Enabling Private Continuous Queries For Revealed User Locations

Chi-Yin Chow and Mohamed F. Mokbel

Department of Computer Science and Engineering, University of Minnesota
{cchow,mokbel}@cs.umn.edu

Abstract. Existing location-based services provide specialized services to their customers based on the *knowledge* of their exact locations. With untrustworthy servers, location-based services may lead to several privacy threats ranging from worries over employers snooping on their workers' whereabouts to fears of tracking by potential stalkers. While there exist several techniques to preserve location privacy in mobile environments, such techniques are limited as they do not distinguish between location privacy (i.e., a user wants to hide her location) and query privacy (i.e., a user can reveal her location but not her query). Such distinction is crucial in many applications where the locations of mobile users is publicly known. In this paper, we go beyond the limitation of existing cloaking algorithms as we propose a new *robust* spatial cloaking technique for snapshot and continuous location-based queries that clearly distinguishes between location privacy and query privacy. By such distinction, we achieve two main goals: (1) supporting private location-based services to those customers with public locations, and (2) performing spatial cloaking on-demand basis only (i.e., when issuing queries) rather than exhaustively cloaking every single location update. Experimental results show that the robust spatial cloaking algorithm is scalable and efficient while providing anonymity for large numbers of continuous queries without hiding users' locations.

1 Introduction

The emergence of the state-of-the-art location-detection devices, e.g., cellular phones, global positioning system (GPS) devices, and radio-frequency identification (RFID) chips, results in a location-dependent information access paradigm, known as *location-based services*. Location-based services provide convenient information access for mobile users who can issue location-based *snapshot* or *continuous* queries to a database server at anytime and anywhere. Examples of *snapshot* queries include "*where is my nearest gas station*" and "*what are the restaurants within one mile of my location*", while examples of *continuous* queries include "*continuously report my nearest police car*" and "*continuously report the taxis within one mile of my car*". Although location-based services promise safety and convenience, they threaten the security and privacy of their customers [1, 2, 3, 4]. With untrustworthy servers, an adversary may access sensitive information about individuals based on their issued location-based queries

queries. For example, an adversary may check a user's habit and interest by knowing the places she seeks. In many real life scenarios, GPS devices have been used in stalking personal locations [5, 6, 7].

To tackle the privacy threats in location-based services, several spatial cloaking algorithms have been proposed for preserving user location privacy (e.g., [8, 9, 10, 11, 12, 13]). The key idea of spatial cloaking algorithms is to blur the exact user location information into a spatial region that satisfies certain privacy requirements. Privacy requirements can be represented in terms of k-anonymity [14] (i.e., a user location is indistinguishable among k users) and/or minimum spatial area (i.e., a user location is blurred into a region with a minimum size threshold).

Unfortunately, existing techniques for preserving location privacy have limited applicability as they do not distinguish between *location* and *query* privacy. In many applications, mobile users do not have the luxury to choose hiding their locations as such locations are publicly known. In other applications, mobile users do not mind to reveal their exact location information, however, they would like to hide the fact that they issue some location-based queries as these queries may reveal their personal interests. So far, none of the existing spatial cloaking algorithms support such distinction between location privacy and query privacy where it is always assumed that users have to hide both their locations and their queries. Examples of applications that call for such distinction between location privacy and query privacy include:

- **Business operation**. A courier business company has to know the locations of its employees to decide which employee is the nearest one to collect a certain package. However, the company is not allowed to keep track of the employees' behavior in terms of their location-based queries. Thus, company employees reveal their location information, but not their query information.
- **Monitoring system**. Monitoring systems (e.g., transportation monitoring) rely on the accuracy of user locations to provide their valuable services. In order to convince users to participate in these systems, certain privacy guarantees should be imposed for users' behavior through guaranteeing the privacy of the users' location-based queries although users' locations will be revealed.
- **Regular working hours**. During daytime, the locations of company employees are publicly known as their office cubes. Yet, such employees still want to maintain their privacy when they issue location-based queries as such queries would reveal their private personal interests.

In this paper, we present a new *robust* spatial cloaking algorithm that clearly distinguishes between *location* privacy and *query* privacy where mobile users can entertain private *continuous* and *snapshot* location-based queries even if their locations are revealed. By such distinction, we achieve two main goals: (1) supporting private location-based services to those customers with public locations, and (2) performing spatial cloaking on-demand basis only (i.e., when issuing queries) rather than exhaustively cloaking every single location update as in traditional spatial cloaking techniques. The main idea of our *robust* spatial cloaking algorithms is to anonymize the link between user locations and location-based queries in such a way that even the user location is known, an adversary would not be able to link the user location to the submitted query. Such paradigm is a radical shift from almost all of existing location privacy techniques that aim to anonymize the user location itself with the assumption that if an adversary cannot get access to the user location, then the adversary cannot know the user query.

To achieve our goals, we go through three main steps. First, we identify two main adversary attacks, namely, *query sampling* and *query tracking* attacks, that take place in almost all existing location privacy techniques when distinguishing between location and query privacy. Second, we identify two main properties, namely, *k-sharing region* and *memorization*, that if satisfied in any location cloaking technique, the underlying technique will be free of the *query sampling* and *query tracking* attacks. Finally, we present our *robust* algorithm that employs the *k-sharing region* and *memorization* properties to support private *continuous* location-based queries for users with public location information. In general, the contributions of this paper can be summarized as follows:

- We introduce a novel query privacy notion in which mobile users are either obligated or willing to reveal their locations, yet they do not want to be identified as the issuer of their location-based queries. Such privacy notion is relaxed from the widely used notion that considers hiding the user location and user query is one process. Several applications can make use of our new notion to enhance the overall quality of location-based services.
- We show that directly applying existing spatial cloaking techniques to the new query privacy notion would immediately result in two privacy attack models, namely, *query sampling* and *query tracking* attacks, that can be used by adversaries to infer the actual querying users.
- We identify two main properties, namely, *k-sharing region* and *memorization* that if applied to any location cloaking technique, would make it free from the introduced attack models.
- We propose a new *robust* spatial cloaking technique that: (a) distinguishes between location privacy and query privacy, (b) employs the *k-sharing region* and *memorization* properties, and (c) supports *continuous* location-based queries.
- We provide experimental evidence that the robust spatial cloaking algorithm is scalable in terms of supporting large numbers of users and continuous queries, efficient in terms of supporting various user privacy requirements, provides high-quality services without compromising users' query privacy.

The rest of the paper is organized as follows: Section 2 highlights the related work. The underlying system model is presented in Section 3. Section 4 outlines the adversary attacks. Section 5 presents the required properties to avoid the identified adversary attacks. Our proposed robust spatial cloaking technique is presented in Section 6. Section 7 delineates extensive experimental evaluation of our proposed techniques. Finally, Section 8 concludes the paper.

2 Related Work

Almost all previous techniques for location privacy do not distinguish between location privacy and query privacy where the main focus is always to preserve the location privacy. Preserving query privacy is done as a by product of preserving the location privacy. For example, if a certain user wants to have her location k-anonymized, then each query issued by such users will be also k-anonymized. Unfortunately, such techniques would not work if the location *cannot* be anonymized. In general, existing research efforts for preserving location privacy can be categorized based on three orthogonal dimensions, namely *employed techniques*, *underlying system architecture*, and *user privacy requirements*:

- **Employed techniques**. Based on the underlying employed technique, location privacy techniques can be classified to either: (a) reporting false locations [15, 16] where the main idea is to cheat the server by either generating a set of n locations in which only one of them is true [15] or aligning the actual location to the nearest prescribed landmark location [16], or (b) spatial cloaking [8, 9, 10, 11, 12, 13, 17, 18] where the main idea is to blur user locations into spatial regions that satisfy certain privacy requirements. In this paper, we focus on spatial cloaking techniques as they are more efficient, accurate, and most commonly used than false location techniques.
- **Underlying system architecture**. Based on the underlying system architecture, location privacy techniques utilize either: (a) a centralized architecture in which a third trusted party is responsible in cloaking the locations of mobile users [8, 9, 10, 11, 12, 13], or (b) a peer-to-peer architecture in which mobile users collaborate with other peers to find cloaked spatial regions [18, 19, 20]. In this paper, we do not have any assumption for the underlying system architecture as our proposed techniques can be applied to both centralized and peer-to-peer architecture.
- **User privacy requirements**. Based on user privacy requirements, location privacy techniques consider at least one of two main privacy requirements: (a) k-anonymity in which the user wants to be indistinguishable among k users [9, 10, 11, 12, 13, 18], and (b) minimum area in which the user wants to have a blurred region with an area size at least A_{min} [8, 12, 18]. Our proposed techniques in this paper support both the k-anonymity and minimum area requirements.

In terms of *continuous* queries, existing research efforts for location privacy techniques (e.g., see [9, 10, 12, 15, 16, 17, 13, 8, 18]) focus only on the case of *snapshot* queries with no direct extension to the case of *continuous* queries .

Our proposed *robust* location privacy technique distinguishes itself from all previous techniques in the following: (1) It distinguishes between location privacy and query privacy, thus it can still provide anonymity to location-based queries even if the user locations are known, (2) It preserves the privacy of *continuous* queries as well as *snapshot* queries, (3) It does not hold any assumption about the system architecture as it can work for both centralized and peer-to-peer architecture.

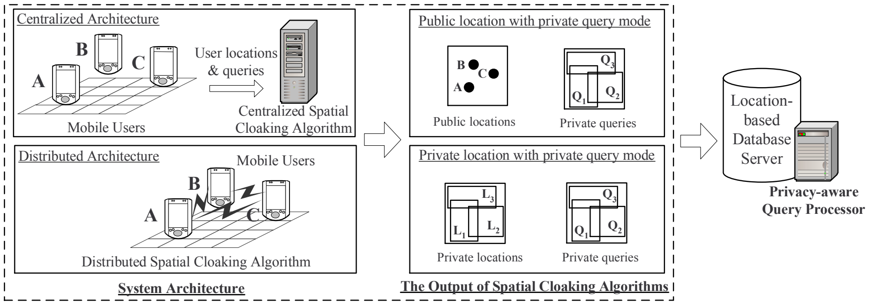


Fig. 1. System architecture and privacy modes

3 System Architecture

Figure 1 depicts the underlying system architecture that can employ our techniques. In general, the system architecture includes three main entities, *mobile users*, *spatial cloaking techniques*, and back-end *database server*.

3.1 Mobile users

To distinguish between user location privacy and user query privacy, mobile users register in the system with a privacy profile (k_l, k_q), where k_l indicates that the user wants her *location* to be k_l-anonymous, i.e., the *cloaked* region for user *location* should contain at least k_l users, while k_q indicates that the user wants her *query* to be k_q-anonymous, i.e., the *cloaked* region for user *query* can be reported by at least k_q users (if all these users want to issue queries). Based on the values of k_l and k_q, we distinguish between two privacy modes:

- *Public location with private query* $(k_l = 1, k_q > 1)$. In this mode, users are willing or obligated to reveal their locations $(k_l = 1)$, yet they do not want adversaries to link their locations to the queries they issue $(k_q > 1)$. Thus, the user location is simply sent to a database server without any perturbation processing while the location information of queries is *cloaked* into spatial regions that satisfy the query privacy requirement before forwarding them to the database server.
- *Private location with private query* $(k_l > 1, k_q > 1, k_q \geqslant k_l)$. In this mode, users want to hide both their locations and query information. However, users have the luxury to request different privacy requirements for their locations and queries. In this case, both user locations and queries are blurred into spatial regions according to the privacy requirements before sending them to the database server.

It is important to note that in all cases, $k_q \geqslant k_l$; otherwise, a user query would degrade the degree of privacy protection of the user location. For example, consider the extreme case of a user with $k_q = 1$, even such user has very high k_l, whenever the user issues a query, the user will be the only person within the query region, thus her personal location can be immediately revealed. Another thing to note is that all existing spatial cloaking techniques implicitly consider only the case of $k_l = k_q$, where no distinction is made between location privacy and query privacy.

For simplicity, we did not include the minimum area requirements for the spatial cloaked region. As have been proposed in the literature, integrating the minimum area requirements can be done simply be aligning the cloaked area from the k-anonymity requirement to a grid area that satisfies the minimum area requirements [12, 18].

3.2 Spatial Cloaking Techniques

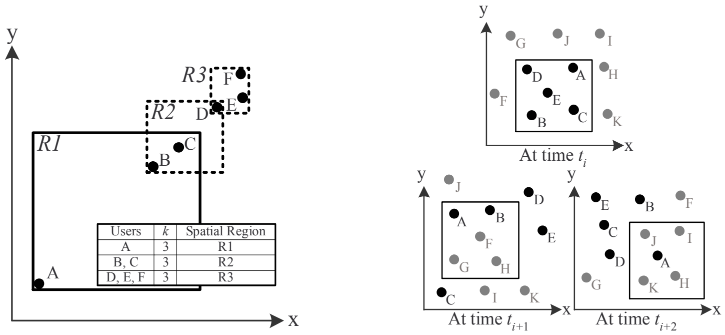
At the core of the system, spatial cloaking techniques are employed to blur the user locations and queries into spatial regions that satisfy each user profile. Our *robust* spatial techniques can be incorporated into either centralized or distributed architecture. In the centralized architecture (top left corner of Figure 1), a third trusted party is employed to blur the user locations and/or queries into cloaked spatial regions while in the distributed architecture (bottom left corner of Figure 1), system users employ a peer-to-peer spatial cloaking algorithm to blur their locations and/or queries into cloaked regions. Regardless of the architecture, the output of the spatial cloaking algorithm has two sets. The first set is either a set of exact point locations in case of public location mode, $k_l = 1$ (represented as black dots A, B, and C in Figure 1) or a set of *cloaked* location regions in case of private location mode, $k_l > 1$ (represented as regions L_1, L_2, and L_3 in Figure 1). The second output set is spatial query regions for the query privacy requirements $k_q \geqslant k_l$ (represented as query rectangles Q_1, Q_2, and Q_3 in Figure 1).

3.3 Database Server

At the back-end of the system, a *privacy-aware query processor* is embedded inside the location-based database server in order to tune its functionalities to deal with cloaked spatial regions for user locations and user queries rather than exact point locations. The details of the privacy-aware quer processor is beyond the scope of this paper where it has been well studied in [12, 21].

4 Privacy Leakage in Spatial Cloaking Techniques

This section presents two privacy attack models that can be used by adversaries to link users with revealed locations to their queries. The first attack, *query sampling*, is applicable for *snapshot* queries while the second attack, *query tracking*, is applicable for *continuous* queries. For these two attacks, we briefly discuss the applicability of the following spatial cloaking algorithms: the adaptive interval cloaking [10], CliqueCloak [9], k-area cloaking [11], Casper [12], hilbASR [13], nnASR [13], and the uncertainty cloaking [8].



Users	k	Spatial Region
A	3	R1
B, C	3	R2
D, E, F	3	R3

(a) Query sampling (b) Query tracking

Fig. 2. Privacy attack models

4.1 Query Sampling Attacks

In many cases, the location distribution of users is not uniform, e.g., there are many users in a shopping mall (a dense area), but there are only a few users in a small cafe near the mall (a sparse area). Thus, the users located in sparse areas become outliers in the system [13]. As a result of such non-uniform user location distribution, most of existing spatial cloaking algorithms tend to generate larger cloaked spatial regions for customers in sparse areas than that of users in dense areas. If the user location information in a sparse area is known, then, using the *query sampling* attack, an adversary can link this location to a certain query.

Attack scenario. Figure 2(a) gives an example of the *query sampling* attack where there are six users A, B, C, D, E, and F. Since almost most of existing spatial cloaking techniques do not distinguish between location privacy and query privacy, users can supply only one value for k anonymity (k=3). The result of the cloaking is that user A has R_1 as its cloaked region, users B and C have R_2 as their cloaked region, while users D, E, and F have R_3 as their cloaked region. In case that user locations are publicly known, an adversary can see that user A is an outlier to the system. Then, from the cloaked region R_1, the adversary can infer that the query is sent by the user A located in the sparse area. The main idea is that if the query has been issued by any other user, the cloaked spatial region must first cover surrounding users in the dense area. As a result, given the knowledge of the user locations, an adversary can link location-based queries to their users.

Analysis. With the exception of CliqueCloak [9] and hilbASR [13], all other spatial cloaking techniques suffer from the *query sampling* attack. For example, the techniques that rely on k-anonymity (the adaptive interval cloaking [10], Casper [12], and nnASR [13]) would simply result in a scenario similar to that of Figure 2(a). On the other hand, spatial cloaking techniques that rely only on a spatial area (e.g., k-area cloaking [11] and uncertainty cloaking [8]) may end up to the case where only one user is located at the cloaked spatial region. Given the public knowledge of the locations of such users, it would be trivial to link a user to its query. Both the CliqueCloak algorithm [9] and hilbASR [13] are free from the *query sampling* attack as these algorithms ensure that a cloaked spatial region R contains at least k users and all these k users report R as their cloaked regions. However, the CliqueCloak algorithm suffers from high computational cost as it can support only k-anonymity up to $k = 10$ [9] while the static version of hilbASR lacks flexibility in supporting various k-anonymous requirements [13].

4.2 Query Tracking Attacks

In continuous queries, mobile users have to continuously report their location information to a database server. Although the location information of a query is cloaked as regions, an adversary could link consecutive time snapshots together to identify the query issuer.

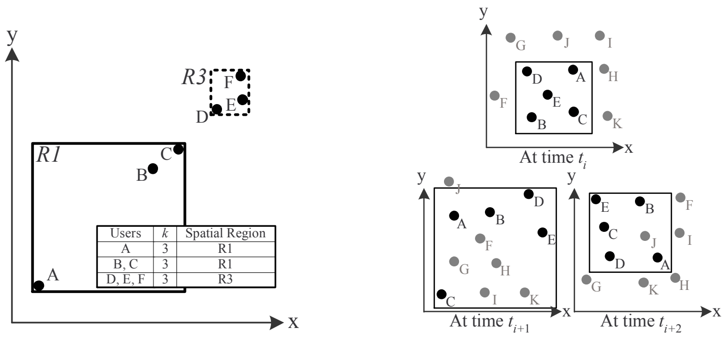
Attack scenario. Figure 2(b) gives an example of the *query tracking* attack where there are eleven mobile users A to K. At time t_i, user A issues a five-anonymous continuous query. Cloaking algorithms would give an area that contains the users A, B, C, D, and E. Assuming uniform distribution, an adversary can only guess that this query is coming from any of the five users within the query area. At time t_{i+1}, mobile users change their locations while the five-anonymous continuous query is still running. An adversary can see that currently the continuous query has the mobile users A, B, F, G, and H. By linking the snapshots of the continuous query at times t_i and t_{i+1}, the adversary can guess that the query issuer is either A or B as these are the only common users between the two snapshots. Similarly at time t_{i+2}, the adversary can conclude that A is the user query issuer as A is the only common users within the query area for the three consecutive snapshots.

Analysis. Since all of our studied cloaking techniques focus only on the case of *snapshot* queries, such algorithms will suffer from the *query tracking* attack.

5 Privacy-Preserving Properties

In this section, we identify two main general properties, namely, *k-sharing region* and *memorization*, that if employed by any spatial cloaking technique, the cloaking technique will be free from *query sampling* and *query tracking* attacks:

- ***k*-sharing region**. Employing the *k-sharing region* property would directly eliminate the *query sampling* attack. The main idea of the *k-sharing region* property is to define a more restrictive k-anonymity requirement: *A cloaked spatial region not only contains at least k users, but the region is also shared by at least k of these users.* Figure 3(a) depicts the result of applying the *k-sharing region* property to the example given in Figure 2(a). Each *cloaked* region R_1 and R_2 is *reported* and *shared* by at least three users. Thus, with the knowledge of user locations and regardless of the user distribution in the space, an adversary cannot link a query to a certain user.
- **Memorization**. Employing the *memorization* property would directly eliminate the *query tracking* attack. The main idea of the *memorization* property



Users	k	Spatial Region
A	3	R1
B, C	3	R1
D, E, F	3	R3

(a) Query sampling (b) Query tracking

Fig. 3. Solutions for privacy attack models

is that the spatial cloaking algorithm has to memorize the users who are contained in the cloaked spatial region of a continuous query at the time when the query is initially issued. Then, with each snapshot of the query, the spatial cloaking algorithm should make sure that such initial users are still with the query cloaked area. Figure 3(b) depicts the result of applying the *memorization* property to the example given in Figure 2(b). The cloaked query regions at all instances of the continuous query at times t_i, t_{i+1}, and t_{i+2} include the five users A, B, C, D, and E. Thus, an adversary cannot narrow down his search to less than the five original users.

It is important to note that both the *k-sharing region* and *memorization* properties are algorithm-independent. So, any spatial cloaking algorithm that has these properties is free from the *query sampling* and *query tracking* attacks.

6 Robust Spatial Cloaking Algorithm

This section presents our *robust* spatial cloaking algorithm that distinguishes between location privacy and query privacy while supporting continuous queries. The main idea is to *group* a set of mobile users together such that the cloaked query region for each mobile user in a *group* G is the spatial region that includes all users in G. The rest of this section is organized as follows: Section 6.1 introduces the *dynamic group* concept which is the main underlying idea of our proposed *robust* spatial cloaking algorithm. Section 6.2 discusses the algorithm details. Section 6.3 proves that the proposed algorithm satisfies both the *k-sharing region* and *memorization* properties.

6.1 Dynamic Group Concept

The main idea of the *dynamic group* concept is to group users together based on their privacy requirements where each group of users has at least one user currently issuing a location-based query. Formally, a group of users should have the following three properties:

1. The number of users in a group is equal to or larger than the most restrictive k-anonymity query requirement among all *querying* users in the group.
2. All users in the same group report the same cloaked spatial region as their cloaked query regions. Such spatial region is the minimum region (aligned to some grid) that includes all users belong to that group.
3. For each group, if there are more than one user issuing the same query, the query is only registered once with the database server.

In general, a user is not allowed to issue a snapshot or continuous query unless the user belongs to a certain *group*. Users may leave their groups once their queries are terminated. Similarly, users may join a new group whenever they want to issue new queries. In the same time, users may be added to or removed from some groups to help other users form a cloaked query area. A user can be either in a *grouped* or *ungrouped* state. Initially, all users are in the *ungrouped* state. Whenever a user joins an existing group or form a new group, the user becomes in the *grouped* state. Only *grouped* users are allowed to issue location-based queries. Each user maintains a tuple $U = (id, L, K_l, K_q, \mathcal{Q}, G)$, where id is a unique user identifier, L is the user's current location, K_l and K_q are the location anonymity and query anonymity privacy requirements, respectively, $\mathcal{Q}$ is a set of queries sent by the user, and G is the identifier of the group where the user is assigned to. Setting G to *null* indicates that the user is in the *ungrouped* state. For each group, we maintain a tuple $G = (id, \mathcal{M}, R, K, \mathcal{Q})$, where id is a unique group identifier, $\mathcal{M}$ is a set of users assigned to G, K is the most restrictive k-anonymity query privacy requirement of all users assigned to G, R is the cloaked spatial region of G, and $\mathcal{Q}$ is a set of queries issued by at least one user assigned to G.

6.2 Algorithm

Our *robust* spatial cloaking algorithm has four main modules: (1) *Query registration* which is called whenever a user wants to issue a snapshot or continuous location-based query, (2) *Query termination* which is called whenever a users wants to terminate its previously issued query, (3) *Group join* which is called from the *query registration* module to find the most suitable group for the querying user, and (4) *Group leave* which is called by the user to act as a cleanup process whenever the users wants to disconnect from the system. Details of these modules as follow:

Query registration. Algorithm 1 depicts the pseudo code of the query registration module. Such module is called only when a user issues a snapshot or continuous location-based query. The goal of *query registration* is to find a suitable group G that matches the querying user location and the query privacy requirements. Then, the cloaked region for the issued query is the minimum spatial region R that includes all users in G. To avoid having mobile users lying

Algorithm 1 Robust Spatial Cloaking: Issue a Query

```
 1: procedure QueryRegistration (Query $Q$, User $U$)
 2: if $U.G$ = null then
 3:     GroupJoin($U$) (See Algorithm 3)
 4: else
 5:     $G \leftarrow U.G$
 6:     if $|G.\mathcal{M}| > U.K_q$ then
 7:         if $Q \notin G.\mathcal{Q}$ then
 8:             $G.\mathcal{Q} \leftarrow G.\mathcal{Q} \cup \{Q\}$
 9:             Send $Q$ to the database server as cloaked region $G.R$
10:         end if
11:     else
12:         GroupLeave($U$)
13:         GroupJoin($U$) (See Algorithm 3)
14:     end if
15: end if
```

on the cloaked area boundary, the minimum spatial region R is aligned to a certain grid. The *query registration* module starts by checking the status of the querying user (Line 2 in Algorithm 1). If the querying user is *ungrouped*, i.e., does not belong to any current group ($U.G = null$), then we call the *Group join* module to find the suitable group for the user (Line 3 in Algorithm 1). Then, the algorithm terminates as the query cloaked region would be computed from the group that the user will join. On the other side, if the querying user is already in the *grouped* state (i.e., belongs to a group G), we check if the current user group does satisfy the user query privacy requirement, i.e., the number of users within G ($|G.\mathcal{M}|$) is greater than the user query anonymity ($U.K_q$) (Line 6 in Algorithm 1). If this is the case, then we check if the user query is already registered by some other users in the same group. If this the case, we do nothing as the current user can share the query answer with other users. However, if the issued query is a new one, we add it to the current outstanding queries of G and send it as cloaked region to the database server (Lines 7 to 10 in Algorithm 1). Finally, if the current user group G does not satisfy the user query privacy requirements, the user has to leave G and joins another group that would be more suitable to the query privacy requirement (Lines 12 to 13 in Algorithm 1). In this case, the query cloaked region will be produced from the new group that the user will join.

Query termination. Algorithm 2 depicts the pseudo code of the query termination module. Such module is called when a user decides to terminate its outstanding continuous query or when the result of the snapshot query is received. The main idea of the algorithm is to update the user and group information with respect to the terminated query, and unregister the query if there are no other group members that are interested in this query. Such process is done in two phases. In the first phase, we clean the group information while in the second phase we clean the user information. For the first phase, we start by checking if there are any other group members in G who are interested in the terminated query Q (Line 3 in Algorithm 2). If this is not the case, we remove Q from the list of outstanding queries in G. Also, we have to unregister Q from the database server. To do such process safely, we set a random timer that upon

Algorithm 2 Robust Spatial Cloaking: Terminate a Query

```
 1: procedure QueryTermination (Query $Q$, User $U$)
 2: $G \leftarrow U.G$
 3: if no other queries in $G$ are interested in $Q$ then
 4:     $G.\mathcal{Q} \leftarrow G.\mathcal{Q} - \{Q\}$
 5:     Set a random timer; when it expires, unregister $Q$ with the database server
 6:     if $G.\mathcal{Q}$ is empty then Annihilate $G$, and mark all users in $G$ as ungrouped; return;
 7: end if
 8: $U.\mathcal{Q} \leftarrow U.\mathcal{Q} - \{Q\}$
 9: if $U.\mathcal{Q}$ is empty then
10:     $G.\mathcal{M} \leftarrow G.\mathcal{M} - \{U.id\}$; $U.G \leftarrow$ null; $G.K \leftarrow \mathsf{max}(\forall\ U_i.K_q, U_i \in G.\mathcal{M} \wedge U_i.\mathcal{Q} \neq \{\emptyset\})$;
11:     if $|G.\mathcal{M}| > G.K$ then
12:         Determine a centroid of all querying users in $G$
13:         Remove $|G.\mathcal{M}| - G.K$ non-querying members that are furthest away from the centroid
14:     end if
15: end if
```

expiration we can safely unregister Q from the server. The key idea of using a random timer is to mix the order of some time-related events in order to avoid an adversary linking the query termination event to some other events, e.g., a user leaves a group or the system (Lines 4 to 5 in Algorithm 2). After terminating Q, if there are no more querying users in G, we annihilate the group G by marking all group members as *ungrouped* while updating their tuples accordingly (Line 6 in Algorithm 2). The second phase (cleaning user information) is invoked only if group G is still outstanding. In this phase, we start by removing the terminated query Q from the list of outstanding queries associated with the user U (Line 8 in Algorithm 2). Then, if the user has no more outstanding queries, we remove the user from the group, mark the user as an *ungrouped* user, and update the group privacy information $G.K$ to be the current maximum privacy requirements of all querying users within G (Line 10 in Algorithm 2). Since, we are updating the maximum group privacy requirement $G.K$, we may end up in having $G.K$ less than the number of current user in G ($|G.\mathcal{M}|$). In this case, G is considered to have an additional $|G.\mathcal{M}| - G.K$ users than what it needs. So, we aim to to release all additional group members as this would mainly reduce the group region area $G.R$ and in the same time allow released users to either form new groups or join other existing groups that could be more suitable to their privacy requirements. To do such process, we remove the $|G.\mathcal{M}| - G.K$ non-querying members that are furthest away from the centroid of all querying users in the group. (Lines 12 to 13 in Algorithm 2). The key idea of using the centroid of all querying members is to minimize the group region area $G.R$ with respect to querying members. Minimizing a group region area would result in better accuracy in the query answer reported from the database server.

Group join. Algorithm 3 depicts the pseudo code of the group join operation. The key idea of this module is to find a group for a user that is suitable to the user query privacy requirement. We start by finding a set of groups $\mathcal{G}$ covering the user location, and then sort them by their group region area in an increasing order (Lines 3 to 4 in Algorithm 3). The main idea of sorting based on the area is to give preference to those groups with minimum region area $G.R$

Algorithm 3 Robust Spatial Cloaking: Group Join

```
 1: procedure GroupJoin (User $U$)
 2: $\mathcal{G}' \leftarrow \{\emptyset\}$;
 3: $\mathcal{G} \leftarrow$ all existing groups $G$ that cover the user location, i.e., $U.L \in G.R$
 4: Sort $\mathcal{G}$ by the area of $G.R$ in an increasing order
 5: for each group $G$ in $\mathcal{G}$ do
 6:     if $|G.\mathcal{M}| + 1 \geqslant U.K_q$ then
 7:         $\mathcal{G}' \leftarrow \mathcal{G}' \cup \{G\}$
 8:         if $U.\mathcal{Q} \in G.\mathcal{Q}$ then $G.\mathcal{M} \leftarrow G.\mathcal{M} \cup \{U.id\}$; return
 9:     end if
10: end for
11: if $\mathcal{G}' \neq$ null then
12:     $G \leftarrow$ the first group in $\mathcal{G}'$
13:     $G.\mathcal{M} \leftarrow G.\mathcal{M} \cup \{U.id\}$; $G.\mathcal{Q} \leftarrow G.\mathcal{Q} \cup \{U.\mathcal{Q}\}$
14:     Send the query in $U.\mathcal{Q}$ to the database server as cloaked region $G.R$
15: else
16:     if the number of ungrouped users $< U.K_q$ then
17:         Suspend the request for a certain period of time
18:         Go to Line 2
19:     end if
20:     Construct a new group $G$
21:     $U.G \leftarrow G; G.\mathcal{M} \leftarrow \{U.id\}$; $G.\mathcal{Q} \leftarrow \{U.\mathcal{Q}\}$; $G.K \leftarrow U.K_q$
22:     Add $G.K$ ungrouped users that are closest to $U.L$ into $G$
23:     Send the query in $U.\mathcal{Q}$ to the database server as cloaked region $G.R$
24: end if
```

Then, we join the user to a group G based on following prioritized cases with the first one is the highest priority while the last one is the lowest priority:

1. If there is a group $G \in \mathcal{G}$ satisfying the user's query privacy requirement, i.e., $|G.\mathcal{M}| + 1 \geqslant U.K_q$ and the user's required query has already registered in G, i.e., $U.\mathcal{Q} \subseteq G.\mathcal{Q}$, we simply assign the user U to G. Notice that, due to the pre-sorting step, if there are several groups with this property, we pick the group G with the minimum are $G.R$ (Lines 5 to 10 in Algorithm 3).
2. If there is a group $G \in \mathcal{G}$ satisfying the user's query privacy requirement, i.e., $|G.\mathcal{M}| + 1 \geqslant U.K_q$ but does not have the query Q among its query list, we assign U to G. In this case, we would need to register Q with G and send Q to the database server as the cloaked region $G.R$. Notice that if there multiple groups G, we would select the one with the minimum area $G.R$ (Lines 5 to 10 in Algorithm 3).
3. Otherwise, we check whether there are enough number of *ungrouped* users to construct a new group for the user. In case that the number of *ungrouped* users is less than the user's query privacy requirement, we suspend the request a prescribed period of time, and then it restarts (Lines 17 to 18 in Algorithm 3). On the other side, if we are able to construct a new group G for the user U, we add $G.K$ (that is equal to $U.K_q$) *ungrouped* users that are closest to the user's location to G, in order to satisfy the user's query privacy requirement (Lines 20 to 22 in Algorithm 3). The reason of adding nearby *ungrouped* users to G is to minimize the group region size. Finally, the algorithm registers the query with a database server with the cloaked region $G.R$ from by the locations of the new group members.

Group leave. Algorithm 4 depicts the pseudo code of the group leave module. Such module is executed when a user U decides to leave the system. The first

Algorithm 4 Robust Spatial Cloaking: Group Leave

```
1: function GROUPLEAVE (User $U$)
2: // $U$ is a tuple of a leaving user
3: $G \leftarrow U.G$
4: if $U.\mathcal{Q}$ is not empty then
5:     for each $Q \in U.\mathcal{Q}$ do
6:         QUERYTERMINATION ($Q$, $U$)
7:     end for
8: end if
9: $U.G \leftarrow$ null; $G.\mathcal{M} \leftarrow G.\mathcal{M} - \{U.id\}$;
10: if $|G.\mathcal{M}| < G.K$ then
11:     Determine a centroid of all querying users in $G$
12:     Add *one ungrouped* users that is closest to the centroid into $G$
13: end if
```

thing to do when is to go through all outstanding queries of U and terminate them one by one (Lines 4 to 7 in Algorithm 4). Then, we set the user group to *null* and remove the user from its current group. It may happen that removing such user would reduce the number of users in the group ($|G.\mathcal{M}|$) to be less than the most restrictive query privacy requirement ($G.K$). If this is the case, we add another *ungrouped* user, if possible, that is the closest to the centroid of querying users (Lines 11 to 12 in Algorithm 4).

6.3 Proof of Correctness

In this section, we prove that the robust spatial cloaking algorithm has the privacy-preserving properties, i.e., *k-sharing region* and *memorization*, and thus is free from *query sampling* and *query tracking* adversary attacks.

Theorem 1. *The robust spatial cloaking algorithm is free from query sampling attacks.*

Proof. For each group G, the number of users in G ($|G.\mathcal{M}|$) is guaranteed to satisfy the most restrictive query privacy requirement among the querying members. As depicted in Line 22 in Algorithm 3 and Line 12 in Algorithm 4, whenever the number of users become less than the most restrictive query privacy requirement, we immediately add more users to the group. Thus, the group region satisfies the query privacy requirement of all querying members. Since we only report the group region $G.R$ as the location information of all outstanding snapshot or continuous queries, so the group region is always shared by all group members. Therefore, the algorithm possesses the *k-sharing region* privacy-preserving property, and it is free from query sampling attacks. □

Theorem 2. *The robust spatial cloaking algorithm is free from query tracking attacks.*

Proof. Let S be the set of members located within the group region $G.R$ at the time when a query Q is initially registered with a database server. If there is no member admission or leave, the subsequent $G.R$ must contain all members in S at time t_i. Thus, the algorithm has the memorization privacy-preserving property.

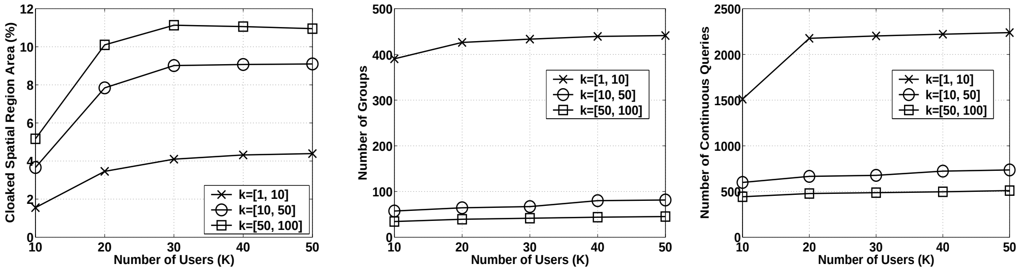
A member admission does not affect the memorization property, because $G.R$ must still contain all members in S. Since each query in a group is only registered with a database server once, even though it is issued by multiple group members. Thus, the memorization property can be generalized to remember all members in a group. For example, in Figure 2(b), at time t_i, the group region contains users A to E, and at time t_{i+1}, the group region contains users A, B, F, G, and H. The adversary cannot link these two snapshots to infer that the querying user is either user A or B. This is because users F, G, and H may be interested in Q, and thus the algorithm does not deregister Q. Also, users C, D, or E could be also the querying user, since the algorithm will deregister a query after a random period time. Thus, the group with member leave implicitly has the memorization property. □

7 Experimental Result

In this section, we experimentally evaluate the robust spatial cloaking algorithm, in terms of scalability and privacy requirements. In all experiments of this section, we use the Network-based Generator of Moving objects [22] to generate a set of moving objects. The input to the generator is the road map of Hennepin County in Minnesota, USA. The output of the generator is a set of moving objects that move on the road network of the given map. We consider continuous nearest-neighbor queries for a randomly selected object type, i.e., "continuously report my nearest *object*". Furthermore, we consider three performance metrics, *cloaked spatial region area*, *number of continuous queries*, and *number of groups*. The *cloaked spatial region area* metric is defined as the ratio of the average area of the cloaked spatial region reported to a database server to the entire system area, while the *number of continuous queries* metric is defined as the total number of query registration. The *number of groups* metric is the total number of groups created by the robust spatial cloaking algorithm.

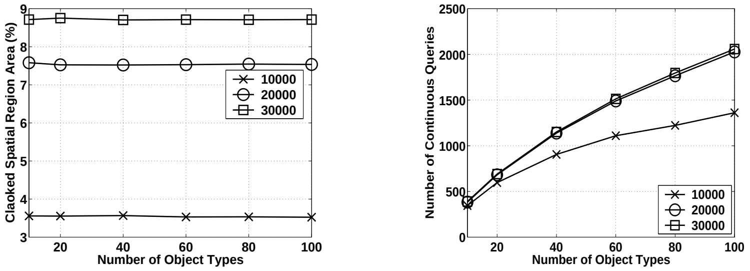
7.1 Scalability

Figures 4 and 5 give the scalability of the robust spatial cloaking algorithm with respect to varying the number of users from 10K to 50K (with three different levels of k-anonymity for queries $[1, 10]$, $[10, 50]$, and $[50, 100]$), and the number of object types from 10 to 100 (with different numbers of users 10K, 20K, and 30K), respectively. Figure 4(a) depicts that the cloaked spatial region area gets larger with increasing the number of users, but the region area is stabilized as the number of users is over 30K. When there are more users, each group generally contains more members. This fact is depicted in Figure 4(b) where the number of groups only slightly increases as there are much more users. To satisfy the memorization privacy-preserving property, the algorithm continuously adjusts a group region to cover all group members. With more members in a group, the group region tends to be larger to cover all the members; therefore, the



(a) Cloaked Region Area (b) No. of Groups (c) No. of Continuous Queries

Fig. 4. Number of users

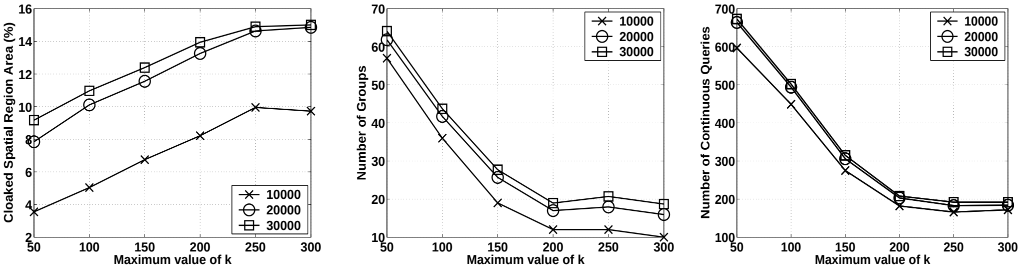


(a) Cloaked Region Area (b) No. of Continuous Queries

Fig. 5. Number of object types

group region area increases as the number of users gets larger. Furthermore, the cloaked spatial region gets larger, when the users have more restrictive privacy requirements. This is because the algorithm needs to assign more users to a group to satisfy the user's query privacy requirement. Figure 4(c) depicts that there are less number of continuous queries, as the users have more restrictive query privacy requirement. With more restrictive query privacy requirement, there are more members in each group that leads to a higher chance for them to share query answer.

Figures 5(a) and 5(b) give the system performance of our proposed robust spatial cloaking algorithm with respect to various number of object types. Figure 5(a) gives that the cloaked spatial region area is not affected by increasing the number of object types. However, the total number of issued continuous queries significantly rises with increasing the number of object types (Figure 5(b)). This is due to the fact that if there are more different object types, more members issue continuous queries for a distinct object type in a group. As a result, each group has to issue more continuous queries to a database server with the increasing number of object types.



(a) Cloaked Region Area (b) No. of Groups (c) No. of Continuous Queries

Fig. 6. Number of object types

7.2 Effect of Privacy Requirement

Figure 6 gives the effect of privacy requirements on system performance by increasing the range k_q values from $[10, 50]$, to $[10, 300]$. With more restrictive query privacy requirements, each group needs more members. Since the cloaked spatial region of a group must contain all members, the region area increases when the privacy requirement gets more restrictive (Figure 6(a)). On the other hand, with more restrictive privacy requirements, more users are assigned to the same group, so the number of groups drops, as illustrated in Figure 6(b). Since there are more members in each group, they have a higher chance to share the query answer. Thus, the total number of issued continuous queries drops (Figure 6(c)).

8 Conclusion

In this paper, we have introduced a new privacy notion in which mobile users can protect their query privacy even if their locations are revealed. Such privacy notion is crucial in many applications where users are obligated or willing to reveal their locations. We show that with such new privacy notion, existing techniques for preserving the privacy of location-based queries would fail as such techniques do not distinguish between location privacy and query privacy. Namely, we identify two privacy attacks models, *query sampling* and *query tracking* that take place upon distinguishing between location privacy and query privacy. Then, we outline two main properties, namely *k-sharing region* and *memorization* that if satisfied by location privacy techniques would make them resilient to the identified attack. Then, we present a *robust* spatial cloaking technique that: (1) clearly distinguishes between location privacy and query privacy, (2) supports *continuous* and *snapshot* location-based queries, (3) employs both the *k-sharing region* and *memorization* properties, hence, free from the identified attacks. Experimental results show that the robust spatial cloaking algorithm is scalable and efficient in terms of large numbers of mobile users, object types, and various privacy requirements.

References

[1] Ackerman, L., Kempf, J., Miki, T.: Wireless Location Privacy: A Report on Law and Policy in the United States, the Europrean Union, and Japan. Technical Report DCL-TR2003-001, DoCoMo Commuinication Laboratories, USA (2003)

[2] Barkhuus, L., Dey, A.K.: Location-Based Services for Mobile Telephony: a Study of Users' Privacy Concerns. In: Proceeding of the IFIP Conference on Human-Computer Interaction, INTERACT. (2003)

[3] Beresford, A.R., Stajano, F.: Location Privacy in Pervasive Computing. IEEE Pervasive Computing **2**(1) (2003) 46–55

[4] Warrior, J., McHenry, E., McGee, K.: They Know Where You Are . IEEE Spectrum **40**(7) (2003) 20–25

[5] Foxs News: Man Accused of Stalking Ex-Girlfriend With GPS. http://www.foxnews.com/story/0,2933,131487,00.html. Sep 4, 2004

[6] USAToday: Authorities: GPS System Used to Stalk Woman. htt://usatoday.com/tech/news/2002-12-30-gps-stalker_x.htm. Dec 30, 2002

[7] Voelcker, J.: Stalked by Satellite: An Alarming Rise in GPS-enabled Harassment. IEEE Spectrum **47**(7) (2006) 15–16

[8] Cheng, R., Zhang, Y., Bertino, E., Prabhakar, S.: Preserving User Location Privacy in Mobile Data Management Infrastructures. In: Proceedings of Privacy Enhancing Technology Workshop. (2006)

[9] Gedik, B., Liu, L.: Location Privacy in Mobile Systems: A Personalized Anonymization Model . In: ICDCS. (2005)

[10] Gruteser, M., Grunwald, D.: Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. In: MobiSys. (2003)

[11] Gruteser, M., Liu, X.: Protecting Privacy in Continuous Location-Tracking Applications. IEEE Security and Privacy **2**(2) (2004) 28–34

[12] Mokbel, M.F., Chow, C.Y., Aref, W.G.: The New Casper: Query Processing for Location Services without Compromising Privacy. In: VLDB. (2006)

[13] Kalnis, P., Ghinita, G., Mouratidis, K., Papadias, D.: Preserving Anonymity in Location Based Services. Technical Report TRB6/06, Department of Computer Science, National University of Singapore (2006)

[14] Sweeney, L.: *k*-anonymity: A Model for Protecting Privacy. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems **10**(5) (2002) 557–570

[15] Hong, J.I., Landay, J.A.: An Architecture for Privacy-Sensitive Ubiquitous Computing. In: MobiSys. (2004)

[16] Kido, H., Yanagisawa, Y., Satoh, T.: An Anonymous Communication Technique using Dummies for Location-based Services. In: ICPS. (2005)

[17] Duckham, M., Kulik, L.: A Formal Model of Obfuscation and Negotiation for Location Privacy. In: Pervasive. (2005)

[18] Chow, C.Y., Mokbel, M.F., Liu, X.: A Peer-to-Peer Spatial Cloaking Algorithm for Anonymous Location-based Services. In: ACM GIS. (2006)

[19] Ghinita, G., Kalnis, P., Skiadopoulos, S.: PRIVÉ: Anonymous Location-Based Queries in Distributed Mobile Systems. In: WWW. (2007, to appear)

[20] Mokbel, M.F., Chow, C.Y.: Challenges in Preserving Location Privacy in Peer-to-Peer Environments. In: Proceedings of the International Workshop on Information Processing over Evolving Networks, WINPEN. (2006. Invited paper)

[21] Mokbel, M.F., Chow, C.Y., Aref, W.G.: The New Casper: A Privacy-Aware Location-based Databse Server (Demonstration). In: ICDE. (2007, to appear)

[22] Brinkhoff, T.: A Framework for Generating Network-Based Moving Objects. GeoInformatica **6**(2) (2002) 153–180