorithm always outputs a fixed interest set for a given input cell, i.e. both $\mathcal{R}$ and $\mathcal{A}$ are many-to-one functions. If any of these functions is many-to-many, then the equation should be modified accordingly.

We also compute the expected overhead as the expected number of POIs in an interest set.

$$E[Overhead] = \sum_{cell\ c_i} |I_{user_i}|\Phi(c_i). \tag{8}$$

IV. Interest Set Computation

The problem statement proposed in section III-B serves as an objective that one may try to achieve while designing a privacy algorithm for local search. However, it is open whether the computational overhead associated with the solution process will be practical on a mobile device. Alternatively, instead of obtaining the optimal regions $C_1, ..., C_m$, we take an engineering approach and divide the set of cells C into pre-defined regions. These regions are created by overlaying the grid with a coarser grid of size $b \times b$. All cells contained within a coarser cell then form a region. In this case, we have $m = \left(\frac{z}{b}\right)^2$ (assume z is a multiple of b) and each region C_i is simply a sub-grid of the original grid. Note that, in this formulation of the function $\mathcal{R}$, we have ignored the obfuscation requirement, under the assumption that the property can be achieved by increasing b. Service loss minimization is also not performed explicitly; nonetheless, we expect the union size of the top-K sets to be relatively low since neighboring cells are collected together to form the regions. The convergence requirement still holds here using the same arguments as in section III-B. The interest set computation for a cell is then performed in two steps—(i) $\mathcal{R}$:determine the sub-grid C_i where the cell belongs, and (ii) $\mathcal{A}$:obtain the union of the top-K POI sets of the cells in C_i.

A. Top-K of a sub-grid

Given the set of POIs P, and a sub-grid C_i of cells, we need a method to calculate the set of top-K POIs for each cell in the grid, and return the union of those sets. A brute force technique to achieve this is trivial, although results in a sluggish user experience. We propose a faster algorithm consisting of the following steps.

1) Using the full set P of POIs, we use a kd-tree based branch-and-bound search algorithm to obtain the union

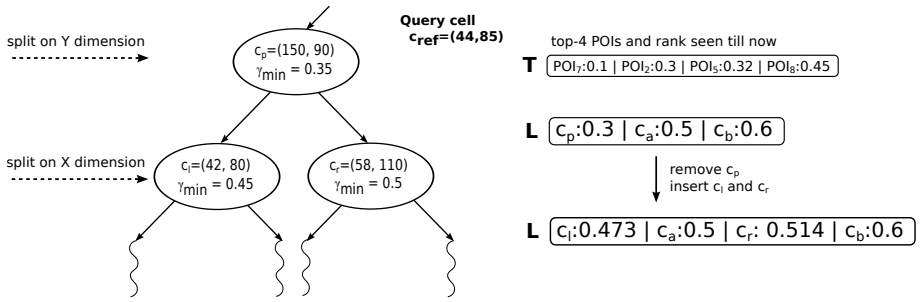

Fig. 1. Example kd-tree for lower bound computation.

set of the top-K POI sets of each cell on the border of C_i. We denote this set as I^{border}.

2) Determine subset $P_{internal}$ of P, the set of POIs that occur inside, or on the border of, region C_i. Create a reduced set of POIs $P_{reduced} = I^{border} \cup P_{internal}$.
3) Compute union set $I^{internal}$ of the top-K POI sets for each non-border cell in C_i. 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 $I^{internal}$ in an incremental manner (one cell at a time) and stop when $P_{internal} \subseteq I^{internal}$, or when the top-$K$ set for all non-border cells have been evaluated. The correctness argument for this step is detailed in section IV-C.
4) The interest set I_i is returned as $I^{border} \cup I^{internal}$.

B. kd-Tree branch-and-bound search

We first construct a kd-tree using the cell co-ordinates of the POIs in P. The construction uses standard mechanisms for determining the root, and left and right children of each subtree. In addition, each node in the tree is augmented with the minimum possible γ 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–(i) a list T representing the top-K nodes (POIs), ordered according to the rank of the nodes with respect to c_{ref}, and (ii) 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 for the subtree rooted at that node. During the search, we explore the nodes in L in their order of appearance (effectively a best first search strategy), and terminate when L becomes empty, or it is determined that no node in the subtree can potentially change the existing T list. Exploring a node involves the steps of checking if the node can be inserted in the T list (based on its rank), computing the lower bounds for the left and right children, and then inserting them in L. We explain the process using an example.

Consider the L list shown in Fig. 1 during a top-4 search for the query cell $c_{ref} = (44, 85)$. We delete the first node in the list and consider the subtree rooted at that node (c_p at $(150, 90)$ in the example). Say the grid is of size 320×320, giving us a normalizing factor of $1/320\sqrt{2}$ for distance calculations. Assuming that the γ value of the POI corresponding to node c_p is 0.35, the rank of c_p is then $\frac{\sqrt{(150-44)^2+(90-85)^2}}{320\sqrt{2}}+0.35 = 0.584$. Therefore, c_p cannot be inserted into the current T list.

We associate two distance estimates to every node c, $d_x(c)$ and $d_y(c)$, representing the minimum possible distance we expect to find in the subtree at c, along the x- and y-dimensions respectively. Note that $\frac{\sqrt{d_x(c)^2+d_y(c)^2}}{320\sqrt{2}} + \gamma_{min}(c)$ then serves as a lower bound of the rank values in the subtree at node c. Both distance estimates are initialized to zero for the root of the tree. For the purpose of demonstration, say $d_x(c_p) = 4$ and $d_y(c_p) = 10$. Observe that the y-coordinate of c_{ref} is less than the y-coordinate of c_p. Given that the y-dimension is used for splitting nodes at c_p, all nodes in the right subtree of c_p have y-coordinate values greater than 90. Hence, the minimum y-distance we will see for nodes in the right subtree of c_p is $d_y(c_r) = |85 - 90| = 5$. A similar argument is not possible for the x-distance; however, we can still retain the estimate from the parent of c_r, effectively giving us $d_x(c_r) = d_x(c_p) = 4$. The lower bound for the subtree rooted at c_r is then $\frac{\sqrt{4^2+5^2}}{320\sqrt{2}} + 0.5 = 0.514$. For the left subtree, we can only retain the values from the parent, i.e. $d_x(c_l) = d_x(c_p) = 4$, $d_y(c_l) = d_y(c_p) = 10$ and the lower bound at c_l is $\frac{\sqrt{4^2+10^2}}{320\sqrt{2}} + 0.45 = 0.473$. The figure also shows where in the list L will c_r and c_l be inserted. Note that once L is updated, we can terminate the search since the first node in L (i.e. c_l) has a lower bound that is larger than the rank of the K^{th} ($K = 4$) node in the list T.

C. Performance improvements

1) Improving the grid 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 $\text{dist}_{norm}(c_s, c_t)$ (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 $\lambda = rank(T[K+1], c_s) - \text{dist}_{norm}(c_s, c_t)$ when computed with respect to c_t. Therefore,

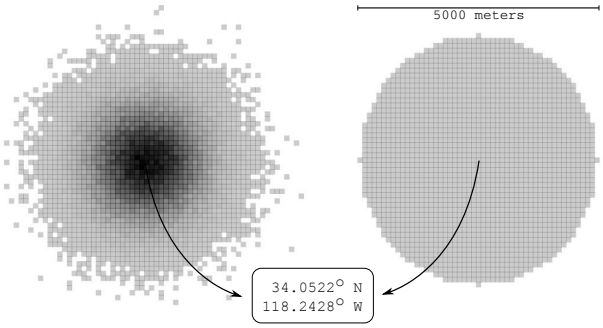

Fig. 2. Uniform-local (right) and gaussian (left) background knowledge. Darker cells imply higher probability.

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 λ, 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, λ should always be calculated using the last fully searched cell.

2) Reducing the search space for non-border cells: An improvement can also be achieved if the number of nodes in the kd-tree itself can be reduced. 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 region, the top-K set matches the top-K set of either one of the neighboring cells. For example, if set T_1 is the top-K set of cell $(1, 1)$ and T_2 is the top-K set of cell $(1, 2)$, then the top-K set of any cell $(1, y); 1 < y < 2$ is either of T_1 or T_2. The physical distance between two cells in our empirical evaluation is 100 meters, which is reasonably small to maintain the accuracy of this heuristic.

V. Empirical Evaluation

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). Default values of $\alpha = 0.8$ and $K = 10$ are used, and all distance computations are Euclidean. 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 are obtained on an Android emulator running a virtual device with a ARM Cortex-A8 processor (∼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. We consider three different forms for the adversary's prior probability distribution–(i) no knowledge–*uniform-global*: equal probability throughout the 320×320 cells, (ii) locality knowledge–*uniform-local*: equal probability in a circle of 25 cells radius, centered at (34.0522^o N, 118.2428^o W); zero everywhere else, and (iii) precise knowledge–*gaussian*: normally distributed probabilities; mean cell at (34.0522^o N, 118.2428^o W) and variance of 50 cells. Fig. 2 depicts the uniform-local and the gaussian knowledge distributions. We assume that the attacker's background knowledge is always correct, i.e. the user can never be at a cell where the attacker's prior probability is zero.

A. Runtime performance

Table I lists the average time to compute the interest set across the different regions for a given coarse grid. For example, a 10×10 coarse grid results in 100 regions (subgrids) of 32×32 cells. The execution time for each region 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 regions (a 5×5 coarse grid) 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 runtimes for the 10×10 coarse grid are a three fold improvement over the emulated values on the ARM device.

In addition to the overhead associated with the interest set computation, our architecture also incurs a communication overhead. This overhead appears in step 4 of the architecture, where the client has to obtain the locations and γ values of the matching POIs inside the broad 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 broad area, then a total of 12KB of data needs to be downloaded before computing the interest set. Assuming a 3G connection with 320 KB/s speed [16], this download will incur an additional 37.5 ms to the process.

B. Expected privacy and overhead

Fig. 3 shows the expected privacy achieved for different coarse grid sizes. The data points in each line correspond to $b = 80, 40, 20, 10, 8$ and 5 cells, from left to right. Recall that the expected inexact privacy is the expected number of cells that the adversary would visit before arriving at the user's actual location. We can transform the value to an area by multiplying with the area of a cell, 0.01 km^2 in this study. The expected exact privacy reflects the probability of *not* arriving at the user's location in one single attempt. We choose three different keywords—starbucks coffee (92 POIs), gas station

TABLE I

AVERAGE TIME (MILLISECONDS) TO COMPUTE INTEREST SET FOR DIFFERENT SIZES OF THE COARSE GRID ($b \times b$). TOP AND BOTTOM VALUES CORRESPOND TO THE ARM PROCESSOR AND THE ATOM PROCESSOR VIRTUAL DEVICES RESPECTIVELY.

search query	no. of POIs	computation time (ms) given a coarse grid					
		80×80	40×40	20×20	10×10	8×8	5×5
bus station	32	10.83 2.0	15.32 2.9	24.41 4.71	43.47 8.62	49.65 9.84	76.96 15.18
farmers market	50	24.99 4.03	31.93 5.4	45.69 8.12	74.04 13.62	92.25 17.32	140.88 26.78
police	84	35.12 5.82	46.72 7.93	71.25 12.41	128.58 22.93	179.54 31.9	396.82 70.38
starbucks coffee	92	33.73 5.5	43.51 7.21	63.26 10.79	117.24 20.57	156.18 27.49	478.98 84.84
grocery	95	34.49 5.89	44.11 7.84	64.43 12.02	119.55 23.03	178.24 34.48	386.67 75.34
restaurant italian	124	46.1 7.29	57.39 9.3	81.74 13.77	155.23 27.02	206.6 36.2	539.14 94.71
liquor store	125	50.29 7.54	64.12 9.88	92.82 14.8	180.66 29.42	277.67 45.38	831.41 135.69
bookstore	126	46.29 7.63	57.12 9.92	80.66 14.83	152.26 29.55	221.03 43.41	650.29 128.76
library	141	58.54 8.67	72.38 11.02	102.61 16.19	194.44 31.38	275.47 44.88	783.86 128.72
night club	149	61.27 8.84	72.56 10.9	96.27 15.35	172 29.59	266.14 46.25	644.26 115.82
clothing store	169	72.09 11.14	86.95 13.99	120.06 20.53	235.17 42.34	320.04 58.63	920.34 171.79
car rental	196	91.37 13.67	107.25 16.89	142.95 23.97	252.14 44.91	352.26 63.83	889.84 165.54
parking	281	136.63 19.55	160.78 23.91	202.45 33.26	363.43 62.34	504.01 87.48	1248.92 220.6
atm	297	131.66 18.62	148.88 21.75	190.53 29.72	339.13 57.46	491.26 85.47	1382.03 250.42
gas station	347	153.38 23.12	169.56 26.32	210.34 34.82	383.49 69.25	565.08 105.49	1580.96 308.41
pharmacy	369	173.49 24.45	195.14 28.08	247.32 36.97	446.25 70.33	649.94 104.13	1846.01 303.26
cafe	608	305.48 44.18	328.5 49.64	385.3 61.73	613.32 107.47	805.09 146.33	2097.42 407.17
bakery	834	444.79 61.7	457.99 66.73	525.32 80.89	803.75 137.3	1078.03 193.21	2776.04 539.39

(347 POIs) and bakery (834 POIs)—to evaluate the quantities in the case of low, medium and high density POIs.

It is reasonably accurate to say that the expected exact privacy for all three POI densities is much above levels of concern, greater than 90% in this case. This implies that it will be difficult for the adversary to accurately make a random "guess" about the user's location using the posterior distribution, even if precise information (gaussian knowledge adversary) on the whereabouts of the user is available to the adversary.

The expected privacy under inexact localization depends primarily on the extent of background knowledge. As expected, the uncertainty about the user's location is significantly less when the adversary has more precise knowledge. Larger values of b help improve the expected privacy to some extent. Note that, for lower values of b (larger sub-grids), the privacy level we observe (high or low) is primarily due to the prior knowledge of the adversary. Larger sub-grids will encompass most of the locality where the adversary's prior knowledge is concentrated. Since the convergence requirement is enforced, the expected privacy value from the posterior distribution will be the same as that from the prior distribution.

The expected interest set size shows some variations across different knowledge forms and POI densities. In general, smaller values of b results in more cells in a region; therefore, more number of top-K sets are merged to create the interest set. The denser the POI, the higher is the number of unique top-K sets. The set sizes are larger for the uniform-local and gaussian knowledge adversaries, since the user is highly likely to be located in central downtown, where the variations in the top-K POIs are also expected to be the most (due to the higher concentration of POIs).

The solid data points in the first three plots signify the case of $b = 10$ (output regions of the $\mathcal{R}$ function are 32×32 cells). Irrespective of the general trends, use of this value results in expected privacy levels of *at least* 2 km^2 (200 cells) and interest sets of around 30. An area of 2 km^2 is equivalent to around 1000 homes with $22,000$ ft^2 lots, which we consider to be a significantly large area for a privacy conscious user. The expected interest set size is larger than what is necessary, but may prove to be useful if the user does not find an acceptable choice in the top-10 results. Retrieval of detailed feature data for 30 POIs is also not considered expensive. Most current applications already retrieve more than that (Google Places allows retrieval of data on up to 60 POIs in a query).

VI. CONCLUSIONS

Privacy preserving techniques for local search have relied heavily on the availability of a trusted third party. In the absence of such third parties, the service architecture itself has to undergo significant modifications to facilitate private local search. In this paper, we propose a novel architecture to perform local search that does not require the sharing of location coordinates with any party. By leveraging the processing power available in current mobile devices, our

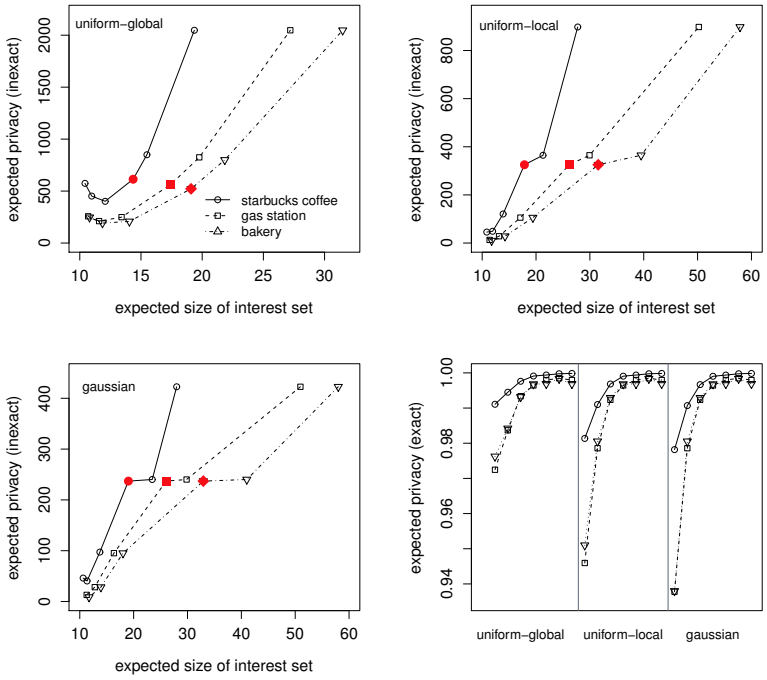

Fig. 3. Expected privacy and expected interest set size trade-off. Data points in each line correspond to $b = 80, 40, 20, 10, 8$ and 5 cells, from left to right.

architecture performs the location related computations within the client device. We demonstrate the feasibility of these computations by designing a fast kd-tree based top-K search algorithm, and enhance its performance by using multiple optimizations. We also propose a formal model to evaluate privacy under this architecture, and show that reasonable levels of location uncertainty can still be induced in even the strongest of adversaries. The algorithms are executed on real-world POI distributions, and timing experiments show that our architecture adds only sub-second delays to prevalent local search architectures. We believe that the results shown here will motivate further research into client centric privacy preservation techniques across other types of location-based applications.

REFERENCES

[1] 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.

[2] 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.

[3] 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*, 2008, pp. 237–246.

[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*, 2009, pp. 436–442.

[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, no. 12, pp. 1719–1733, 2007.

[6] 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.

[7] 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.

[8] 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*, 2010, pp. 325–339.

[9] 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*, 2011, pp. 247–262.

[10] 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*, 2010, pp. 115–118.

[11] 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*, 2012, pp. 617–627.

[12] A. R. Beresford and F. Stajano, "Mix Zones: User Privacy in Location-Aware Services," in *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, 2004, pp. 127–131.

[13] 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*, 2005, pp. 88–97.

[14] A. Khoshgozaran and C. Shahabi, "Blind Evaluation of Nearest Neighbor Queries Using Space Transformation to Preserve Location Privacy," in *Proceedings of the 10th International Conference on Advances in Spatial and Temporal Databases*, 2007, pp. 239–257.

[15] B. O'Clair, D. Egnor, and L. E. Greenfield, "Scoring local search results based on location prominence," *US Patent 8046371 B2*, 2011.

[16] M. Sullivan, "3G and 4G Wireless Speed Showdown: Which Networks Are Fastest?" *PC World*, April 2012. [Online]. Available: www.pcworld.com/article/253808-6