# Preserving User Location Privacy in Mobile Data Management Infrastructures

Reynold Cheng[1], Yu Zhang[2], Elisa Bertino[2], and Sunil Prabhakar[2]

[1] The Hong Kong Polytechnic University,Hung Hom, Kowloon, Hong Kong
csckcheng@comp.polyu.edu.hk
[2] Purdue University,West Lafayette, IN 47907-1398, USA
{zhangyu,bertino,sunil}@cs.purdue.edu

**Abstract.** Location-based services, such as finding the nearest gas station, require users to supply their location information. However, a user's location can be tracked without her consent or knowledge. Lowering the spatial and temporal resolution of location data sent to the server has been proposed as a solution. Although this technique is effective in protecting privacy, it may be overkill and the quality of desired services can be severely affected. In this paper, we suggest a framework where uncertainty can be controlled to provide high quality and privacy-preserving services, and investigate how such a framework can be realized in the GPS and cellular network systems. Based on this framework, we suggest a data model to augment uncertainty to location data, and propose imprecise queries that hide the location of the query issuer and yields probabilistic results. We investigate the evaluation and quality aspects for a range query. We also provide novel methods to protect our solutions against trajectory-tracing. Experiments are conducted to examine the effectiveness of our approaches.

## 1 Introduction

Positioning technologies like the Global Positioning Systems (GPS), GSM, RF-ID and the Wireless LAN have undergone rapid developments in recent years [1, 2]. These new technologies allow locations of users to be determined accurately, and enable a new class of applications known as Location-Based Services (LBS). An important LBS is the E-911 application mandated by the U.S. (correspondingly E-112 in Europe), which requires cell phone companies to provide an accurate location (i.e., within a few hundred feet) of a cell phone user who calls for emergency help [2]. In another application, a user may want to know the waiting time for a table in a restaurant close to her. A user may also wish to be notified when her friend is located within her walking distance. All these applications require an extensive use of location data [3].

Although LBS applications hold the promise of safety, convenience, and new business opportunities, the ability to locate users and items accurately also raises a new concern – intrusion of *location privacy*. According to [4], location privacy is "the ability to prevent other parties from learning one's current or past location".

Using locationing technologies, a service provider can track the whereabouts of a user and discover her personal habits. These pieces of sensitive information can be sold to third parties without the user's consent or knowledge. It is often feared that government agencies can monitor the behavior of individuals, or know the places they have visited. Preventing location privacy from being invaded is thus of utmost importance.

Recently several solutions for location privacy protection have been proposed. Some researchers suggest the use of "policies", in which the service provider is required to state explicitly how user's location information can be used [5–7]. In another proposal, a user "cloaks" her information before sending it to the LBS, by providing her location at a lower resolution in terms of time and space [2, 4]. In other words, rather than giving a precise location and time instant, a larger region covered in a time frame is reported. This solution, known as *location cloaking*, provides the user with more flexibility in controlling her information.

By reducing the granularity of spatial and temporal information, location cloaking allows a user's privacy to be better protected. Unfortunately, this scheme can also reduce the quality of service provided by the LBS. This is simply because the LBS does not have the most accurate information to provide the best service. Consider a remote cab service that allows a subscriber to call for a cab nearby. If the subscriber reports her precise location, the service provider can find her the closest cab, and can tell the cab driver how to reach the customer. However, if only a vague location is given, it may take more time for a cab to reach the customer. Thus, by adjusting the accuracy of the location information sent to the service provider, a tradeoff can be achieved between: (1) privacy of the user, and (2) quality of a service requested. To the best of our knowledge, the interaction of these factors has only been briefly mentioned in [8, 9]. In this paper, we propose a framework to study this problem more extensively. We also describe how this framework can be realized in commonly-used systems like the GPS and cellular network systems.

We then focus on data modeling and query evaluation issues in this system. We propose an intuitive model for representing cloaked location data, and present a metric for measuring the privacy of location cloaking. Moreover, we study the evaluation of a particular query called the *Location-based Range Query* (LRQ), where a user issuing an LRQ is notified of any object of interest within a fixed distance from her current location. This kind of query is commonly used in location-based applications [10]. We propose an "imprecise" version of the Location-based Range Query, namely ILRQ, which processes cloaked location data. Since the location of the query issuer is also inexact, the query result is "imprecise". That is to say, there is no single, definite answer to the query. To capture the answer imprecision, ILRQ provides probabilistic guarantees to the query answers that indicates the degree of confidence about these answers. For example, an answer for the ILRQ is $\{(S_1, 0.4), (S_2, 0.8)\}$, which means that users $S_1$ and $S_2$ have probabilities of 0.4 and 0.8 respectively of satisfying the query. We will present algorithms for computing probabilistic answers for ILRQ, based on computational geometry techniques.

Another important use of "probabilistic answers" is to quantify the ambiguity of a query answer due to the inexactness of cloaked location data. We define metrics for evaluating the quality of a service, allowing a user to decide whether she should adjust the granularity of her cloaked location information in order to obtain a better service.

We also address the issues of inference attacks, where future locations can be inferred based on tracing the movement in the past. We present two approaches, namely *patching* and *delaying*, in order to prevent the user's location from being deduced, thereby reducing the impact of this kind of threats.

The rest of this paper is organized as follows. We first describe related works in Section 2. We propose a framework to capture data uncertainty, privacy and quality of service in Section 3. In Section 4, we formally present the data cloaking model. Section 5 presents an algorithm for evaluating an ILRQ, and Section 6 describes service quality metrics for this query. Experimental results are presented in Section 7. We conclude the paper in Section 8. Appendix A discusses how our framework can be used in real systems, and Appendix B investigates a type of inference attacks to our system and their corresponding solutions.

## 2 Related Works

Data privacy has been a subject of active research in recent years. Pfitzmann et al.[11] explain various terms related to privacy, such as "linkability" and "anonymity". The notion of $k$-anonymity, first proposed in [12], has been widely studied. The main idea of $k$-anonymity is to make a data attribute indistinguishable with $k-1$ other values. Recent work in $k$-anonymity include [13], which presents efficient query algorithms for a $k$-anonymity model; and [14], which uses $k$-anonymity for privacy and copyright protection.

The notion of $k$-anonymity has also been used in a number of works that study location privacy [2, 4, 15]. How $k$-anonymity is defined depends on the particular type of LBS application. According to [4], LBS applications can be non-anonymous, pseudonymous and anonymous. A non-anonymous LBS needs a user's location information and her true identity; a pseudonymous LBS needs a user's pseudonym but not her real identity; an anonymous LBS does not require a user's true identity.

For an anonymous LBS, Gruteser et al. define $k$-anonymity in the context of location cloaking [2], which is the method that we will study in this paper. In location cloaking, a middleware is proposed to transform each tuple $(x, y, t)$ (i.e., location $(x, y)$ at time $t$) to $([x_1, x_2], [y_1, y_2], [t_1, t_2])$ where $([x_1, x_2], [y_1, y_2])$ is the rectangular area within which $(x, y)$ is found, between the time interval $[t_1, t_2]$. The middleware then sends the transformed tuple to the system. Here the $k$-anonymity requirement is that the time interval $[t_1, t_2]$ there must be at least $k$ users in the same spatial vicinity $([x_1, x_2], [y_1, y_2])$. Notice that this location cloaking technique is not limited to anonymous applications [2]; it can also be applied to location data in order to enhance the privacy of non-anonymous and

pseudonymous LBS applications. In [16], a personalized version of $k$-anonymity is proposed.

Although $k$-anonymity is a simple metric and can be applied to different LBS applications, it has several problems. First, the scheme may not be used if there are fewer than $k$ users in the system. Secondly, even if there are more than $k$ users, they may span in a large area over an extended time period, in which case the cloaked location can be very large and cause a severe degradation of service quality, due to the large amount of ambiguity in the cloaked location. In this paper, we suggest the size of a cloaked location to be controlled by the user's policy, which can be independent of the number of users inside the uncertainty region. This can also avoid the cloaked location from being too large, which can adversely affect the service quality.

In [9], Atallah et al. studied the idea of perturbing the location of a query issuer for protecting location privacy, and its effect on the accuracy of nearest-neighbor queries. Here we investigate the provision of probabilistic guarantees for range queries.

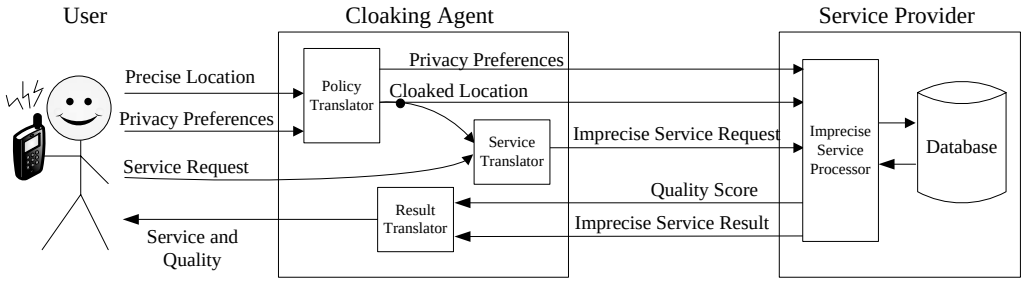


**Fig. 1.** Managing Privacy and Service Quality with the Cloaking Agent.

## 3 A Framework for Capturing Location Privacy and Service Quality

In this section, we will describe a system model that connects privacy, cloaked information and service quality. This model forms the basis for subsequent discussions. Figure 1 illustrates this system model. Its main idea is to allow the user to specify her location, service request and privacy requirements to the **cloaking agent**, which then produces the cloaked location (i.e., a larger region that contains the user's true location) and an "imprecise" service request. On receiving these pieces of information, the service provider processes the request and sends back the service and feedback to the user. The cloaking agent can either be implemented in the user's device, or provided by a third-party system.

In Figure 1 we see that a user can first specify her privacy preferences through a privacy language. This language, that we are planning to develop, allows a subject to specify her privacy preferences with respect to:

- Locations – a user may specify that when being near to a given object, cloaking is required, and the accuracy requirements. Locations can be logical or physical; logical position may identify a specific spatial entity (like the "Purdue Hospital") or a set of spatial entities (like "Any Hospital").
- Other users and service providers – a user may also specify that her presence be made known (or hidden) to specific users and service providers. For example, a physician while being in the hospital may require that her position be notified only to the head nurse and only to emergency services.

The user's privacy preferences are then passed to the **policy translator** inside the cloaking agent. The policy translator produces a cloaked location based on the precise location of the user as well as her privacy requirements. For instance, if the user's requirement is "generate a cloaked location that covers five buildings when I am in Area $X$", the policy translator produces the corresponding cloaked location when it detects the user is in Area $X$. The policy translator also forwards to the service provider the user's privacy preference concerning other users and service providers if needs.

Based on the cloaked location and the service request, the service translator produces an "imprecise" service request that processes cloaked data. For example, the LRQ is a service request from the user, and the service translator transforms the LRQ to ILRQ, an imprecise service request that processes cloaked location data. Both the cloaked location and the imprecise service request are then shipped to the **imprecise service processor**, which stores the cloaked location in a spatial-temporal database and processes the service request. Since location values are imprecise, the service processor produces a "probabilistic service result" i.e., answers are augmented with probability to indicate the confidence of their presence [17]. For example, the result of ILRQ contains user names together with their probabilities. In addition, a score indicating the quality of the service is generated. We will revisit the query and quality evaluation details in Sections 5 and 6.

Both the probabilistic service result and the quality score can be transferred directly to the user, or optionally to the **result translator** inside the cloaking agent. The main purpose of the result translator is to hide the technical details of the probabilistic service result (e.g., probability, quality scores), and converts the answers to a higher-level form that casual users can understand. For an ILRQ, the translator can choose to return only the names for which there is a high confidence (e.g., $p_j > 0.8$) and not return any probability value. It can also describe to the user the quality as `LOW`, `MEDIUM` and `HIGH` for quality score ranges between $(0, 0.2], (0.2, 0.8], (0.8, 1]$ respectively, instead of requiring the user to interpret the numerical values. Based on the recommendation from the cloaking agent, the user can then decide if the degree of privacy should be reduced.

In Appendix A we will describe how this framework may be deployed in practical systems like the GPS and cellular network systems. Let us now focus

on data modeling, query evaluation, quality and privacy protection issues for this system.

## 4 Cloaking and Privacy

In this section, we present a formal model of location cloaking, based on which location privacy is defined.

### 4.1 Location Cloaking Model

As mentioned before, a cloaked location data is essentially a region that contains the user's true location [2, 4]. To define this notion formally, assume that the system has $n$ users with names $S_1, S_2, \ldots, S_n$. Suppose that the location of each user $S_i$ at time $t$ is $L_i(t)$, then location cloaking can be defined as follows.

**Definition 1. Location Cloaking:** *At time $t$, a user $S_i$ reports to the service provider a closed region called uncertainty region, denoted $U_i(t)$, such that $L_i(t)$ has an even chance of being located inside $U_i(t)$.*

In other words, the probability density function (pdf) of the user's location within the uncertainty region is $\frac{1}{\text{Area}(U_i(t))}$. If the user moves outside the uncertainty region given earlier, she has to report to the service provider a new uncertainty region. A user may also send an uncertainty region if she wants to issue a query that requires her current location information (e.g., a location-based range query).

Our solution does not limit the geometric shape of the uncertainty region. The only requirement is that the uncertainty region is closed. The rationale is that the evaluation of probabilities and quality scores needs integration over the uncertainty regions, as illustrated in subsequent sections. The user can define the geometric shape required through the privacy language discussed in Section 3. For example, if the uncertainty region is a circle of radius $r$, the cloaking agent can select the center of the circle randomly with distance less than $r$ from the user's actual location.

We remark that the user assumes the service provider only knows her position accurate to the size of the uncertainty region she has provided. This may not be necessarily true, since the service provider can use various methods to reduce the uncertainty about the cloaked location. We investigate this issue in Appendix B. Furthermore, the service provider may be able to infer a higher pdf to some areas in the cloaked region if it has some information about it, instead of assuming a uniform distribution within $U_i(t)$. For now, we assume the service provider holds the same view of the cloaked location data as that of the user.

### 4.2 Measuring Privacy of Cloaking

By "injecting" different amount of spatial uncertainty to her location, cloaking provides a simple way for a user to control the release of her private information

to untrusted parties. The degree of privacy can be measured in two ways: (i) size of uncertainty region and (ii) coverage of sensitive area.

**1. Size of uncertainty region.** By providing a larger uncertainty region, the spatial resolution of a location is reduced, making the user's location more difficult to be guessed. The size of the uncertainty region can thus be used to reflect the degree of privacy: the larger the region size, the more the privacy.

**2. Coverage of sensitive region.** The second means of quantifying privacy depends on the location of the user. To see this, assume the size of the uncertainty region is fixed. Suppose the user is inside a hospital (which she does not want people to know about this), and her uncertainty region has a fraction of 90% overlap with the hospital. One can easily guess she is in the hospital. On the other hand, if the user is shopping in a mall, she may not be very concerned even if her location is known.

From this example, we can see that whether the user's located in a "sensitive region" (e.g., hospital, nightclub) affects the degree of privacy. Based on this observation, we define the "coverage" of sensitive region for user $S_i$ as follows:

$$\text{Coverage} = \frac{\text{Area}(\text{sensitive regions of } S_i \cap U_i(t))}{\text{Area}(U_i(t))} \tag{1}$$

In general, the higher the coverage, the lower the privacy. In the previous example, the coverage is 90%, and thus the user can be easily guessed that she is in the hospital. Thus the uncertainty region should be enlarged in order to assure that the user's location cannot be easily associated with a sensitive region.

It is also worth mention that the definition of sensitive region is user-specific. For example, while for a physician a hospital may not be a sensitive region, the same cannot be said about a patient. The system described in Section 3 should allow the users to specify what places are sensitive to them.

Although cloaking lessens the threat to location privacy, it can affect the *quality* of service provided. In particular, since the service provider does not receive accurate location information, it may not be able to generate a precise answer, leading to a lower service quality. In the next section, we develop algorithms to quantify the "imprecision" of query answers through the use of probability values. Section 6 further illustrates how the query answer quality can be evaluated based on the probabilistic answers.

## 5 Evaluation of Imprecise Queries

We now study query evaluation over cloaked location data. We focus on the *location-based range query* (LRQ), which is a range query with the range depending on the position of the query issuer. We propose an "imprecise" form of this query for evaluating cloaked data, and techniques for evaluating it.

### 5.1 Precise and Imprecise Queries

Suppose the current time instant is $t_c$. A user of the system, $S_i$ (where $i \in [1, n]$), issues a snapshot query on the uncertainty regions $U_j(t_c)$ of objects $S_j$ (with

$j \neq i$ and $j \in [1, n]$). For ease of presentation, let us use $S$ to refer to $S_i$. We denote her location $L(t_c)$ and uncertainty region $U(t_c)$ as $L$ and $U$ respectively. Moreover, $U_j(t_c)$ is represented as $U_j$ unless stated otherwise. Given a circle $C$ with $L$ and $r$ as its center and radius respectively, we can define a location-based range query as follows.

**Definition 2.** *A* **Location-based Range Query (LRQ)** *returns* $\{S_j | j \neq i \wedge j \in [1, n]\}$*, where user* $S_j$ *is located within* $C$*, at the current time instant.*

Essentially, LRQ asks the question: "who is within $r$ units from me?". As an example, Figure 2(a) shows that $S$ issued an LRQ (a range query with $L$ as the center and $r$ as the radius), and obtains $S_4$ as the only answer. Notice that the query range of the LRQ is defined to be a circle for the purpose of easier explanation; our methods can be generalized to deal with range queries with any geometric shape. Moreover, the system needs to know the current location of $S_i$, and it also knows the positions of the objects being queried. Location privacy of the database is not protected.

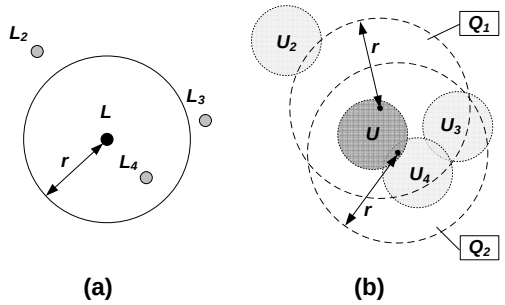


**Fig. 2.** Location-based Range Query: (a) exact locations, and (b) cloaked locations.

Location cloaking can alleviate the threat to location privacy. Instead of supplying exact locations, users only supply their cloaked locations in order to achieve higher anonymity, or lower entropy [18]. How should a query be defined and executed over this kind of data? To address this question, we term the version of LRQ that employs cloaked location data as the **Imprecise Location-based Range Query**. The word "imprecise" comes from the fact that both the locations of the query issuer and those of objects being queried are ambiguous. [3] This query is formally defined as follows:

[3] Although we assume objects being queried are moving objects, static objects like buildings can also be queried. These objects have no ambiguity in locations. However, our queries can still be applied by modeling the uncertainty regions of these objects as points.

**Definition 3.** *An* **Imprecise Location-based Range Query (ILRQ)** *returns a set of tuples* $\{(S_j, p_j)|j \neq i \wedge j \in [1, n]\}$*, where* $p_j > 0$ *is the non-zero probability that user* $S_j$ *is located within* $C$ *at the current time instant.*

Figure 2(b) shows a scenario where an ILRQ is computed over cloaked locations, with two range queries, namely $Q_1$ and $Q_2$, issued at two different locations in $U$. For $Q_1$, the answer is $\{(S_2, 0.2),(S_3, 0.6),(S_4, 0.7)\}$, while for $Q_2$, the answer is $\{(S_3, 0.9), (S_4, 1)\}$. After considering the probabilities of objects satisfying the queries issued at all possible points in $U_1$, the answer is:

$$\{(S_2, 0.1), (S_3, 0.7), (S_4, 0.9)\}$$

The probabilities allow the user to place confidence in the answers, which is the consequence of evaluating cloaked (or imprecise) location values. Depending upon the requirements of the application, one may choose to report only the object with the $k$ highest probability value, or only those objects whose probability values exceed a minimum threshold. Our work will be able to work with any of these models. Next, let us examine how an ILRQ can be evaluated.

### 5.2 Evaluation of ILRQ

The ILRQ is evaluated in three phases:

1. Pruning Phase, which removes all the objects that do not have any chance of satisfying the ILRQ.
2. Transformation Phase, which converts an ILRQ into subqueries, and
3. Evaluation Phase, which computes probabilistic answers for the ILRQ.

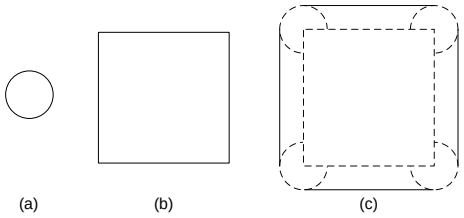


**Fig. 3.** Illustrating the Minkowski Sum.

**1. Pruning Phase.** In the first step, we retrieve objects from the database that are candidates for the query answer. It prunes away objects that have no chance of satisfying the query, and hence allows us to manipulate a smaller set of objects for further processing (e.g., calculating their probabilities).

There are a number of ways to achieve the pruning effect. First of all, the service provider can exploit the privacy preferences of the users as suggested in Section 3. For example, user $S_j$ may specify that she does not want user $S$ to know that they are close to each other. Moreover, $S$ may have defined the list of the people of interest to her, and she only requires the subset of the names of these people to appear her query answer. By using these policies, the service provider can filter away objects before further processing.

The second pruning technique is specific to ILRQ. Notice that the objects that do not need to be considered for an ILRQ are those that have zero chance of being located inside the query range $C$. To prune away these objects, we must take into account all the possible locations of the query issuer $S$. Here we employ the concept of the *Minkowski Sum*, which is commonly studied in computational graphics and motion planning. Given two point sets or polygons, $A$ and $B$, the Minkowski Sum is defined as follows [19]:

$$A \oplus B = \{x + y | x \in A, y \in B\}$$

Figure 3 illustrates an example, where the Minkowski Sum of the circle in (a) and the square in (b) is shown in (c). Conceptually, the Minkowski Sum is the union of all translations of $A$ by a point $y$ located in $B$.

We can view the Minkowski Sum of the query range $C$ and the uncertainty region $U$, that is, $U \oplus C$, as the union of all range queries, by considering all the possible positions of $S$ who resides somewhere in $U$. If the uncertainty region of any object being queried does not overlap $U \oplus R$, we can be assured that this object does not have any chance of satisfying any range query issued at any position in $U$. Thus we can use $U \oplus R$ as a query range to obtain objects that have non-zero probability of satisfying the query (i.e., their uncertainty regions overlap with the query range). The objects retrieved from this range query will be the ones that are processed in the subsequent phases.

Standard algorithms for computing the Minkowski Sum are discussed in [19], and here we do not present the details due to space limitation. The worst-case complexity of these algorithms, given a circle and a polygon with $e$ sides, can have a complexity of $O(e^2 \log e)$ (if the polygon is non-convex).To reduce the complexity of these algorithms, notice that the Minkowski Sum is used solely to improve the efficiency of our methods. It is not necessary that we obtain the exact shape, although that may result in getting false hits that do not contribute to the query answers. If $U$ is a non-convex polygon, we can first approximate it with a convex polygon that circumscribe it (e.g., compute its convex hulls, which needs $O(e \log e)$ times for a $e$-sided polygon) [19]. We can also circumscribe the circular query range, $C$, with a $m$-sided polygon (e.g., a square). The resulting Minkowski sum then produces a convex polygon with at most $m + e$ edges, and requires only $O(m + e)$ time to compute. A range query using this approximate version of Minkowski Sum can then be issued to produce a set of objects (denoted by $K$) for the next phase.

**2. Transformation Phase.** In LRQ, the query range of the user $S$ is a circle $C$ with radius $r$ and center $L$. If the user transmits her cloaked location,

the query range is no longer $C$, since the service provider has no idea of where $L$ exactly is. The service provider does know that $L$ is within $U$, so it transforms the query into sub-queries over all possible locations of $S$. In other words, at each point $(u, v) \in U$, a query is issued to find out which user(s) from the set $K$ is(are) within the region $C'(u, v)$, where $C'(u, v)$ is the circle with radius $r$ centered at $(u, v)$. The result of ILRQ is essentially the union of the results of the range queries issued at each point in $U$.

**3. Evaluation Phase.** This phase computes the actual probability that each object in $K$ satisfies the ILRQ. Notice that the probability $p_j(u, v)$ of user $S_j$ satisfying $S$'s request at point $(u, v) \in U$ is given by

$$p_j(u, v) = \frac{\text{Area}(U_j \cap C'(u, v))}{\text{Area}(U_j)} \tag{2}$$

where $U_j \cap C'(u, v)$ is the common region between $U_j$ and $C'(u, v)$. Essentially, $p_j(u, v)$ is the fraction of $U_j$ that overlaps $C'(u, v)$.

The total probability of $S_j$ satisfying the ILRQ issued by $S$ is given by the integration of the product of the pdf of user $S$'s location at $(u, v)$ (i.e., $\frac{1}{\text{Area}(U)}$) and $p_j(u, v)$ over all $(u, v) \in U$. Therefore,

$$p_j = \int_U \frac{1}{\text{Area}(U)} p_j(u, v) du dv \tag{3}$$

$$= \frac{\int_U \text{Area}(U_j \cap C'(u, v)) du dv}{\text{Area}(U)\text{Area}(U_j)} \tag{4}$$

by substituting $p_j(u, v)$ with Equation 2. The probability so computed serves indicates the confidence placed on the answer. For example, in Figure 2(b), $p_2$ is only 0.1, showing that $S_2$ is unlikely to be the answer, while $S_3$ and $S_4$ have a much higher chance (0.5 and 0.9 respectively) of being the answers.

In practice, $p_j$ can be evaluated with numerical integration techniques. The basic idea is to collect a set of sampling points $(u, v)$ for $U$. Then $p_j(u, v)$ is computed for each sampling point. The sum of these $p_j(u, v)$ values divided by $\text{Area}(U)$ yields the approximate value of $p_j$. Here an important issue is to use suitable number of sampling points to achieve an accurate answer. We have implemented numerical integration in our experiments (Section 7), and have also determined the number of sampling points experimentally. In our technical report, we have also shown how the evaluation can be expressed as a SQL query in a spatial database [20].

## 6 Quality of Imprecise Queries

Due to the inherent imprecision in location data and the query itself, an imprecise query returns probabilistic answers, which can be ambiguous. Here we investigate the notion of quality for imprecise queries that quantifies query ambiguity. It also serves as an indicator to the query issuer to see whether she should adjust

the degree of her location uncertainty. For ease of exposition, we again use the simplified notions as described in Section 5.1.

The quality of an imprecise query is affected by the uncertainty of the query issuer's location. In an ILRQ, for example, the position of the query issuer $S$, which is part of the query, is only known to be accurate within the uncertainty region $U$. As a result, there can be numerous possible answers, where only one of them is correct. However, the final answer returned to the user is the union of all possible answers, and this reduces the overall answer quality.

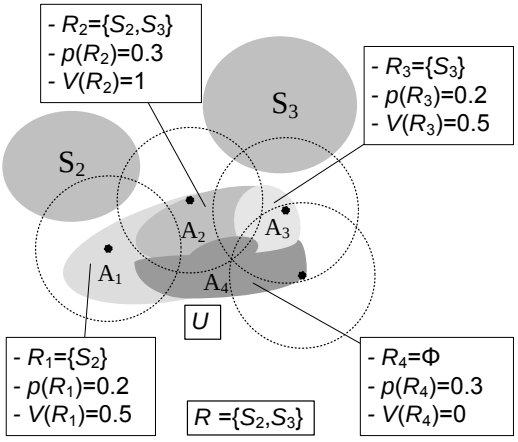


**Fig. 4.** Illustrating the query score of ILRQ.

To illustrate, consider Figure 4 which shows the uncertainty region $U$ of $S$, and also the uncertainty regions of two objects being queried, $S_2$ and $S_3$. Some sub-queries of the ILRQ issued by $S$ (in dotted line) are also shown. When the same query $C$ is issued at different points in $U$, the query results can also vary. For example, the range query issued in the region $A_1$ yields $S_2$ as the result, while for the query issued in the region $A_3$, the answer is $S_3$. In reality, $S$ is only positioned in one point in $U$, so the true answer is only one of the possible answers. However, the final answer returned to $S$ contains both $S_2$ and $S_3$. Thus, the inconsistency of results can affect the overall quality of a query answer.

Our goal is to provide a quality metric to quantify the quality of the query result due to the ambiguity of the query issuer. The metric should take into account (1) how many different answers are produced and (2) how different these answers are. Intuitively, the quality should be the best when the uncertainty region $U$ is reduced to a single point, since in that case there is only one single answer. In Figure 4, for instance, if the uncertainty region is simply the point in $A_2$, then the set $\{S_2, S_3\}$ is the only answer.

Let $R$ be the set of answers that are actually returned to the user. In Figure 4, $R$ equals to $\{S_2, S_3\}$. Our approach is to partition $U$ into subregions according to their answer sets. For example, in Figure 4, $U$ is partitioned into four subregions, namely $A_1, A_2, A_3$ and $A_4$. Moreover, if a query is issued at any one point in $A_1$, the answer set $R_1$, i.e., $\{S_2\}$, is produced. Notice that for all values of $i, j \in [1, 4]$ (with $i \neq j$), some objects in $R_i$ and $R_j$ can be the same, but $R_i - R_j \neq \emptyset$. The final answer $R$ is the union of all the answer sets $R_1, \ldots, R_4$.

Now we can measure the similarity between the answer returned $R$, and each sub-result $R_k$. Specifically, we define the *precision* of $R$ with respect to $R_k$ as

$$V(R_k) = \frac{|R_k|}{|R|} \tag{5}$$

where $V(R_k)$ indicates the amount of "impurities" injected to $R_k$ in order to become $R$. As $V(R_k)$ increases from 0 to 1, $R_k$ is closer to $R$. Thus $V(R_k)$ serves as a score for measuring the closeness of $R$ to $R_k$, if $S$ is located in $A_k$. In Figure 4, if $S$ is located in $A_1$, the score obtained for this query answer, i.e., $V(R_1)$, is $\frac{1}{2}$.

However, $S$ has only a certain probability of being located in $A_1$. Let this value be $p(R_1)$, which is also the probability that $S_1$ yields $R_1$ as the answer. The product $p(R_1) \cdot V(R_1)$ can then be viewed as the precision weighted by the probability of $S_1$ having $R_1$ as the true answer.

In general, given that $U$ is partitioned into $B$ subregions according to the difference in their answers, the *query score* of ILRQ can be measured by the sum of the weighted precision values at all the subregions of $U$,

$$\text{Query score } = \sum_{k=1}^{B} p(R_k) V(R_k) \tag{6}$$

which varies between 0 (lowest quality) and 1 (highest quality). For example, the query score shown in Figure 4 is $0.2 \cdot 0.5 + 0.3 \cdot 1 + 0.2 \cdot 0.5 + 0.3 \cdot 0 = 0.32$.

An important question is how to compute $p(R_k)$, i.e., the probability that $S$ gets the answer $R_k$. Let us assume that each sub-query of ILRQ issued at point $(u, v)$ returns a set of answers $Q'(u, v)$. Then,

$$p(R_k) = \int_{(u,v) \in U \wedge R_k = Q'(u,v)} \frac{1}{Area(U)} du dv \tag{7}$$

that is, the integration of the uniform pdf over all points in $U$ that evaluate the same result $R_k$.

The answer quality metric allow a user to trade-off privacy for a potentially better answer quality. In particular, the query score depends on the size of the uncertainty region – a larger uncertainty region potentially yields more distinct answers and lower query scores. Therefore, a low query score indicates that the user may reduce the size of her uncertainty region and resubmit the query.

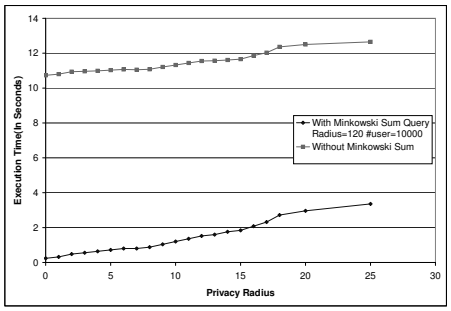


**Fig. 5.** Execution time vs. Privacy

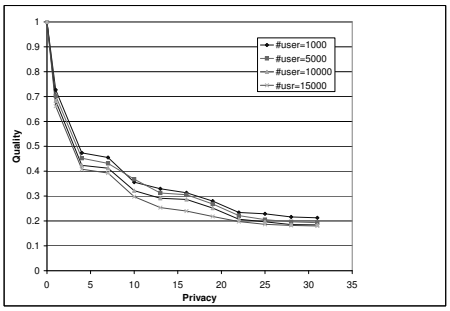


**Fig. 6.** Quality vs. Privacy

## 7 Experimental Results

We have performed an extensive simulation study on the behavior of location cloaking. Our experiments are based upon data generated by the City Simulator 2.0 [21] developed at IBM. The City Simulator simulates the realistic motion of 10,000 people moving in a city. The input to the simulator is a map of a city. We used the sample map provided with the simulator that models a city of size $840 \times 1260$ square units, with 71 buildings, 48 roads, six road intersections and one park. The simulator models the movement of objects within the buildings, as well as their relatively faster movement on the roads and highways.

Each object reports its location to the server at an average rate of $0.5\text{sec}^{-1}$. An ILRQ is generated by randomly choosing a user as the query issuer. The ILRQ has a range of radius $r$ of 150 units. Each user has an uncertainty region with a radius of $U_i(t).r$, equal to 30 units. The radius is denoted as "privacy" in the graphs shown. In our experiments, since both the query range and the query issuer ($S_i$)'s uncertainty region are circles, their Minkowski Sum is simply a circle centered at the query issuer's location, with a radius of $(r + U_i(t).r)$. Each data point is the average value over 200 location update cycles. We use the radius of uncertainty region as a measure of the location privacy of user – a larger radius implies a higher degree of privacy.

### 7.1 Effectiveness of Location Cloaking

**Quality and Performance.** In Figure 5 we see that an increase in privacy value lengthens the execution time of ILRQ. With a higher privacy (or uncertainty region of the query issuer), the ranges of sub-queries cover a larger area. Thus more objects are involved in computation, resulting in a higher execution time. We can also see from the same figure that when the Minkowski Sum is used as an initial filtering phase, the performance improves by at least a factor of three. Thus, the use of the Minkowski Sum in the first step is effective in improving the query evaluation performance.

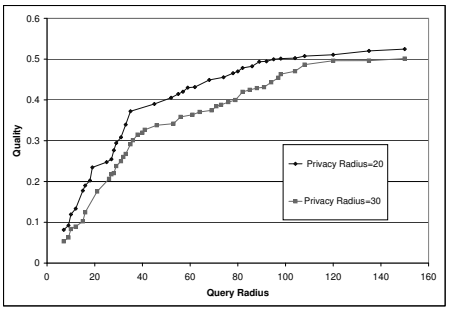


**Fig. 7.** Quality vs. Query Radius

**Quality and Privacy.** We investigate the effect of location privacy on query score of the ILRQ. Figure 6 shows the result for different number of users. The quality is 1 (highest) when there is no privacy at all. As privacy (i.e., uncertainty region area) increases, the query score drops. This is because the larger uncertainty region increases the number of distinct query answers, thereby lowering the query score. We also observe the difference in quality when the number of users varies. In general, for the same privacy value, a larger population produces a lower score, since more distinct answer sets are produced.

**Quality and Query Size.** Next, we study the effect of query size on answer quality. Figure 7 illustrates that answer quality increases with query size. With a fixed privacy value (uncertainty radius), an increase in the query size creates less distinct answer sets. For example, when the query range has a very large radius (140) compared with the radius of the uncertainty region (which is equal to 20), the query ranges created will render many similar answers, since the difference in the answers to the queries at different points in the uncertainty region is small. As shown in the same graph, at a larger uncertainty region radius (30), the relative difference between the uncertainty size and privacy is smaller than when the radius is 20, and thus the quality is lower too.

## 8 Conclusions

In this paper we suggested a framework to connect privacy, information cloaking and service quality. We proposed imprecise queries, which hide the identity of the query issuer and enable evaluation of cloaked information. We studied an evaluation algorithm and quality metrics of moving range queries. We also performed an extensive simulation to investigate the behavior of our proposed methods.

## References

1. Warrior, J., McHenry, E., McGee, K.: They know where you are. IEEE Spectrum **40**(7) (2003) 20– 25

2. Gruteser, M., Grunwald, D.: Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. In: Proc. 1st Intl. Conf. on Mobile Systems, Applications, and Services. (2003)
3. Varshney, U.: Location management for mobile commerce applications in wireless internet environment. ACM Transactions on Internet Technology **3**(3) (2003)
4. Beresford, A.R., Stajano, F.: Location Privacy in Pervasive Computing. IEEE Pervasive Computing **2**(1) (2003) 46–55
5. Snekkenes, E.: Concepts for personal location privacy policies. In: Proceedings of the 3rd ACM conference on Electronic Commerce, ACM Press (2001) 48–57
6. Hengartner, U., Steenkiste, P.: Protecting Access to People Location Information. In: Proc. 1st Intl. Conf. on Security in Pervasive Computing. (2003)
7. Hengartner, U., Steenkiste, P.: Access control to information in pervasive computing environments. In: Proc. 9th USENIX Workshop on HotOS. (2003)
8. Cheng, R., Prabhakar, S.: Using uncertainty to provide privacy-preserving and high-quality location-based services. In: Workshop on Location Systems Privacy and Control, MobileHCI 04. (2004)
9. Atallah, M., Frikken, K.: Privacy-preserving location-dependent query processing. In: Proc. ACS/IEEE Intl. Conf. on Pervasive Services (ICPS). (2004)
10. Mokbel, M., Xiong, X., Aref, W.: SINA: Scalable incremental processing of continuous queries in spatio-temporal databases. In: Proc. ACM SIGMOD. (2004)
11. Pfitzmann, A., Hansen, M.: Anonymity, unobservability, psuedonymity, and identity management - a proposal for terminology. (2004)
12. Sweeney, L.: k-anonymity: a model for protecting privacy. Intl. Journal on Uncertainty, Fuzziness and Knowledge-based Systems **10**(5) (2002)
13. LeFevre, K., DeWitt, D., Ramakrishnan, R.: Incognito: efficient full-domain k-anonymity. In: Proc. ACM SIGMOD Intl. Conf. (2005)
14. Bertino, E., Ooi, B., Yang, Y., Deng, R.: Privacy and ownership preserving of outsourced medical data. In: Proc. IEEE ICDE. (2005)
15. Gruteser, M., Liu, X.: Protecting privacy in continuous location-tracking applications. IEEE Security and Privacy **2**(2) (2004)
16. Gedik, B., Liu, L.: A customizable k-anonymity model for protecting location privacy. In: ICDCS. (2005)
17. Cheng, R., Kalashnikov, D., Prabhakar, S.: Evaluating probabilistic queries over imprecise data. In: Proc. ACM SIGMOD. (2003)
18. Serjantov, A., Danezis, G.: Towards an information metric for anonymity. In: Privacy Enhancing Technologies: 2nd Intl. Workshop, PET 2002. (2002)
19. Berg, M., Kreveld, M., Overmars, M., Schwarzkopf, O.: Computational Geometry – Algorithms and Applications, 2nd ed. Springer Verlag (2000)
20. Cheng, R., Zhang, Y., Bertino, E., Prabhakar, S.: Querying private data in moving-object environments. Technical Report CERIAS TR #2005-45, Purdue U (2005)
21. Kaufman, J., Myllymaki, J., Jackson, J.: IBM City Simulator Spatial Data Generator 2.0 (2001)
22. Stallings, W.: Wireless Communications and Networks. Prentice Hall (2005)
23. Wong, V., Leung, V.: Location management for next-generation personal communications network. IEEE Network (2000)

## A Applicability to Real-World Systems

Can the framework proposed in Section 3 be deployed in systems like the GPS and cellular network systems? Here we describe how this is possible for two major classes of systems:

– **Systems that do not track a user's location regularly.** Typical examples are the GPS and RF-ID systems. In particular, the GPS receiver that resides in the user's device obtains location information from 24 satellites and gets a location precision ranging from a few to several hundred meters [3]. Our framework is readily used here, where the user's device can generate location data spontaneously.
– **Systems that track a user's location regularly.** The cellular network system (such as GSM and PCS) are representative examples, where the cellular network system has to identify the user's location (in terms of a hexagonal cell with radius ranging from 0.1 to 1 km [22]) in order to deliver phone calls. If the system is trusted, the user can contact it directly before sending her location to the service provider [15].
Here we describe how a user can place less trust on the cellular network system compared to [15]. In this solution, the user only lets the cellular network know a rough position of her, by specifying a cloaked location that is represented by a set of neighboring cells. The cloaking agent should map the cloaked location into cells before sending them to the system. This may result in an increase in connection cost, due to the additional effort in finding out the user in a particular cell for making a connection. Without going into further details, we would like to point out that such a scheme (that allows a user to notify the cellular network her approximate positions) has already been implemented in cellular networks [23]. Thus it is possible to deploy our framework in these systems.

## B Threat Analysis and Solutions

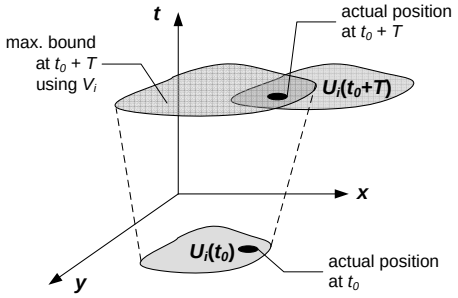


**Fig. 8.** Illustrating trajectory tracing.

In this paper, we have assumed the service provider holds the same amount of location uncertainty as reported by the user. In this section, we investigate how the service provider can improve its knowledge about the user's location.

Specifically, we describe a technique called "trajectory tracing" that can reduce the effectiveness of cloaking in Section B.1. We then describe solutions to reduce this threat in Section B.2, and present experimental results in Section B.3.

### B.1 Trajectory Tracing

Suppose the service provider saves all the cloaked locations it has received from a user. The service provider may also capture the maximum velocity information of the user through her movement record, her vehicle, etc. We now show that it is possible for the service provider to reduce the size of uncertainty region reported by the user.

Specifically, let the maximum velocity of a user $S_i$ be $V_i$. Assume $S_i$ sent her last cloaked location at time $t_0$, i.e., $U_i(t_0)$, and then again after $T$ time units, i.e., $U_i(t_0 + T)$. Using the values of $V_i$ and $U_i(t_0)$, the service provider can know the possible location of $S_i$. As shown in Figure 8, the service provider derives the bound enclosing $S_i$'s location at time $t_0+T$ (called *maximum bound*) Thus, even if $S_i$ says she is located somewhere in $U_i(t_0+T)$, her possible location is actually limited within the overlapping region between $U_i(t_0 + T)$ and the maximum bound, which is smaller than $U_i(t_0 + T)$. Notice that this is an accumulative effect, since the service provider can derive an even smaller bound based on the overlapping region.

If the service provider is able to reduce the location uncertainty of the user, the location privacy may be threatened. In non-anonymous applications, for example, the user's identity is also transmitted to the service provider, and hence the service provider can associate with the user's identity with a smaller uncertainty region. Even in anonymous applications, the service provider can use a map to help identifying the owner of the location. For instance, if the uncertainty region has significant overlap with a residential area, the owner of the location may be discovered and her movement can then be tracked. We propose two techniques, namely **patching** and **delaying**, in order to alleviate this problem.

### B.2 Patching and Delaying

The first solution is to combine the cloaked locations released in the past with the current cloaked location before it is sent. We call this technique *patching*. Figure 9(a) illustrates this concept. At time $t_0 + T$, in place of $U_i(t_0 + T)$, the user $S_i$ sent the region $U_i'(t_0 + T) = U_i(t_0) \cup U_i(t_0 + T)$. As a result, the "loss" of uncertainty in $U_i(t_0 + T)$ due to trajectory tracing is "compensated" by the inclusion of $U_i(t_0)$, which is assured to be within the maximum bound. Essentially, the spatial accuracy of the location is further relaxed. Notice that this may cause a degradation in query quality due to the increase in uncertainty, as shown in our experimental results.

Another solution is based on relaxing the timing requirement, which we termed "*delaying*". The idea is to suspend the request until the cloaked location fits into the maximum bound. As shown in Figure 9(b), $U_i(t_0 + T)$ is not

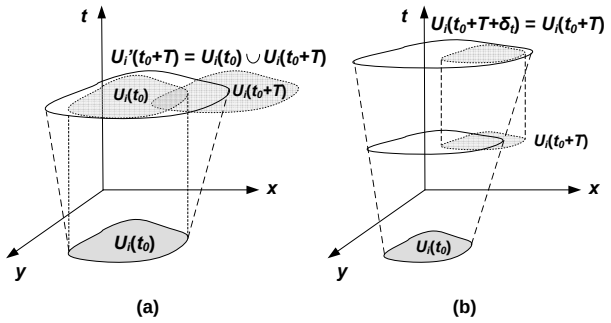


**Fig. 9.** (a) Patching and (b) Delaying.

sent until after $\delta_t$ more time units, when $U_i(t_0+T)$ is guaranteed to be within the maximum bound. The value of $\delta_t$ can be estimated based on distance between the edges of the adjacent uncertainty regions, and the value of the maximum velocity. The advantage of this scheme over patching is that the extent of the cloaked location remains unchanged and so the quality is not affected. However, the response time of the query can be increased due to the delay introduced, which can be an important Quality-of-Service parameter in time-critical applications.

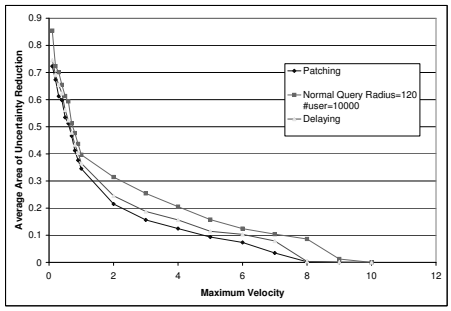


**Fig. 10.** Uncertainty vs. Velocity

### B.3 Effectiveness of Patching and Delaying

We examine experimentally the behaviors of the patching and delaying policies. We compare these two policies with the case when these two policies are not used (denote this case as *normal*). We first investigate the impact of the knowledge about the maximum velocity of the query issuer on the amount of uncertainty reduced. Figure 10 shows that regardless of whether patching or delaying is used,

the area of uncertainty region reduced is lower with increase in the maximum velocity. As illustrated in Figure 8, a faster velocity increases the amount of overlap of the maximum bound with the next reported uncertainty region, and thus the portion of uncertainty reduced also decreases. While both patching and delaying have lower amount of uncertainty reduced than *normal*, patching performs the best due to the enlargement of the uncertainty region sent.

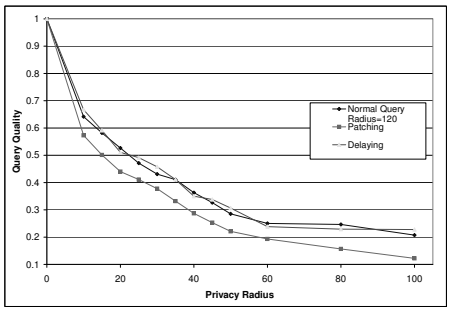


**Fig. 11.** Quality vs. Privacy

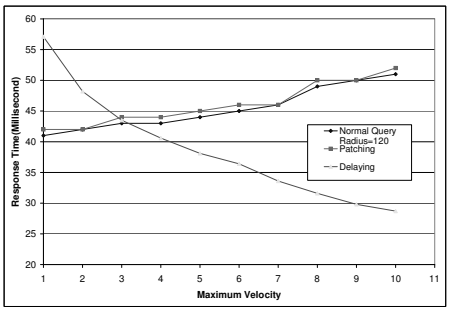


**Fig. 12.** Time vs. Velocity

Next, we examine the effect of privacy of the query issuer on query quality. Figure 11 shows that as privacy increases, the three policies suffer from different degrees of quality degradation. While both *normal* and delaying perform similarly, patching performs the worst. This is because patching augments the previously reported uncertainty region, which is enlarged with a larger privacy radius. This results in a more ambiguous query issuer location and a poorer query quality. This is the price paid for a lower reduction of uncertainty area, as we have described previously.

Finally, Figure 12 illustrates the response time (i.e., the duration between the time that the user submits a query and the time that the query is finished). Delaying outperforms both *normal* and patching at a maximum velocity higher than 3. This is because the maximum bound can cover the newly reported uncertainty region faster (c.f. Figure 9). However, delaying does not work well when the maximum velocity is small, since the cloaking agent has to wait for a longer time until the uncertainty region can be covered by the maximum bound. In these cases, it may be necessary to limit the waiting time, and the uncertainty region is sent even though it may not be fully covered by the maximum bound.