Location Privacy Techniques in Client-Server Architectures

Christian S. Jensen[1,2], Hua Lu[2], and Man Lung Yiu[2]

[1] Google Inc., Mountain View, CA 94043, USA
[2] Department of Computer Science, Aalborg University, Denmark
{csj,luhua,mly}@cs.aau.dk

Abstract. A typical location-based service returns nearby points of interest in response to a user location. As such services are becoming increasingly available and popular, location privacy emerges as an important issue. In a system that does not offer location privacy, users must disclose their exact locations in order to receive the desired services. We view location privacy as an enabling technology that may lead to increased use of location-based services.

In this chapter, we consider location privacy techniques that work in traditional client-server architectures without any trusted components other than the client's mobile device. Such techniques have important advantages. First, they are relatively easy to implement because they do not rely on any trusted third-party components. Second, they have potential for wide application, as the client-server architecture remains dominant for web services. Third, their effectiveness is independent of the distribution of other users, unlike the k-anonymity approach.

The chapter characterizes the privacy models assumed by existing techniques and categorizes these according to their approach. The techniques are then covered in turn according to their category. The first category of techniques enlarge the client's position into a region before it is sent to the server. Next, dummy-based techniques hide the user's true location among fake locations, called dummies. In progressive retrieval, candidate results are retrieved iteratively from the server, without disclosing the exact user location. Finally, transformation-based techniques employ cryptographic transformations so that the service provider is unable to decipher the exact user locations. We end by pointing out promising directions and open problems.

1 Introduction

The Internet is rapidly becoming mobile. An infrastructure is emerging that encompasses large numbers of users equipped with mobile terminals that posses geo-positioning capabilities (e.g., built-in GPS receivers) and data communication capabilities. Thus, location-based services (LBS) are increasingly becoming available. These return results relative to the users' locations. An example service returns the gas station nearest to the location of a user. Another example is a service that returns all restaurants within 2 km of the user's location.

To receive such services, the users must disclose their locations to the service provider. Users may be uncomfortable disclosing their exact locations to an untrusted service provider that may misuse the knowledge of the users' locations [1]. We view location privacy as an enabling technology for the diffusion of the mobile Internet and the proliferation of location-based services. By offering users the ability to choose different levels of location privacy, users are encouraged to use mobile services more often.

Some existing location privacy solutions assume the presence of a *centralized* third-party anonymizer that is aware of all users' locations. This trusted anonymizer serves as an intermediary in-between the users and the service provider. However, such an anonymizer may not always be practical, and it may itself present security, performance, and privacy problems. For example, the anonymizer represents a single-point-of-attack for hackers. Also, the anonymizer is prone to becoming a performance bottleneck because it may need to serve a large number of users.

In contrast, the techniques covered in this chapter assume a client-server architecture without any third-party anonymizer. We therefore call these *decentralized* solutions. The decentralized solutions are motivated by several considerations. First, the client-server architecture is widely used by today's location-based services. This popularity affords decentralized solutions wide applicability.

Second, a mobile terminal in a decentralized solution does not need to keep an anonymizer up to date with its location at all times; the terminal only issues queries to the server on demand. The anonymizer of a centralized solution needs to maintain up-to-date locations of all mobile terminals in order to perform cloaking for the small fraction of users that are issuing queries at any point in time.

Third, the setting of this chapter is based on the seemingly realistic assumptions that an adversary knows what the service provider knows, i.e., the identity of the user who issues a query and the parameters and result of the query. Specifically, we assume that users must register with the service provider to receive services; and we assume that users are not required to report their latest locations continuously.

In the next section, we provide an overview of decentralized solutions found in the literature.

2 Overview of Client-Server Solutions

The privacy models of existing solutions can be broadly classified into two types: identity privacy and location privacy.

The *identity privacy* model [2] assumes that (i) an untrusted party has access to a location database that records the exact location of each user in the population of users and (ii) that service users are anonymous. If a service user discloses her exact location to the untrusted party, that party may be able to retrieve the user's identity from the location database. In this setting, which this chapter does not consider, the location of a user is obfuscated in order to preserve the anonymity of the user.

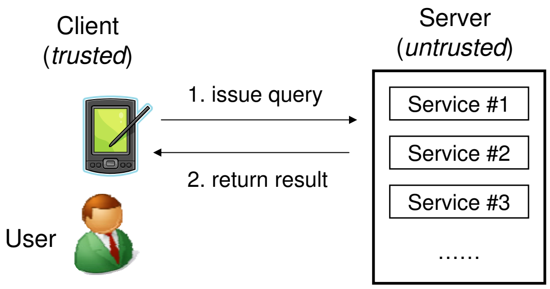


Fig. 1. Client-Server Architecture

This chapter is devoted to the *location privacy* model, which assumes that untrusted parties know the user's identity, but not the user's location. This model fits well with services where a user must log in before using the services. Examples include location-based services available in Google's Android Market[1]. Also, FireEagle[2] by Yahoo! enables users to share their locations with their friends, allowing them to specify the preciseness of the shared locations (e.g., exact location, city of the location, or undisclosed location).

Under the above model, we study privacy solutions that simply assume a *client-server* architecture and that apply to *snapshot* queries based on *the user's location*. In other words, we consider neither the privacy of continuous queries nor of a user's trajectory. Figure 1 illustrates the client-server architecture, in which the client is trusted, but the server (including its services) is not trusted. It does not rely on peer-to-peer communication among the clients, and nor does it employ a trusted third-party anonymizer.

Existing solutions for the location privacy model can be classified into four categories.

- *Query enlargement* techniques [3, 4, 5, 6, 7] (Section 3) enlarge the client's exact position into a region before sending it to the server.
- *Dummy-based* techniques [8, 9] (Section 4) generate dummies (i.e., fake locations) at the client and then send them together with the exact user location to the service provider, thus hiding the user location among the dummies.
- *Progressive retrieval* techniques [10, 11, 12] (Section 5) iteratively retrieve candidate results from the server, without disclosing the exact user location.
- *Transformation-based* techniques [13, 14] (Section 6) employ cryptographic transformation so that the service provider is unable to decipher the exact user locations, while providing the clients with decryption functionality so that they can derive the actual results.

Table 1 offers a summary of specific location privacy solutions that belong to the above categories. Six features are covered: (i) the nature of the domain space, (ii) the privacy measure, (iii) the types of queries supported, (iv) whether

[1] http://www.android.com
[2] http://fireeagle.yahoo.net

Table 1. Features of Various Location Privacy Techniques

Method	**Domain Space**	**Privacy Measure**	**Supported Queries**	**Exact Result**	**Accuracy Guarantee**	**Impl. Difficulty**
[3]	Euclidean	Area-based	Range	Yes	Yes	Medium
[4]	Euclidean	Area-based	Range, kNN	Yes	Yes	Medium
[5, 6]	Euclidean	Area-based	Range, kNN	Yes	Yes	Medium
[7]	Euclidean	Area-based	Proximity	No	No	Medium
[8]	Euclidean	Size-based	Range, kNN	Yes	Yes	Low
[9]	Euclidean	Size and Area	Range, kNN	Yes	Yes	Low
[10, 11]	Network	Size-based	1NN	Yes/No	Yes/No	Medium
[12]	Euclidean	Distance-based	kNN	Yes	Yes	Low
[12]g	Euclidean	Distance-based	kNN	No	Yes	Low
[13]	Euclidean	Full-domain	kNN	No	No	Medium
[14]	Euclidean	Full-domain	1NN	Yes	Yes	High

exact results can be retrieved, (v) whether result accuracy guarantees are given (for approximate results), and (vi) the difficulty of implementing the solution.

The domain space used by Duckham and Kulik [10, 11] is modeled by a graph that represents a road network. All the other work focus on the Euclidean space. No existing solution is applicable to both Euclidean space and network space simultaneously.

The privacy measure, i.e., the means of quantifying the privacy afforded a user, of the solutions can be classified into four categories. First, in the *area-based* measures [3, 4, 5, 6, 7], the privacy of the user is measured by the area (or a derivative of it) of the region that contains the user's location. Second, the *size-based* measures [8, 10, 11] simply express the privacy as the cardinality of a discrete set of locations that contains the user's location. The work of Lu et al. [9] employs a hybrid that builds on the size-based and area-based measures. Third, the *distance-based* privacy measures [12] capture the expected distance of the user's location from the adversary's estimate. Fourth, the *full-domain* privacy measures [13, 14] ensure that the adversary cannot learn any information on the user's location, as it is transformed into another space.

An interesting issue is to examine whether a particular privacy model is applicable to other solutions. Among the solutions covered, the full-domain measure is applicable only to the solutions in references [13, 14]. The distance-based measure is applicable to the solutions in references [3, 4, 5, 6, 8, 9, 10, 11]. It can also be noted that the area-based measures cannot be applied to the solutions in references [8, 10, 11] that use a discrete set of points, whereas the size-based measure is inapplicable to the solutions in references [3, 4, 5, 6] that use a single continuous region for cloaking.

The typical queries that underlie location-based services are the range query and the k-nearest neighbor query. Given a dataset P (of points of interest, or data points) and a query region W, the *range query* retrieves each object $o \in P$ such that o intersects with W. Given a set P and a query point q, the *k-nearest* *neighbor query* retrieves k objects from P such that their distances from q are minimized. it follows from Table 1 that some solutions support range queries only, some support k-nearest neighbor queries only, and some support both. It is worth noticing that the methods in references [10, 11, 14] support only the nearest neighbor query (i.e., the special case with $k = 1$). In addition, proximity based queries (e.g., finding those of my friends that are close to me) are supported [7].

We cover two aspects that relate to the quality of a query result: whether it either is or contains the exact result, and, if not, whether an accuracy guarantee is provided. We observe that most of the existing solutions guarantee that their results are supersets of the actual results, thus allowing the client to obtain the exact result. The solutions of Duckham and Kulik [10, 11] ensure that the exact result is returned only if the user agrees to reveal a sufficiently accurate obfuscation of her location. The table uses "Yes/No" to capture this conditional property. Otherwise, the solution does not guarantee the accuracy of the returned result (thus the corresponding "Yes/No"). Yiu et al. [12] propose a solution that offers exact results and thus accuracy guarantees. In addition, an extension that utilizes so-called granular search for improving performance returns approximate results with user-controlled accuracy guarantees. In the table, this extension is called [12]g. The work of Khoshgozaran and Shahabi [13] does not provide result accuracy guarantees, and it cannot support exact result retrieval.

The aspect concerns the difficulty of implementing and deploying the proposed solutions. The solutions in references [8, 9, 12] are easy to implement as they reuse existing location-based operations that can be assumed to be available in location based servers. The solutions in references [3, 4, 5, 6, 7, 10, 11] have medium implementation difficulty as they apply specialized geometric search algorithms. The method of Khoshgozaran and Shahabi [13] also has medium implementation difficulty because a Hilbert curve transformation function needs to be used by the client. The solution of Ghinita et al. [14] has high implementation difficulty as both the client and the server have to run a protocol for private information retrieval.

3 Query Enlargement Techniques

A straightforward way of protecting an exact user location in a service request is to replace the user location by a region that contains the location. We call the solutions that adopt this tack *Query Enlargement Techniques*. Unlike centralized cloaking solutions, the query enlargement techniques considered here do not require any trusted third-party component.

3.1 Cloaking Agent-Based Technique

Cheng et al. [3] assume a setting in which the data points are not the typical, static points of interest such as restaurants, but are the locations of other users. thus, user requests are intended to retrieve private data rather than public data, as do all other techniques covered in this chapter.

In this setting, the service quality may degrade when the spatial and temporal information sent to the service provider is at a coarse granularity. Motivated by this, Cheng et al. [3] proposed a framework for balancing the user location privacy and quality of service requested.

Architecture
The proposed architecture is illustrated in Figure 2. It encompasses of a crucial component, the *cloaking agent*. The cloaking agent is not necessarily a third-party component—it can also be implemented directly on the client side, i.e., on the user's device. For this reason, we cover this technique.

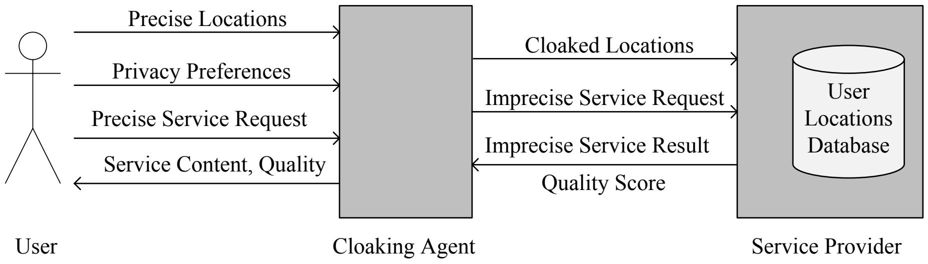


Fig. 2. Cloaking Agent-Based Architecture for Privacy and Service Quality Tradeoff

In particular, the cloaking agent receives precise locations and privacy preferences from a user, introduces uncertainty into the user's locations according to the privacy preferences, and reports the uncertain locations to the database at the service provider side.

When the user issues a service request with an exact location, the request is passed to the cloaking agent where it is translated into an imprecise service request with a cloaked location obtained according to the user's privacy preferences as known by the agent.

The imprecise service request is then sent to the service provider where it is processed using the uncertain user locations stored in its database, yielding an imprecise service result. The imprecise result, together with a score quantifying the service quality, is then sent back to the cloaking agent. The cloaking agent delivers the service result and the quality measurement to the user, who is allowed to adjust the privacy preferences based on the service and quality received.

Privacy Model
Cheng et al. [3] base the specification of location privacy preferences on a probabilistic location cloaking model. Assume that n users, namely $S_1, S_2, \ldots, S_n$, are registered in the system. Let $L_i(t)$ be the exact location of user S_i at time t. Instead of reporting $L_i(t)$, the user S_i reports a closed uncertainty region $U_i(t)$ to the service provider, such that $L_i(t)$ has a uniform probability distribution in $U_i(t)$.

A user is allowed to control the degree of location privacy in two ways. First, a user can specify the desired area of the uncertainty region, i.e., $Area(U_i(t))$. In general, the larger the value of $Area(U_i(t))$, the higher the location privacy. The idea is that it is more difficult for the adversary to determine the user's exact location $L_i(t)$ the larger the uncertainty region $U_i(t)$ becomes.

Second, a user can specify the desired coverage of sensitive regions. When the user is in a sensitive region, e.g., at a psychology clinic, she does not want to release the location information. However, if the user's uncertainty region happens to overlap with the clinic by a high percentage (e.g., 90%), it becomes easy for the adversary to guess that the user is at the clinic. To overcome this problem, a user may specify a coverage value based on the equation below that is not to be exceeded.

$$Coverage = \frac{Area(sensitive\ regions \cap U_i(t))}{Area(U_i(t))}$$

Query Processing

Cheng et al. focus on the processing of range queries. Based on their probabilistic location cloaking model, a range query from user S_i is translated by the cloaking agent into an imprecise location-based range query (ILRQ). An ILRQ issued at time t returns the set $\{(S_j, p_j) \mid j \neq i \wedge j \in [1, n]\}$, where $p_j > 0$ is the probability that S_j is located within $U_i(t)$ at time t. As mentioned, such a query concerns other users' locations, not public points of interest such as restaurants.

An ILRQ is processed by the service provider in three phases: (i) The pruning phase eliminates objects whose uncertainty regions do not overlap with the ILRQ. (ii) The transformation phase transforms an ILRQ into subqueries. For each possible location $(u, v) \in U_i(t)$ of S_i, a subquery is generated to find those unpruned objects whose uncertainty regions overlap with the circle centered at (u, v) and with radius r (specified in the original query), denoted as $C((u, v), r)$. (iii) The evaluation phase evaluates each subquery, by computing the actual probability that each remaining object satisfies the ILRQ. The probability of object S_j satisfying a subquery located at (u, v) is given as:

$$p_j(u, v) = \frac{Area(U_j(t) \cap C((u, v), r))}{Area(U_j(t))}$$

The results of all subqueries are combined as the answer to the original ILRQ.

From the location privacy preference specification above, it is easy to see that better user location privacy results from using a larger uncertainty region. However, simply increasing the uncertainty region inevitably hurts the service quality. Specifically, the use of larges uncertainty regions tends to retrieve more objects with lower probabilities. To enable trade-offs between privacy and service quality, a service quality metric is proposed.

Result Quality

Assume that an ILRQ from user S is partitioned into B subqueries that correspond to B locations among A_1 to A_B. Let the probability that S is located at A_k $(1 \leq k \leq B)$ be $p_k(S)$. The result of the subquery at A_k is R_k, while $R = \bigcup_{k=1}^{B} R_k$. The quality score of the ILRQ is defined as follows:

$$Query\ score = \sum_{k=1}^{B} p_k(S) \cdot \frac{|R_k|}{|R|}$$

The score varies between 0, the lowest quality, and 1, the highest quality. When a user S receives a query result and its corresponding score, she can adjust privacy preferences stored in the cloaking agent according to her expectation and the score value.

Benefits and Limitations

The proposed solution has two advantages. First, it allows flexibility on the architecture, as the cloaking agent can be part of the client or can be a separate third party. Second, it offers quantifies location privacy and service quality, which allows users to make trade-off according to their needs.

Nevertheless, the cloaking agent based solution also suffer from some disadvantages. First, it is assumed that the service provider knows all possible locations where a user can be. This is exploited in the query transformation and query quality score calculation. If there are many such locations, the query transformation, the query evaluation, and the quality score calculation can all be very expensive. And if there are few such locations, the location privacy is not well protected. Second, it may be difficult for a user to understand well the exact meaning of service quality scores, which therefore may reduce the utility of such scores.

3.2 Spatial Obfuscation Techniques

Ardagna et al. [4] propose a straightforward and intuitive way to express user location privacy preferences using obfuscated circles. Due to assumed measurement accuracy limitations, a user location is represented as a circular region $C((x_c, y_c), r_{meas})$ (i.e., centered at (x_c, y_c) and with radius r_{meas}). The possible user locations are assumed to be uniformly distributed within that region.

Measurement of Privacy and Accuracy

To support multiple location obfuscation techniques, an attribute λ is first introduced to represent a *relative privacy preference*, which is derived according to the following formula:

$$\lambda = \frac{max(r_{meas}, r_{min})^2}{r_{meas}^2} - 1$$

Here, r_{meas} represents measurement accuracy, i.e., the radius of a measured circular region modeling the user location; r_{min} is the minimum distance specified by a user to express her privacy preference [11]. For example, “1 mile” indicates that the user requires her location to be represented by a circular region with a radius of at least 1 mile. The term $max(r_{meas}, r_{min})$ is used because it is possible that $r_{min} < r_{meas}$ because the measurement accuracy may be unknown to the user.

When $r_{meas} \geq r_{min}$, $\lambda = 0$, indicating that the user privacy preference is already satisfied as the measured radius exceeds the preferred minimum distance. In this case, no location obfuscation is needed. When $r_{meas} < r_{min}$, $\lambda > 0$, reflecting the degree to which the location accuracy is to be degraded to protect the user according to the user's privacy preference. In this case, obfuscation is needed.

To measure the accuracy of an obfuscated region, a technology-independent metric, called *relevance*, is defined as a value $\mathcal{R} \in (0, 1]$. The relevance is 1 when the user location has the best accuracy, and its value is close to 0 when the user location is considered too inaccurate to be used by the service provider. The value $1 - \mathcal{R}$ is accordingly the *location privacy* offered by an obfuscated location.

The privacy management solution of Ardagna et al. embodies two crucial relevance values. The *initial relevance* ($\mathcal{R}_{Init}$) is the measure of the accuracy of a user location as obtained using some positioning technology. The *final relevance* ($\mathcal{R}_{Final}$) is the measure of the accuracy of the final obfuscated region that satisfies a relative privacy preference λ. let r_{opt} be the measurement radius corresponding to the best accuracy of a positioning technology. The initial and final relevance are calculated as follows:

$$\mathcal{R}_{Init} = \frac{r_{opt}^2}{r_{meas}^2} \qquad \mathcal{R}_{Final} = \frac{\mathcal{R}_{Init}}{\lambda + 1}$$

Obfuscation Operators

To derive $\mathcal{R}_{Final}$ from $\mathcal{R}_{Init}$, three basic obfuscation operators are defined on circular regions. First, the *Enlarge* operator (E) enlarges the radius of a region. Second, the *Shift* operator (S) shifts the center of a region. Third, the Reduce operator (R) reduces the radius of a region.

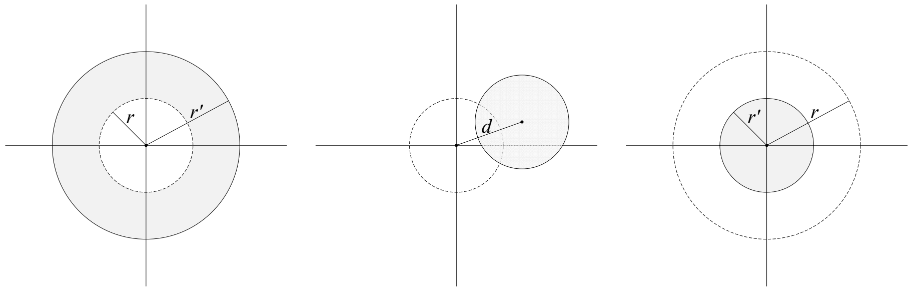
An example of E obfuscation operator is illustrated in Figure 3(a). Here the initial radius r is increased to $r' > r$. Let $\mathcal{R}$ and $\mathcal{R}'$ be the relevances before and after the operator, respectively. Then $\mathcal{R}'$ is derived from $\mathcal{R}$ as follows:

$$\mathcal{R}' = \frac{f_{r'}(x, y)}{f_r(x, y)} \cdot \mathcal{R} = \frac{r^2}{r'^2} \cdot \mathcal{R}$$

In the formula, $f_r(x, y)$ ($f_{r'}(x, y)$) is the joint probability density function (pdf) of an exact user location to be in the circular region indicated by r (r'). Note that $\mathcal{R}' < \mathcal{R}$ as $r' > r$. Therefore, $1 - \mathcal{R}' > 1 - \mathcal{R}$, which means that the location privacy is increased by the Enlarge operator.

An example of the S obfuscation operator is illustrated in Figure 3(b). Here the initial center is shifted by a vector $(\Delta x, \Delta y)$ of length d. The relevance of the result of applying the operator is derived as follows:

$$\begin{aligned} \mathcal{R}' &= P((x_u, y_u) \in C_{Init} \cap C_{Final}) \cdot P((x, y) \in C_{Init} \cap C_{Final}) \\ &= \frac{Area(C_{Init} \cap C_{Final})^2}{Area((x_c, y_c), r)^2} \cdot \mathcal{R} \end{aligned}$$



(a) Enlarge (b) Shift (c) Reduce

Fig. 3. Basic Obfuscation Operators

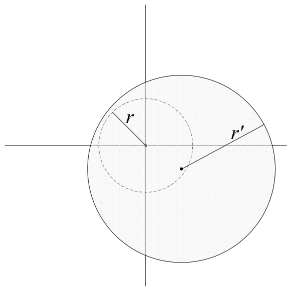


Fig. 4. Composite Obfuscation Operation: S followed by E

Here, $P((x_u, y_u) \in C_{Init} \cap C_{Final})$ is the probability that the exact user location belongs to the intersection of the two circular regions; $P((x, y) \in C_{Init} \cap C_{Final}$ is the probability that a random location selected from the whole obfuscated region is within the intersection. In addition, (x_c, y_c) represents the original center and r is the original radius.

Finally, an example of the R obfuscation operator is shown in Figure 3(c). Here the initial radius r is reduced to $r' < r$. The analysis of this case is symmetric to that of the E operator, and we obtain the following relationship between the relevance of the argument and the result:

$$\mathcal{R}' = \frac{r'^2}{r^2} \cdot \mathcal{R}$$

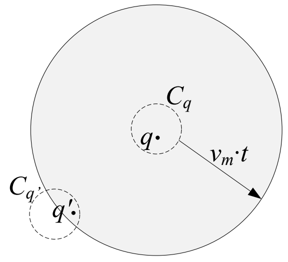
Composite obfuscation is achieved by combining two operators. As operators E and R are inverse to each other, there are four kinds of composite operators: E followed by S, S followed by E, R followed by S, and S followed by R. An example of S followed by E is shown in Figure 4.

3.3 The iPDA Solution

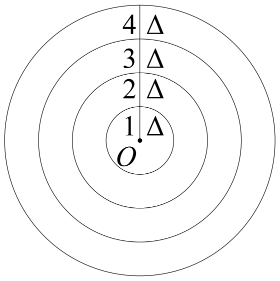
Xu et al. [5] propose a client-based solution that enables pr<u>i</u>vacy-<u>p</u>reserving location-based <u>d</u>ata <u>a</u>ccess, called iPDA. The basic idea behind iPDA is to transform a location-based query to a region-based query using an optimal location cloaking technique that is fully implemented on the client side. When a point-based query is transformed to a region-based query, the query result of the latter is a superset of that of the former. Xu et al. show that compared to any other shape with the same area, a circular enlarged region minimizes the size of the superset query result.

Mobility Analysis Attack

iPDA is designed to address the *mobility analysis attack*. Referring to Figure 5(a), a user first issues a query at location q, whose cloaking region is C_q. After a period



(a) Mobility Analysis Attack



(b) Rings for Locations

Fig. 5. iPDA Example

of time t, the user issues another query at location q' with a cloaking region $C_{q'}$. An adversary who knows the user's maximum speed v_m can infer that the user cannot be located within the white subregion of $C_{q'}$ when query q' is issued. Rather, the user must be within the intersection of $C_{q'}$ and the gray circular region expanded from C_q by the distance $v_m \cdot t$.

Privacy Model

iPDA is intended to generate cloaking regions for clients issuing queries repeatedly such that at any time a query is issued, the client's exact location uniformly distributed within the cloaking region. A general movement pattern is assumed to be known to both the client and the server. To carry out a numerical analysis, the plane of movement is divided into a set of rings with a common center O, as shown in Figure 5(b). Each ring, except the innermost one, which is actually a circular region, is of a sufficiently small width Δ.

Assume that at the time of the previous query, the cloaking region was centered at O and had radius $r = K \cdot \Delta$, and $R = L \cdot \Delta$ denotes the longest distance a user can travel between two queries. Both K and L are integers. Two probabilities can be defined based on the set of rings indicated by r and R.

The probability that the user's new location is in the i'th ring is captured by $U(i) = \int_{(i-1)\cdot\Delta}^{i\cdot\Delta} u(x)dx$, where $u(x)$ is the density function for the probability that the distance between the new user location and O is x. The conditional probability that the user's new location is in the i'th ring given that the center of the new cloaking region is in the j'th ring is denoted by $Q(i|j)$. Two expressions of $Q(i|j)$ are derived for the two cases $j < K$ and $j \geq K$ [5]. As a result, the location privacy against a mobility analysis attack is indicated by the value of $Q(i|j)$.

To enable a trade-off between query result accuracy and the cost of communication between client and server, queries are allowed to be blocked. A blocked query is not sent to the server; the result of the previous query is instead reused to derive the new result. If a blocked query comes from a ring $i \leq K$, the old result is a superset of the new result. This means that the query accuracy is not reduced while the communication cost is avoided. If a blocked query comes from a ring $i > K$, the old result is no longer a superset of the new result. This means that the communication cost is saved at the expense of the query accuracy. This also leads to a set of linear equations:

$$\sum_{j=max\{1,i-K+1\}}^{min\{L-K+1,i+K-1\}} \left(\frac{Q(i|j)}{U(i)} \cdot v_j\right) \leq 1 \quad i = 1, 2, \ldots, L$$

$$v_j = \sum_m (P(j|m) \cdot U(m)) \geq 0 \quad j = 1, 2, \ldots, L-K+1$$

With these constraints, two linear programming techniques are developed to either maximize the query accuracy or minimize the query communication cost [5].

Query Processing

iPDA transforms a traditional kNN query to a query that requires the k nearest neighbors of a circle's perimeter Ω and the interior of Ω. Thus, a proposal for the server-side processing of k circular range nearest neighbor queries is also part of iPDA. A general heuristic is to access objects (or index nodes when the objects are indexed) in ascending order of their minimum distances to Ω. For disk-resident data, two pruning heuristics are proposed based on the distance and topology between Ω and index nodes.

Benefits and Limitations

Although the cloaking technique in iPDA is claimed to be optimal, discretization is needed before iPAD can be implemented using some numerical method. This reduces the attractiveness of iPAD. Nevertheless, iPDA addresses the mobility analysis attack and allows a user to issue consecutive queries while enjoying location privacy. Further, iPAD offers users the flexibility of either maximizing query accuracy or minimizing query communication cost when the original query is cloaked. As a last remark, a demonstration system is available that implements iPDA in a practical setting of a GPS-enabled Pocket PC and spatial database supported servers [6].

3.4 Server-Side Processing of Enlarged Queries

As range queries and k nearest neighbor queries are fundamental, they may be expected to be supported by location-based service servers. After applying query enlargement, an original location-based query is converted into a region-based query.

Region-Based Range Query

The starting point is a range query applied to an uncloaked user location. This query retrieves all data points within the range, which might be a circle or rectangle centered at the user location. The region-based range query occurs when the user location is cloaked by a region, again perhaps a circle or a rectangle. The region-based range query then returns all data points within the original range, but with any point in the cloaking region being the query point.

The resulting query can still be viewed as a range query, so the query is relatively straightforward to compute. For example, if the cloaking region is a (axis-aligned) rectangle and the query range is a (axis-aligned) rectangle, the region-based range query becomes rectangular (axis-aligned) range query.

Region-Based kNN Queries

The ensuing discussion assumes the nearest neighbor query (i.e., the special case with $k = 1$), but it is easily generalized to the k nearest neighbor query for any positive integer k.

An original point nearest neighbor query is transformed to a *range nearest neighbor query* (*RNN*) [15]. Given a range W, the *RNN* query returns the union of the nearest neighbors for each point in W, i.e., $RNN(W) = \{NN(p) \mid p \in W\}$. This definition implies that any object within W belongs to $RNN(W)$, as it is its own nearest neighbor. For any object o outside W, it holds [15] that:

$$o \in RNN(W) \Leftrightarrow \exists p \in \text{Border(W)}(o = NN(p))$$

Therefore, it suffices to compute $RNN(W)$ in two steps: (i) performing a range query with the region W, and (ii) computing the nearest neighbor of any point p located at the border of W.

In case the range W is a rectangle, the second operation can be further reduced into four *line segment based nearest neighbor* (LNN) queries [15] that can be evaluated by a continuous nearest neighbor (CNN) algorithm [16] that retrieves the nearest static points for any point on a given line segment. Hu and Lee [15] propose solutions for LNN queries on both memory-resident and disk-resident data.

An alternative server-side algorithm for processing the range nearest neighbor query is also available [17].

3.5 Location Privacy in Proximity-Based Services

Proximity-based services differ from services that rely on range and kNN queries. For example, a "friend finder" is a typical proximity-based service in which a user

A is alerted if a friend is within a specified distance δ_A of the user's current location. In this section, we introduce a privacy technique that employs location enlargement.

Setting

Mascetti et al. [7] propose techniques that offer location privacy in proximity based services. A service provideris assumed that receives (enlarged) user location updates, maintains (enlarged) user locations, and processes service requests from users by finding their nearby friends. The spatial domain of interest is abstracted as a *granularity* that consists of a number of non-overlapping *granules*. The granules are heterogenous in size and shape, and each is identified by an index.

For a given user, both the service provider and buddies (other users in the system) can be adversaries. Therefore, each user A needs to specify two granularities: G_A^{SP} defines the minimum location privacy requirement for the service provider, and G_A^U defines the requirement for the buddies. In either granularity, each granule is a minimum uncertain region.

***SP-Filtering* Protocol**

Within the setting described above, the authors propose three privacy-aware communication protocols. We consider the *SP-Filtering* protocol, which is the only one that does not require peer-to-peer communication.

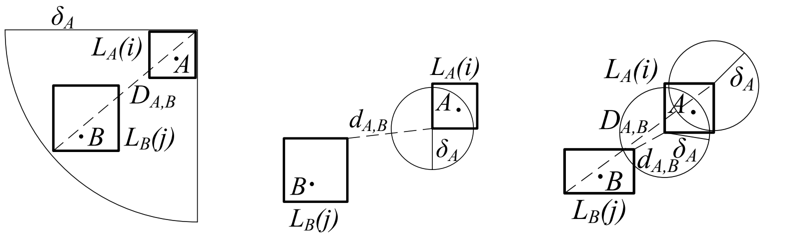
With this protocol, user A sends to a generalized location to the service provider when she updates her current location. Let A be located in the granule $G_A^{SP}(i)$, which is known by the service provider. The user's generalized location $L_A(i)$ is defined as follows:

$$L_A(i) = \bigcup_{i' \in \mathbb{N} \mid G_A^U(i') \cap G_A^{SP}(i) \neq \emptyset} G_A^U(i'),$$

which is the union of those G_A^U granules that intersects with $G_A^{SP}(i)$. Any other user B updates her location $L_B(j)$ similarly, where j is the index of the G_B^{SP} granule in which B is located.

Given two users A and B with generalized locations $L_A(i)$ and $L_B(j)$, the service provider determines the minimum distance ($d_{A,B}$) and maximum distance ($D_{A,B}$) between them. Based on $d_{A,B}$ and $D_{A,B}$, the service provider determines whether B is in the proximity of A. The three cases illustrated in Figure 6 occur. Here, where the actual locations are represented as dots while the generalized locations are rectangles.

If $D_{A,B} < \delta_A$, as shown in Figure 6(a), B must be in the proximity of A regardless of the exact locations of A and B are within their generalized location rectangles. In this case, the service provider informs A that "B is within proximity." If $d_{A,B} > \delta_A$, as shown in Figure 6(b), B has no chance to be within the proximity of A. In the last case, $d_{A,B} \leq \delta_A \leq D_{A,B}$, as shown in Figure 6(c). Here, it is uncertain whether B is within proximity of B. Thus, the service provider informs A that "B is possibly within proximity."



(a) $D_{A,B} < \delta_A$ (b) $d_{A,B} > \delta_A$ (c) $d_{A,B} \leq \delta_A \leq D_{A,B}$

Fig. 6. Example for SP-Filtering Protocol

Limitations and Extensions

The simple SP-Filtering protocol faces a dilemma. On the one hand, the granularity G_A^{SP} should be coarse for the purpose of location privacy. On the other hand, a coarse G_A^{SP} lowers the service accuracy, as the service provider is unable to determine whether buddy B is within proximity of A. Consequently, Mascetti et al. [7] propose two additional protocols, which, however, rely on peer-to-peer communication to derive a better result when an uncertain answer is returned.

4 Dummy-Based Techniques

We can regard the query enlargement techniques as *continuous*, in the sense that a user's location is enlarged into a closed region. In contrast, dummy-based techniques are *discrete* because the user's location is covered by multiple dummy locations, or dummies for short. All dummies, together with the user's location, are sent to the service provider, which is then unable to identify the user's real location. If a total of k locations are sent in a request, the service provider is then unable to identify the user's real location with a probability higher than $1/k$. This way, dummy-based techniques protect a user's location privacy.

We review two dummy-based location privacy protection techniques: a basic dummy-based technique [8] that offers limited control of the generation of dummies; and an enhanced dummy-based technique [9] that offers such controls and also takes into account the notion of privacy area to offer location privacy.

4.1 Basic Dummy-Based Technique

Format of a Message

In the basic dummy-based approach [8], a service request has the following format:

$$S = \langle u, L_1, L_2, \ldots, L_k \rangle$$

Here, u is a user identifier and $\langle L_1, L_2, \ldots, L_k \rangle$ is a set of locations consisting of the real user location and generated dummies. When the service provider receives a service request S, it processes the request for each location L_i $(1 \leq i \leq k)$ according to the service type required, and then returns an answer R as follows:

$$R = \langle (L_1, D_1), (L_2, D_2), \ldots, (L_k, D_k) \rangle$$

Here, D_i is the services content for location L_i. When the user receives R, the service content D_r for the real query location L_r ($1 \leq r \leq k$) is selected as the result. Note that the real value of r is known only to the user.

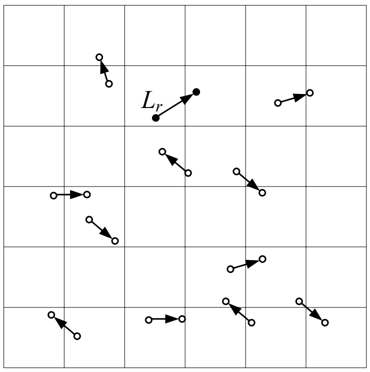
Dummy Generation Algorithms

Given a region P, the service provider knows the (dummy) location cardinality in P as users send in requests. The location cardinality in P can vary from one time t to the next $t + 1$. If the cardinality difference for consecutive time points is too large, it is possible that the dummies move irregularly when compared with the user's actual movement. This, according to [8], causes the risk that adversaries may be able to identify user's real locations.

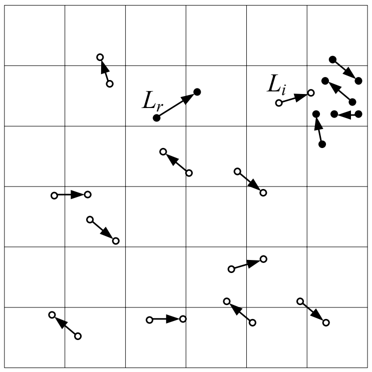
Thus, two dummy generation algorithms are proposed for users that issue service requests at each time step as they move [8]. In both algorithms, the first batch of dummy locations for a user are generated at random. The two algorithms then differ in how they generate subsequent dummy locations.

The *Moving in a Neighborhood* (MN) algorithm generates the next location of a dummy solely based on the current location of the dummy. Figure 7(a) illustrates the MN algorithm. It contains 11 dummies whose locations are represented by circles. The real user location L_r is drawn as a small dot. The arrow between each pair of locations indicates the movement during a time step. The new location of a dummy only depends on its previous location.

The *Moving in a Limited Neighborhood* (MLN) differs from the MN algorithm in that it takes into account the density of the region in which a newly generated dummy resides. Assuming that a user device is capable of obtaining the positions of other users, MLN regenerates a dummy if it finds that the dummy's generated location is in a region with too many users. Figure 7(b) gives an example of the MLN algorithm. During a time step, dummy L_i basically gets its new location



(a) MN Algorithm



(b) MLN Algorithm

Fig. 7. The MN and MLN Algorithms for Dummy Generation

according to the MN algorithm. However, the new location falls in a small range (a grid cell here) where there are four real locations of other users. Let the desired threshold of the region density be set at four. The MLN algorithm regenerates the new location for L_i at random until it is not in a dense region.

A drawback of the MLN algorithm is its reliance on the assumption that a user can obtain the real locations of nearby users. We remark that such a capability can itself cause location privacy issues.

Communication Cost Savings

In addition to dummy generation, communication cost reduction is also addressed [8]. As described, an original request is represented as $\langle u, (x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)\rangle$, where (x_r, y_r) $(1 \leq r \leq k)$ is the exact user location. Such a request ensures that the service provider is unable to identify the real user location with a probability higher than $1/k$. The request contains $8k$ bytes of location data if a single coordinate value takes 4 bytes (e.g., a float type value).

If all the x coordinates are put together followed by all the y coordinates in the request message, i.e., $\langle u, (x_1, x_2, \ldots, x_k), (y_1, y_2, \ldots, y_k)\rangle$, the message can be viewed as representing k^2 locations, namely the locations (x_i, y_j) $(1 \leq i, j \leq k)$. Thus, to ensure a probability of $1/k$, a request needs only use $2\sqrt{k}$ coordinates, or $8\sqrt{k}$ bytes of location data.

4.2 Enhanced Dummy-Based Technique

Although the MN and MLN algorithms are able to generate dummies that ensure that the service provider is unable to identify the real user location with a probability higher than $1/k$, they do not take into account the notion of the distances between the (dummy) locations. In particular, the region covered by all (dummy) locations, called the *privacy region*, is of importance because the area of that region indicates the difficulty for an adversary of tracking down the user. Neither MN nor MLN are capable of controlling the area of the privacy region.

Privacy Requirement

Motivated by this, Lu et al. [9] propose a privacy-area aware, dummy-based technique (PAD) for location privacy protection. PAD allows a user to specify privacy preference as $\langle k, s\rangle$, where k is the total number of locations in a request sent to the service provider and s is the area of the privacy region containing these k locations. Such a preference states that the service provider must be unable to identify the real user location with a probability higher than $1/k$ and must be unable to position the user in a region with area smaller than s.

Simply increasing the number of dummies in a request does not necessarily produce a larger privacy region. Therefore, new algorithms are needed to satisfy the privacy preference $\langle k, s\rangle$. Thus, two privacy-area aware dummy generation algorithms are proposed [9].

Circle-Based Dummy Generation

The circle-based dummy generation constrains all (dummy) locations, including the real user location, to a circle centered at position pos' with radius r.

Figure 8(a) shows an example where $k = 9$ and pos is the real user location. Each pair of clock-wise consecutive positions and pos' determines an angle θ. All positions are distributed in such a way that all θ values are equivalent.

The area $\widetilde{s}$of the hull of all positions is the sum of the areas of k triangles:

$$\widetilde{s} = \sum_{i=1}^{k} \frac{1}{2} \cdot r_i \cdot r_{i+1} \cdot sin\theta = \frac{1}{2} \cdot \sum_{i=1}^{k} r_i \cdot r_{i+1} \cdot sin\frac{2\pi}{k},$$

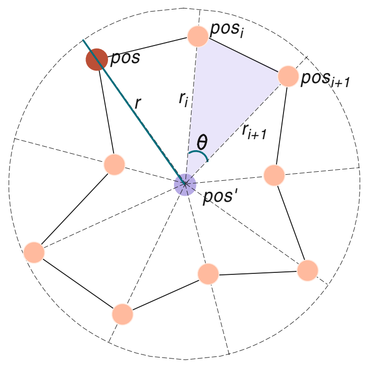
where r_i is $dist(pos_i, pos')$. As there are only k locations in addition to pos', $pos_{k+1} = pos_1$ and $r_{k+1} = r_1$. Note that the hull is not necessarily convex and that $\widetilde{s} \le \widehat{s}$, where $\widehat{s}$ is the area of the corresponding convex hull. Assuming that all positions have identical distance to pos', the hull determined by them must be convex. Thus, taking into account the privacy area requirement s, the following holds:

$$\widetilde{s} = \widehat{s} = \frac{1}{2} \cdot k \cdot r_i^2 \cdot sin\frac{2\pi}{k} = s$$

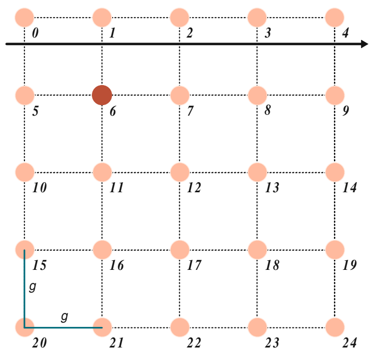
Solving this produces an upper bound $r = \sqrt{(2 \cdot s)/(k \cdot sin\frac{2\pi}{k})}$. Let $r_{min} = \rho \cdot r$, where $0 < \rho \le 1$. The following holds:

$$\widehat{s} \ge \widetilde{s} \ge \frac{1}{2} \cdot k \cdot r_{min}^2 \cdot sin\frac{2\pi}{k} = \frac{1}{2} \cdot k \cdot (\rho \cdot r)^2 \cdot sin\frac{2\pi}{k} = \rho^2 \cdot s$$

This indicates a lower bound of the privacy area of the k positions, i.e., $\widehat{s} \ge \rho^2 \cdot s$. Thus, the virtual center pos' is determined at random such that $dist(pos, pos') \in [\rho \cdot r, r]$. As a result, by carefully choosing ρ, a guarantee can be gained on the privacy area of the location privacy query generated based on a virtual circle. For example, if we choose $\rho = \sqrt{3}/2$, we can ensure that the resulting privacy area is not smaller than three quarters of s.



(a) Circle Based



(b) Grid Based

Fig. 8. Privacy-Area Aware Dummy Generation Examples

Grid-Based Dummy Generation

The circle-based algorithm aims to approximate the privacy area requirement s. In contrast, the grid-based algorithm always generates dummies whose privacy area is no smaller than the required s. The grid-based dummy generation works as follows. A (virtual) uniform, square grid is created, such that (i) it has k vertices, (ii) its area is equal to s, and (iii) the user position *pos* is one of the k vertices. The $k-1$ other vertices are dummy locations, to be sent to the server together with the user position *pos*.

Figure 8(b) shows an example of the grid-based dummy generation, where $k = 25$ and 24 dummies are generated. All locations, including the real user location, are indexed in row-major order, and vertex 6 is the real user location. The side length of a grid cell is $g = \sqrt{s}/(\sqrt{k}-1)$. The coordinates of all dummies are determined by their indexes relative to the real user location.

When dummies are generated based on the virtual grid, the upstream communication cost can be further reduced compared to [8]. Instead of sending all coordinate values, it is possible to send the grid configuration only in the request. The configuration of a uniform grid is given by 3 parts: the top-left corner location (8 bytes), the side length of each square grid cell (4 bytes), and the number of grid cells in the horizontal/vertical direction (1 byte). Therefore, the location information in a request with the grid configuration consumes 13 bytes.

5 Progressive Retrieval Techniques

The solutions in this section are called as *progressive retrieval techniques* because they progressively retrieve potential result objects from the server until it is guaranteed that the exact result can be found or the user chooses to terminate the search with an approximate result.

5.1 Graph-Based Obfuscation

Duckham and Kulik [10, 11] study location privacy in the context of a graph, which is employed to model a road network. We first introduce their graph model and then elaborate on the procedure for processing a query.

Graph Model and Equivalence Class

In the graph model, each (possible) location l_i refers to a graph vertex. An edge between two locations l_i and l_j has an associated weight $w(l_i, l_j)$. The *network distance* $dist_N(l_i, l_j)$ between any two locations l_i and l_j is defined as the length of the shortest path between the two, i.e., the sum of the weights along the shortest path.

It is assumed that the data points (and the query object) are located at vertices. Figure 9a depicts a graph with seven locations $(l_1, l_2, \cdots, l_7)$. The numbers next to the edges indicate their weights. The dataset P contains two data points p_1 and p_2, which are located at l_7 and l_1, respectively. The network distance $dist_N(l_2, l_4)$, for example, is computed as $1 + 4 = 5$.

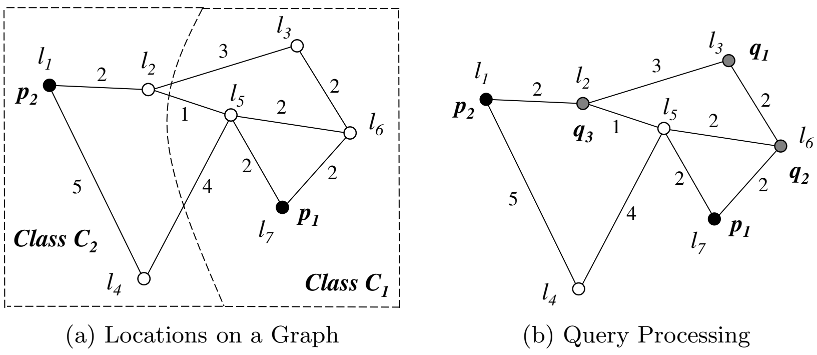


Fig. 9. Obfuscation in a Graph Setting

By using the set P, the sets of locations can be partitioned into disjoint sets of *equivalence classes* such that each location in the same equivalence class has the same data point as its nearest neighbor. In Figure 9a, the locations l_3, l_5, l_6, and l_7 belong the same equivalence class (i.e., class C_1) because each of them takes p_1 as its nearest neighbor. The other locations belong to class C_2.

Obfuscation Set

Observe that the server maintains both the graph and the dataset P, whereas the user only knows the graph and her exact location q. Instead of submitting q to the server, the user needs to specify an *obfuscation set* Q, which is a set of graph vertices. The set Q is said to be *accurate* if it contains q, and it is said to be *imprecise* if it has more than one vertex [10]. The default setting is to choose a set Q that is both accurate and imprecise. Nevertheless, the user is allowed to choose a set Q that is inaccurate or precise. The consequence of using an inaccurate Q is that the actual query result is not guaranteed to be found. Intuitively, the user enjoys a high level of privacy when Q has a high cardinality or the user's exact location q is far from all the members of Q. However, it remains an open question how to combine both the accuracy and preciseness aspects into a unified, quantitative notion of privacy.

Negotiation Protocol for Query Processing

The user issues a nearest neighbor query by sending the obfuscation set Q to the server and following a *negotiation protocol.* If all vertices of Q belong to the same equivalence class (say, the class C_i), the server returns the data point of C_i as the result. Otherwise, the server negotiates with the user for a more precise obfuscation set Q' that is a proper subset of Q. If the user agrees to provide such a set Q', then thee protocol is applied recursively. If not, the server determines the largest equivalence class (say, the class C_j) that overlaps Q and returns the data points of C_j as the result.

We proceed to consider the query example in Figure 9b, with the obfuscation set $Q = \{q_1, q_2, q_3\}$. Since Q intersects with more than one equivalence class (i.e., classes C_1 and C_2), the server asks whether the user can provide a more precise obfuscation set Q'.

If the user prefers not to provide Q', the server checks whether the majority of the vertices of Q belong to the class C_1 or C_2. Since the majority of vertices of Q (i.e., q_1 and q_2) belong to class C_1, the server returns the point p_1 as the result.

In case the user accepts to provide a more precise set Q', say, $Q' = \{q_1, q_2\}$, the negotiation protocol is executed on Q' recursively. In this example, all vertices of Q' belong to the class C_1, so the server returns p_1 as the result.

Negotiation Strategies

Duckham and Kulik [10] also suggest strategies for the client to automate the negotiation process on the user's behalf. For example, their O-strategy reveals the equivalence class that covers the user upon the first negotiation; their C-strategy iteratively discards border locations from the obfuscation set; and their L-strategy provides the server with an inaccurate but precise location as the obfuscation set. In addition to these basic strategies, some advanced negotiation strategies are also discussed. Experimental results demonstrate that the O-strategy is able to achieve the best privacy protection [10].

Benefits and Limitations

As a remark, the above work is the first to study location privacy in the setting of a graph model. The negotiation protocol is a novel approach that enables the user to interactively control the trade-off between location privacy and query efficiency. From the viewpoint of user-friendliness, a user wishes to specify her desired privacy value without understanding the negotiation protocol and participating in negotiations. An open issue is to design a fully automatic negotiation policy for choosing the initial obfuscation set and the negotiation strategy.

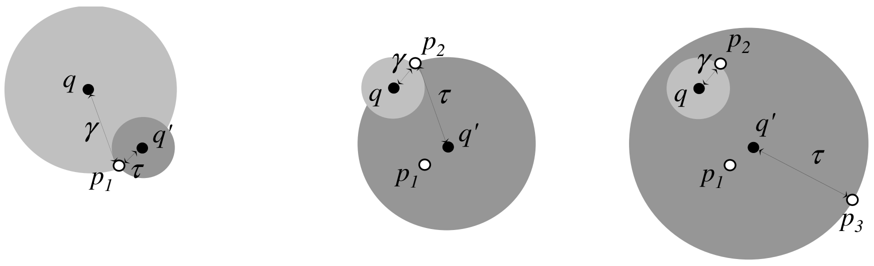
The negotiation protocol has medium difficulty of implementation as it requires the server to compute the equivalence classes (of the dataset P) that intersect the user's obfuscation set Q (see Figure 9).

5.2 SpaceTwist

Yiu et al. [12] propose a client-based algorithm, called *SpaceTwist*, for retrieving the user's k nearest neighbors from the server without revealing the user's exact location q. In the following, we first describe the running steps of SpaceTwist, then examine its privacy model. At the end, we study the trade-offs among the privacy, performance, and result accuracy of SpaceTwist.

Query Execution of SpaceTwist

The server takes as input a "fake" location q' that differs from q. The location q' can be generated at the client side if the user specifies her exact location q and the distance $dist(q', q)$ between q' and q. For instance, the user sets $dist(q', q) =$ 500 m if she wants to obtain privacy at the level of a city block. In fact, the ability to vary $dist(q', q)$ enables a trade-off between location privacy and query efficiency. A location q' being far from q offers high privacy, but also leads to high query cost.



(a) First Point (b) Second Point (b) Third Point

Fig. 10. Query Processing in SpaceTwist

The algorithm then requests the server to retrieve data points in the ascending order of their distances from q'. This operation is known as incremental nearest neighbor retrieval [18], and it has been studied extensively in the literature. In order to reduce the number of communication packets, multiple points retrieved consecutively on the server are shipped to the client in a single packet. Let β be the number of points that can fit into a packet.

During execution, the algorithm maintains a result set W and two variables τ and γ. The variable τ represents the largest distance between any retrieved point and q' so far. The variable γ denotes the distance between q and its k k nearest neighbor, with respect to the set of points retrieved so far. Initially, W is the empty set, $\tau = 0$, and $\gamma = \infty$. Whenever a point p_i is retrieved, τ is updated to $dist(q', p_i)$. In case q is closer to p_i than some point in W, the algorithm updates both the set W and γ to reflect the best k nearest neighbors found so far. The algorithm guarantees that the actual k nearest neighbors are available to the user when the condition $\gamma + dist(q', q) \leq \tau$ is satisfied. When this condition is met, the algorithm terminates.

We proceed to illustrate the running steps of the SpaceTwist algorithm using the example in Figure 10. Assume that we have $k = 1$ and $\beta = 1$. After retrieving point p_1 (see Figure 10a), the best result is set to p_1. Both τ (dark gray circle) and γ (light gray circle) are updated. When point p_2 is retrieved (see Figure 10b), τ is updated. As q is closer to p_2 than the previous result (i.e., p_1), the best result becomes p_2 and γ is updated. Next, point p_3 is retrieved (see Figure 10c) and τ increases. Since $\gamma + dist(q', q) \leq \tau$ (i.e., the dark gray circle contains the light gray circle), the algorithm terminates the search on the server and returns p_2 as the nearest neighbor of q.

Privacy Model

The privacy study of the SpaceTwist algorithm [12] assumes that the adversary knows: (i) the point q' and the value k, (ii) the set of retrieved points from the server, and (iii) the termination condition of SpaceTwist. The goal of the adversary is to utilize the above information for determining whether a location q_c can be a *possible* user location. Note that q_c is not necessarily the same as the *actual* user location q.

Let m be the number of packets received by the client, and let their points (in their retrieval order) be $p_1, p_2, \cdots, p_{m\beta}$. It is shown that a possible user location q_c must satisfy both of the following inequalities [12]:

$$dist(q_c, q') + \min_{1 \le i \le (m-1)\beta}^{k} dist(q_c, p_i) > dist(q', p_{(m-1)\beta})$$

$$dist(q_c, q') + \min_{1 \le i \le m\beta}^{k} dist(q_c, p_i) \le dist(q', p_{m\beta}),$$

where the term $\min^k_{1 \le i \le m\beta} dist(q_c, p_i)$ represents the distance between q_c and the k^{th} nearest neighbor, with respect to the first $m \cdot \beta$ points retrieved.

The inferred privacy region Ψ is then defined as the set of all such possible locations q_c. Assume that both the point q' and the value k are fixed. It is worth noticing that the use of any location q_c in Ψ causes the SpaceTwist algorithm to retrieve the exact same sequence of points $(p_1, p_2, \cdots, p_{m\beta})$ as does the actual user location q. Thus, the adversary cannot observe the difference of q from the other points of Ψ based on the behavior of SpaceTwist.

The *privacy value* that quantifies the privacy obtained is then defined as the average distance between q and any point in Ψ:

$$\Upsilon(q, \Psi) = \frac{\int_{z \in \Psi} dist(z, q)\, dz}{\int_{z \in \Psi} dz}$$

Although the region Ψ can be inferred by both the user and the adversary, only the user can derive the privacy value $\Upsilon(q, \Psi)$ (which requires the knowledge of

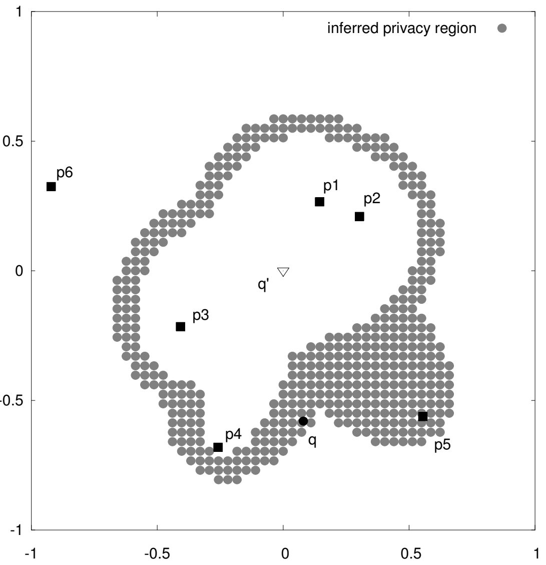


Fig. 11. Inferred Privacy Region

q). Figure 11 shows an example where the algorithm terminates after retrieving 6 data points from the server. The retrieved points $p_1, p_2, \cdots, p_6$ are labeled by their retrieval order. The nearest neighbor of q is the point p_4. The inferred privacy region corresponds to the gray region, which is an irregular ring that contains q.

Trade-offs among Location Privacy, Query Performance, and Query Accuracy

Yiu et al. [12] study a relaxed notion of the k nearest neighbors query. Given a query point q, a distance threshold ϵ, and a dataset P, the ϵ-relaxed k nearest neighbors query accepts a k-sized set W as the result if the maximum distance between q and W is upper-bounded by the sum of ϵ and the distance between q and the actual k nearest neighbor in P. The motivation of this approximate query is that a user (e.g., due to limited communication bandwidth) may be willing to accept a result that is not too far from her location q, if the cost can be reduced significantly.

A technique is then presented that computes the ϵ-relaxed k nearest neighbors query [12]. It employs a virtual grid structure to prune unnecessary points by utilizing the flexibility of the ϵ-relaxed k nearest neighbors query. It is shown that, the above technique guarantees the accuracies of the query results (within the ϵ bound), while saving communication cost (m) and improving the privacy value (Υ).

6 Transformation-Based Techniques

This section introduces *transformation-based* techniques that transform the original location-based query problem (e.g., range search, k nearest neighbors queries) into a search problem in another space.

6.1 Hilbert Curve Transformation

Transformation

Khoshgozaran ans Shahabi [13] propose to evaluate the k nearest neighbor query at an untrusted server by transforming all data points and the query point into one-dimensional numbers. This approach achieves complete privacy and constant query communication cost. However, it does not guarantee the accuracy of a query result.

The transformation function is implemented by a Hilbert curve function. The instance of the function being used is defined by an encryption key $\mathcal{EK}$ that consists of 5 values: a translation offset (X_t, Y_t), a rotation angle θ, a curve order O, and a scaling factor F. Let $\mathcal{H}(\mathcal{EK}, p_i)$ denote the Hilbert value of the data point p_i. An analysis suggests that there exists an exponential number of possible encryption keys and that it is infeasible for an adversary to infer the exact encryption key being used [13]. Without knowing the encryption key $\mathcal{EK}$, it is infeasible for the adversary to recover a data point p_i from its Hilbert value $\mathcal{H}(\mathcal{EK}, p_i)$.

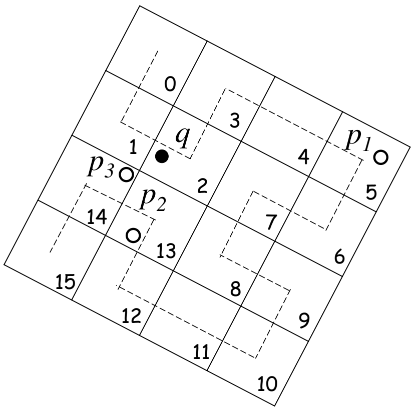


Fig. 12. Example of Hilbert Curve Transformation

Figure 12 depicts three data points p_1, p_2, and p_3 in the plane. The Hilbert curve is shown as a dashed curve, and each (square) cell is labeled with its Hilbert value. For example, the point p_1 is mapped to the value $\mathcal{H}(\mathcal{EK}, p_1) = 5$. Similarly, the values $\mathcal{H}(\mathcal{EK}, p_2) = 13$ and $\mathcal{H}(\mathcal{EK}, p_3) = 14$ are computed. These Hilbert values are then uploaded to the server. Observe that the server stores only three values (5, 13, 14), not the original data points.

Query Processing

At query time, the user applies the same encryption key to transform her location q into its Hilbert value $\mathcal{H}(\mathcal{EK}, q)$. Then the user requests the server to return k Hilbert values that are closest to the query Hilbert value. Next, the user applies the inverse transformation to obtain the result points from the retrieved Hilbert values.

We proceed to demonstrate an example of processing the $(k = 1)$ nearest neighbor query in Figure 12. At the client side, the query point q is mapped to the Hilbert value $\mathcal{H}(\mathcal{EK}, q) = 2$. Then the client asks the server to return the Hilbert value closest to 2. The server returns the value 5, the client decodes this back to the data point p_1, and it returns p_1 as the query result.

It is worth noticing that the retrieved point (say, p_1) is not necessarily the actual nearest neighbor of q (i.e., p_3). In fact, the solution does not provide any guarantee on the retrieved point.

6.2 Private Information Retrieval

Ghinita et al. [14] propose a solution for private nearest neighbor search by applying a *computationally private information retrieval protocol*. The core idea is that the client can efficiently test whether a large number is quadratic residue (QR) or quadratic non-residue (QNR); however, the adversary (e.g., the service provider) cannot efficiently do so. Under modulo arithmetic, QR and QNR are analogous to a bit value 0 and 1, respectively.

In the pre-processing phase, the Voronoi cell $V(p_i)$ of each data point p_i is computed. Then the domain space is partitioned into a regular grid of $G \times G$ cells, and each grid cell c_j stores an m-bitmap to represent the data points whose Voronoi cells intersect the spatial region of c_j.

At query time, the client first locates the cell c_q that contains the user location q. It then computes a sequence of G large numbers such that the number corresponding the column of c_q is a QNR and the others are QR. The server then performs modulo arithmetic on the above sequence of numbers by using the bit values in the grid. Essentially, the server obtains a sequence of G numbers, corresponding to the content of cells at the same column as c_q. The client then picks the number in the retrieved sequence that is in the same row as c_q. If that number is QNR, the original bit value in that cell is 0; otherwise, it is 1. This procedure is repeated for each of the m bit positions in order to obtain the complete content (of data points) stored in c_q.

This solution always returns the actual query results, and it is proven to be computationally secure (i.e., achieving perfect privacy) [14]. The solution is specially designed for the nearest neighbor query; its extension to the k nearest neighbors query has not been studied. A drawback of the solution is that it incurs high execution time on the server-side, limiting the server from being concurrently used by massive amount users. Empirical studies show that it takes 20 seconds to process the exact nearest neighbor query using a single-CPU server [14].

7 Promising Directions and Open Problems

Location privacy solutions for the client-server architecture are of high interest in the sense that this architecture is simple and widely deployed. While good advances have been made, several general directions for future research exist.

First, it is possible to extend techniques covered in this chapter to 2-dimensional space with obstacles, which are regions where service users cannot be located. Such obstacles usually are known to the server or adversaries, who can make use of this information to increase their success of guessing the real user location. For example, the dummy generation algorithms should not use any locations within those obstacles as dummies. For query enlargement techniques, the intersection between an enlarged query and the obstacles should be minimized, or taken into account, to ensure location privacy. One possibility would be to build on the formal model of bettini et al. [2] for obfuscation-based techniques.

Second, it is relevant to extend the techniques covered in this chapter to simultaneously support Euclidean space (with obstacles) and road network space. For instance, it is relevant to consider the extension of some works [10, 11] to also apply to Euclidean space, and the extensions of other works [3, 4, 5, 6, 8, 9, 12, 13, 14] to also apply to road network space.

Third, while most techniques covered in this chapter consider only snapshot queries, it is of interest to offer more proposals to support also continuous queries that can be issued by mobile users. For query enlargement techniques, it is of interest to find more efficient server-side query processing approaches for enlarged queries.

Fourth, transformation-based techniques covered [13, 14] are promising in terms of the privacy they offer. However, they are not readily deployable in existing settings. A recent approach [19], though not designed for location privacy applications, devises a keyed transformation function such that its output domain remains to be the 2-dimensional space. This way, existing spatial indexes and query processing techniques are leveraged for the query processing. Unfortunately, this approach is applicable to range queries only, and it assumes that all the clients share the same key value. It remains an open problem to extend this approach to become a solution to the query location privacy problem.

Last but not least, there is a need for a location privacy solution that can be accepted by both the clients and the service provider. In practice, the service provider may be interested in certain aggregate statistics over the whole population of users (e.g., "finding the region with the highest density of users at 5 p.m."), rather than tracking the clients' exact locations. It is challenging to develop a solution that support the above aggregate query and yet satisfies the users' privacy requirements.

Acknowledgments

This work was conducted in part while Christian S. Jensen was on sabbatical at Google Inc. from his position at Aalborg University.

References

1. Voelcker, J.: Stalked by Satellite: An Alarming Rise in GPS-enabled Harassment. IEEE Spectrum 47(7), 15–16 (2006)
2. Bettini, C., Mascetti, S., Wang, X.S., Jajodia, S.: Anonymity in Location-Based Services: Towards a General Framework. In: MDM, pp. 69–76 (2007)
3. Cheng, R., Zhang, Y., Bertino, E., Prabhakar, S.: Preserving User Location Privacy in Mobile Data Management Infrastructures. In: Privacy Enhancing Technology Workshop, pp. 393–412 (2006)
4. Ardagna, C.A., Cremonini, M., Damiani, E., di Vimercati, S.D.C., Samarati, P.: Location Privacy Protection Through Obfuscation-Based Techniques. In: DBSec, pp. 47–60 (2007)
5. Xu, J., Du, J., Tang, X., Hu, H.: Privacy-Preserving Location-based Queries in Mobile Environments. Technical Report, Hong Kong Baptist University (2006)
6. Du, J., Xu, J., Tang, X., Hu, H.: iPDA: Supporting Privacy-Preserving Location-Based Mobile Services. In: MDM, pp. 212–214 (2007)
7. Mascetti, S., Bettini, C., Freni, D., Wang, X.S., Jajodia, S.: Privacy-aware proximity based services. In: MDM, pp. 1140–1143 (2009)
8. Kido, H., Yanagisawa, Y., Satoh, T.: An Anonymous Communication Technique using Dummies for Location-based Services. In: IEEE International Conference on Pervasive Services (ICPS), pp. 88–97 (2005)

9. Lu, H., Jensen, C.S., Yiu, M.L.: PAD: Privacy-Area Aware, Dummy-Based Location Privacy in Mobile Services. In: MobiDE, pp. 16–23 (2008)
10. Duckham, M., Kulik, L.: Simulation of Obfuscation and Negotiation for Location Privacy. In: Cohn, A.G., Mark, D.M. (eds.) COSIT 2005. LNCS, vol. 3693, pp. 31–48. Springer, Heidelberg (2005)
11. Duckham, M., Kulik, L.: A Formal Model of Obfuscation and Negotiation for Location Privacy. In: Gellersen, H.-W., Want, R., Schmidt, A. (eds.) PERVASIVE 2005. LNCS, vol. 3468, pp. 152–170. Springer, Heidelberg (2005)
12. Yiu, M.L., Jensen, C.S., Huang, X., Lu, H.: SpaceTwist: Managing the Trade-Offs Among Location Privacy, Query Performance, and Query Accuracy in Mobile Services. In: ICDE, pp. 366–375 (2008)
13. Khoshgozaran, A., Shahabi, C.: Blind Evaluation of Nearest Neighbor Queries Using Space Transformation to Preserve Location Privacy. In: Papadias, D., Zhang, D., Kollios, G. (eds.) SSTD 2007. LNCS, vol. 4605, pp. 239–257. Springer, Heidelberg (2007)
14. Ghinita, G., Kalnis, P., Khoshgozaran, A., Shahabi, C., Tan, K.L.: Private Queries in Location Based Services: Anonymizers are not Necessary. In: SIGMOD, pp. 121–132 (2008)
15. Hu, H., Lee, D.L.: Range Nearest-Neighbor Query. IEEE TKDE 18(1), 78–91 (2006)
16. Tao, Y., Papadias, D., Shen, Q.: Continuous Nearest Neighbor Search. In: VLDB, pp. 287–298 (2002)
17. Mokbel, M.F., Chow, C.Y., Aref, W.G.: The New Casper: Query Processing for Location Services without Compromising Privacy. In: VLDB, pp. 763–774 (2006)
18. Hjaltason, G.R., Samet, H.: Distance Browsing in Spatial Databases. TODS 24(2), 265–318 (1999)
19. Yiu, M.L., Ghinita, G., Jensen, C.S., Kalnis, P.: Outsourcing Search Services on Private Spatial Data. In: ICDE, pp. 1140–1143 (2009)