ISSN: 2277-5528

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES
&
MANAGEMENT

Defend seclusion against Location based personal recognition in mobile services

A.Nagarjuna ,PG student ,QCET,Nellore,sunnyavadi@gmail.com
CH.Subbarao,AssociateProfessor,QCET,Nellore,chepurusubbarao@gmail.com
P.Babu, Associate professor,QCET,Nellore, babu123mca@gmail.com

Abstract—Privacy protection has recently received considerable attention in location-based services. A large number of location cloaking algorithms have been proposed for protecting the location privacy of mobile users. In this paper, we consider the scenario where different location-based query requests are continuously issued by mobile users while they are moving. We show that most of the existing k-anonymity location cloaking algorithms are concerned with *snapshot* user locations only and cannot effectively prevent location-dependent attacks when users' locations are continuously updated. Therefore, adopting both the location k-anonymity and cloaking granularity as privacy metrics, we propose a new incremental clique-based cloaking algorithm, called ICliqueCloak, to defend against location-dependent attacks. The main idea is to incrementally maintain maximal cliques needed for location cloaking in an undirected graph that takes into consideration the effect of continuous location updates. Thus, a qualified clique can be quickly identified and used to generate the cloaked region when a new request arrives. The efficiency and effectiveness of the proposed ICliqueCloak algorithm are validated by a series of carefully designed experiments. The experimental results also show that the price paid for defending against location-dependent attacks is small.

Index Terms—Location privacy, mobile data management, location-based services.

1 INTRODUCTION

WITH advances in wireless communication and mobile positioning technologies, location-based services (LBSs) have been gaining increasingly popularity in recent years. This is evident from a recent report from ABI Research, which forecasts that LBS revenue is expected to reach an annual global total of \$13.3 billion by 2013 [2]. But on the other hand, the privacy threat of revealing a mobile user's personal information through his/her location has become a key issue to be concerned. For example, the EU Commission has regulated the use of location data in its Directive on Privacy and Electronic Communications.[1]

A lot of research has been conducted concerning how to enjoy location-based services while protecting the location privacy of mobile users [12], [13], [19], [25], [29], [31]. For example, using her PDA phone, Alice wants to find out "the nearest hospital with specialty in ophthalmology" while hiding her exact location (e.g., being in a clinic or at home) and the sensitive information that it is her (Alice) who made this query. A straightforward method is to replace her identity with a pseudonym before sending the query to the service provider. But this is not enough. Location information included in the query can be used as a *quasi-identifier* to re-identify the user [20], [26], [33], [34]. Suppose the query was issued from Alice's home; it can then be linked to Alice with some background knowledge (e.g., telephone directory). We consider the location privacy is under threat when an adversary can obtain unauthorized access to raw location data and sensitive information due to location disclosing [28].

To address the location privacy issue, *location k-anonymity* [19] and *cloaking granularity* [31] are two commonly used privacy metrics:

- **Location k-Anonymity**. A mobile user is considered location k-anonymous if and only if the location information sent to the service provider is made indistinguishable from that of at least k-1 other users. To achieve location k-anonymity, exact user locations are extended to cloaked regions such that each region covers at least k users.
- **Cloaking Granularity**. It requires the area of cloaked region to be larger than a user-specified threshold.[2]

While the location k-anonymity protects the user identity (out of k users), it may not be able to prevent the location disclosure (e.g., a cloaked region covering k users in populated areas could be very small). On the other hand, the cloaking granularity prevents the location disclosure but cannot defend against attacks for user identifies in the cases where user locations are publicly known and there is only one user in the cloaked region [19].

1. It demands that location data may only be processed when it is made anonymous or with the consent of the user for the duration necessary for the provision of a service [9].

2. The location privacy can be better protected with a larger cloaking region, which however may degrade the quality of service.

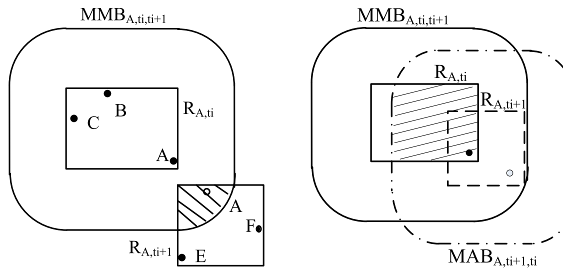


Fig. 1. Example of location-dependent attacks (k=3)

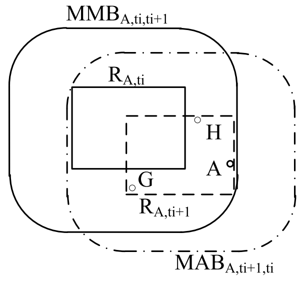


Fig. 2. ICliqueCloak algorithm (k=3)

Most of the existing privacy-aware algorithms (e.g., [13], [19], [22], [31]), which comply with location k-anonymity model, are concerned with *snapshot* user locations only. They have not considered the effect of continuous location updates. This may result in serious privacy breaches when different one-shot queries are frequently issued by a mobile user.[3] If an attacker (e.g., the service provider) can collect the historical cloaked regions of a user as well as the mobility pattern (e.g., speed limit), the location privacy of the user might be compromised. Continuing with the above example, Alice gets the address of the hospital; and at some time on the way to the hospital, Alice wants to gas up her car. Thus, she issues another query "list all the gas stations in $5km$ around me." As shown in Fig. 1(a), users A (Alice), B, and C are cloaked into region R_{A,t_i} at time t_i, when she issues the hospital query; users A, E, and F are cloaked into region $R_{A,t_{i+1}}$ at time t_{i+1}, when she asks for the gas stations. If an attacker knows the last cloaked region R_{A,t_i} and the maximum speed v_A (e.g., speed limit of the road), it can be inferred that the possible location of A at t_{i+1} should be limited to the **m**aximum **m**ovement **b**oundary (MMB) $MMB_{A,t_i,t_{i+1}}$, a round rectangle that extends R_{A,t_i} by a radius of $v_A \cdot (t_{i+1} - t_i)$. As a consequence, the attacker can deduce that A must be located in the overlapped area of $MMB_{A,t_i,t_{i+1}}$ and $R_{A,t_{i+1}}$ (i.e., the shaded area) at t_{i+1}. In the worst case, if the overlapped area is just a location point, the exact user location will be disclosed. Similarly, the user's previous location at an earlier time could also be under attack. As shown in Fig. 1(b), the location of A would be limited to the intersection area of the **m**aximum **a**rrival **b**oundary (MAB) MAB_{A,t_{i+1},t_i} and R_{A,t_i}, where MAB_{A,t_{i+1},t_i} is a round rectangle that extends $R_{A,t_{i+1}}$ by a radius of $v_A \cdot (t_{i+1} - t_i)$. We call the above types of attacks *location-dependent attacks*.

Location-dependent attacks have been studied in some previous work [7], [14], [37]. However, the prior solutions in [7], [14], [37] only considered the cloaking granularity as the privacy metric, which, as discussed earlier, may fail to protect the user identity in case there is only one user in the cloaked region. Thus, in this paper, we adopt *both* the cloaking granularity and location k-anonymity as privacy metrics. We propose a new location cloaking algorithm, called *ICliqueCloak*, to incorporate the effect of continuous location updates in the process of location cloaking. As illustrated in Fig. 2, at time t_{i+1}, the cloaking algorithm is aware of R_{A,t_i} and $MMB_{A,t_i,t_{i+1}}$, and attempts to find the cloaked region for A within $MMB_{A,t_i,t_{i+1}}$. Supposing that G and H are found in $MMB_{A,t_i,t_{i+1}}$ at t_{i+1}, A, G, and H can form a cloaking set and generate a cloaked region $R_{A,t_{i+1}}$. Thus, even if the attacker knows each user's speed limit, he/she still cannot tell the exact location of A in R_{A,t_i} and $R_{A,t_{i+1}}$.

3. It is noted that the contents of the queries issued at different time instances might be different. However, we assume that the pseudonym id of the user remains the same for these different queries. We will elaborate this issue in Section 3.2.

We use a graph model to formulate this problem. Each location-based query request is represented by a node in the graph; an edge exists between two nodes only if they are within the MMB of each other and can be potentially cloaked together. To meet the location k-anonymity requirement, the problem becomes to find k-node cliques in the graph such that all the nodes within a clique form a cloaking set. To reduce the computational complexity, we propose to maintain the *maximal cliques* incrementally. That is, all maximal cliques are identified at the beginning of the process; they are then incrementally maintained based on three classes: *positive candidates*, *negative candidates* and *non-candidates*. Thus, a qualified clique can be quickly identified and used to generate the cloaked region when a new request arrives.

We conduct a series of experiments to evaluate the performance of the proposed ICliqueCloak algorithm using both location data generated from a well-known road network simulator [36] and adapted from a real dataset [1]. Experimental results show that ICliqueCloak is efficient in terms of various performance metrics including the cloaking time, the request processing time, and the cloaking success rate, while its anonymization cost is only slightly increased in comparison with the existing algorithm.

The remainder of this paper is organized as follows. We review the related work on privacy protection in LBS in Section 2. The problem under study is formally defined in Section 3. A new efficient cloaking algorithm called ICliqueCloak is proposed in Section 4. Section 5 presents the performance evaluation results of our proposed algorithm. Finally, the paper is concluded in Section 6.

2 RELATED WORK

In this section, we review the existing work in terms of attack models. We survey the techniques for preventing snapshot location attacks, location-dependent attacks, query tracking attacks, and trajectory attacks in Sections 2.1–2.4, respectively.

2.1 Preventing Snapshot Location Attacks

When exact snapshot locations are disclosed, two kinds of attacks may happen: *location linking attacks* [19] and *query sampling attacks* [8]. *Location linking attacks* refer to the scenario where the location information included in a user query is used as a *quasi-identifier* to re-identify the user. For example, if a location exclusively belongs to some owner, the corresponding query can thus be linked to the location owner. The *location k-anonymity* model was proposed to prevent this kind of attacks [19]. The basic idea is to extend an exact user location to a cloaked region that covers at least k users. In [19], a Quad-tree-based cloaking algorithm is used to generate cloaked regions. In [31], each user can specify the minimum tolerable area of a cloaked region as well as the smallest privacy level *k*, and a variant Quad-tree is used to compute cloaked regions. More recently, [11] proposed a cloaking algorithm called *hilbASR*, which makes use of Hilbert curve to approximate the spatial proximity between query requests.

However, even if the locations are cloaked, an adversary may still be able to link a query to its user in case user locations are publicly known. For example, suppose there are three users A, B, and C, and they issue three queries Q_A, Q_B, and Q_C, respectively. Assume k=2. Following the location k-anonymity model, user A might be cloaked into a region R_1 that covers A and B, and users B and C might be cloaked into another region R_2 that covers both of them. Then, an adversary can infer that Q_A must be issued by A since only Q_A is with R_1 and only A must be covered by R_1 (B can be covered by R_1 or R_2). This kind of attacks is called *query sampling attacks* [8]. To deal with such attacks, [8] suggested employing *k-sharing regions*, i.e., a cloaked region should not only cover at least *k* users, but the region is also shared by at least *k* of these users.

In [13], a clique-based cloaking algorithm, called *CliqueCloak*, was proposed to find *k-sharing* cloaked regions. It models user-specified privacy requirements using an undirected graph and identifies the cloaking sets through finding cliques in the graph. Our work also employs a graph model to facilitate location cloaking, but differs from [13] in several aspects. First, the underlying problems are different. The clique-based cloaking algorithm in [13] is concerned with QoS support in privacy protection for snapshot locations and hence could suffer from location-dependent attacks, whereas the aim of our cloaking algorithm is to protect location privacy against location-dependent attacks. Second, the methods for finding cliques in a graph are different. In [13], every time a new request arrives, the algorithm needs to recursively search the neighbors of the node representing the request. In contrast, in our approach we incrementally maintain the maximal cliques in the graph; thus a cloaking set is found in some maximal clique instead of recursively searching the neighbors. Third, the requests with privacy levels higher than that of the newly arrived request cannot be cloaked in [13]. This limitation, however, does not exist in our approach: those requests still have a chance to be cloaked successfully as long as their k is less than the clique size.

2.2 Preventing Location-Dependent Attacks

Location-dependent attacks (illustrated in Fig. 1) are the focus of this paper. This problem was first pointed out in [7] and [10]. To prevent location-dependent attacks, [7] proposed two simple solutions, namely *patching* and *delaying*. The first solution, called *patching*, enlarges the current cloaked region to cover the last one so that the overlapped area with the MMB is at least as large as the last cloaked region. The drawback is that the size of the cloaked region would increase significantly as time evolves. The second solution, called *delaying*, suspends the request by Δt time until the MMB grows large enough to fully contain the current cloaked region. However, the user may have already changed her location and is no longer in the cloaked region at time $t_{i+1} + \Delta t$. [14] also proposed to postpone requests, and they considered the scenario where the attacker has prior knowledge about the placement of sensitive regions on a map. [10], [37] developed a mobility-aware cloaking technique by considering mobility patterns in location cloaking. However, different from our work, the privacy metric employed in these previous studies is only the granularity of cloaked regions (without considering the location k-anonymity).

Another related work is [38], which employed *entropy* of information theory to measure the location anonymity level by considering the probabilities of users being in a cloaked region. As entropy does not care whether user locations are actually different, the exact user location would be disclosed if all *k* users are at the same location. To overcome the limitations of the previous work, in this paper we employ both the cloaking granularity and location *k*-anonymity as privacy metrics.

2.3 Preventing Query Tracking Attacks

The location privacy of continuous queries has been considered in [8]. For a continuous query, the query results would be continuously returned for a designated time period (called *query lifetime*). For example, consider a sample query ”finding the nearest gas station in the next five minutes.” The query lifetime is five minutes. *Query tracking attacks* become possible if a user is cloaked with different users at different time instances during the query lifetime [8]. Consider a continuous query CQ_A issued by a user A. Suppose A is cloaked with B into a region R_{t_1} at time t_1, and cloaked with C into a region R_{t_2} at time t_2. Then, by linking the snapshots at time t_1 and t_2, an adversary may infer that CQ_A must be issued by A if CQ_A is the only common query between these two snapshots and A is the only user residing in both R_{t_1} and R_{t_2}. In order to defend against query tracking attacks, an idea is to exploit the *memorization* property — the same set of users should always be cloaked together during the query lifetime [8]. Clearly, query tracking tracks are different from the location-dependent attacks considered in this paper. Even if the users are prevented from query tracking attacks by applying the *memorization* property to each continuous query, they may still suffer from location-dependent attacks. Suppose two different continuous queries are issued by a mobile user at some time interval. If the user is cloaked with different sets of users for these two continuous queries, the location-dependent attack shown in Fig. 1 may still happen. On the other hand, if the user is always cloaked with the same set of users, the cloaked region would eventually expand to the whole service region when the users move apart and issue more and more queries over time.

2.4 Preventing Trajectory Attacks

When a trajectory is published, the owner might be inferred by attackers, even though the identifier has been removed. This type of attacks is called *trajectory attacks*. The problem of trajectory anonymization is to publish trajectories in such a way that the anonymity of each trajectory is preserved, while the utility of published data is maximized. The existing work can be classified into two categories: trajectory anonymization in free space [3], [5], [15], [17], [21], [35], [39], [40] and in constrained space [18], [27]. Our work also applies to free space, but differs from the trajectory anonymization problem. In addition, trajectory anonymization is typically an off-line process. In contrast, location cloaking considered in this paper is an online process that is invoked during processing of location-based queries. Moreover, trajectory anonymization requires a series of locations on a trajectory to be cloaked all at once. But in our location cloaking problem, user locations are cloaked on the fly along with new requests. Therefore, existing methods for anonymizing trajectories are not applicable to our problem.

3 PRELIMINARIES

In this section, we formally define the problem under study. Section 3.1 describes the system architecture. The privacy model and the cloaking set are defined in Sections 3.2 and 3.3, respectively.

3.1 System Architecture

Like most existing work [4], [16], [24], [31], we consider a system consisting of many mobile users, a trusted anonymizing proxy, and an un-trusted service provider (see Fig. 3). A mobile user sends location-based query requests (e.g., “finding the nearest gas station”), in the form of (id, l, $\mathcal{P}$, q), to the anonymizing proxy through an authenticated and encrypted connection, where id is the real user identity, l is the user location, $\mathcal{P}$ contains the privacy parameters (to be detailed in Section 3.2), and q represents the query content.[4]

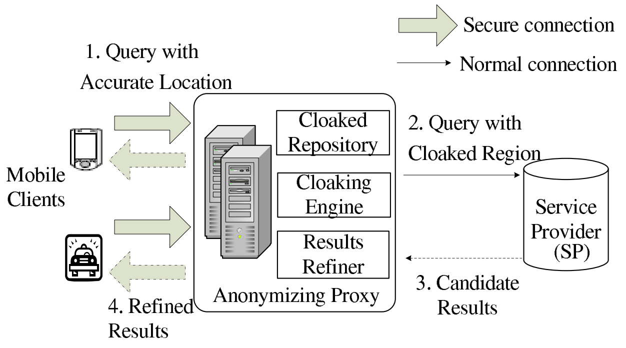


Fig. 3. System architecture

The anonymizing proxy consists of a cloaking engine, a cloaked repository, and a results refiner. Upon receiving a location-based query request, we check whether the user had made any queries before. If this is a first-time user, the cloaking engine replaces the user's id with a new pseudonym id'; otherwise it replaces id with the user's existing pseudonym id'. Next, the query request is forwarded to the cloaking process to generate a cloaked region R in accordance with the user's privacy requirements. When the cloaking succeeds, the cloaked request is saved in the cloaked repository in the form of (id, id', $\mathcal{P}$, R_{t_i}, t_i), where R_{t_i} is the user's cloaked region at time t_i. Afterwards, the anonymizing proxy forwards the modified query request (id', R, q) to the service provider.

On the service provider side, upon receiving a location-based query request (id', R, q), it will search and return all candidate results that are potentially query results of some location point within R. After that, these candidate results will be further refined by the anonymizing proxy using the user's exact location l. Finally, the refined results will be securely returned to the mobile user. In this paper, we focus on the location cloaking algorithm, which is concerned with how to extend a location l to a cloaked region R without violating user-specified privacy requirements.

3.2 Privacy Model and Attacks

We now define the privacy model.

Definition 1. ((k, A, dt)-Location Privacy Model*). In order to accommodate personalized privacy requirements, each user can specify three parameters for protecting the location privacy [13], [19], [31]:*

4. We assume in this paper that a new request is not issued until the last one is serviced. In other words, at each time instance, a user can be associated with only one query request. Therefore, we do not distinguish ”request”, ”query”, and ”user” whenever they are clear in the context.

- *k: It represents the anonymity level in the location k-anonymity model. More specifically, each cloaked region should cover at least k different users. The larger is the value of k, the more privacy is offered.*
- *A_{min}: It specifies the minimum area that a cloaked region should have. This is to prevent the cloaked region from being too small for highly populated areas.*
- *dt: It is the maximum tolerable cloaking delay, which is a QoS parameter. The larger is the dt value, the worse is the service quality, since the user will have a higher chance of moving away from the location where the query was issued.*

It is noted that the maximum area of a cloaked region could also be used as a QoS parameter. Nevertheless, to simplify our privacy model, we do not require limiting the maximum area of a cloaked region, but instead use the area of the cloaked region as a measure of the anonymization cost (see Definition 6). As will be shown in Section 5.7, the computed cloaked regions are of acceptable size (1-4 km^2 in most cases).

Before giving the formal definition of location-dependent attacks, we elaborate some assumptions for the attacker.

Definition 2. *(*Knowledge of the Attacker*). Any party owning the following knowledge can be a potential attacker:*
- *a set of historical cloaked regions;*
- *the maximum moving speed of the user.*

In the previous example shown in Fig. 1(a), the service provider could be a potential attacker since it knows cloaked regions R_{A,t_i} and $R_{A,t_{i+1}}$, as well as the maximum movement speed of the user, which might be inferred from the speed limit of the road and/or the user type. For example, if the user is driving, the speed may not exceed 150 *km/h*; and if the user is walking, the speed cannot be higher than 5 *km/h*.

Definition 3. *(*Location-dependent Attacks*). Assume that*
- *R_{u,t_i} and R_{u,t_j} are user u's cloaked regions at times t_i and t_j, respectively;*
- *the maximum speed of user u is v_u.*

*The **m**aximum **m**ovement **b**oundary (MMB_{u,t_i,t_j}) of u at t_j is a round rectangle that extends R_{u,t_i} by a radius of $v_u \cdot (t_j - t_i)$. Denote by MM_{u,t_i,t_j} the intersection area between R_{u,t_j} and MMB_{u,t_i,t_j}:*

$$MM_{u,t_i,t_j} = MMB_{u,t_i,t_j} \bigcap R_{u,t_j}.$$

*Similarly, the **m**aximum **a**rrival **b**oundary (MAB_{u,t_j,t_i}) of u at t_j is a round rectangle that extends R_{u,t_j} by a radius of $v_u \cdot (t_j - t_i)$. Denote by MA_{u,t_j,t_i} the intersection area between R_{u,t_i} and MAB_{u,t_j,t_i}:*

$$MA_{u,t_j,t_i} = MAB_{u,t_j,t_i} \bigcap R_{u,t_i}.$$

If any inequality below holds:
- *$MM_{u,t_i,t_j} \neq R_{u,t_j}$ for any t_i and t_j,*
- *$MA_{u,t_j,t_i} \neq R_{u,t_i}$ for any t_i and t_j,*

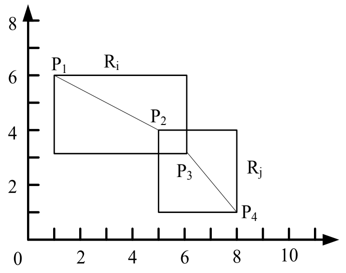


Fig. 4. MaxMin Distance

the location privacy of u might be compromised. This attack is termed as location-dependent attack.

In the previous example of Fig. 1(a), $MM_{A,t_i,t_{i+1}}$ of user A is the shaded area. Since $MM_{A,t_i,t_{i+1}} \neq R_{A,t_{i+1}}$, A could suffer from location-dependent attacks. For another example in Fig. 1(b), since the shaded area $MA_{A,t_{i+1},t_i} \neq R_{A,t_i}$, the location of A at time t_i may also be disclosed.

We remark that, in order to protect the user from location-dependent attacks, the anonymizing proxy may assign different pseudonym ids for different queries of the same user such that the attacker cannot correlate consecutive cloaked regions. However, this is not favorable in practice since it would disable the provisioning of personalized services and complicate the billing of service charges [5], [30]. On the other hand, even without knowing the identities, the attacker is still able to identify the requests from the same user by employing muti-target tracking (MTT) [32] or clustering based on query interest and spatial locality. Therefore, in this paper, we simply assume that a user is always assigned the same pseudonym id for all the queries the user issued.

3.3 Cloaking Set

Definition 4. *(*MaxMin Distance*). Let R_i and R_j be two cloaked regions. The MaxMin distance from R_i to R_j is defined as:*

$$MaxMinD(R_i, R_j) = \max_{p \in R_i} \min_{q \in R_j} distance(p, q).$$

$MaxMinD(R_i, R_j)$ implies the maximum distance between a point $p \in R_i$ and its closest point $q \in R_j$. For regions R_i and R_j in Fig. 4, MaxMinD(R_i, R_j)=$|P_1P_2|$=$2\sqrt{5}$ and MaxMinD(R_j, R_i)=$|P_4P_3|$=$2\sqrt{2}$.

From Definition 3, if R_j (R_i) is free of location-dependent attacks, it should be fully covered by MMB_{u,t_i,t_j} (MAB_{u,t_j,t_i}), which indicates MaxMinD(R_i, R_j) $\leq v_u \cdot (t_j - t_i)$ (MaxMinD(R_j, R_i) $\leq v_u \cdot (t_j - t_i)$). Therefore, we define cloaking set as follows:

Definition 5. *(*Cloaking Set*) Let CS be a user set and its **m**inimum **b**ounding **r**ectangle (MBR) be R_{u,t_i} at time t_i. Denote the previous cloaked region of each user u by $R_{u,t_{i-1}}$. CS is a cloaking set if and only if for any user $u \in CS$,*

1) *$MaxMinD(R_{u,t_i}, R_{u,t_{i-1}}) \leq v_u \cdot (t_i - t_{i-1})$;*

2) *MaxMinD*$(R_{u,t_{i-1}}, R_{u,t_i}) \leq v_u \cdot (t_i - t_{i-1})$;
3) *the privacy level* $k_u \leq |CS|$;
4) *the minimum area* $A_{min_u} \leq Area(MBR(CS))$.

The first two conditions ensure that the cloaked region at any time is free of location-dependent attacks; the third condition is to protect the user identity by following the k-anonymity requirement; and the fourth condition ensures that the area of the cloaked region is not too small in a populated area. Taking the user set $\{A, G, H\}$ in Fig. 2 as an example. $R_{A,t_{i+1}}(R_{A,t_i})$ is totally covered by $MMB_{A,t_i,t_{i+1}}(MAB_{A,t_{i+1},t_i})$; thus the first two conditions in the above definition are satisfied. Suppose k_A=k_G=k_H=3 and $A_{min_A} = A_{min_G} = A_{min_H} = 0$. Thus, the last two inequations are also established. Therefore, $\{A, G, H\}$ can be a cloaking set and its MBR $R_{A,t_{i+1}}$ is the cloaked region.

Similarly to the existing work [6], [8], [23], we employ the average area of the cloaked region for each query as a measure for anonymnization cost.

Definition 6. (Anonymization Cost) *Consider a series of cloaking sets CU. For any cloaking set $U \in CU$, its total anonymization cost is the product of the MBR area of U and the number of users in U:*

$$Total\_Cost(U) = Area(MBR(U)) \times |U|.$$

Thus, the average anonymization cost of CU is:

$$Average\_Cost(CU) = \frac{\sum_{U \in CU} Total\_Cost(U)}{\sum_{U \in CU} |U|}.$$

In this paper, we consider cloaking of the current user location with respect to the last cloaked region concerning location-dependent attacks. Earlier cloaked regions are not considered, as prior work [14], [37] has proved that the location-disclosure safety property is transitive.

4 ICLIQUECLOAK ALGORITHM

In this section, we first give the overview of ICliqueCloak algorithm in Section 4.1, and then elaborate each step of the algorithm in Sections 4.2–4.5.

4.1 Overview of ICliqueCloak

Intuitively, we want to use the MaxMin distance to find the safe cloaked region R_{u,t_i} for a newly arrived request u at time t_i. But on the other hand, we require R_{u,t_i} as the input in computing the MaxMin distance. Therefore, the basic idea of our algorithm is to find a candidate cloaking set first, and then extend some sides of the cloaked region until every user in the cloaking set is free of location-dependent attacks.

The proposed ICliqueCloak algorithm involves four main steps. First, upon the arrival of a new request u, the existing requests that are in u's MMB and vice versa are detected and modelled in an undirected graph.

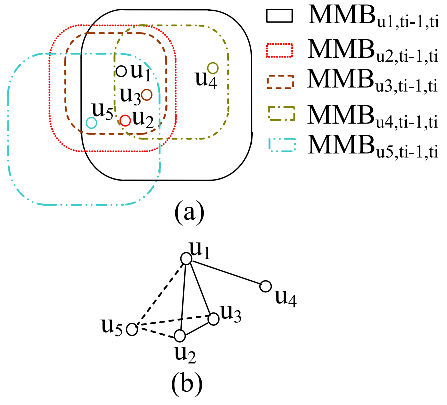


Fig. 5. Illustration of graph model

Then, a cloaking set that satisfies location k-anonymity, if any, is found from the undirected graph, and the MBR of the cloaking set is considered a candidate cloaked region. Next, the candidate cloaked region is checked whether it needs to be adjusted in order to prevent from location-dependent attacks. Finally, the graph will be updated accordingly if the cloaking is successful or some request(s) are found expired.

Now, let's formally define the graph model.

Definition 7. (Graph Modeling). *Let $G(V, E)$ be an undirected graph where V is a set of nodes representing the users who submitted location-based query requests, and E is a set of edges. Assume that the last cloaked region of user u is $R_{u,t_{i-1}}$ at t_{i-1}, and that the current time is t_i. There exists an edge e_{uw} between two nodes u and w, if and only if*

- $u \neq w$,
- *u is covered by MMB_{w,t_{i-1},t_i},*
- *w is covered by MMB_{u,t_{i-1},t_i}.*

These three conditions collectively ensure that the first condition of Definition 5 is satisfied as they have different ids and are within each other's MMB.

To find cloaking sets satisfying location k-anonymity in a graph $G(V, E)$, which is the third condition in Definition 5, it has been shown in [13] that this problem is equivalent to the problem of finding k-node cliques (i.e., k-node complete subgraphs) in the graph. Once a k-node clique is found, all the users within the clique may form a candidate cloaking set CS_{t_i} and the MBR of their locations can be used as the candidate cloaked region CR_{t_i}. Then, for each user $u \in CS_{t_i}$, compute its MaxMinD$(R_{u,t_{i-1}}, CR_{t_i})$. If the second condition of Definition 5 is violated, CR_{t_i} is enlarged until its new MAB covers $R_{u,t_{i-1}}$. The overview of ICliqueCloak is given in Algorithm 1.

We use a simple example to illustrate the algorithm. Fig. 5(a) shows four requests u_1, u_2, u_3, u_4 from different users with their MMB_{u_i,t_{i-1},t_i} at time t_i. Each MMB_{u_i,t_{i-1},t_i} is extended from the previous cloaked region $R_{u_i,t_{i-1}}$. We can see that u_1 is covered by MMB_{u_2,t_{i-1},t_i} and MMB_{u_3,t_{i-1},t_i}, and that u_2 and u_3 are both covered by MMB_{u_1,t_{i-1},t_i}. Thus, an edge exists between u_1 and u_2, as well as between u_1 and u_3.

```
Algorithm 1 Overview of ICliqueCloak
Input: a set of requests awaiting for anonymization, a new query request $u$
Output: a set of cloaked requests
1: Step 1: incrementally update the max-clique set for the new request $u$ (Section 4.2)
2: Step 2: find the cloaking set $CS_{t_i}$ satisfying location $k$-anonymity from the max-clique set (Section 4.3)
3: Step 3: generate the cloaked region for $CS_{t_i}$ (Section 4.4)
4: Step 4: update the max-clique set upon request cloaking or expiration (Section 4.5)
```

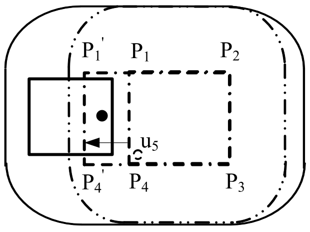


Fig. 6. Illustration of extending cloaked region

Similarly, an edge exists between u_1 and u_4, as well as between u_2 and u_3, as shown in Fig. 5(b). Now suppose that a new request u_5 arrives; new edges between u_1 and u_5, u_2 and u_5, and u_3 and u_5 are added to the graph (dashed lines in Fig. 5(b)). As a result, a clique $\{u_1, u_2, u_3, u_5\}$ can be found. Assume that $k_{u_1} = k_{u_2} = k_{u_3} = k_{u_4} = 4$ and $k_{u_5} = 3$. The size of the clique is 4, which is no smaller than any k value. Meanwhile, assuming $Area(MBR(\{u_1, u_2, u_3, u_5\})) \geq \max_{u \in \{u_1,u_2,u_3,u_5\}} A_{min_u}$, the MBR of $\{u_1, u_2, u_3, u_5\}$ can be a candidate cloaked region CR_{t_i} for this set of users.

Next, for each user in $\{u_1, u_2, u_3, u_5\}$, the algorithm checks whether the user's previously cloaked region at time t_{i-1} is covered by MAB of CR_{t_i}. For simplicity, we take user u_5 as an example. As shown in Fig. 6, the rectangle with solid lines is the cloaked region $R_{u_5,t_{i-1}}$ of u_5 at time t_{i-1}, and the rectangle with dotted-dashed lines (i.e., $P_1P_2P_3P_4$) is its cloaked region R_{u_5,t_i} at t_i. The round rectangle with dotted-dashed lines is the $MAB_{u_5,t_i,t_{i-1}}$ of u_5 from t_i to t_{i-1}. From the figure, we can see that $MAB_{u_5,t_i,t_{i-1}}$ cannot fully cover $R_{u_5,t_{i-1}}$. As such, u_5 would suffer from location-dependent attacks. Therefore, the cloaked region $P_1P_2P_3P_4$ is extended, where the edge of P_1P_4 moves to $P_1'P_4'$. As a result, $R_{u_5,t_{i-1}}$ can be fully covered by the new MAB (the round rectangle with solid lines). Further assuming that the new cloaked region $P_1'P_2P_3P_4'$ is still in each user's MMB, it will then be returned as the cloaked region for $\{u_1, u_2, u_3, u_5\}$. Finally, u_1, u_2, u_3, and u_5 are removed from the graph.

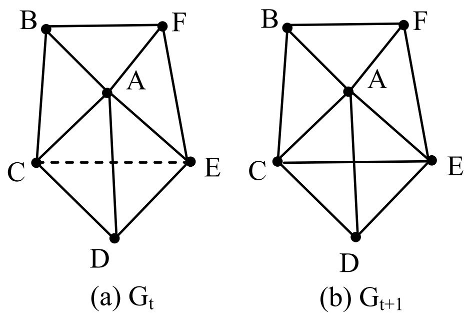


Fig. 7. Example of adding a new edge

4.2 Incremental Maximal Cliques

In this and subsequent subsections, we detail each step of Algorithm 1. To find a candidate cloaking set in the graph upon the arrival of a new request, the cloaking algorithm proposed in [13] exhaustively searches the graph for cliques covering the new request. In the following, we present a new more efficient cloaking algorithm based on incremental maintenance of maximal cliques.

For a graph without any edges, each node is a 1-node clique. Denote this set of maximal cliques by $MCSet$. $MCSet$ will then be dynamically maintained with more edges added. To facilitate our presentation, we first give a few preliminaries.

Definition 8. (t-Parameterized Graph). *Consider an undirected graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. Define $G_0 = (V, \emptyset)$. Traversing each edge $e_{uw} \in E$, G can be parameterized as*

$$G_t = (V, E_t),$$

where $t = 1, 2, \ldots, |E|, E_t$ is the set of edges visited so far, $E_t - E_{t-1} = e_{uw}$.

For a t-parameterized graph G_t, let C_t be the set of maximal cliques and C_{u_t} be the set of maximal cliques which contain node u. Before a new edge e_{uw} is added, the cliques in C_t can be partitioned into three classes:

- the cliques containing node u (C_{u_t});
- the cliques containing node w (C_{w_t});
- the cliques containing neither u nor w.

It is easy to see that adding edge e_{uw} to the graph can alter only the maximal cliques in C_{u_t} or C_{w_t}.

Theorem 1. *The maximal cliques which contain neither u nor w are still maximal after edge e_{uw} is added.*

Taking Fig. 7 as a running example, there are six nodes in the graph G_t, $V = \{A, B, C, D, E, F\}$. The set of maximal cliques C_t is $\{\{A, B, C\}, \{A, E, F\}, \{A, C, D\}, \{A, D, E\}, \{A, B, F\}\}$ before edge CE is added (see Fig. 7(a)). The maximal clique $\{A, B, F\}$ is not contained in C_{C_t} or C_{E_t}. After edge CE is added, from Fig. 7(b) we can see that the clique $\{A, B, F\}$ is still maximal for G_{t+1}.

Following Theorem 1, for incremental updating of maximal cliques when a new edge e_{uw} is added, we only need to consider the cliques in C_{u_t} and C_{w_t}. In particular, some of the cliques involving both u and w may form a new (maximal) clique owing to the new edge e_{uw}. To find those cliques, let us first examine two properties of maximal cliques.

Property 1: *Any subset of a maximal clique is also a clique.*

Property 2: *The intersection of two maximal cliques is also a clique.* For example, $\{A, C, D\} \cap \{A, D, E\} = \{A, D\}$ is also a clique.

Now we define the intersection of two max-clique sets.

Definition 9. (Intersection of Max-Clique Sets). *Given two max-clique sets C_{u_t} and C_{w_t},*

$$C_{u\_t} \Cap C_{w\_t} = \{c_i \cap c_j | c_i \cap c_j \not\subset c'_i \cap c'_j, c_i, c'_i \in C_{u\_t}, c_j, c'_j \in C_{w\_t}\}.$$

For our running example in Fig. 7, $C_{C\_t} \Cap C_{E\_t} = \{\{A, B, C\}, \{A, C, D\}\} \Cap \{\{A, E, F\}, \{A, D, E\}\}$. We have $\{A, B, C\} \cap \{A, E, F\} = \{A\}, \{A, B, C\} \cap \{A, D, E\} = \{A\}, \{A, C, D\} \cap \{A, E, F\} = \{A\}, \{A, C, D\} \cap \{A, D, E\} = \{A, D\}$. Since $\{A\} \subset \{A, D\}$, we get $C_{C\_t} \Cap C_{E\_t} = \{\{A, D\}\}$.

Theorem 2. *For any clique $c \in C_{u\_t} \Cap C_{w\_t}$, $c \cup \{u, w\}$ is a new maximal clique after edge e_{uw} is added.*

Proof: Let $c' = c \cup \{u, w\}$. It is easy to verify that c' is a clique. Next, we prove c' is a *maximal* clique by contradiction. Assume on the contrary c' is not a maximal clique. There must exist a node $n \notin c'$, n is connected to u and w as well as all nodes in c. Thus, by definition, $\{n\} \cup c$ is also a clique. As a result, $c \notin C_{u\_t} \Cap C_{w\_t}$ since $c \subset (\{n\} \cup c)$. This contradicts with our condition $c \in C_{u\_t} \Cap C_{w\_t}$. Therefore, the theorem follows. □

For the example of Fig. 7, since $C_{C\_t} \Cap C_{E\_t} = \{\{A, D\}\}$, we have a new maximal clique $\{A, D\} \cup \{C, E\} = \{A, D, C, E\}$ after edge e_{CE} is added.

Finally, we need to check whether the cliques in C_{u_t} and C_{w_t} are still maximal. For any clique $c_i \in C_{u\_t} \cup C_{w\_t}$ and each $c_k \in C_{u\_t} \Cap C_{w\_t}$, if $c_i \subset c_k \cup \{u, w\}$, c_i is no longer a maximal clique in C_{t+1} for G_{t+1}. For the example of Fig. 7, $\{A, C, D\}$ and $\{A, D, E\}$ are no longer maximal cliques as we now have a new maximal clique $\{A, D, C, E\}$.

As a summary, given the graph shown in Fig. 7(a), after edge e_{CE} is added, the maximal cliques contained in Fig. 7(b) include those maximal cliques containing neither C and E (i.e., $\{A, B, F\}$), $c \cup \{C, E\}$ where $c \in C_{C\_t} \Cap C_{E\_t}$ (i.e., $\{A, D, C, E\}$), and those retaining as maximal cliques in C_{C_t} and C_{E_t} (i.e., $\{A, B, C\}$ and $\{A, E, F\}$).

Algorithm 2 gives the formal procedure for incrementally updating the max-clique set when a new query request arrives. First of all, a new node u is added to the graph and initialized as a 1-node clique (Line 1). In Lines 2-3, if the user of u had not issued any query before, the last cloaked region is set to be the whole service area so that its MMB can cover all existing nodes. Then, we find u's neighbors according to the conditions specified in the graph definition (see Definition 7), and push all edges connecting u and its neighbors to a queue named $EdgeQueue$ (Lines 4-5). Next, in Lines 6-18, we process all edges in $EdgeQueue$ one after one, and incrementally maintains the set of maximal cliques as described in the last few paragraphs.

Algorithm 2 Incremental updating max-clique set

Input: max-clique set $MCSet$, a new request u
Output: updated max-clique set $MCSet$

```
1:  add a new clique $\{u\}$ to $MCSet$
2:  **if** the user of $u$ had not issued any query before **then**
3:      set $u$'s last cloaked region to the whole service area
4:  find $u$'s neighbors according to Definition 7
5:  push all edges connecting $u$ and its neighbors to $EdgeQueue$
6:  **while** $EdgeQueue$ is not empty **do**
7:      $MCSet' = \emptyset$
8:      pop up the first edge $e_{uw}$ from $EdgeQueue$
9:      find clique sets $C_{u\_t}$ and $C_{w\_t}$ in $MCSet$
10:     compute $C = C_{u\_t} \Cap C_{w\_t}$
11:     **for** each $c \in C$ **do** add $c \cup \{u, w\}$ to $MCSet'$
12:     $C = MCSet'$
13:     **for** each $c_i \in C_{u\_t} \cup C_{w\_t}$ **do**
14:         **for** each $c_k \in C$ **do**
15:             **if** $c_i \not\subset c_k$ **then** add $c_i$ to $MCSet'$
16:     **for** each $c_i \in MCSet - C_{u\_t} - C_{w\_t}$ **do**
17:         add $c_i$ to $MCSet'$
18:     $MCSet = MCSet'$
```

Note that Algorithm 2 involves a lot of set operations on the maximal cliques. Thus, in the actual implementation, we represent each maximal clique by a bit vector to ease the computation. For example, suppose that there are five nodes in the graph $\{A, B, C, D, E\}$. Given a maximal clique of $\{A, C, D\}$, it is represented by a bit vector of <10110>. The length of the bit vector is equal to the number of nodes in the graph. Even for a large system with 10K users, only 1.2 Kbytes are needed to store a bit vector, which is acceptable to today's computer.

As for the time complexity, in each while-loop iteration, the most expensive operations are computing $C_{u\_t} \Cap C_{w\_t}$ (Line 10) and identifying non-maximal cliques (Lines 13-15), both of which take $O(|MCSet|^2)$ time. Since $EdgeQueue$ has a size of $|V|$ in the worst case, where $|V|$ is the number of nodes in the graph, the worst-case time complexity of Algorithm 2 is $O(|V| \cdot |MCSet|^2)$. Nevertheless, as we observe in the experiments (Section 5), $|EdgeQueue|$ and $|MCSet|$ are typically small ($10-200$ for over 10K users); the practical complexity of Algorithm 2 is not high.

4.3 Finding the Cloaking Set

After incrementally updating the max-clique set, the cliques where the new request is involved might be candidates for the cloaking set. They can be classified into three classes: *positive candidates*, *negative candidates*, and *non-candidates*.

Definition 10. *Given a candidate clique c, define $|c|$ to be its size, MBR(c) the area of the minimum bounding rectangle for all requests in c, max_k (min_k) the maximum (minimum) privacy level k specified by the requests in c, and $\widehat{A}_{min}$ the maximum A_{min} specified by the requests in c. Denote the newly arrived request by u and its privacy level by k_u. We have:*

- *c is a* positive candidate *if $|c| \geq max\_k$ and $MBR(c) \geq \widehat{A}_{min}$;*
- *c is a* negative candidate *if $max(k_u, min\_k) < |c| < max\_k$ and $MBR(c) \geq \widehat{A}_{min}$;*
- *c is a* non-candidate *if $MBR(c) < \widehat{A}_{min}$ or $|c| \leq max(k_u, min\_k)$.*

For a positive candidate clique, all requests contained in it forms a candidate cloaking set to generate the cloaked region (Section 4.4), since they satisfy both the location *k*-anonymity and minimum area requirements. If there are more than one positive candidate, the largest clique will be chosen.

However, for a negative candidate clique, the location *k*-anonymity is not satisfied for requests with a privacy level *k* higher than the clique size. Therefore, we try to transform a negative candidate to a positive one or a non-candidate by greedily removing some request(s). To do so, the algorithm first sorts the requests in the clique by their descending order of privacy level *k*. Then, it repeatedly removes the user with the highest privacy level until ($|c| \geq max\_k$ and $MBR(c) \geq \widehat{A}_{min}$) or $MBR(c) < \widehat{A}_{min}$. If the former condition is satisfied, this negative candidate has been transformed to a positive candidate. On the other hand, if the latter condition is satisfied, it has been transformed to a non-candidate. For example, suppose that we have $c = \{A, B, C, D, E, F\}$, where $k_A = 8$, $k_B = k_C = 5$, $k_D = 4$, $k_E = k_F = 2$. The clique size $|c|$ = 6. Assume that E is the newly arrived request. Since $|c| < max\_k = 8$, *c* is a negative candidate. As k_A has the highest value of *k*, *A* is dropped from the clique. After that, the clique size becomes 5, which is no smaller than the current max_k = 5. Thus, the remaining users $\{B, C, D, E, F\}$ can form a cloaking set if their MBRs also satisfy the minimum area requirement. The detailed procedure for finding the cloaking set in a negative candidate is described in Algorithm 3. The time complexity of Algorithm 3 is $O(|c| \cdot \log |c|)$.

In summary, given a new request, we examine all maximal cliques involving the new request in descending order of clique size, until a positive candidate is found. The procedure for classifying candidates and finding the cloaking set is described in Algorithm 4. For the time complexity, the sorting process in Line 2

Algorithm 3 Finding cloaking set in negative candidate clique

```
Input: negative candidate clique $c$
Output: candidate cloaking set $CS_{t_i}$
1: sort the requests in $c$ in descending order of their privacy level $k$
2: while $|c| < max\_k$ and $MBR(c) \geq \widehat{A}_{min}$ do
3:     drop the request with the highest $k$ from $c$
4:     update $|c|$, $max\_k$, MBR($c$), and $\widehat{A}_{min}$
5: if $|c| \geq max\_k$ and $MBR(c) \geq \widehat{A}_{min}$ then
6:     $CS_{t_i}$ = the set of current requests in $c$
7: else
8:     $CS_{t_i} = \emptyset$
```

Algorithm 4 Finding the cloaking set

```
Input: max-clique set $MCSet$, the new request $u$
Output: candidate cloaking set $CS_{t_i}$
1: $canCR = \{c \in MCSet \mid u \in c\}$
2: sort $canCR$ in descending order of clique size
3: for each $c$ in $canCR$ do
4:     find $max\_k$, $min\_k$, $\widehat{A}_{min}$ for $c$
5:     if $|c| \geq max\_k$ and $\widehat{A}_{min} \leq MBR(c)$ then
6:         $CS_{t_i}$ = the set of requests in $c$; break
7:     if $|c| < max(k_u, min\_k)$ or $MBR(c) < \widehat{A}_{min}$ then
8:         not a candidate; continue
9:     else
10:        $CS_{t_i}$ = returned result of invoking Algorithm 3
11:        if $CS_{t_i} \neq \emptyset$ then break
```

takes $O(|canCR| \cdot \log |canCR|)$ time. In each for-loop iteration, the most expensive operation is finding the cloaking set in a negative candidate clique (Line 10), which has a complexity of $O(|c| \cdot \log |c|)$. In the worst case, $|c| = |V|$. Thus, the time complexity of Algorithm 4 is $O(|canCR| \cdot (\log |canCR| + |V| \cdot \log |V|))$.

4.4 Generating the Cloaked Region

Recall that a cloaking set should satisfy four conditions (see Definition 5). However, the cloaking set obtained from Algorithm 4 satisfies all the conditions except the second one. In order to resolve this, we may need to extend the boundary of the cloaking set, such that for each request $u \in CS_{t_i}$, its previous cloaked region $R_{u,t_{i-1}}$ is covered by the new MAB. The algorithm works as follows. After we get the cloaking set CS_{t_i}, its MBR is computed as the candidate cloaked region CR_{t_i}. For each request $u \in CS_{t_i}$, compute d = MaxMinD($R_{u,t_{i-1}}, CR_{t_i}$). If $d > v_u \cdot (t_i - t_{i-1})$, $\delta_d = d - v_u \cdot (t_i - t_{i-1})$. Then, CR_{t_i} is extended by δ_d on the direction where $R_{u,t_{i-1}}$ locates w.r.t. CR_{t_i}. Finally, the enlarged region is returned as the cloaked region R_{t_i} of the cloaking set. The pseudo-code is given in Algorithm 5.

Again take user u_5 in Fig. 6 as an example. Let the cloaked region at t_{i-1} be $R_{t_{i-1}}$. First, the rectangle $P_1P_2P_3P_4$ is computed as the MBR of CR_{t_i}. We find

Algorithm 5 Generating the cloaked region

```
Input: cloaking set $CS_{t_i}$
Output: cloaked region $R_{t_i}$
1: $CR_{t_i}$ = the MBR of $CS_{t_i}$
2: for each request $u$ in $CS_{t_i}$ do
3:     $\delta_d$ = MaxMinD$(R_{u,t_{i-1}}, CR_{t-i}) - v_u \cdot (t_i - t_{i-1})$
4:     $dirs$ = directions where $R_{u,t_{i-1}}$ locates w.r.t. $CR_{t_i}$
5:     for each direction $dir$ in $dirs$ do
6:         if $md_{dir} < \delta_d$ then
7:             $md_{dir}$ = projected distance of $\delta_d$ on $dir$
8: extend $CR_{t_i}$ to $R_{t_i}$ by $md_{dir}$ on each $dir$
9: for each request $u$ in $CS_{t_i}$ with new boundary $R_{t_i}$ do
10:     if $R_{t_i}$ is not covered by $u$'s $MMB_{u,t_{i-1},t_i}$ then
11:         return false
12: return $R_{t_i}$ as the cloaked region
```

that $R_{t_{i-1}}$ is on the *left* of $P_1P_2P_3P_4$. Compute δ_d = MaxMinD($R_{t_{i-1}}$, $P_1P_2P_3P_4$) $- v_{u_5} \cdot (t_i - t_{i-1})$. Then, the boundary edge P_1P_4 should be shifted to the left by δ_d, so that u_5 can be free of location-dependent attacks. Note that for a request from a first-time user, its R_{i-1} is regarded as the whole service space, thus the MaxMin distance is 0. Next, Lines 9 to 11 of Algorithm 5 are to check whether the new boundary of the extended region is still covered by each user's previous MMB. If not, this cloaking process fails. Otherwise, it succeeds and proceeds to update the max-clique set, as will be detailed in the next subsection.

For the time complexity, Algorithm 5 has two for-loops (Lines 2-7 and Lines 9-11), each of which takes $O(|CS_{t_i}|)$ time. Since $|CS_{t_i}| = |V|$ in the worse case, the time complexity of Algorithm 5 is $O(|V|)$.

4.5 Update the Max-Clique Set for Leaving Requests

The requests will leave the system after they have been successfully cloaked or failed to be cloaked due to expiry of their tolerable cloaking delays. To efficiently identify expired requests, we employ a min-heap to keep track of the set of alive requests, where the key is the expiry time.

For both successfully cloaked requests and expired requests, they are seen as leaving requests. When they leave, the max-clique set should be updated accordingly. Specifically, for each leaving request, it should be removed from the maximal cliques where it is involved. Recall that a subset of a maximal clique is still a clique, but not guaranteed a maximal clique. Thus, we need to check whether the updated cliques are still maximal. To do so, we examine each of such cliques against other cliques remaining in the system. If it is a subset of any other clique, it is not a maximal clique and should be removed from the max-clique set. For example, assume $MCSet = \{\{A, B, C\}, \{A, C, D\}\}$. After the request D expires, it should be removed from $\{A, C, D\}$. $\{A, C, D\}$ is updated to $\{A, C\}$, which is no longer a maximal clique since it is a subset of $\{A, B, C\}$. Hence, $MCSet$ = $\{\{A, B, C\}\}$ after removing D. The procedure for updating the max-clique set due to leaving requests is described in Algorithm 6. The time complexity of Algorithm 6 is $O(|MCSet|^2)$.

Algorithm 6 Updating max-clique set for leaving request

```
Input: max-clique set $MCSet$, leaving request $u$
Output: updated max-clique set $MCSet$
1: remove $u$ from the min-heap
2: for each $c$ in $MCSet$ where $u \in c$ do
3:     for each $c'$ in $MCSet$ do
4:         if $c \subset c'$ then
5:             remove $c$ from $MCSet$; break
```

5 Performance Evaluation

In this section, the effectiveness and efficiency of our proposed ICliqueCloak are experimentally evaluated under various system settings. We first describe the experiment setup in Section 5.1, followed by the performance evaluation results presented in Sections 5.2–5.7.

5.1 Experiment Setup

We compare three algorithms, namely IClique, MMB-Clique, and OptClique. IClique is short for our proposed ICliqueCloak. MMBClique is revised from the cloaking algorithm proposed in [13] with a new definition of node connectivity, where the tolerable maximum cloaking region is replaced with the MMB to prevent from location dependent attacks. MMBClique is included for comparison because we are interested in finding out *what and how much* is improved by our proposed incremental cloaking algorithm. OptClique is a modified version of IClique without considering the effect of location-dependent attacks. That is, in OptClique the MMB is set to be infinite (such that all users are connected with each other in the underlying graph). As such, OptClique *cannot* prevent from location-dependent attacks; but it is included for comparison to show the *cost* required for defending against location-dependent attacks. As none of the existing cloaking algorithms (e.g., [7], [10], [14]), that are immune to location-dependent attacks, employ the location k-anonymity model, we do not include these existing cloaking algorithms for comparison. The evaluation metrics we used include the cloaking success rate, the anonymization cost (Definition 6), the cloaking time, and the processing time for successful requests.

To the best of our knowledge, due to privacy and commercial interest reasons, no real large-scale moving-objects datasets have been publicly released to date. Therefore, in most of our experiments, we use the well-known Thomas Brinkhoff Network-based Generator of Moving Objects [36] to generate the moving objects in the system. The input of the generator is the road map

TABLE 1
Default System Settings

Parameter	Default setting
Number of users	50,000
Speed profile	Medium
Privacy level k	Randomly chosen from [2, 10]
Tolerable cloaking delay	$0.1s$
A_{min}	0.005% to 0.01% of the space
Query Interval	60s

of Oldenburg County. In the default setting, a total of 50,000 moving objects are generated at the beginning of the simulation. In the generator, three speed profiles can be configured for the moving objects: slow, medium, and fast.[5] The default speed setting is medium. The time of the first query request for each user is randomly chosen from [0, $60s$]. The query interval is set at $60s$. Thus, about 1/60 of all users issue queries in each second. By default, every request sets its privacy level k with a random number in the range of [2, 10]. The tolerable cloaking delay is set shorter than the query interval, which guarantees that a new request is not issued until the last request is successfully cloaked or expired. For each request, A_{min} is set to $0.005\%-0.01\%$ of the space. Table 1 lists the default system settings.

In addition to the simulated data, we also adapt the real trucks data [1] to validate the effectiveness of our IClique algorithm. All cloaking algorithms are implemented in C++ and run on a desktop PC with a dual AMD 780MHz processor and 2GB main memory.

5.2 Impact of Privacy Level

In this section, we investigate the impact of privacy level k on the performance of cloaking algorithms from two aspects: increasing the privacy levels and enlarging the privacy level range.

In the first set of experiments, we fix the privacy levels at a similar range, but increase both the lower and upper bound, which implies that the privacy requirement of every request becomes more constrained. From Fig. 8(a), it is interesting to observe that the success rates of OptClique and IClique increase slightly with increasing privacy level. They reach up to 99% when the privacy level range is [10, 20]. The reason is that both OptClique and IClique find cloaking sets from the maximal cliques, and prefer to return the largest clique (see Lines 3-11 in Algorithm 4) when there are more than one positive candidate. Thus, they are very suitable for cloaking requests with higher privacy requirements. Comparing OptClique and IClique, only a success rate of 2% is sacrificed for protecting against location-dependent attacks in IClique in the worst case. In contrast, MMBClique has the worst performance. Its success rate decreases significantly with increasing k. This is mainly because MMBClique becomes much slower to find a cloaking set with a larger k (see Fig. 8(c)). As a result, more requests in MMBClique are likely to expire before they can be successfully cloaked. This signifies the importance of the proposed incremental clique maintenance method.

Fig. 8(b) shows the average anonymization cost for all cloaking algorithms. As expected, the cost increases when the privacy level increases, because more requests need to be cloaked together to meet a higher privacy requirement. MMBClique shows a smaller anonymization cost than the other two algorithms for the first two settings. The reason is that given a number of requests with various privacy levels, MMBClique prefers to cloak the neighboring requests with lower privacy levels; but OptClique and IClique favor the requests with higher privacy levels and hence incur some more anonymization cost. However, when the privacy level increases to [10, 20], the cost of MMBClique exceeds that of the other two algorithms. As explained above, in this case much more requests in MMBClique expire and farther requests have to be cloaked together.

The cloaking time of a request is the time used to update the max-clique set, find the cloaking set, and generate the cloaked region, but does not include the time awaiting to be cloaked. As shown in Fig. 8(c), the cloaking times of all algorithms increase due to a more constrained privacy requirement when the privacy level increases. Obviously, OptClique and IClique require a much shorter cloaking time than MMBClique, since both of them can quickly find the cloaking set from the set of incrementally maintained maximal cliques. We also measure the processing time for successful requests, which is the time period from when the request is received to when the request is successfully cloaked. It includes the cloaking time and the time awaiting for cloaking. As shown in Fig. 8(d), in most cases the waiting time dominates in the overall processing time, and the average processing time increases with increasing privacy level. In particular, MMBClique cannot scale up to a large privacy level and its processing time worsens dramatically.

Next, we evaluate the impact of enlarging the privacy level range, which implies not only more constrained but also more diversified privacy requirements. From Figs. 9(b)-9(d), the anonymization cost, the cloaking time, and the processing time show similar trends as those observed in Figs. 8(b)-8(d). However, different from Fig. 8(a), in Fig. 9(a) the success rates of OptClique and IClique decrease with much more diversified privacy levels. Nevertheless, they are still above 84% and 82%, respectively, even when the maximum privacy level reaches at 15. In contrast, the success rate of MMBClique drops more quickly. When the maximum privacy level is 15, its success rate drops down to about 60%. The main reason is that MMBClique finds the cloaking set only from the neighbors whose privacy levels are less than that of the new request. Thus, a request with a

5. According to the generator [36], each setting of speed is five times faster than the previous one on average. Denote by v_{min} (v_{max}) the minimum (maximum) speed in each speed setting. Then, 80% object speeds are selected from [v_{min}, $\frac{v_{max}}{3}$].

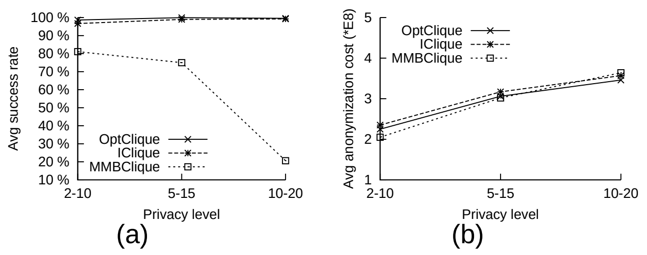
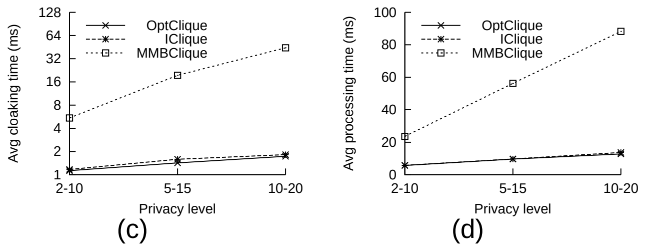



Fig. 8. Different privacy levels with similar diversity: (a) Success rate (b) Anonymization cost (c) Cloaking time (d) Processing time

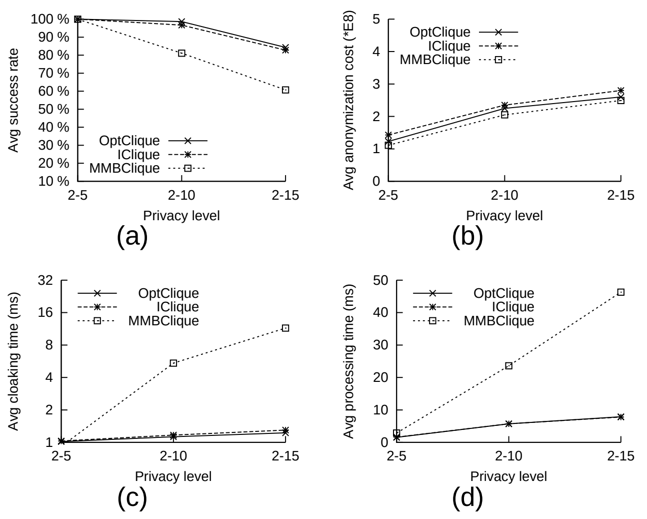


Fig. 9. Different diversities of privacy levels: (a) Success rate (b) Anonymization cost (c) Cloaking time (d) Processing time

high privacy level is hard to be cloaked successfully in MMBClique.

5.3 Impact of Tolerable Cloaking Delay

This section examines the impact of varying tolerable cloaking delay dt, while the privacy level k is randomly chosen between 2 to 10. As shown in Fig. 10(a), the success rates of all cloaking algorithms generally increase with prolonging dt, as each request has more time to wait for successful cloaking. However, for OptClique and IClique, their cloaking time is very short so that the requests have little chance to expire due to cloaking delay. Thus, prolonging dt for them has only little effect on the success rate. For example, when dt=0.05s, the success rate of IClique is 95%, and it is slightly increased to 98% when dt is prolonged to 2s. In contrast, the

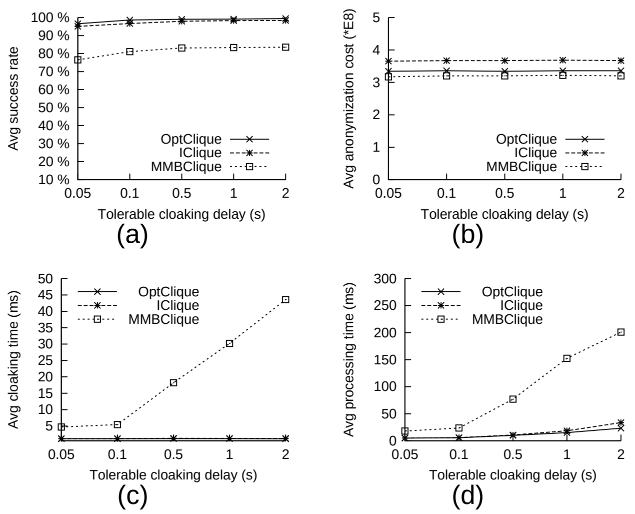


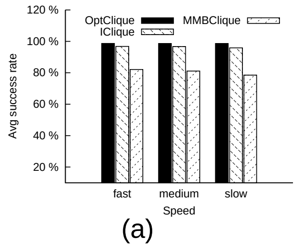
Fig. 10. Different tolerable cloaking delays: (a) Success rate (b) Anonymization cost (c) Cloaking time (d) Processing time

cloaking success rate of MMBClique greatly increases from 77% to 83% when dt is prolonged from 0.05s to 0.5s, and becomes stable after that.

Fig. 10(b) shows that for all the three cloaking algorithms the average anonymization cost is not affected much by prolonging the tolerable cloaking delay. Fig. 10(c) shows that the average cloaking time under different tolerable cloaking delays. A longer tolerable cloaking delay dt means that a request can stay longer in the system, thereby increasing the connectivity of the modelled graph. From Fig. 10(c), we can see that the average cloaking time of MMBClique significantly increases with prolonging dt. This is because it requires more time to find a clique involving the new request for a more highly connected graph. However, for both OptClique and IClique, a longer dt does not imply much more maximal cliques. Thus, their average cloaking time is not affected much and fixed at around 1.1ms and 1.3ms, respectively, for all dt settings tested. As more requests could be waiting with prolonging dt, Fig. 10(d) shows that the average processing times of all algorithms increase as expected.

5.4 Impact of Movement Speed

As the size of MMB is a factor that affects the cloaking success rate and anonymization cost directly, IClique, OptClique, and MMBClique are also evaluated under different moving speeds. A slower speed means a smaller size of MMB for a user, which indicates the condition for finding a cloaked region is much more constrained. It consists of two aspects: one is the condition of having an edge between two nodes, and the other is the free space in MMB for a candidate cloaked region to extend. From Fig. 11(a), we can see that the speed has no effect on the cloaking success rate of OptClique, since its MMB is set to be infinitely large regardless of the moving speed. Its success rate can still reach 98% with slow



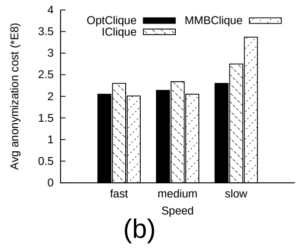
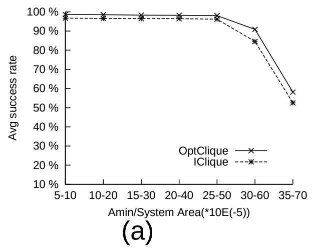
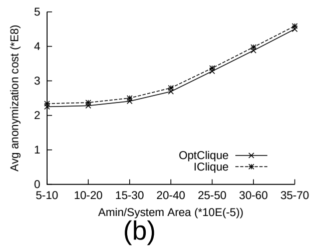


Fig. 11. Different speed profiles: (a) Success rate (b) Anonymization cost





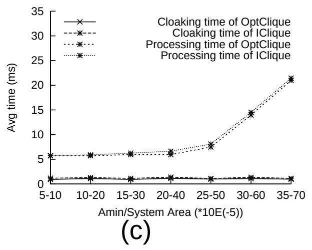


Fig. 12. Different A_{min} Settings: (a) Success rate (b) Anonymization cost (c) Cloaking time and processing time

speeds. In contrast, the cloaking success rates of IClique and MMBClique drop slightly with decreasing the speed. However, as shown in Fig. 11(b), their anonymiation cost increases greatly with slowing down the speed. The reason is that a request needs to find farther neighbors to form the cloaking set when the speed is slower. Thus, the price paid for defending against location-dependent attacks is higher for a slower moving speed.

5.5 Effect of A_{min}

Recall that our privacy model consists of the privacy level k, the minimum area A_{min}, and the tolerable cloaking delay dt. The effects of k and dt have been evaluated in the previous sections. The parameter A_{min} is not used in the graph model, but used when a candidate result is to be verified whether it is a positive candidate. Therefore, it is expected that A_{min} has little influence on the cloaking time for each successful request. What might be affected by A_{min} is the processing time, the cloaking success rate, and the anonymization cost. Since MMBClique [13] does not consider the A_{min} requirement, in this section we evaluate the effect of A_{min} for IClique and OptClique.

Fig. 12(c) validates that the cloaking time is not affected much by the setting of A_{min} as expected. The average cloaking time is about 1.2*ms*, fluctuating between 1.1*ms* and 1.3*ms*. It is interesting to observe in Fig. 12(c) that the processing time is not affected much when A_{min} is not very large. But the processing time increases significantly at the last three A_{min} settings. We observe a similar phenomenon from Fig. 12(a), where the cloaking success rate remains constant for different A_{min} settings except the last three ones. This implies that IClique has to find a larger cloaked region to meet an increased A_{min} requirement when A_{min} exceeds a threshold, and this also lengthens the waiting time. However, having a larger cloaked region may violate the MMB constraint, and thus more requests fail to be cloaked. Fig. 12(b) shows that the average anonymization cost increases linearly to meet an increased A_{min} requirement.

5.6 Scalability

We now evaluate the effect of number of users on the performance of cloaking algorithms. The number of users indicates two aspects: the user density of the service area and the workload of the anonymizing system. We vary the number of users from 10,000 to 100,000.

As shown in Fig. 13(a), the success rates of all the three algorithms slightly decrease with increasing number of users. This is mainly because of the increased workload (see Fig. 13(c)), which makes more requests to expire and fail to be cloaked. Comparing the three cloaking algorithms, MMBClique has the worst success rate for all the settings tested. Fig. 13(b) shows the effect of user number on the average anonymization cost. Increasing the number of users implies increasing the user density. When the user density is higher, each cloaked region tends to be smaller. As expected, the average anonymization costs of all the three algorithms generally reduce when the the number of users increases.

The cloaking time and processing time are shown in Figs. 13(c) and 13(d), respectively. It is clear that OptClique and IClique outperform MMBClique for all the settings. The reason is as explained earlier. MMBClique finds the cloaking set from the cliques centered around the new request online. However, IClique and Optclique exploit the intermediate searching results, namely incrementally maintained maximal cliques, and find the cloaking set from these maximal cliques. Thus, a new request only needs to check the maximal cliques where it is contained as a member. For both OptClique and IClique, their cloaking time increases gradually with increasing number of users, since more users imply more maximal cliques and longer search time. For MMBClique, its cloaking time is also lengthened due to the need to search a larger graph when the number of users increases.

5.7 Real Dataset Results

As mentioned earlier, no real large-scale moving-objects datasets are publicly available. Thus, we adapt the Athens trucks dataset available at [1]. As the original trucks dataset contains the movement trajectories of 50

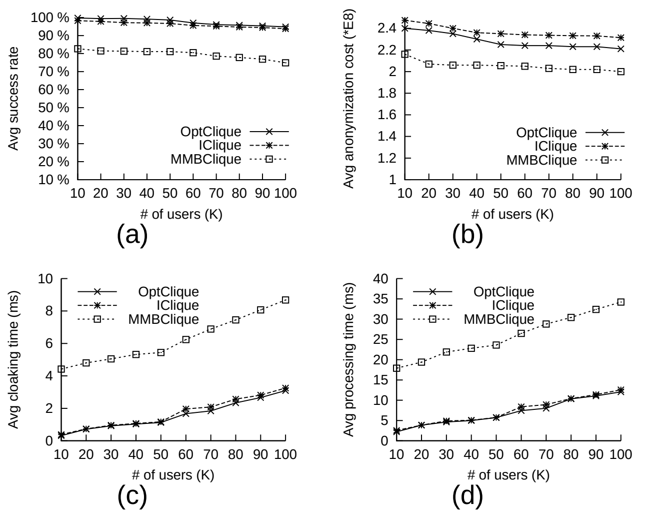


Fig. 13. Different dataset sizes: (a) Success rate (b) Anonymization cost (c) Cloaking time (d) Processing time

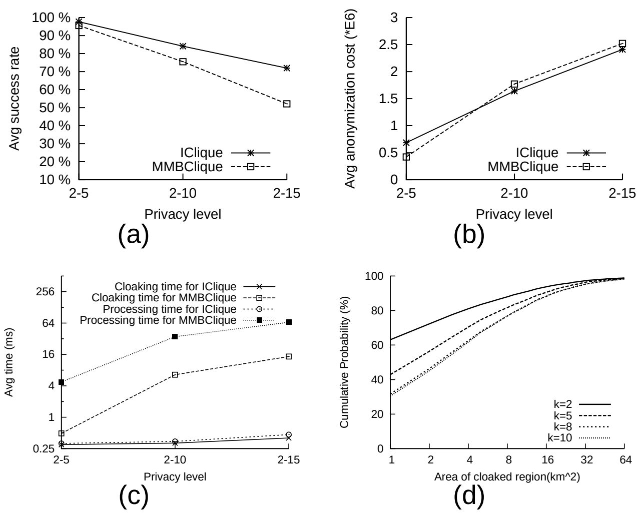


Fig. 14. Real dataset results: (a) Success rate (b) Average anonymization cost (c) Cloaking time and processing time (d) CDF of Cloaking Area

trucks only, it is too small for our performance evaluation. Hence, we randomly select 80,000 trajectory segments (each with 500 location updates) from the original trajectories to represent 80,000 users. At each timestamp, 12.5% of the users make query requests. That is, the number of effective users at each timestamp is 10,000. We report the evaluation results in Figs. 14(a)-14(d).

The performance trends on the success rate and cloaking/processing time are similar to that of the simulation results in Section 5.2. As observed from Fig. 14(a), the success rate of MMBClique decreases significantly with increasing privacy level. Its success rate is dropped to 52% when $k \in [2, 15]$. In contrast, IClique still gets a success rate of about 71% at the same setting. Fig. 14(c) shows that IClique performs even better than MMBClique for larger k values in terms of the cloaking and processing time. When $k \in [2, 15]$, IClique is about 36 (144) times faster than MMBClique in terms of the cloaking (processing) time.

Fig. 14(b) compares the average anonymization costs of IClique and MMBClique. Similar to the results obtained from the simulation data (Fig. 9(b)), both of IClique and MMBClique get a worse performance when the privacy level increases. MMBClique has a smaller anonymization cost than IClique when $k \in [2, 5]$. However, MMBClique has a slightly higher cost than IClique when $k \in [2, 10]$ and $k \in [2, 15]$, which is different from the results obtained in Fig. 9(b). The is mainly because compared to the simulation data, here MMBClique has an even lower cloaking success rate for these two settings. As a consequence, a new request has to find farther neighbors to form a cloaking set.

In Fig. 14(d), we show the cumulative distribution function (CDF) results of the cloaking area. As can be seen, over 60% of the cloaking areas are less than $1km^2$ for a low privacy level (k=2), less than $2.5km^2$ for a medium privacy level (k=5), and less than $4km^2$ for a high privacy level (k=8 or k=10). We believe that such a cloaking region size would be acceptable in terms of server query processing overhead while protecting location privacy of mobile users for most LBS applications.

6 Conclusions

In this paper, we investigated cloaking algorithms that protect location privacy against location-dependent attacks. We showed that most of the existing location cloaking algorithms cannot effectively defend against location-dependent attacks as they are concerned with snapshot user locations only. To address this problem, we have employed a graph model to formalize the problem and transformed it to the problem of finding k-node cliques in the graph. We have proposed an incremental clique-based cloaking algorithm called ICliqueCloak to generate cloaked regions. A series of experiments has been conducted to evaluate ICliqueCloak under various system settings. The experimental results show that the price paid for location-dependent attacks is small. The average processing time is only 5.7*ms* and the cloaking success rate is about 97% for most cases, which validate the efficiency and effectiveness of the proposed ICliqueCloak algorithm.

References

[1] Athens Trucks Data. Available at http://www.rtreeportal.org/, 2006.

[2] ABI research. Available at http://www.abiresearch.com/press/1097-Mobile+Location+Based+Services+Revenue+to+Reach+$13.3+Billion+Worldwide+by+2013, 2008.

[3] O. Abul, F. Bonchi, and M. Nanni, ”Never walk alone: Uncertainty for anonymity in moving objects databases,” *Proc. of ICDE*, pp. 376-385, April 2008.

[4] B. Bamba and L. Liu, ”Supporting anonymous location queries in mobile environments with privacygrid,” *Proc. of WWW*, 2008.

[5] C. Bettini, X. S. Wang, and S. Jajodia, ”Protecting privacy against location-based personal identification,” *Proc. of SDM*, pp 185-199, 2005.

[6] K. Bharath, G. Ghinita, P. Kalnis, ”Privacy-preserving publication of user locations in the proximity of sensitive sites”, *Proc. of SSDBM*, July 2008.

[7] R. Cheng, Y. Zhang, E. Bertino, and S. Prabhakar, ”Preserving user location privacy in mobile data management infrastructures,” *Proc. of Privacy Enhancing Technology Workshop (PET’06)*, 2006.

[8] C. Chow and M. F. Mokbel, ”Enabling private continuous queries for revealed user locations,” *Proc. of SSTD*, 2007.

[9] DIRECTIVE 2002/58/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 12 July 2002, *Official Journal of the European Communities*, pp. 37-47, 2002.

[10] J. Du, J. Xu, X. Tang, and H. Hu, ”iPDA: Enabling privacy-preserving location-based services,” *Proc. of MDM*, 2007.

[11] G. Ghinita, P. Kalnis, and S. Skiadopoulos, ”Prive: Anonymous location-based queries in distributed mobile systems,” *Proc. of WWW*, 2007.

[12] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K. Tan, ”Private queries in location based services: anonymizers are not necessary,” *Proc. of SIGMOD*, 2008.

[13] B. Gedik and L. Liu, ”Location privacy in mobile systems: A personalized anonymization model,” *Proc. of ICDCS*, pp. 620-629, 2005.

[14] G. Ghinita, M. L. Damiani, and C. Silvestri, ”Preventing velocity-based linkage attacks in location-aware applications,” *Proc. of ACM GIS*, 2009.

[15] G. Gidofalvi, X. Huang, and T. B. Pedersen, ”Privacy-preserving data mining on moving objects trajectories,” *Proc. of MDM*, 2007.

[16] A. Gkoulalas-Divanis, P. Kalnis, and V. S. Verykios, ”Providing k-anonymity in location based services,” *SIGKDD Explorations Newsletter*, vol. 12, no. 1, June 2010.

[17] A. Gkoulalas-Divanis and V. S. Verykios, ”A privacy-aware trajectory tracking query engine,” *SIGKDD Explorations Newsletter*, vol. 10, no. 1, pp. 40-49, 2008.

[18] A. Gkoulalas-Divanis, V. S. Verykios, M. F. Mokbel. ”Identifying unsafe routes for network-based trajectory privacy.” *Proc. of SDM*, 2009.

[19] M. Gruteser and D. Grunwald, ”Anonymous usage of location-based services through spatial and temporal cloaking,” *Proc. of MobiSys*, pp. 163-168, 2003.

[20] M. Gruteser and B. Hoh, ”On the anonymity of periodic location samples,” *Security in Pervasive Computing*, vol. 3450, pp. 179-192, 2005.

[21] B. Hoh, and M. Gruteser, ”Protecting location privacy through path cloaking,” *Proc. of SecureComm*, 2005.

[22] H. Hu and J. Xu, ”Non-exposure Location Anonymity,” *Proc. of*

[23] H. Hu, J. Xu, S. T. On, J. Du, and J. K. Ng. ”Privacy-Aware Location Data Publishing.” *ACM Transactions on Database Systems (TODS)*, 35(3), July 2010.

[24] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias, ”Preventing location-based identity inference in anonymous spatial queries,” *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 19(12): 1719-1733, 2007.

[25] H. Kido, Y. Yanagisawa, and T. Satoh, ”Protection of location privacy using dummies for location-based services,” *Proc. of the 25th Int’l Conf. on Distributed Computing Systems (ICPS)*, 2005.

[26] J. Krumm, ”Inference Attacks on Location Tracks,” *Proc. of the 5th International Conference on Pervasive Computing*, 2007.

[27] K. Lee, W. C. Lee, H. V. Leong, B. Zheng, ”Navigational path privacy protection: Navigational path privacy protection,” *Proc. of CIKM*, 2009.

[28] L. Liu, ”From data privacy to location privacy: Models and algorithms,” *Proc. of VLDB*, pp. 1429-1430,2007.

[29] H. Lu, C. S. Jensen, and M. L. Yiu, ”A3D: Anonymity area aware, dummy-based location privacy in mobile services,” *Proc. of MobiDE*, 2008.

[30] S. Mascetti, C. Bettini, X. S. Wang, D. Freni, and S. Jajodia, ”Preserving anonymity in location-based services when requests from the same issuer may be correlated,” TR, University of Milan, Italy, 2007.

[31] M. F. Mokbel, C. Y. Chow, and W. G. Aref, ”The new casper: Query processing for location services without compromising privacy,” *Proc. of VLDB*, 2006.

[32] D. Reid, ”An algorithm for tracking multiple targets,” *IEEE Transactions on Automatic Control*, 24(6): 843C854, 1979.

[33] P. Samarati and L. Sweeney, ”Protecting privacy when disclosing information: K-anonymity and its enforcement through generalization and suppression,” *Int’l Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5): 571-588, 2002.

[34] L. Sweeney, ”K-anonymity: A model for protecting privacy,” *Int’l Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5): 557-570, 2002.

[35] M. Terrovitis and N. Mamoulis, ”Privacy preservation in the publication of trajectories,” *Proc. of MDM*, 2008.

[36] Thomas Brinkhoff Network-based Generator of Moving Objects. Available at http://www.fh-oow.de/institute/iapg/personen/brinkhoff/generator/, 2008.

[37] J. Xu, X. Tang, H. Hu, and J. Du, ”Privacy-Conscious Location-Based Queries in Mobile Environments,” *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, March 2010.

[38] T. Xu and Y. Cai, ”Location anonymity in continuous location-based services,” *Proc. of GIS*, 2007.

[39] T. Xu, and Y. Cai, ”Exploring historical location data for anonymity preserving in location-based services,” *Proc. of INFOCOM*, 2008.

[40] T. H. You, W.-C. Peng, and W. C. Lee, ”Protecting moving trajectories with dummies,” *Proc. of PALMS*, 2007.