2010

Location cloaking for location privacy protection and location safety protection

Ge Xu
Iowa State University

Recommended Citation

Xu, Ge, "Location cloaking for location privacy protection and location safety protection" (2010). *Graduate Theses and Dissertations*. 11628.
https://lib.dr.iastate.edu/etd/11628

Location cloaking for location privacy protection and location safety protection

by

Ge (Toby) Xu

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Science

Program of Study Committee:
Ying Cai, Major Professor
Ahmed Kamal
Leslie Miller
Wallapak Tavanapong
Wensheng Zhang

Iowa State University

Ames, Iowa

2010

DEDICATION

To my parents and my wife -

Without whose love and support I would not have been able to complete this work

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

I am extremely lucky that I get support, encouragement, and inspiration from many people. Without them this work would not have been possible.

My greatest gratitude goes to my advisor Dr. Ying Cai for his guidance and consistent support. His knowledgeable and wise discussions have inspired the ideas of all the work in this dissertation. When facing obstacles, I am lucky that he has always been there helping me find the right direction. From him I have learned how to find problems, generate solutions, make presentations, and overall how to be a researcher in computer science. Thank you, Dr. Cai!

I would also like to thank other members of my committee, Dr. Ahmed Kamal, Dr. Leslie Miller, Dr. Wallapak Tavanapong and Dr. Wensheng Zhang for their time and input. Specifically, I would like to thank their insightful comments which help me clarify my ideas and highlight my contributions in this dissertation.

I am greatly thankful to my lab mates, Kihwan Kim, Ashwin S. Natarajan, and Patricio Galdames for their collaboration during this research and contributions to this dissertation. I wish to thank Jiang Tian, Taiming Feng, Xia Wang, Chuang Wang, Ru He, Wei Zhang, Hsin-yi Jiang for their support and assistance during my study.

Finally, my deep gratitude goes to my wife (Liu Hong) and my parents (Liguang Du and Rongsheng Xu) for their love, sacrifice and support during my life.

ABSTRACT

Many applications today rely on location information, yet disclosing such information can present heightened privacy and safety risks. A person's whereabouts, for example, may reveal sensitive private information such as health condition and lifestyle. Location information also has the potential to allow an adversary to physically locate and destroy a subject, which is particularly concerned in digital battlefields.

This research investigates two problems. The first one is location privacy protection in location-based services. Our goal is to provide a desired level of guarantee that the location data collected by the service providers cannot be correlated with restricted spaces such as home and office to derive who's where at what time. We propose 1) leveraging historical location samples for location depersonalization and 2) allowing a user to express her location privacy requirement by identifying a spatial region. With these two ideas in place, we develop a suite of techniques for location-privacy aware uses of location-based services, which can be either sporadic or continuous. An experimental system has been implemented with these techniques. The second problem investigated in this research is location safety protection in ad hoc networks. Unlike location privacy intrusion, the adversary here is not interested in finding the individual identities of the nodes in a spatial region, but simply wants to locate and destroy them. We define the safety level of a spatial region as the inverse of its node density and develop a suite of techniques for location safety-aware cloaking and routing. These schemes allow nodes to disclose their location as accurately as possible, while preventing such information from being used to identify any region with a safety level lower than a required threshold. The performance of the proposed techniques is evaluated through analysis and simulation.

CHAPTER 1. Introduction

With the continuous price dropping and miniaturization of positioning systems such as GPS, more and more applications in wireless networks have taken advantage of location information of wireless users and devices in their design and development. However, disclosing location information can presents heightened privacy and safety risks. In the aspect of privacy, physical destinations such as medical clinics may indicate a person's health problems. Likewise, regular stops at certain types of places may be linked directly to one's lifestyles or political associations. In the aspect of safety, knowing the position of a wireless device allows an adversary to locate and physically destroy the subject, which is particularly concerned in digital battlefields.

Our research in this thesis aims to address the above threats presented by location exposure. Specifically, we investigate two problems. The first one is *location privacy protection in the context of location-based services (LBSs)*. Too use an LBS, a user needs to submit her location to the service provider, which may not be trustworthy in keeping the information in confidential. Even if a user replaces her real-world identity with a pseudonym, the anonymous location information may still be correlated with restricted spaces such as house and office for subject re-identification. Our research focuses on this problem known as *restricted space identification*, and investigates location depersonalization for the purpose of location privacy protection. Specifically, given an anonymous location disclosed in a service request, we want to prevent an adversary from deriving who was in the location at the time of the service request.

Toward this goal, we propose to explore users' historical location samples, each called a *footprint*, for location depersonalization. A spatial region with K different footprints means it has been visited by K different people. When a user requests a service, her location is reported as such a region instead of her accurate position. Therefore, even if an adversary manages to identify all these visitors using restricted spaces, he will not know which of them was inside the area at the time of the service request. In addition to location depersonalization, we address the challenge of modeling location privacy requirement. With the traditional K-anonymity model, a user needs to specify a value of K as her privacy requirement. This is problematic, because privacy is about feeling, and it is awkward for one to scale her feeling using a number. Our solution circumvents this problem by allowing a user to identify a public region, such as a shopping mall, which she would feel comfortable that it is reported as her location should she request a service inside it. This region is then used as her privacy requirement – each location disclosed on her behalf needs to be at least as popular as that area. Compared to choosing a number of K, this *feeling-based* strategy provides a much more intuitive way for users to express their privacy requirement. With the above ideas in place, we present a suite of cloaking algorithms to depersonalize users' location disclosed in both sporadic and continuous LBSs. We evaluate the performance of our techniques via simulations. Moreover, we have implemented an experimental prototype for feasibility and practicality study.

The second problem investigated in this thesis is *location safety protection in the context of ad hoc networks*. Many applications and protocols (e.g., LAR (1), DREAM (2), GPSR (3)) designed for ad hoc networks take advantage of node location information for functionality and scalability. Little work, however, has been done to deal with the safety threat introduced when nodes disclose their location information in a hostile environment. Knowing a spatial region contains a set of sensors, an adversary may comb through the area to locate and destroy all of them. This threat is different from location privacy intrusion in the sense that here the adversary is not interested in finding the individual identities of the nodes in a spatial region, but simply wants to locate and destroy them.

The specific goal of thwarting the location safety threat is to make it practically infeasible for an adversary to find nodes' accurate position based on the location information they disclose in communications. An adversary can always comb through an entire region to destroy the nodes located inside it, but if the area is very large, the cost can be prohibitively high. As such, we define the *safety level* of a spatial region as the inverse of its node density. The higher safety level a spatial region has, the less attractive it is for the adversary to attack the nodes inside. With this concept, we developed a set of distributed algorithms for nodes to cloak their location in both stationary and mobile ad hoc networks. Our strategy is to partition the network domain into a number of safe subdomains that is as small as possible, and let each node take the subdomain where it resides in as its cloaking box. To make subdomains as small as possible, each subdomain is recursively split as long as the resulted subdomains are all safe. We evaluate the performance of our techniques through both mathematical analysis and simulation.

The above cloaking techniques protect nodes' location safety by reducing their location resolution. This, unfortunately, has a significant impact on the geographic routing protocols in ad hoc networks. We show that the routing operation of packet forwarding may allow an adversary to refine a node's location resolution, thus reversing the effect of location cloaking on safety protection. To address this issue, we introduce a new concept called *safe link*. A network link is said to be a safe link if the packet delivery through the link does not allow an adversary to refine the sender and receiver's location resolution. Based on the concept, we first propose a verification technique that allows a node to determine whether or not a link is safe based on the received signal strength. Then, we develop a secure geographic routing protocol (LSR). LSR constructs a routing path using only safe links, and it delivers packets to destination nodes without knowing their accurate location information. To our knowledge, LSR is the first ad hoc routing technique designed with built-in mechanisms to support location safety protection.

The rest of this thesis is organized as follows. We discuss the background and related work in Chapter 2. We present our research for location privacy and location safety protection in Chapter 3 and Chapter 4 respectively. Finally, we conclude this thesis in Chapter 5.

CHAPTER 2. Related work

This chapter surveys existing techniques closely related to our work. These pieces of work were also proposed to protect users' sensitive information from being revealed in the disclosed locations in communications. In Section 2.1, we present regulation and policy-based approaches for safeguarding personal location information. In Section 2.2, we discuss existing techniques proposed for anonymous uses of LBSs. In Section 2.3, we discuss a novel approach which can be used for location privacy protection since it does not need users disclose their location for requesting LBSs. In Section 2.4 and 2.5 we address trajectory perturbation and trajectory anonymization used to prevent a user being identified from a location trajectory revealed in continuous LBSs respectively. In Section 2.6, we present privacy protection in opportunistic sensing and monitoring. In Section 2.7, we shift our focus on ad hoc networks, and surveys existing anonymous routing protocols.

2.1 Regulation and policy-based approaches for location data protection

Various efforts have been made toward safeguarding personal location data. On the legislation front, laws and regulations governing collection and distribution of the location information of wireless subscribers have been or are in the process of being enacted in a number of regions, including the United States, the European Union, and Japan (4). Technical standards for location data transmission have been investigated by the Internet Engineering Task Force's Geopriv working group (5). The group focuses on protocol designs that allow mobile devices to communicate their location in a private and secure fashion.

Several policy-based approaches (e.g., (6; 7; 8)) have also been developed for personal location management, by which users can configure their mobile devices when and to whom their location information can be released. The work in (6) focuses on Automotive Telematics, which are information intensive applications using mobile vehicles as the sensing, computing and communication platform. Dynamic data generated by automobiles creates unique challenges for privacy protection. Unlike static data, which has to be collected only once by any interested party, dynamic data has to be collected repeatedly by a telematic service provider to keep it up-to-date. Thus, continuously reporting location information to the untrustworthy service providers presents a serious threat on users' privacy. In the proposed framework, user can configure their privacy policy to withhold their precise location information, and only report accumulative information. For example, users are not allowed to report their accurate positions, but can report the mileage of their recent travel.

The framework LocServ proposed in (7) lets users apply general policies to control distribution of their information. The policies let users restrictly access to their location information in several ways: 1) Service type. A user only reports her location in requesting certain types of service. 2) Time limit. A user only reports her location at certain time periods, which can vary on workdays and holidays. 3) Location limit. A user only reports her location in specified geographic areas. In addition, a standard xml format is defined for users to submit their privacy policy statement to validators which validate the user preference against system policy.

The work in (8) considers a ubiquitous computing environment which contains a number of services. Each service has a policy proxy which beacons the service description and data collection policies to every user entering the environment. Upon receiving a beacon, the user's personal privacy proxy extracts the policies from the beacon and compares them with the user's own privacy preference. After the comparison, the privacy proxy will help user to decide whether accept or decline the service. Specifically, a service is acceptable only if it can support mechanisms to encode the location information in the collected data and can enforce access restrictions based on the location of the person wanting to use the data.

Although these approaches protect location privacy from various aspects, they do not work when users have to release their location information to a party that is not trustworthy in keeping the data in confidential.

2.2 Anonymous uses of LBSs

Just like regular Internet access, a user may not want to be identified as the subscriber of some LBS, especially when the service is sensitive. To achieve anonymous uses of LBSs, the user's location disclosed to the service provider has to be prevented from being linked to her identity. This problem was first investigated in (9), and a solution is proposed by introducing the concept of location K-anonymity in the context of LBSs. Location K-anonymity demands that the user's location information reported to the service provider should be indistinguishable from at least $K-1$ other users. To achieve K-anonymity protection, the proposed scheme reduces the accuracy of the location information along spatial and/or temporal dimensions before it is disclosed to the service provider. Specifically, when a client requests a service, a quadtree-based algorithm is applied to compute a cloaking box that contains the client and at least $K-1$ others, and then uses this cloaking box as the client's location to request the service. If the resolution is too coarse for quality services, temporal cloaking is applied, i.e., delaying a user's service request. When more mobile nodes come near to the user, a smaller cloaking area can then be computed. The concept of location K-anonymity has since been improved by a series of work. In the rest of this section, we review them in different categories.

Customizable location K-anonymity: The work in (10) address the disadvantages of the cloaking technique proposed in (9). First, (9) uses a system-wide static K value to anonymize all users, which is unrealistic in practice as users tend to have varying privacy requirements under different contexts. Instead, this paper introduces a *customizable* K-anonymity model that allows each user to specify her own value of K. In addition, the quadtree-based algorithm used in (9) suffers from poor service quality because the generated cloaking boxes have low resolution, while in this paper a CliqueCloak algorithm is proposed to compute cloaking boxes as small as possible. The algorithm first constructs a graph, in which a vertex corresponds to a user's location, and an edge between two vertices means the two corresponding users can share a cloaking box. Then, the algorithm searches for cliques of K vertices and the minimum bounding rectangle (MBR) of them is computed as the users' cloaking box.

The work in (11) and (12) consider preventing an adversary from identifying a subject based on her historical moving pattern. These pieces of work introduce the notion of *historical K-anonymity* which defines the requirement to preserve location K-anonymity for a trajectory of service queries submitted by the same user. Based on this definition, a spatial-temporal cloaking algorithm is proposed. For each location update in a trajectory, a three dimensional (two spatial dimension and one temporal dimension) MBR that is crossed by $K-1$ other users is built and these MBRs form a cloaking trajectory. In addition, a probabilistic unlinking technique is presented which prevents adversaries from linking the service requests submitted by a same user at different time.

Query processing with reduced location resolution: In an anonymized service query received by a LBS provider, the location is cloaked into a spatial region. The cloaked location information brings up the challenges of query processing, i.e., how to provide efficient and accurate LBSs based on the knowledge of the spatial region rather than the exact location information. This issue motivates a series of work (13; 14; 15). In (13), a probabilistic model is proposed to process the queries with cloaked location data. It generates *imprecise* answers to the user, each of which is a tuple (S, P) where S is the service content and P is the probability that the answer satisfies the corresponding query. In addition, it defines several metrics for evaluating the quality of a service based on the imprecise answers. These metrics allow a user to decide whether she should adjust her cloaked location in order to obtain a better service.

The work in (14) addresses the challenge of processing queries over private data, i.e., the location information of the querying target is also inaccurate. Specifically, this paper focuses on nearest neighbor (NN) query processing, and a grid-based algorithm is proposed to find the minimum set of candidates for an NN query. The main idea of the algorithm is to initially select a set of *filter* objects that can be used to prune the search over the whole set of object. With the filter objects, the algorithm can identify the spatial region which covers all potential answers to an NN query regardless of the exact location of objects in their cloaking boxes.

The work in (15) focuses on k nearest neighbor (kNN) query processing, and it addresses the challenge of query processing when the cloaked location is a circular region. An algorithm called *CkNN-Circ* is proposed to compute the candidate list of query results. Specifically, the algorithm partitions the circumference of the circular cloaking region into disjoint arcs, and associates to each arc the data objects nearest to it. In addition, it shows that compared with query processing on rectangular cloaking boxes, *CkNN-Circ* has a higher overhead but it can reduce the number of candidates, which means that using circular cloaking box is preferable in the situation when communication cost is more important than processing cost.

Anonymous uses of LBSs in P2P environments: In all the above techniques, a central anonymization server is used as a trusted middle-ware between mobile nodes and service providers. The server tracks the movement of mobile nodes and computes cloaking boxes upon requests. On the other hand, some pieces of work (16; 17; 18; 19; 20) have investigated anonymous uses of LBS in fully distributed mobile peer-to-peer environments. Compared to the centralized framework, the cloaking box computation in P2P system does not rely on the anonymization server. Thus, it is free from the problem of server bottleneck or single point of failure. In (16), before a user send a request to the LBS provider, she finds a group of peers in her neighborhood via single-hop and/or multi-hop routing. Then, the spatial cloaked region that covers the entire group of peers is computed as her location for requesting service. The proposed P2P cloaking algorithm operates in two modes: 1) *on-demand* in which mobile clients execute the algorithm only when they need to request an LBS; 2) *proactive* in which mobile clients periodically look around to find the desired group of peers for cloaking. The on-demand mode has lower power consumption, while the proactive mode can achieve faster service response. The P2P system proposed in (17) is called MOBIHIDE. It manages the mobile users with a hierarchical distributed hash table based on Chord architecture (21). In addition, it employs the Hillbert space-filling curve to map the 2-D user locations to a 1-D Chord space. With the assist of such curve, peers can compute their cloaking boxes by choosing random groups of K users (including the service user) that are consecutive in the 1-D space.

The work in (18; 19) assumes that users' actual positions are publicly known. Thus, to protect users' anonymity, each cloaking box should have the *K-sharing* property, i.e., it must satisfy that 1) at least K users are contained by the cloaking box and 2) at least K of these users share the same cloaking box. The system PRIVE in (18) uses the Hilbert transformation to generate a sorted 1-D sequence of all users. Then, it constructs fixed partitions of K users each, and the minimum bounding rectangle (MBR) of all the users in a partition is the cloaking box for these users. To generate the fixed partitions, PRIVE implements an overlay network which resembles a distributed B^{+}-tree. Since every time the search for a cloaking box starts from the root of the tree, the peer at the root can be overloaded. The work in (19) focuses on the anonymity protection in continuous LBSs, where the adversary can attack users' anonymity based on their historical movement. At the initial timestamp, the proposed algorithm computes the service user a cloaking box which contains at least K users. Then, at a subsequent timestamp, the algorithm computes a new cloaking box which encloses the same set of users. The drawback of this technique is obvious. As users move, the resulting cloaking box can grow very large, leading to prohibitively low service quality.

Despite their difference, all the above P2P systems assume mobile nodes trust each other and require nodes to disclose exact location to their neighbors. In contrast the technique (20) allows nodes to collaborate in computing cloaking boxes without having to reveal their exact location. Specifically, the cloaking process contains two phases. In the first phase, K users (including the service user) are grouped together according to the proximity location information, the distance between a user and its 1-hop neighbors. In the second phase, the bounding box of the group of users is obtained without exposing their accurate positions. To achieve this goal, the proposed technique extend Secure Multi-party Computation (SMC) (22), and a public function is used for all the users in the group to evaluate a candidate bound with their private positions while ensuring no users can learn additional information other than the evaluation results.

All the above techniques described in this section are designed to preserve users' anonymity in service uses, but not their location privacy. Each cloaking box contains a set of users who are currently inside the area. By correlating with restricted spaces, an adversary has the potential to identify all these users. The adversary may not know which of them requests the service, but knows they are all inside the area at the time when the service is requested, thus violating their location privacy. As compared to a single user's location, revealing the presence of a group of people together in a small area is even more threatening – it is well said that ”where you are and whom you are with are closely correlated with what you are doing” (23).

2.3 Non-location exposure uses of LBS

The work in (24) proposes a novel framework to let a user directly download location-based information from a service provider without having to report her location information (either accurate position or cloaking box). The proposed technique does not need an anonymizer. It is based on the theory of Private Information Retrieval (PIR) (25), which allows a user to privately retrieve information from a server database, without letting the server learn what particular information the user has requested. Specifically, this framework implements PIR for nearest neighbor (NN) service requests based on the Hilbert curve and on an R-tree variant. It shows that for a server database which contains n point of interests (POIs), the user can securely find her NN by downloading $O(\sqrt{n})$ of them.

This technique can be used to provide users location privacy protection, since a user does not need to disclose any location information to the service provider. *For each query, however,* *this scheme requires a mobile client to download about the square root of the total amount of data stored in the service provider.* This requirement will present a major burden to a mobile client when the database is large.

2.4 Trajectory perturbation

A user's time-series sequence of location updates creates a trajectory. One can associate each location update with a different pseudonym; but using different pseudonyms or simply not using identifiers at all may not be effective, because successive location samples are highly correlative and could be re-linked using trajectory tracking methods. For example, Multi Target Tracking (MTT) (26) algorithms are a well-studied technique to link subsequent location samples to individual users who periodically report anonymized location information.

Beresford and Stajano first proposed the concept of *mix zone* (27) for trajectory perturbation. Specifically, the network domain is partitioned into application zones and mix zones. Each application zone is a region registered by an LBS, and a user can report her location to the LBS provider whenever she steps into the region. On the other hand, a mix zone is a region not registered by any LBS, and a mobile user does not report its location when she is inside a mixed zone. When there are multiple nodes inside the same mix zone, they exchange their pseudonyms. After exiting the mix zone, these nodes start to use new pseudonyms in location updates, making it hard for an adversary to link incoming and outgoing paths of these nodes.

The above approach is restricted in many applications because it relies on pre-defined spatial regions for pseudonym exchange, and users do not report their locations in mix zones. Hoh and Gruteser (28) proposed another approach through path confusion. A trusted anonymizer is employed to track the movement of mobile users. When it finds some users' paths are within some threshold, it switches their pseudonyms. In addition, it replaces users' original locations with perturbed location samples such that the adversary (applying MTT algorithms) will confuse the tracks and follows the wrong users. Specifically, the proposed technique formalizes the perturbation using an entropy-based model, and uses this model to generate perturbed location samples in order to maximize the chance of confusion.

Despite their difference, these approaches reduce, but cannot prevent, location privacy risks. A partial trace, or just a single location sample, can be sufficient for an adversary to identify a user, thus knowing her whereabouts.

2.5 Trajectory anonymization

Anonymizing the trajectories of a given set of moving objects has been investigated recently. In (29), it is shown that even if the users' identities are removed in a trajectory database, the adversary can still assemble a user's trajectory according to his partial knowledge, i.e., a portion of location samples in the trajectory. The proposed anonymization technique suppresses location information in the original trajectory database wherever privacy leaks occur and converts it to a secure published database. It shows that finding the optimum set of location samples to suppress is NP-hard, and a greedy algorithm is proposed to ensure that the adversary cannot correctly infer the owner of any unknown location sample with certain probability threshold, while maximizing the similarity of the original trajectories to their corresponding transformations.

In (30) Abul et al exploits the impact of position uncertainty on the trajectory anonymization. Due to the imprecision of moving objects' whereabouts (e.g., caused by GPS error), the trajectory of a moving object is no longer a polyline, instead it is a cylindrical volume, where its radius δ represents the possible location imprecision. The main contribution of this paper is the introduction of the concept (K, δ)-anonymity, which anonymizes a trajectory by having at least K moving objects appearing within the cylindrical volume of radius δ of every moving object in the same period of time. To ensure (K, δ)-anonymity a clustering algorithm is proposed based on *space translation*. The paper first proves that the problem of achieving (k, δ)-anonymity by space translation with minimum distortion is NP-hard, and then proposes a greedy algorithm which represents the best trade-off between effectiveness and efficiency.

The work in (31) shares a similar idea of (30), but addresses the impact of trajectory anonymization on applications which rely on the published location data. The anonymized trajectories are 3D cylindrical volumes, but most data mining and statistical applications work on atomic trajectories which consist of location samples with accurate coordinates. To address this issue, a location reconstruction technique is applied, which regeneralizes users' locations by randomly sampling from the cloaking boxes in a trajectory. The experiment results show that although the reconstructed locations are different from the originated ones reported by users, they only produces slight effect on the performance of data mining applications.

Fung et al study the privacy threats caused by publishing RFID data in (32). It is shown that even if names and social security numbers has been removed from the published RFID data, an adversary may identify a target victim's record or infer her sensitive value by matching a priori known visited locations and timestamps. Since RFID data is high-dimensional and sparse, the challenge of anonymizing RFID moving objects data lies on how to improve the cloaking resolution. The proposed LKC-privacy model ensures that every RFID moving path with length not greater than L is shared by at least $K-1$ other moving paths and the confidence of inferring any pre-specified sensitive values is not greater than C.

In all the above trajectory anonymization schemes, each disclosed trajectory is traversed by a set of users at the same time. As such, they share the same problem as the techniques developed for anonymous uses of LBS.

2.6 Privacy-aware opportunistic sensing and monitoring

Opportunistic sensing and monitoring systems (e.g., (33; 34; 35)) have been proposed to leverage users' mobile devices to measure environmental context. In these systems, applications can task mobile nodes in a target region to report context information (e.g., traffic conditions, pollution reading) from their vicinity. However, the location information revealed in the report can put the privacy of users at risk. In (36), Kapadia et al present a privacy-aware opportunistic sensing system called *AnonySense*, which features a two-layer protection of users' privacy. In the first layer, the proposed technique partitions the network domain into many tiles, each being a region that K users typically visit within a short time interval, and lets each node report its location at a granularity of tiles. In the second layer, reports are aggregated to ensure that several reports are combined before sending context information to applications. As a result, this system allows applications to deliver tasks to anonymous nodes and eventually collect reports from anonymous nodes. *It is unclear, though, how mobile nodes are updated with the latest tessellation information. The proposed system also assumes that each report is an independent event. It does not protect privacy when a user's location updates form a trajectory*.

In parallel to the above work, Hoh et al address the location privacy risk in traffic monitoring system (37), trying to shorten the time period that the adversary can successfully track a probe vehicle. To achieve this goal, they proposed a system based on Virtual Trip Lines (VTLs). A VTL is a geographic marker that indicates where a vehicle needs to make a traffic report (with its location). For privacy protection, these markers are placed to avoid particularly sensitive areas. Their distances are also made large enough to prevent a user's consecutive location updates from being re-linked as a trajectory. In addition, a distributed temporal cloaking scheme was proposed which reduces timestamp accuracy to guarantee K-anonymity protection. Specifically, it replaces a VTL timestamp with a time window during which at least K updates were generated from the same VTL. *However, this approach cannot be used for location privacy protection in LBS because the placement of VTLs is pre-determined.*

2.7 Anonymous routing in ad hoc networks

An ad hoc network consists of a set of nodes, either stationary or mobile, which communicate with each other through packet relaying. Because of low cost and easy setup, such networks are often deployed in hostile environments such as enemy terrain where no communication infrastructures exist. In addition, many applications developed in ad hoc networks are security sensitive, such as military battlefield operations, homeland security scenarios, and rescue missions. As a result, security issues in ad hoc routing have drawn intensive attention recently.

Traffic analysis is one of the most serious security attacks against ad hoc routing. By tracing the network routes and the en-route nodes, the adversary can infer sensitive information about the applications and the communicating parties, such as nodes' identities, locations and moving patterns. To thwart the above threat, a number of anonymous routing protocols have been developed in literature. In this section we review them in different categories.

Protecting traffic pattern in ad hoc routing: The adversary in some ad hoc network has global observation of the network traffic and wants to identify the routing paths of data packets. Knowing traffic pattern of communications the adversary can deduce sensitive information such as military actions. To address this threat, Jiang et al in (38) proposed Dynamic Mix Method (DMM) which is extended from the traditional Chaum's mix method (39) used to hide sender and receiver of email. They assume that there are a number of mix nodes distributed over the whole network. When a node has a packet pending, it searches for the mix node to forward the packet by executing a mix discovery protocol called Optimal Mix (OM). This protocol is similar to Dynamic Source Routing (DSR) (40), and it constructs a routes between source and destination with a set of mix nodes in between. Since each mix node re-encrypts the packets it forwards, the routes are untraceable for the adversary.

Kong et al in (41) proposed an anonymous on-demand routing protocol called ANODR which does not rely on the predetermined mix nodes in the network. For route discovery, an Onion-based (42) routing algorithm is proposed to construct an anonymous route between the source and destination. During data delivery, each hop en route is associated with a random route pseudonym. As a result, the data packets on different routing paths are mixed at each forwarding node so that it is hard for the adversary to find out where a packet flow comes from and where it goes to.

Different from the above two papers, the work in (43) address the anonymous routing in MAC-layer communications. It considers the case that there co-exist multiple groups of nodes in the network. Nodes within a same group communicate with each other and they do not want to be identified by nodes of other groups. Thus, the challenge is how to construct communication links between group members without revealing their identities. The proposed technique employs a trusted authority who assigns each node a sufficiently large set of collision-resistant pseudonyms. These pseudonyms are chosen to substitute real ID in communications in order to prevent adversary from tracking. In addition, a pairing-based secret handshake scheme is used to anonymously authenticate two nodes in the same group and establish the corresponding communication link.

Protecting source location in ad hoc routing: Another track of research (44; 45; 46) considers the sensor networks that are deployed for detecting and monitoring valuable sources. The sensors around a source continuously send reports to the sink, while an adversary can trace the data flow hop by hop backward to discover the source. In (44; 45), Ozturk et al show that a simple strategy to address this issue is to let some fake sources generate messages at the same time in order to confuse the adversary. However, it consumes too much energy and is not suitable for sensor networks. On the other hand, they introduce phantom flooding which attempts to direct messages from a source to different locations of the network so that the adversary cannot receive a steady stream of messages to track the source. Specifically, a message is first unicasted in a random fashion (referred to as random walk) within the first h hops, and then the message is flooded to the sink. Thus, each message traverses a different path to the sink.

The work in (46) addresses that the scheme based on random walk in the above two papers can prevent adversary from tracing back to the source, but the generated routing path may be much longer than the shortest path to the sink, which leads to undesirable delivery time. To cope with this problem, this paper proposes the cyclic entrapment method (CEM). In this approach, several loops are generated after the deployment of the sensor network and before sources send any messages to the base station. When a message is being routed along a path from the source to the sink and it encounters one of these pre-configured loops, the encountered loop will be activated and will begin cycling fake messages around the loop. Therefore, when an adversary arrives at this spot, he cannot distinguish the real message and fake messages, and then becomes unable to track back further.

Protecting nodes' location in position-based routing: In general, the techniques in the above two categories consider topology-based routing, and they assume the adversary locates nodes based on their signal strength (e.g., using triangulation (47; 48; 49)). In contrast, the work in (50; 51) focuses on position-based routing, and it assumes the adversary can compromise some node and thus access the location information of nodes that they disclose in routing protocols. Since location information is the driving factor in position-based routing, it is more challenging to prevent the adversary from deriving nodes' sensitive information from their locations.

The work in (50) aims to unlink the nodes' identities to their locations during geographic routing. An anonymous greedy forwarding (AGFW) technique is proposed to achieve this goal. In this approach, each node maintains an anonymous neighbor table, where the neighbors' identities are not known but pseudonyms are used instead. During data delivery, the packet header includes a *trapdoor* field which can only be opened by the destination. When a node receives a packet, it can determine if it is the destination using the trapdoor. Otherwise, it forwards the packet to its neighbor closest to the destination.

AGFW unlinks a node's identity to its locations using pseudonyms. However, similar to the traditional geographic routing protocols (e.g. GPSR (3)), it requires each node periodically broadcast its accurate position to its neighbors. In (51), Wu et al argue that such periodical heart-beat makes a node highly traceable and it's much easier to obtain a node's ID based on its trajectory. To address this issue, they propose an anonymous geographic routing protocol called AO2P, which does not require the time-based position report. During packet forwarding, only the destination's position is open, and a contention-based scheme is used to choose the next hop. Specifically, once a previous hop sends out a packet, its neighbors compute their distance to the destination, and the ones closer to the destination have a higher probability to win the contention and become the forwarder. This approach can suppress the location information revealed in routing, meanwhile it can generate routes with small number of hops.

Despite their differences, anonymous routing techniques aim to prevent an adversary from identifying important nodes in the network. They do not deal with the safety threat imposed by the exposure of nodes' location information.

CHAPTER 3. Location privacy protection in LBSs

In this chapter, we present our research that investigates location privacy protection in the context of location-based services. As we discussed in Chapter 1, location information exposed in service requests presents users significant privacy threats. To address this issue, our research objective is to prevent a user's location information, either a single location sample or a time-series sequence of them, from being correlated with restricted spaces to derive *who's where at what time*. Towards this goal, we make following contributions.

1) We propose using historical location samples, each called a *footprint*, for location depersonalization. A location or a trajectory with K different footprints means it has been visited by K different people. Even if an adversary manages to identify all these people, he will not know who was there at the time of the service request, thus preserving the user's location privacy. 2) We address the challenge of modeling location privacy requirement. With the traditional K-anonymity model, a user needs to specify a value of K to request a desired level of privacy protection. This is problematic, because privacy is about feeling, and it is difficult to scale one's feeling using a number. Our solution circumvents this problem by allowing a user to identify a spatial region, such as a shopping mall, which she would feel comfortable that it is reported as her location should she request a service inside it. This region is then used as her privacy requirement – each location disclosed on her behalf needs to be at least as popular as that region. Compared to choosing a number of K, this *feeling-based* strategy provides a much more intuitive way for users to express their privacy requirement. 3) With the above ideas in place, we present a suite of algorithms for efficient location cloaking. These techniques allow users to entertain LBSs, either sporadic or continuous, while providing them

a desired level of location privacy protection. We evaluate the performance of our techniques via simulation under various conditions using location data synthetically generated based on real road maps. For feasibility and practicality evaluation, we have also implemented an experimental prototype that supports location privacy aware uses of LBSs.

3.1 Feeling-based privacy modeling

Our research aims at preventing location information disclosed for LBS requests from being used to derive who's where at what time. More specifically, given a cloaking box b reported at time t for an LBS, we want to prevent an adversary from identifying who was in b at time t by correlating b with restricted spaces such as home and office, which are public-accessible information. Unlike existing work aimed at supporting anonymous service uses, we do not consider observation attack. When a user is under direct observation, she does not have location privacy anyway. As mentioned early, ensuring each cloaking box contains a number of current users can protect users' anonymity in service uses, but not their location privacy. To circumvent this problem, our idea is to leverage historical location samples for cloaking. Given a cloaking box that has been visited by a number of people, even if an adversary manages to identify all these people, he will not know who was in the box at the time of service request.

To customize the level of privacy protection, a user can specify a value of K: each cloaking box disclosed on her behalf must have at least K different visitors. A larger value of K makes it harder to link the box to some specific user, thus meaning a higher level of protection. While this traditional K-anonymity model (52; 53) is simple to implement, choosing an appropriate K value can be difficult. Why would a user feel that her privacy is well-protected if $K = 20$, but not if $K = 19$? Ultimately, privacy is about feeling, and it is awkward for one to scale her feeling using a number. A user can always choose a large K to ensure a sufficient privacy protection, but this will result in unnecessary reduction of location resolution. A very coarse location will make it difficult to provide a meaningful LBS. In addition to this inherent K-anonymity problem, another issue has to do with the robustness in protection. Ensuring each location has been visited by at least K different users may not provide privacy protection at the level of K. Indeed, it can achieve so only when these K users have an equal chance of visiting the region, i.e., they leave the same amount of footprints in the area. In reality, a spatial region may be visited by many people, but some may have a dominant presence. For example, if an LBS is requested from an office, then the office staff is more likely to be the service requestor, even if the office has many visitors.

Instead of a K value, a user can specify a spatial region, which we will refer to as a *public region*, and request that the location disclosed on her behalf be at least as popular as that region. For example, a user may choose a shopping mall in town as her public region. As compared to choosing a number, choosing a public region provides a much more intuitive way for a user to express her privacy requirement. We refer to this approach as *feeling-based* privacy modeling. The challenge now is how to measure the popularity of a spatial region. As mentioned above, simply using the number of visitors for popularity measurement is not sufficient, because the presence of these visitors in the space may not be uniformed. To address this problem, we borrow the concept of *entropy* from Shannon's information theory (54). Suppose we can collect location samples from cellular phone users. These location samples, each called a *footprint*, can then be used to measure the popularity of a spatial region as follows.

Definition 1. *Let R be a spatial region and $S(R)$ = $\{u_1, u_2, \cdots, u_m\}$ be the set of users who have footprints in R. Let n_i $(1 \leq i \leq m)$ be the number of footprints that user u_i has in R, and $N = \sum_{i=1}^{m} n_i$. We define the* entropy *of R as $E(R) = -\sum_{i=1}^{m} \frac{n_i}{N} \log \frac{n_i}{N}$, and the* popularity *of R as $P(R) = 2^{E(R)}$.*

The value of $E(R)$ can be interpreted as the amount of additional information needed for the adversary to identify the service user from $S(R)$ when R is reported as her location in requesting an LBS. According to the above definition, we have $1 < P(R) \leq m$. $P(R)$ has the maximum value m when every user in $S(R)$ has the same number of footprints in R. On the other hand, $P(R)$ has the minimum value when one user in $S(R)$ has $N-m+1$ footprints in R while each of the rest has only 1. We have the following two observations. First, $P(R)$ is higher if m is larger. In other words, a region is more popular if it has more visitors. Second, $P(R)$ has a lower value if the distribution of footprints is more skewed. If some users are dominant in the region, $P(R)$ will be much less than m. In this case, R needs to be enlarged to contain more users in order to have a required popularity.

Let R be a user's public region. When the user requests a sporadic LBS, where the request can be seen as an independent event, we can find a cloaking box that 1) contains the user's current position, 2) has a popularity that is no less than $P(R)$, and 3) is as small as possible, and then report this box as the user's location. When the user requests a continuous LBS, a time-series sequence of cloaking boxes will be reported that form a trajectory. In this case, simply ensuring that each cloaking box has a popularity no less than $P(R)$ does not protect the user's location privacy at her desired level. This is due to the fact that the adversary can narrow down the list of possible service users by finding the common visitors of these cloaking boxes. To prevent such attack, we must use the footprints of the common set of users, instead of all visitors of the regions, in computing the popularity of each cloaking box. We define the popularity of a spatial region with respect to a given set of users as follows.

Definition 2. *Given a spatial region R, and a user set $U = \{u_1, u_2, \cdots, u_{m'}\} \subseteq S(R)$, the* entropy *of R with respect to U is $E_U(R) = -\sum_{i=1}^{m'} \frac{n_i}{N'} \log \frac{n_i}{N'}$, where n_i is the number of footprints that u_i has in R, and $N' = \sum_{i=1}^{m'} n_i$. The* popularity *of R with respect to U is $P_U(R) = 2^{E_U(R)}$.*

When a sequence of cloaking boxes are generated on a user's behalf, we must ensure that the popularity of each cloaking box with respect to the common set of visitors is no less than that of the user's public region. In other words, the trajectory formed by these cloaking boxes must be a *P-Popular Trajectory* (PPT), which is formally defined below:

Definition 3. *Let $T = \{R_1, R_2, \cdots, R_n\}$ be a sequence of cloaking boxes generated for a user, and $S(R_i)$ ($1 \leq i \leq n$) the set of people who have footprints in R_i. We say T is the user's PPT if for each R_i, it satisfies that 1) R_i covers the user's position at the time when R_i is disclosed, and 2) $P_S(R_i) \geq P(R)$, where $S = \bigcap_{1 \leq i \leq n} S(R_i)$ and R is the public region specified by the user.*

Given a trajectory $T = \{R_1, R_2, \cdots, R_n\}$, we define its resolution to be $|T| = \frac{\sum_{i=1}^{n} Area(R_i)}{n}$, where $Area(R_i)$ denotes the area of box R_i. For location privacy protection, a trajectory formed by the location samples disclosed on a user's behalf must be a PPT. Meanwhile, its resolution needs to be as fine as possible to guarantee the quality of the required LBS services.

3.2 Location cloaking techniques

With the feeling-based privacy model in place, we present our location cloaking techniques for location privacy protection in this section. We first give an overview of the system architecture and database indexing. Then, we present a suite of cloaking algorithms to support location privacy protection in both sporadic LBSs and continuous LBSs. For the latter, we discuss trajectory cloaking under two scenarios: 1) *In-advance* cloaking when the client's moving trajectory is predetermined before the service session begins; 2) *On-the-fly* cloaking when the moving trajectory is unknown beforehand.

3.2.1 System overview

Similar to existing work (e.g., (9), (10), (14)), we assume mobile clients communicate with LBS providers through a trusted central location depersonalization server (LDS) managed by the clients' cellular service carriers, as shown in Figure 3.1. For LBSs that require user authentication (e.g., for service charges), we assume anonymous authentication (e.g., (55), (56), (57)) is used. The carriers offer the depersonalization services as a value-added feature to their clients, and supply the LDS with an initial footprint database that contains location samples collected from their clients (e.g., through regular phone calls). These location samples will be used to compute the popularity of a spatial region and for trajectory cloaking. Hereafter, we will use terms location sample and footprint interchangeably. The footprint database will be expanded with the location data obtained from mobile users in their requests of LBSs.

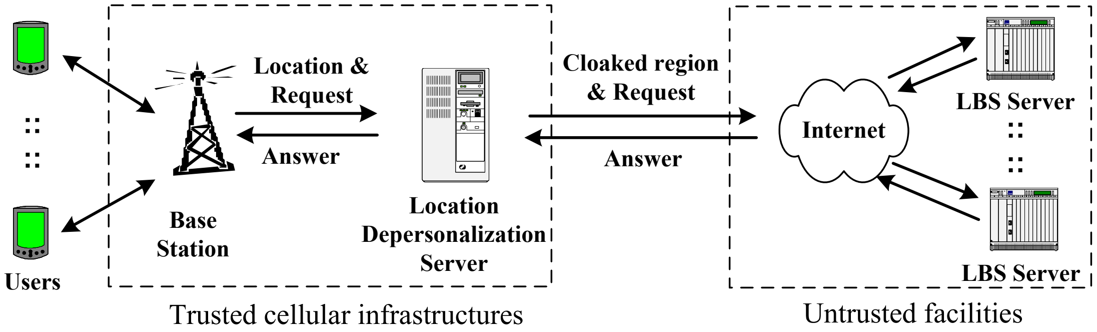


Figure 3.1 System architecture

We assume the adversaries have access to anonymous location data collected by LBSs and are interested in finding who was where at what time by correlating such information with restricted spaces such as office and home addresses. For LBSs, which may involve a large number of users and have a global coverage, such restricted space identification is probably the most realistic and economic way for location privacy intrusion. Unlike service anonymity protection, we do not consider observation attack (9). If an adversary has direct observation over the region where a user locates, the user does not have location privacy anyway.

Our research considers location cloaking for both sporadic LBSs and continuous LBSs. In the former, each location update of a user is independent to others. The LDS needs to cloak it with a region no less popular than the user specified public region. In the latter, location updates of a user in a service are correlated and they form a trajectory. The LDS needs to cloak it with a PPT defined in Section 3.1. To facilitate the cloaking process, we use the following structure of manage the historical location data.

We partition the network domain recursively into cells in a quad-tree style. The partitioning stops when the size of cells becomes less than a threshold (our implementation sets each cell to be at least $100 \times 100\, meter^2$). All the cells generated in the partitioning form a pyramid structure as shown in Figure 3.2. Suppose the partitioning stops at the h^{th} recursion, then the pyramid has a height of h. The top level in the pyramid is level 1 and has only one grid cell that covers the whole network domain. Each grid cell except the ones at the bottom level is composed of four cells at the next lower level, which we refer to as its child cells.

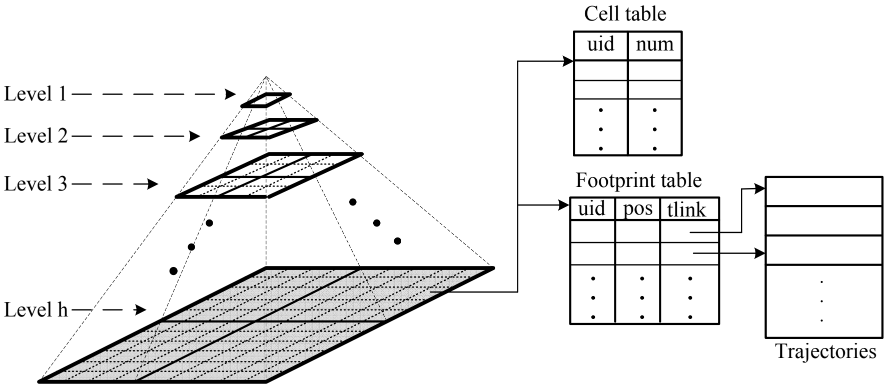


Figure 3.2 Data structure

Each cell at the bottom level h keeps a footprint table and a user table. The footprint table stores the footprints the cell contains, and each tuple of the table is a record of $(uid, pos, tlink)$, where uid is the identity of the mobile user that a footprint belongs to, pos is the coordinates of a footprint, and $tlink$ is a pointer that links to the corresponding trajectory stored in the database. The user table records the number of footprints a user has in the cell, and each tuple of the table is a record of (uid, num), where num is the number of footprints the user has in the cell. For each cell not at the bottom level, we also keeps a user table, which is derived from the user tables corresponding to its four child cells.

3.2.2 Single location sample cloaking

For instant LBSs, a mobile client configures her privacy requirement by specifying a public region R, and report it with its location p to the LDS. In response, the LDS calculates the popularity of the public region $P(R)$, and computes a cloaking box which contains c and has a popularity no less than $P(R)$. For the sake of service quality, the size of the cloaking box should be as small as possible. Cloaking footprints is different from cloaking neighboring users since different footprints may belong to the same user. A cloaking box containing K footprints may not have a popularity of K. Thus, existing cloaking techniques using current neighbors cannot be directly applied for cloaking footprints. In this section, we present a heuristic algorithm to the cloaking box with a resolution as fine as possible. The pseudo code is given in Algorithm 1.

Algorithm 1 Cloak(p, $P(R)$)

```
{Phase I: compute searching box}
$b' \leftarrow$ the cell that contains $p$
**while** $P(b') < P(R)$ **do**
    {get cells at bottom level adjacent to $b'$}
    $E \leftarrow Adjacent(b', h)$
    {merging the cells in $E$ with $b'$}
    $b' \leftarrow b' \bigcup E$
    update user table of $b'$
**end while**
{Phase II: compute cloaking box}
$k = \lceil P(R) \rceil$
$F \leftarrow k$ closest footprints belonging to different users
$U \leftarrow$ corresponding users of footprints in $F$
$b \leftarrow$ MBB of footprints in $F$
**while** $P(b) < P(R)$ and $U \neq S(b')$ **do**
    $k' = \lceil P(R) - P(b) \rceil$
    **for** $i = 1$ to $k'$ **do**
        $f \leftarrow$ the footprint closest to $p$ not belonging to $U$
        $F \leftarrow F \bigcup \{f\}$
        $U \leftarrow U \bigcup$ {user corresponding to $f$}
    **end for**
    $b \leftarrow$ MBB of footprints in $F$
**end while**
**if** $U = S(b')$ and $P(b) < P(R)$ **then**
    $b \leftarrow$ MBB of all footprints in $b'$
**end if**
return $b$
```

The cloaking algorithm consists of two phases: (I) the LDS finds a searching box b' with a popularity no less than $P(R)$ according to the user tables of the cells in the bottom level; (II) using the footprints in the searching box b', the LDS computes the cloaking box b which has a popularity no less than $P(R)$ and has an area as small as possible. In phase I (line 1-9), the LDS first sets the searching box b' as the cell at the bottom level of the pyramid which contains the client's position p, and computes its popularity $P(b')$ according to its user table. If $P(b') < P(R)$, which means b' is not popular enough, the LDS expands the searching box b' by merging it with its adjacent cells at the bottom level, and updates its user table by computing the union of the user tables of the cells in b'. The searching box is expanded repeatedly until its popularity is no less than $P(R)$ (line 3-9).

In phase II (line 10-27), the LDS first finds $k = \lceil P(R) \rceil$ closest footprints to p each belonging to a different user, and records the footprint set as F and the corresponding user set as U. Then the LDS computes the minimum bounding box (MBB) b of the k footprints and compute its popularity. If $P(b) < P(R)$, the LDS finds $k' = \lceil P(R) - P(b) \rceil$ closest footprints to p each belonging to a different user who is not in U, and updates F and U by adding these footprints and corresponding users. Next, the LDS recalculates b as the MBB of the footprints in F and the above process is run repeatedly until $P(b) \geq P(R)$, or $U = S(b')$ which means all people visited b' have been counted (line 15-23). If $U = S(b')$ but $P(b) < P(R)$, the LDS computes the cloaking box b as the MBB of all footprints in b'.

3.2.3 In-advance trajectory cloaking

In a continuous LBS, a mobile client makes a time series sequence of location updates which form a trajectory. In response, the LDS needs to generate a cloaking box for each location update, and make sure these cloaking boxes together form a *P-Popular Trajectory* (PPT). Comparing with single location cloaking, trajectory cloaking is more challenging. As mentioned in Section 3.1, simply having each cloaking box as popular as the public region cannot ensure the protection of the client's location privacy. Instead, the LDS has to make each cloaking box popular enough with respect to a common set of user. In the rest of this section, we focus on how to address this challenge in trajectory cloaking. We first discuss in-advance cloaking scenario, in which the service user knows her moving route beforehand.

As the moving route is predetermined, when the service user submits the service request, she also reports a *base trajectory* $T_0 = \{p_1, p_2, \cdots, p_n\}$ to the LDS, in which p_i is a location sample on her route where she will update her location. In response, the LDS computes a PPT $T = \{b_1, b_2, \cdots, b_n\}$, where b_i is a cloaking box containing p_i. During the service session, when the client arrives p_i, b_i will be reported to the LBS provider in requesting service. In this subsection, we present a trajectory cloaking algorithm that generates a PPT with resolution as fine as possible.

According to its definition, to generate a PPT, the LDS has to find a common set of users, which we refer to as *cloaking set*, and use their footprints to compute cloaking boxes in the PPT. It may first appear that the LDS can determine the cloaking set, denoted as S, by finding the set of users who have footprints closest to the starting point of the service user. This simple solution minimizes the size of the first cloaking box. However, as the service user moves, the users in S may not have footprints that are close to her current position. As a result, the size of the cloaking boxes may become larger and larger, making it difficult to guarantee the quality of LBS. To address this challenge, our idea is to find those users who have footprint close to all the location samples in the base trajectory T_0 and use them to create the cloaking set. Based on this idea, we develop the following approach to compute the cloaking set. The pseudo code is given in Algorithm 2. The LDS first finds out all cells at the bottom level of the pyramid that overlap with T_0's location samples. These cells, denoted as $b'_i (1 \leq i \leq n)$, are marked as *searching boxes* (line 2-4). According to the cells' user tables, the LDS then retrieves the users, say U, that have visited all of these searching boxes, and compute their popularity with respect to U. Among all the searching boxes, if there exists at least one whose popularity with respect to U is less than $P(R)$, the LDS expands the searching scope by merging each searching box and its adjacent cells together as a new searching box. Then, according to the user tables of the searching boxes, the LDS recalculates the user set U who have visited all of them. This process is repeated until all searching boxes' popularity with respect to U is no less than $P(R)$ (line 6-15). Then, U is selected as the cloaking set.

Algorithm 2 Select-In-Advance(T_0, $P(R)$)

```
 1: $\{$Base trajectory $T_0 = \{p_1, p_2, \cdots, p_n\}\}$
 2: for $i = 1$ to $n$ do
 3:     $b'_i \leftarrow$ cell that contains $p_i$
 4: end for
 5: $U \leftarrow \bigcap_{1 \leq i \leq n} S(b'_i)$
 6: while $\exists i \in [1, n] | P_U(b'_i) < P(R)$ do
 7:     for $i = 1$ to $n$ do
 8:         $\{$get cells at bottom level adjacent to $b'_i\}$
 9:         $E_i \leftarrow Adjacent(b'_i, h)$
10:         $\{$merging the cells in $E_i$ with $b'_i\}$
11:         $b'_i \leftarrow b'_i \bigcup E_i$
12:         update user table of $b'_i$
13:     end for
14:     $U \leftarrow \bigcap_{1 \leq i \leq n} S(b'_i)$
15: end while
16: return $U$
```

After choosing the cloaking set, the LDS generates a PPT by computing a cloaking box for each location sample in T_0 using the footprints of users in U. Given a location $p_i, 1 \leq i \leq n$, the LDS can simply call the function in Algorithm 1 to compute the cloaking box. The only difference is that the LDS counts only footprints of users in U.

3.2.4 On-the-fly trajectory cloaking

In this subsection we discuss on-the-fly cloaking scenario, in which the service user does not know her moving route beforehand. When the service user requests an LBS, she also informs the LDS a *travel bound* B, a rectangular spatial region that bounds her travel during the service session. In response, the LDS randomly generates a service session ID and contacts the service provider. After establishing a service session, the service user periodically reports her current location to the LDS. For each location update, the LDS computes a cloaking box which contains the service user's current location, and exports this box along with the session ID to the corresponding LBS provider. The LDS must ensure that the trajectory created by the sequence of cloaking boxes is a PPT that satisfies the user's privacy requirement.

Similar to in-advance cloaking, the LDS has to find a *cloaking set* in order to generate the PPT. But on-the-fly cloaking is more difficult. The challenge is that the service user's route is not predetermined, and thus the LDS cannot figure out whose footprints will be closer to the service user during her travel. Therefore, it is hard to find a PPT with a fine resolution. To address this challenge, our idea is to find those users who have visited most places in the service user's travel bound B and use them to create the cloaking set. As these users have footprints spanning the entire region B, it will help generate a PPT with a fine resolution.

Recall that in Section 3.2.1 we present a pyramid structure which manages all historical location data in the network domain. Here we say a user is *l-popular within B*, if she has footprints in every cell at level l that overlaps with B. According to the pyramid structure, cells at level with a larger l have a finer granularity. This implies that given an l-popular user, the larger the value of l is, the more popular the user is. Figure 3.3 shows an example in which a network domain is partitioned into a 4-level pyramid (There are 1, 4, 16, 64 cells at each level respectively from top to bottom). It also shows a travel bound B and the footprints inside it. The footprints in different colors belong to different users. u_1, u_2, and u_3 are three 2-popular users within B because they have footprints in the two cells at level 2 of the pyramid which overlap with B; u_2, u_3 are two 3-popular users within B since they have footprints in all four cells at level 3 that overlap with B; only u_3 is 4-popular since she is the only one who has footprints in all the sixteen cells at level 4 that overlap with B.

Based on the above definitions, we now present a simple and effective algorithm that can find an appropriate cloaking set which can assist generating a PPT with fine resolution. The pseudo code is given in Algorithm 3. In this algorithm, the LDS sorts the users in $S(B)$ according to their popularity at level l, and selects the most popular users in $S(B)$ as the cloaking set, starting from the bottom to top of the pyramid. Let C_l denote the set of cells at

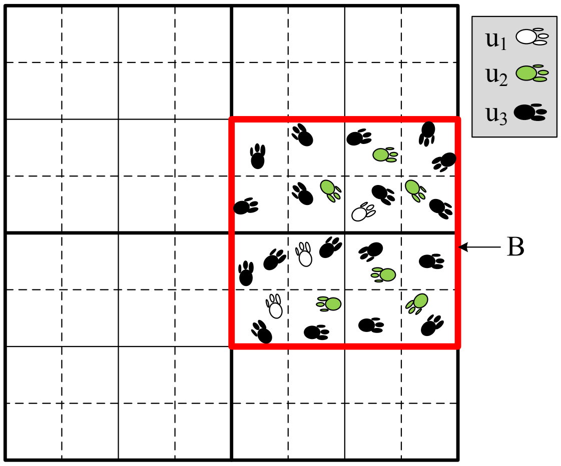


Figure 3.3   A travel bound and footprints inside

level l in the pyramid, C'_l the set of cells in C_l that overlap with B, and S_l the set of users who are l-popular within B. The LDS first finds S_h. Since level h is the bottom level, these users are the most popular users in $S(B)$. To find S_h (i.e., the users who have visited all the cells in C'_h), the LDS simply joins the user tables of these cells on column uid (line 6-7). Next, the LDS computes the popularity of B with respect to S_h using their footprints in B. If the popularity $P_{S_h}(B)$ is less than $P(R)$, it means that cloaking with the footprints of the users in S_h cannot provide the desired level of privacy protection for the service user. In this case, the LDS considers the cells one level higher, i.e., level $h - 1$ (line 9), and computes S_{h-1} and $P_{S_{h-1}}(B)$ similarly. This procedure is repeated until at some level l the popularity $P_{S_l}(B)$ is no less than $P(R)$ (line 3-10).

The above algorithm goes over the pyramid level by level from bottom to top. If a user is l-popular within B, it must also be $(l-1)$-popular within B. Thus, each time the algorithm checks the cells at a higher level, the cloaking set is expanded to include more users. As long as $P(R) \leq P(B)$ (i.e., a user's public region is at most the same popular as that of her travel bound), the algorithm will find a sufficient number of visitors within B for the cloaking set. In the worst case, all users in $S(B)$ are included in the cloaking set. On the other hand, if $P(B) < P(R)$, the LDS does not need to find a cloaking set. It can simply compute a spatial

Algorithm 3 Select-On-The-Fly(B, $P(R)$)

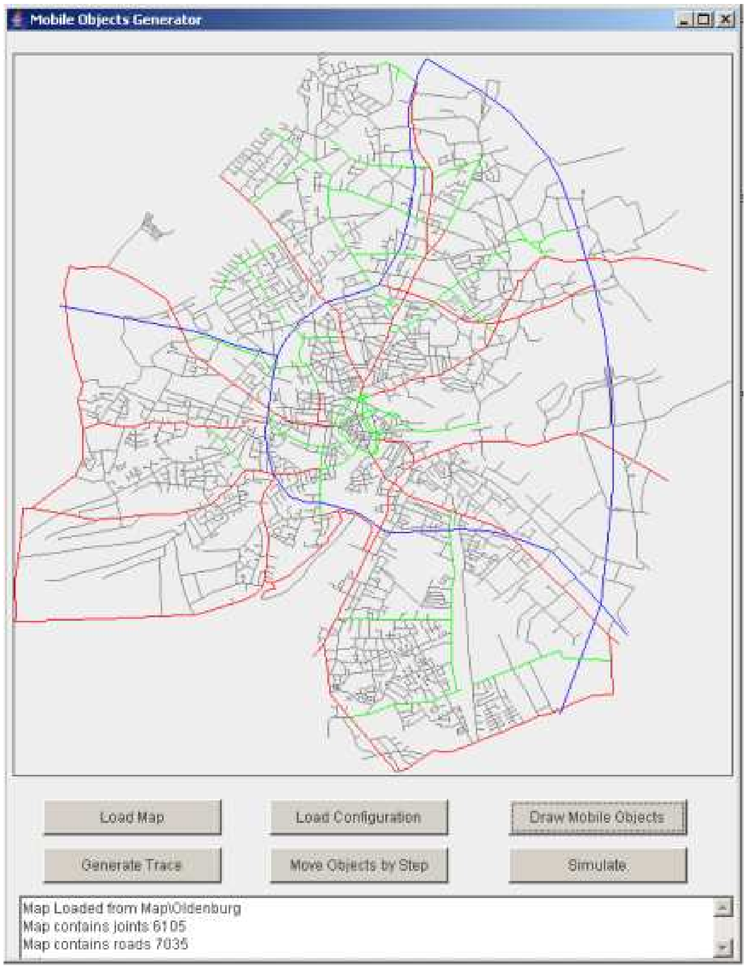
```
$U \leftarrow \emptyset$ {$U$ keeps the cloaking set}
$l \leftarrow h$
**while** $U \subset S(B)$ **and** $P_U(B) < P(R)$ **do**
    {Get cells at level $l$ overlapping with $B$}
    $C'_l \leftarrow Overlap(C_l, B)$
    {Join user tables of $C'_l$ by column $uid$}
    $T \leftarrow Join(C'_l, uid)$
    $U \leftarrow S_l \leftarrow T.uid$
    $l \leftarrow l - 1$
**end while**
return $U$
```

region that contains B and has a popularity no less than $P(R)$, and always report this region as the user's location as long as it moves inside B.

Similar to in-advance cloaking, during the service session, the LDS will generate a PPT by computing a cloaking box for each location update from the service user using the footprints of users in the cloaking set U. Given a location p, the LDS can simply call the function in Algorithm 1 to compute the cloaking box. The differences lie on two points. First, the LDS only counts footprints of users in U. Second, when expanding the searching box (Algorithm 1 line 4-7), only the adjacent cells in B are merged.

3.3 Simulations

In this section, we evaluate the effectiveness of the proposed technique under various conditions using location data synthetically generated based on a real road map. We modify the simulator *Network-based Generator of Moving Objects* (58) to generate mobile nodes and simulate their movement on the real road map of Oldenburg, Germany, a city about 15×15 km^2. The GUI of the simulator is shown in Figure 3.4. We extract four types of roads from the road map, primary road (interstate expressway), secondary road (state road), connecting road and neighborhood road as defined in census TIGER/Line (59). In our simulation, mobile nodes change their speeds at each intersection, and the moving speed on a road follows a normal distribution determined by the road type. The mean speeds and the standard deviations of moving speeds on all road types are listed in Table 3.1. We generate a footprint database that contains a certain number of trajectories, which are assigned to 2000 users. The number of trajectories each user has follows a normal distribution with a standard deviation 0.1. These trajectories are indexed using the grid-based approach discussed in the Section 3.2.1. We evaluate the performance of our techniques for both single location cloaking and trajectory cloaking, which we will present in the rest of this section respectively.



```
Map Loaded from Map\Oldenburg
Map contains joints 6105
Map contains roads 7035
```

Figure 3.4 Generator of moving objects

Table 3.1 Traffic parameters

Road type	Mean speed	Standard deviation
Primary	$100km/h$	$20km/h$
Secondary	$60km/h$	$15km/h$
Connecting	$45km/h$	$10km/h$
Neighborhood	$30km/h$	$5km/h$

3.3.1 Single location sample cloaking

In this study, we investigate the performance of single location cloaking algorithm. For each simulation, we generate 300 service requests. Every service request contains the service user's position which is randomly selected from the location samples in the database, and a public region which is a square region that contains the position. For each request, a cloaking box is computed using Algorithm 1, and exposed as the service user's location. We are interested in two performance metrics. One is *cloaking area*, defined to be the average area of cloaking boxes generated for the set of request in a simulation. The other one is *privacy level*, defined to be the average popularity of the cloaking boxes. We varied the size of a public region, measured by the side length of the square region, from 50 to 250 meters, and plotted the performance results in Figure 3.5. Figure 3.5 (a) shows that when the size of the public region increases, the average cloaking area increases. This is due to the fact that a larger public region is likely to contain more people's footprints and have a higher popularity. Thus, a cloaking box needs to be larger to satisfy a higher level of privacy requirement. In Figure 3.5(b), besides the privacy level, we draw another line which indicates the average number of users who have visited the cloaking box computed for a request. As we can see, both lines are incremental with respect to the size of public region, but the number of visitors is always larger than the popularity of the cloaking box. This result shows that the number of visitors of a region is not a good measure of the privacy level the region can provide for the service user.

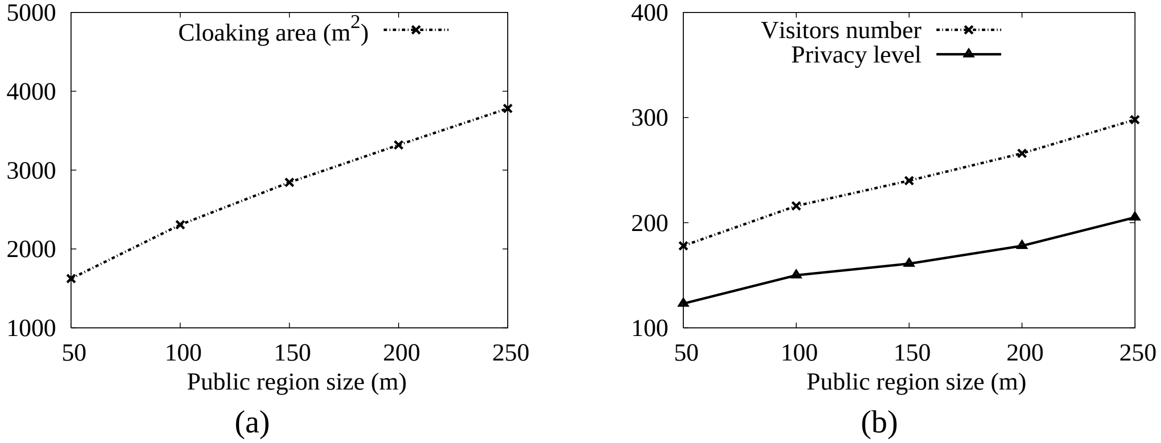


Figure 3.5 Performance of single location cloaking on the effect of privacy requirement

3.3.2 Trajectory cloaking

We evaluate the performance of both in-advance cloaking and on-the-fly cloaking. For comparison purpose, we have implemented two other approaches. The first one, which we will refer to as *Naive*, assumes the location updates made a service user are independent to each other. For each location update, Naive just applies Algorithm 1 to compute a cloaking box, and reports it as the service user's location in her service request. Note that this scheme may not protect a user's location privacy at her desired level when she makes a time-series sequence of location updates. The second approach is referred to as *Plain* hereafter. This scheme determines the cloaking set for the service users by finding the footprints closest to her start position. After fixing the cloaking set, Algorithm 1 is applied to compute the cloaking boxes for the service user during her entire service session. For each simulation, we generate a number of service sessions. Every session contains a user specified public region, a travel bound, and the user's moving route which is the fastest path between a start and a destination selected in the travel bound. For in-advance cloaking, we select a location sample every 100 meters along the moving route and these samples form the user's base trajectory, and the PPT is computed according to the base trajectory using Algorithm 2. For on-the-fly cloaking, we assume the moving routing is un-predetermined, and the PPT is computed according to the travel bound using Algorithm 3.

In our study, we are mainly interested in the following two performance metrics. One is *cloaking area*, defined to be the average area of cloaking boxes in a cloaking trajectory. The other one is *protection level*. Given a cloaking trajectory, we measure its protection level using the ratio between the average popularity of its cloaking boxes with respect to the common set of users who have visited all of them and the popularity of the user specified public region. Clearly, the protection level must be at least 1, otherwise the cloaking trajectory fails to protect the service user's location privacy at the required level. In the following subsections, we report how the performance of the techniques is affected by various factors.

Table 3.2 Simulation parameters

parameter	range	default	unit
Users #	2000	2000	$unit$
Public region size	50 - 250	150	$meter$
Trajectory #	$100K - 300K$	$200K$	$unit$
Travel bound size	$2 - 6$	4	km
Travel distance	$2 - 6$	4	km
Service requests #	300	300	$unit$
Minimum cell size	100×100	100×100	$meter^2$

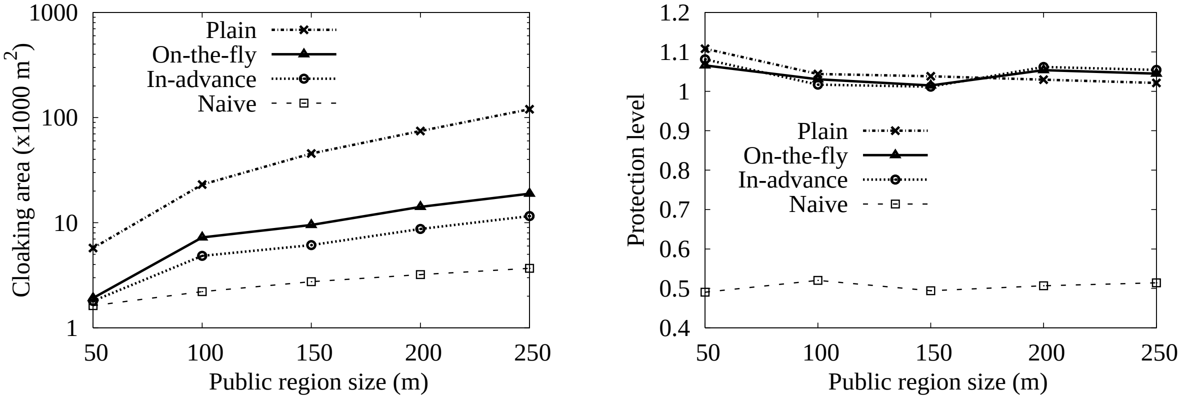
3.3.2.1 Effect of privacy requirement

This study investigates the impact of privacy requirement (i.e., the popularity of the public region specified by a service user) on the performance of the three techniques. We generated 300 service requests. Each request has a travel bound of a $4 \times 4\ km^2$ square region, and the travel distance of the corresponding user during her service session is $4\ km$. Each service user specifies her public region as a square region which contains her start position. The size of a public region, measured by the side length of the square, is varied from 50 to 250 meters.

The performance results are plotted in Figure 3.6. Figure 3.6 (a) shows that when the size of the public region increases, the average cloaking area under all the schemes increases. This is due to the fact that a larger public region is likely to contain more people's footprints and have a higher popularity. To satisfy a higher level of privacy requirement, a cloaking box needs to be larger to include more people. This study also shows that Plain always has a much larger cloaking area as compared to the other approaches. This scheme does not take user popularity into consideration when selecting a user's cloaking set. When some unpopular users are selected in a cloaking set, the cloaking boxes generated for the future movement of a service user will become larger and larger in order to contain all users in the cloaking set. Moreover, we can see in Figure 3.6 (a) that On-the-fly has larger cloaking area than In-advance. This is due to the fact that In-advance cloaking selects the cloaking set according to the predetermined base trajectory, and thus the users in the cloaking set must have footprints close to the moving route. On the other hand, On-the-fly cloaking selects the cloaking set using their footprints in the travel bound, and thus it cannot guarantee they have footprints close to the moving route. Comparing with the other three schemes, Naive has the smallest cloaking area. This scheme does not consider the correlation of the cloaking boxes in a trajectory, just cloaking each location with a bounding box that is as small as possible and has a popularity no less than that of the public region. The problem is, simply ensuring that each cloaking box satisfies the privacy requirement does not protect a user's privacy at her specified level. This is confirmed in Figure 3.6 (b). It shows that the protection level of Naive is constantly lower than 1. As for the other three schemes, they all guarantee that the actual protection level is no less than required.

3.3.2.2 Effect of travel distance

In this study, we investigated the impact of travel distance on the performance of the three techniques. In each simulation run, we set the public region as a $150 \times 150\ m^2$ square, and



(a) (b)

Figure 3.6 Effect of privacy requirement

generated 300 service requests. The travel distance is varied from 2 km to 6 km, and accordingly the side length of travel bound is varied from 2 km to 6 km. The performance results are shown in Figure 3.7 (a) and (b). Figure 3.7 (a) shows that under all schemes except Naive, the average cloaking area increases as the travel distance increases. However, In-advance performs the best while Plain performs the worst. The reason behind is explained as follows. When the travel distance is larger, the trajectory of the service user tends to traverse through a larger region. It is more difficult to find a cloaking set that all the users have footprints close to every location update in the moving route. Since In-advance cloaking always finds the users who have footprints closest to the base trajectory, the generated PPTs tend to have a finer cloaking resolution. As for On-the-fly and Plain, cloaking is not based on the predetermined base trajectory. In general, the more unpopular users included in the cloaking set, the more difficult it is to generate a PPT with fine resolution. Plain performs worse because in average it includes more unpopular users in a cloaking set. On the other hand, the cloaking area under Naive remains almost constant as the travel distance changes. It is due to the fact that Naive assumes each location update is an independent event. For each location update, it simply finds the nearest footprints to cloak. As such, the cloaking area is irrelevant to the number of location updates in the trajectory. Again, this approach cannot be used for location privacy protection when a user has to report her location periodically in a service session. Figure 3.7 (b) shows the protection level of Naive decreases as the travel distance increases. Since each location update is cloaked independently in Naive, a longer trajectory tends to have a less number of users who have visited all cloaking boxes in the trajectory, and thus has a lower popularity with respect to this common set of users. In contrast, the privacy level under none of the other three schemes is much affected by the variance of travel distance.

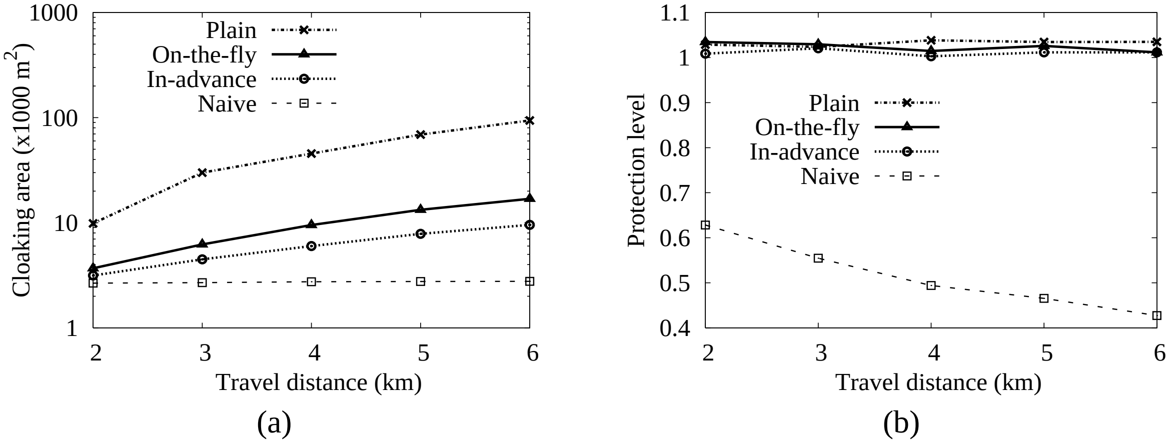


Figure 3.7 Effect of travel distance

3.3.2.3 Effect of footprint database size

This study investigates the impact of the number of trajectories in the footprint database on the performance. We varied the number of trajectories in the database from 100,000 to 300,000. The performance results are plotted in Figure 3.8 (a) and (b). It is shown in Figure 3.8 (a) that all schemes have better cloaking results when the database contains more trajectories. Clearly, more historical trajectories mean that more footprints collected in a fixed spatial region. As a result, a smaller cloaking box may be populous enough to meet the privacy requirement. By adding a service user's moving route to the database for future cloaking, our technique can generate better cloaking results. This feature makes it especially attractive for large-scale LBS that consists of a large number of users. Figure 3.8 (b) again shows that the protection level of Naive is constantly lower than 1. On the other hand, the protection level under all the other schemes is always above 1.

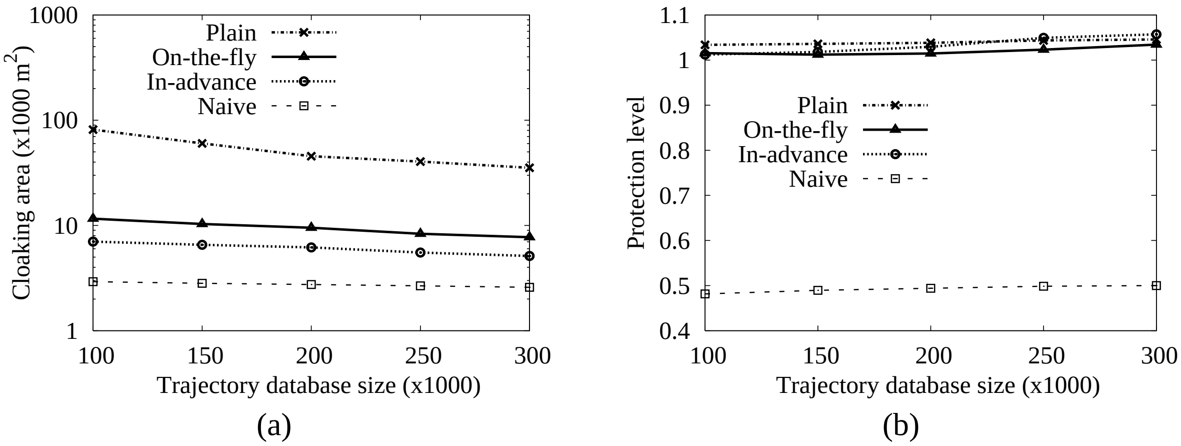


Figure 3.8 Effect of database size

3.4 Experiments

We have implemented an experimental system based on the technique presented in the previous sections. The prototype, called *location privacy aware gateway* (LPAG), has two software components, client and server. Client is implemented in C# using .Net Compact Framework 1.0. It runs on Windows Mobile 2003 platform and we have tested it with two types of mobile devices, HP IPAQ 6515 and HP IPAQ 4310 as shown in Figure 3.9. The former is a smart phone with a built-in 4-channel GPS receiver. The device communicates with the server through AT&T's GPRS wireless data service. As long as it is within the region covered by the carrier's service network, it can stay connected to the server which is located in our lab. The other type of client device, namely HP IPAQ 4310, is a regular pocket PC which connects with the server through our university's campus wireless network, which limits its roaming area to be within our campus. To make it position-aware, we bundle it with an external 16-channel GPS receiver, which provides position information through bluetooth connection. The server component is implemented in C# using .Net Framework 1.0. It manages the historical location data and corresponding indices using MySQL 5.0, and cloaks mobile clients' location updates using the proposed techniques when they request LBSs. In a seperate research project, we have a implemented a location-based service system called *ePostit* (60). This system allows one to publish a geo-referenced note, each associated with a geographic region and delieved to a user when the user arrive at the region. In our experiment, we also plant a number of spatial messages in our campus and let a user entertain the services provided by ePostit through the LPAG.

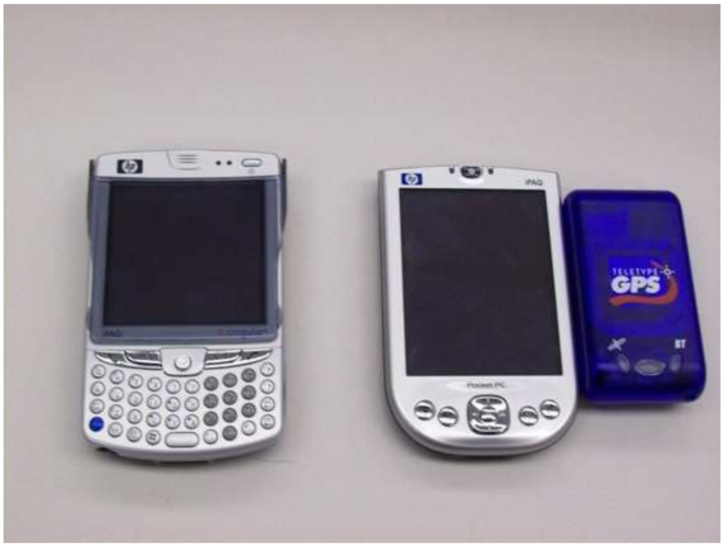
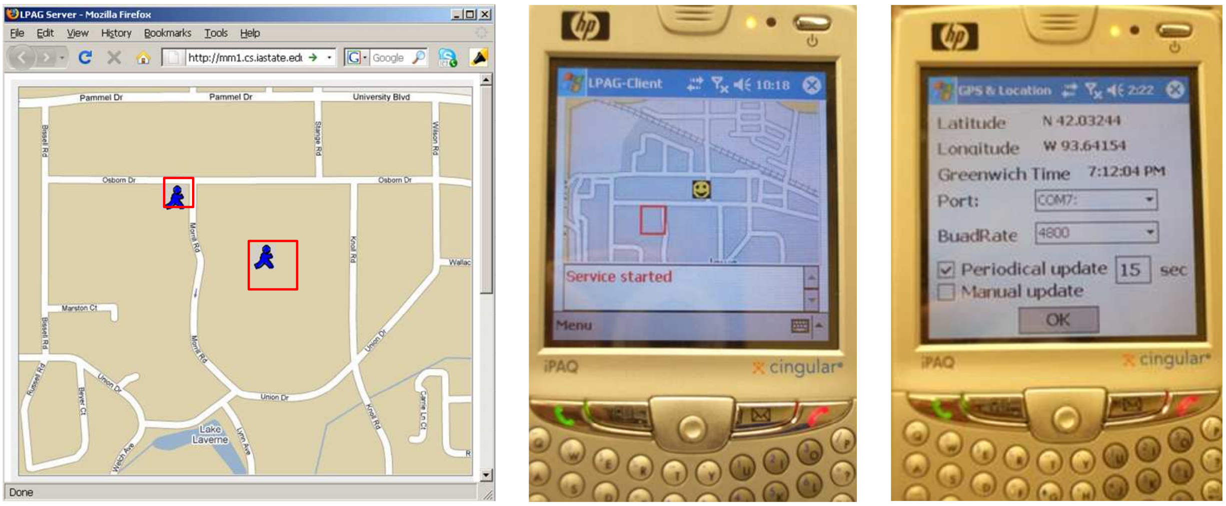


Figure 3.9 Client devices

Our test of LPAG consists of a location sampling phase, during which we collect users' footprints for location depersonalization. We create a number of client accounts, and carry the devices and have a walk around the campus, during which the devices makes periodical location update to the server. After a trajectory is collected, we randomly choose a client from the accounts created before, assign the trajectory to the client, and save it in the trajectory database in the server. In our testing of LPAG, we specify a rectangular region in the campus as the public region, and have a walk in the campus with a mobile device. During the walk, we send a sequence of queries to the server, each with our current position. For each query, the server generates a cloaking box using the proposed technique, and forwards it to ePostit. In response, the service provider delivers all the messages whose bounding boxes overlap with the cloaking box to the server, and the server forwards to the client only the ones whose bounding box contains the client's current position. In the following subsections, we introduce our system's user interfaces and discuss the experimental results collected in our field tests.



Latitude N 42.03244
Longitude W 93.64154
Greenwich Time 7:12:04 PM
Port: COM7:
BuadRate 4800
☑ Periodical update 15 sec
☐ Manual update

(a) (b) (c)

Figure 3.10 Server and client interface

3.4.1 Server and client user interface

Figure 3.10 (a) shows the server interface. Every time the server receives a query from a client, it computes a cloaking box as the client's location in requesting the service. Then, the server displays the cloaking box and the client's position on the map. As the example shown in this figure, two clients and their cloaking boxes are displayed on the campus map.

When a mobile device is powered on, the client finds out the current position and then connects to the server. After initialization, the screen shows a local map as its background and marks the client's position by a small face icon (see Figure 3.10 (b)). At the beginning of a service session, the client can set the public region by clicking the touch screen to specify its top-left corner and bottom-right corner, and embed the public region in the query packet. In the example shown in Figure 3.10 (b), the client specifies the library as her public region which is marked by the red rectangle. In our experiments, the travel bound is set as the whole campus. Then, during the session, the client can choose to periodically update her location or manually update whenever she wants (see Figure 3.10 (c)).

3.4.2 Experimental results

We first examine the system resources used by our code running on mobile devices.

CPU utilization: We measure the CPU utilization of our client code on the smartphone using Xda pps (61), which allows one to monitor the CPU usage of all the processes running on a smart device. When the device is idling with no movement, the CPU utilization is about 1%, indicating that reading GPS position (every one second) does not take much computation. When the client moves around but does not make any location update, we observe that the CPU utilization is in between $4\% - 12\%$, as our code redraws the client's position on the map. When the client communicates with the server (e.g., location update, message delivery), the CPU utilization is in between $10\% - 25\%$.

Memory and storage: Our client executable is only 120KB by itself. Since it is built on the .NET Compact Framework 1.0 and OPENNETCF 1.4, additional 2.5MB and 580KB files from the two platforms are needed, respectively. When running, our system has a memory footprint of 5.1MB, which is less than 10% of available main memory on HP IPAQ6515 (57.78MB) and HP IPAQ4310 (56.77MB). On both devices, our code can run simultaneously with other applications such as media player and Internet explorer.

We also examine two performance metrics which affect the usability of our system.

GPS accuracy: Because of position deviation of the GPS receiver, the position reported to the server may be different from the actual position of a client. If the position deviation is large, the bounding box computed by the server may not contain the client's position, and the client may get the false query result (missing or downloading wrong messages). In our experiments, we have tested the accuracy of the two types of GPS in the campus area. The smartphone we use has a built-in 4-channel GPS, while the external GPS bundled with the pocket PC has 16 channels. To calculate the position, a GPS receiver needs to have signals from at least 4 satellites. In general, the more channels available, the more accurate position it can compute. Our tests show that the 16-channel GPS has 5 meters error in average and 8 meters error in maximum. While the 4-channel GPS performs worse. It has 7 meters error in average and 14 meters error in maximum. These tests indicate that in the worst case the server should expand the boundary of the cloaking box by 15 meters to ensure the cloaking box contains the client's actual position, and the bounding box of a message should not be smaller than $15m \times 15m$.

Response time: The interval between the time a client sends a query and the time she receives the query result consists of four parts: 1) the time it takes to deliver the query from the client to the server, 2) the time the server uses to compute the cloaking box, 3) the time for the server to send the cloaking box to the service provider and receive candidate messages from the service provider, 4) the time it takes to download the resulting messages from the server to the client. Our experiments show that the server computes the cloaking box usually in less than 10 ms. In addition, the transmission speed between the server and the service provider is also very fast ($> 4MB/s$) since they are connected with a high speed LAN. The bottleneck is the communication between the client and the server, i.e., part 1) and 4). The smartphone we use connects to our server via AT&T's GPRS, while the Pocket PC connects to our server via our campus's WLAN. In our test, we create a number of messages, some with simple text messages (1-5KB) and short audio clips (10-30KB), while the rest with video clips (100-300KB). Our tests show that for messages with simple text and audio clips, the smartphone and pocket PC can download them with a delay of less than 1 second and 3 seconds, respectively;

for the messages with video clips, the pocket PC has a minimal delay of 5 seconds while the smartphone has a latency of more than 20 seconds. This study indicates that for cellular phones, our system is more appropriate for light-weight messages. Fortunately, this will not be a problem with the continuous development of broadband wireless services provided by the cellular carriers.

CHAPTER 4. Location safety protection in ad hoc networks

In this chapter, we present our research which investigates location safety protection in ad hoc networks. As discussed in Chapter 1, the location safety threat is considered particularly in digital battlefield, where enemy can locate and destroy a network node using its location information revealed in communications. This threat is different from location privacy intrusion in the sense that here the adversary is not interested in finding the individual identities of the nodes in a spatial region, but simply wants to locate and destroy them. In terms of countermeasure, the two threats actually appear to be opposite to each other. For privacy protection, we want a subject to be accompanied by as many others as possible. This, however, is counterintuitive from the safety point of view, because the more nodes a spatial region contains, the more attractive for an adversary to destroy them all together.

Our research goal of location safety protection is to allow nodes to reveal their location information, yet make it practically infeasible for the adversary to locate them based on such information. Towards this objective, we make the following contributions. 1) We proposed to reduce location resolution of nodes' disclosed location to achieve a desired level of safety protection. Instead of revealing its accurate position, a node can report that it is inside some geographic region – cloaking box. 2) We address the challenge of computing a cloaking box with cost-effective solutions in the context of both stationary and mobile ad hoc networks. 3) We investigate the impact of location cloaking on the geographic routing protocols, and we proposes a novel location secure routing (LSR) protocol which is able to deliver data packets efficiently, as well as prevent adversary from compromising nodes' location safety according to the routing information. In the rest of this chapter, we first present our cloaking techniques in Section 4.1, and then we present the location secure routing protocol in Section 4.2.

4.1 Location cloaking for location safety protection

We consider an ad hoc network formed by a set of sensor nodes deployed in a hostile environment, where communications among the nodes may be open to an adversary. The adversary is interested in collecting the location information revealed by the nodes, then locating and destroying them. This safety threat cannot be fully defeated by the means of message encryption (50) and anonymous routing (38; 41; 43). Encryption makes a message intelligible only to its destination, but works only if the intended recipient is trustworthy. Anonymous routing shares the same problem. By making routes untraceable, it prevents an adversary from identifying important targets like a command post through the collection and analysis of data flow and traffic pattern. But it does not thwart the direct threats imposed by the exposure of nodes' location information. In reality, a node may be destroyed whenever it is located, regardless of its importance.

Knowing that a certain region contains a set of sensors, the adversary can always comb through the whole area to uncover them. However, if the area is very large, the searching cost can be prohibitively high. In light of this observation, we propose reducing location resolution to achieve a desired level of safety protection. Instead of revealing its accurate position, a node can report that it is inside some geographic region, which we will refer to as a cloaking box. Given a cloaking box, we measure its *safety level* as the ratio of its area and the number of nodes inside it. The higher safety level a cloaking box has, the less attractive for the adversary to search the nodes inside it. When a cloaking box's safety level exceeds some threshold, it can be considered safe for release – the adversary would not be able to search because of high cost.

The challenge is how to compute cloaking boxes. First, each cloaking box must satisfy the minimal safety level requirement, meanwhile it must be as small as possible in order to minimize the impact of reduced location resolution on the efficiency of network operating and applications. Second, nodes must be able to compute their cloaking boxes without having to reveal their accurate position. Finally, a sequence of cloaking boxes must not be correlated to identify an area whose safety level is below the requirement. This is due to the fact that the adversary may be able to collect many cloaking boxes. It may first appear that a node can simply broadcast to query its nearby nodes about their location, and then identify the smallest region that meets the safety requirement as its cloaking box. This strategy, however, requires nodes to report their exact location, a security breach that should not be allowed in a hostile environment. In addition, determining the query broadcast region itself is difficult. When the node queries a region, it actually reveals that it is inside the region. If the region's safety level is not sufficient, the nodes inside the region are all in danger.

To address the above challenges, our idea is partitioning the network domain into a number of *safe* subdomains that is as small as possible, and let each node take the subdomain where it resides as its cloaking box. A subdomain is safe if its safety level is no less than the minimal required level. To make subdomains as small as possible, each subdomain is recursively split as long as the resulted subdomains are all safe.

Based on the above idea, we have developed a novel cloaking technique for location safety protection. Next, we will introduce the concept of safety level and the requirement of location cloaking, followed by our cloaking algorithms for both static and mobile ad hoc networks. As we will show, our algorithms do not require nodes to reveal their accurate position. In fact, we guarantee that the safety requirement is satisfied for all location information they disclose. We also show that even if all cloaking boxes reported by nodes are known to the adversary, they cannot be correlated to refine an area that violates the safety requirement.

4.1.1 Safety level

We assume the adversary locates the nodes by collecting and analyzing their location disclosed in communications, but not combing through the network domain (e.g., physically explore the domain and detect communicating nodes within its radio range). To avoid from being located, nodes reduce the resolution of their location information. Specifically, whenever a node has to disclose its location, it reports a *cloaking box*, a geographic region that contains its current position. Given a cloaking box b, we define its *safety level* as $S(b) = \frac{A(b)}{N(b)}$, where $A(b)$ denotes the area of b, $N(b)$ denotes the population of b, i.e., the number of wireless nodes inside b.

Reducing location resolution can provide a desired level of safety protection to an ad hoc network. We say the network is protected at a safety level of θ if during its life time, the adversary cannot find any region in the network domain whose safety level is less than θ. Simply ensuring that each cloaking box reported by nodes has a minimal safety level of θ does not guarantee that the network is protected at θ level, because the adversary may collect multiple cloaking boxes and correlate them for attack. Consider three cloaking boxes showed in Figure 4.1. Even if each box has a safety level of θ, when combined, the safety level of their concatenated region is less than θ.

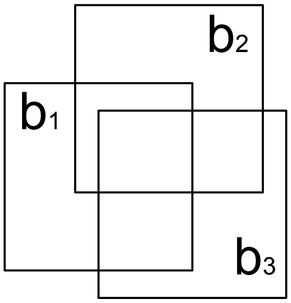


Figure 4.1   Correlation attack

Let b be a cloaking box to be disclosed by a node at time t, and $B = \{b_1, b_2, \cdots, b_m\}$ be the set of all cloaking boxes revealed by nodes in the network before time t. To protect the network's safety at θ level, the following two conditions must be satisfied:

1. $S(b) \geq \theta$;

2. b cannot be combined with any subset of B, say $B' = \{b'_1, b'_2, \cdots, b'_k\} \subseteq B$, such that $S(\bigcup_{j=1}^{k} b'_j \bigcup b) \geq \theta$

The first condition ensures that each cloaking box has a safety level no less than θ. The second condition is to guarantee that any combination of cloaking boxes also has a safety level no less than θ. In other words, the adversary cannot correlate b with any cloaking boxes disclosed previously to identify a region whose safety level is less than θ.

Clearly, a cloaking box needs to be as small as possible, because coarse location information will result in inferior performance of routing protocols and applications. On the other hand, it must not reduce the desired level of network safety protection. The challenges of computing such a cloaking box are twofold. First, it must be computed without having to request other nodes' accurate position. In fact, whenever a node disclose its position, it can only report a cloaking box, which must also meet the two safety requirements. Second, a node generally cannot know all cloaking boxes other nodes have disclosed, making it difficult to prevent the correlation attack.

4.1.2 Location cloaking techniques

In this section, we present our algorithms for cloaking box computing in stationary and mobile ad hoc networks. Without loss of generality, we assume the network is deployed in a rectangular domain D and the required level of network safety protection is θ, where $S(D)$ (i.e., the safety level of D) $\geq \theta$. Our basic idea is to partition domain D recursively into a number of subdomains, each having a safety level of at least θ. The partitioning of a subdomain is always along its longer dimension. If two dimensions are equal, then always split on the horizon. Figure 4.2 illustrates the processing of partitioning. Initially, D is partitioned into two equal subdomains, R_1 and R_2. The partitioning is recursive in the sense that a subdomain is further partitioned into two new subdomains as long as their safety levels are no less than θ.

For instance, R_1 is further partitioned into R_{11} and R_{12}, and so R_{12} into R_{121} and R_{122}. The process of partitioning creates a binary partitioning tree (BP-tree), which records the domain hierarchy. The root of the BP-tree is of depth one and each partitioning creates a new level. At the end of partitioning, each node takes the leaf subdomain, which cannot be partitioned further, as its cloaking box. An important characteristic of a BP-tree is that, given the root domain D and a leaf partition P, a node can compute all subdomains in the path from D to P. For example, a node in R_{121} knows that path $D \to R_1 \to R_{12} \to R_{121}$.

We now consider how to do such partitioning in a fully distributed environment, where each node needs to compute its cloaking box without knowing other nodes' exact position. In the following subsections, we first present our solution for a stationary ad hoc network, and then extend it for a mobile ad hoc network, where the movement of nodes requires the domain partitioning to be adjusted dynamically.

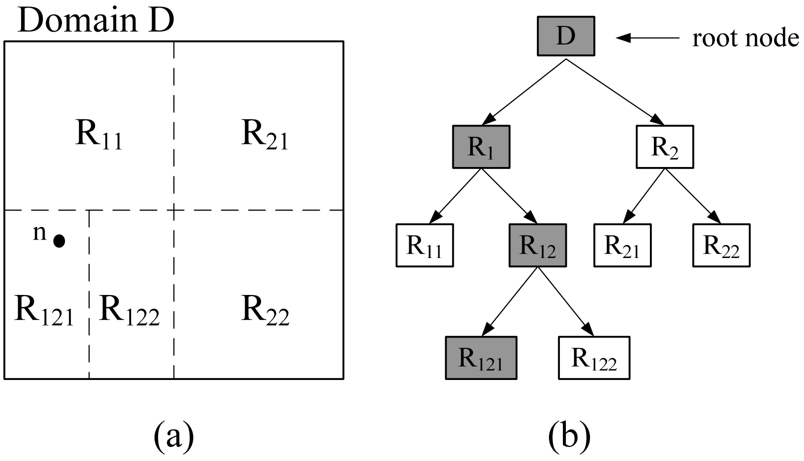


Figure 4.2 Example of BP-tree

4.1.2.1 Cloaking for stationary ad hoc networks

Suppose nodes are deployed simultaneously (e.g., distributed by an airplane) in a target domain D. For each node, its initial partition P is set to be D. Starting at time 0, the nodes refine their partition distributively as follows round by round. In each round, each node broadcasts a *PLUS* packet within its own partition P. The format of the packet is $(PLUS, seqnum, P)$, where $seqnum$ is a global unique number used for dropping the redundant packet during the broadcast (62). By counting the number of different packets it receives, a node knows the number of nodes inside its current partition P. Let i be the number of the different packets. P's safety level can then be computed as $S(P) = \frac{A(P)}{i}$. If $S(P) < 2 \cdot \theta$, the node takes P as its cloaking box (i.e., leaf subdomain). Otherwise, it divides its current partition P into two equal halves (according to the rules discussed earlier) and sets P to be the one that contains its current position. When a new round of refinement starts, it broadcasts $(PLUS, seqnum, P)$ to find the number of nodes inside P and computes the safety level of P. Each node repeats this process until the safety level of its current partition is less than $2 \cdot \theta$.

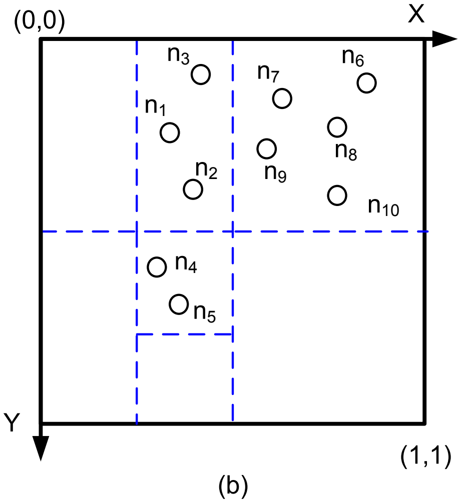


Figure 4.3 Example of location cloaking in stationary ad hoc networks

Algorithm 4 is the pseudo code for each node to find its cloaking box. All nodes start to execute this code right after they are deployed (at time t_0). Since nodes have no idea of the actual position of neighbors, we assume simple flooding is used for regional broadcast. That is, when a node receives a packet $(PLUS, seqnum, P)$, it rebroadcasts the packet if it is inside P and has not seen this packet (based on seqnum). Otherwise, it simply drops the packet. Note that in each round of partitioning, a node needs to wait for a certain time period T. This is to collect PLUS packets sent by other nodes in the same partition. Since the broadcast delay is proportional to node population, we can preset the initial waiting time to some value (e.g., T_0)

and then cut 50% each round.

Algorithm 4 Cloak for stationary ad hoc networks

```
S-Cloak($D$, $\theta$) // executed by each node
$p \leftarrow$ my current position
$seqnum \leftarrow$ my unique id
$P \leftarrow D$
$T \leftarrow T_0$
while true do
  send packet $(PLUS, seqnum, P)$
  wait(T) {wait until PLUS packets collection are finished}
  $i \leftarrow$ the number of PLUS packets collected
  $s \leftarrow \frac{A(P)}{i}$ {Get $P'$s safety level}
  if $s \geq 2\theta$ then
    $P \leftarrow Partition(P, p)$ {get the half partition that contains $p$)
    $T \leftarrow \frac{T}{2}$
  else
    return $P$
  end if
end while
```

In the above process, a node reveals its location information in its *PLUS* packets. For each $(PLUS, seqnum, P)$ packet, P's safety level is guaranteed to be no less than θ. Let P' be P's parent partition, we have $S(P') \geq 2 \cdot \theta$, since a node will not proceed to the next round of partitioning unless the safety level of its current partition is at least $2 \cdot \theta$. Thus, even if all nodes in P' are actually located inside P, P's safety level is still no less than θ. As a result, each partition disclosed satisfies the safety requirement. The recursive domain partitioning also makes correlation attack impossible because, given any two partitions P_1 and P_2, either they do not overlap at all or one contains the other completely. As an example, consider Figure 4.3. Suppose ten nodes are distributed in a square domain of $[0, 1]^2$, and the safety requirement is $\theta = \frac{1}{32}$. We present a partition in the form of $[(x_1, y_1), (x_2, y_2)]$, where (x_1, y_1) denotes the position of its top left corner, and (x_2, y_2) for the bottom right corner. In the first and second rounds, the safety levels of the partitions are all larger than 2θ. In the third round, each node in partition [(.5, 0), (1, .5)] finds out that the safety level of its partition is $\frac{1}{20}$, which is less than 2θ. Thus, they stop the partition process and use [(.5, 0), (1, .5)] as their cloaking box. Similarly, in the fourth round, nodes n_1, n_2 and n_3 determine [(.25,0),(0.5.5)] as their cloaking box, and in the fifth round, node n_4 and n_5 take [(.25, .5),(.5, 1)]. As we can see, nodes in a denser region tend to have a larger cloaking box.

It is possible for an adversary to compromise some node and let it send multiple PLUS packets to falsify cloaking results. This does not jeopardize the safety level of a cloaking box, but artificially enlarges the cloaking box size. Such attack can be prevented by authentication techniques (63; 43). The idea is to add a certificate field in the PLUS packet. The certificate field allows a node to verify the sender of a packet and thus detect if multiple PLUS packets have been sent from a same node. Alternatively, a compromised node may stay silent, not sending any PLUS packets. In this case, the number of PLUS packets received is less than the actual population inside a partition. This approach, however, does not reduce the safety level of those uncompromised nodes since they rely on the actual PLUS packets they receive for cloaking. Ultimately, the adversary is interested in locating and destroying the uncompromised nodes.

4.1.2.2 Cloaking for mobile ad hoc networks

Location cloaking in the presence of node mobility is more challenging. Right after the network is deployed, each node can find its initial cloaking box using the domain partitioning technique discussed in the previous subsection. One minor change is the minimal safety level that governs when a subdomain should be partitioned further. Given a partition P, its safety level $S(P)$ downgrades when a node moves into it. Given the required safety level θ, the maximum number of nodes allowed in P is $n_{max} = \lfloor \frac{A(P)}{\theta} \rfloor$. Thus, to ensure $S(P)$ no less than θ before P is merged with some other partitions, $S(P)$ must be at least $\alpha \cdot \theta$, where α, referred to as P's safety coefficient, is equal to $\frac{n_{max}}{n_{max}-1}$. Therefore, the domain partitioning procedure goes to a subdomain P, only if the safety level of its parent partition is no less than $2\alpha \cdot \theta$.

We now consider how to adjust the domain partition dynamically as nodes move in and out of their cloaking boxes. After the initial partitioning, each node knows its partition and uses that as its cloaking box. When nodes move, they monitor their own movement against their current partition. If a node, say M, moves out of its current partition P, it notifies the nodes inside of P by broadcasting them a $LEAVE$ message, $(LEAVE, seqnum, P)$. When receiving such a message, each node inside P computes P's new safety level. If $S(P) > 2\alpha\cdot\theta$, it calls *S-Cloak($P$, θ)* to split P, and determines its new cloaking box.

Algorithm 5 Cloak for mobile ad hoc networks

```
M-Cloak($\theta$)//exucte by each node
{monitor my movement against current partition $P$}
if crossing the boundary of $P$ then
  //update $P$
  send packet $(LEAVE, seqnum, P)$
  $N(P) \leftarrow N(P) - 1$
  $s \leftarrow \frac{A(P)}{N(P)}$
  if $s \geq 2\alpha \cdot \theta$ then
    wait(T) {wait until LEAVE broadcast is finished}
    $S - Cloak(P, \theta)$ {split $P$}
  end if
  //find new cloaking box
  {listen and eavesdrop ADVERTISE packet for $P'$}
  send packet $(JOIN, seqnum, P')$
  $N(P') \leftarrow N(P') + 1$
  $s \leftarrow \frac{A(P')}{N(P')}$
  if $s \leq \alpha \cdot \theta$ then
    wait(T) {wait until JOIN broadcast is finished}
    while true do
      $P' \leftarrow$ parent partition of $P'$
      calculate safety level of $P'$ {as the same as is S-Cloak}
      if $S(P') \geq \alpha \cdot \theta$ then
        set $P'$ as new cloaking box
      end if
    end while
  end if
end if
```

In addition to notifying its leaving, M also needs to find its new partition and announce its coming to the nodes in the partition. We assume that each node in a partition P' periodically broadcasts an ADVERTISE message, $(ADVERTISE, seqnum, N(P'), P')$, where P' is its current partition. After receiving an ADVERTISE message with P' that contains its current position, M takes P' as its current cloaking box, and then broadcasts a JOIN message, $(JOIN, seqnum, P')$. Upon receiving a JOIN message, each node inside P' computes the new safety level of P'. If $S(P') < \alpha \cdot \theta$, P' is not safe enough, each node takes the parent partition of P' as its new cloaking box, and then broadcasts a MERGE message, $(MERGE, seqnum, P'')$, where P'' is the parent partition of P'. When the nodes in P'' receive the MERGE message, they calculate the safety level of P'' by broadcasting PLUS message in P''. If $S(P'') > \alpha \cdot \theta$, the nodes inside P'' take P'' as their new cloaking box. Otherwise, they repeat this merge process until they find a partition whose safety level is at least $\alpha \cdot \theta$.

The pseudo code of the cloaking update algorithm is given in Algorithm 5. To illustrate this process, we use the same example in section 4.1.2.1. Due to the effect of the safety coefficient α, the initial cloaking box of n_4 and n_5 is [(.25,.5), (.5, 1)] as shown in Figure 4.4 (a), which is different from that in the stationary network. Suppose the node n_4 moves out of its cloaking box. When n_5 receives the LEAVE message sent by n_4, it recalculate the safety value of its cloaking box as $\frac{1}{8}$, which is larger than $2\alpha \cdot \theta = \frac{1}{12}$. Thus, this partition is split in half and n_5 will find its new cloaking box as [(.25,.5), (.5,.75)]. In addition, after receiving the JOIN from n_4, n_1, n_2 and n_3 recalculate the safety value of their cloaking box as $\frac{1}{32}$ which is smaller than $\alpha \cdot \theta = \frac{1}{24}$. Thus, they broadcast MERGE messages in the parent partition of the current cloaking box. As a result, n_1, n_2, n_3 and n_4 will use partition [(0,0), (.5,.5)] as their new cloaking box, which is shown in Figure 4.4 (b).

In the above process, a node reveals its location information in packets LEAVE, ADVERTISE, JOIN and MERGE. For each packet of $(LEAVE, seqnum, P)$, the location information in the packet cannot be used to locate the sender because the sender is outside P. In addition, an ADVERTISE packet is broadcast in the sender's cloaking box, whose safety level is guaranteed to be no less than θ; a MERGE packet is broadcast in the parent partition of sender's cloaking box, whose safety level is also no less than θ. Comparing with LEAVE, ADVERTISE and MERGE, JOIN packets involve a safety problem. When an adversary eavesdrops a JOIN packet of $(JOIN, seqnum, P)$, it knows the population of P increases by one.

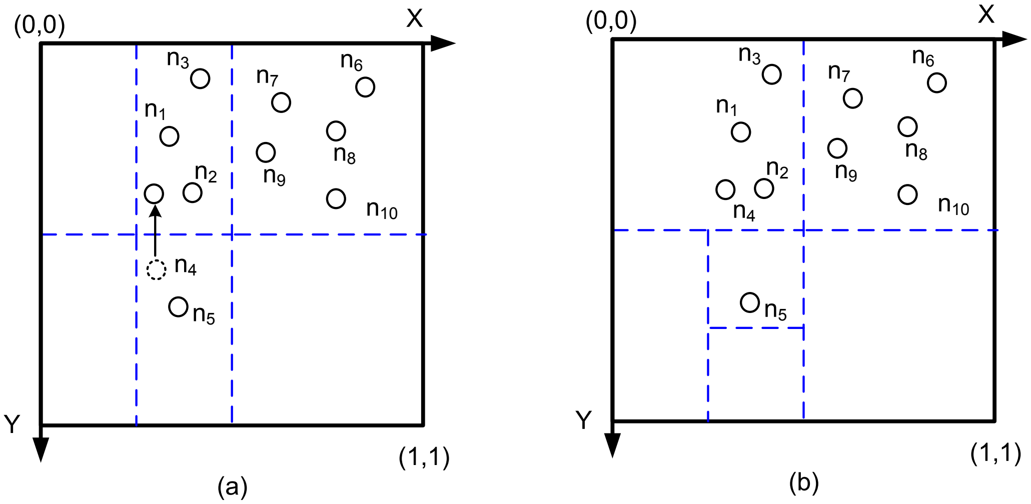


Figure 4.4 Example of location cloaking in mobile ad hoc networks

Thus, if the adversary receives multiple JOIN packets in a short period, it may be able to infer that the safety level of P downgrades below θ. To cope with this problem, a node should not send JOIN packet immediately after it eavesdrops an ADVERTISE message. Instead, we let a node wait for a random time period before it sends JOIN packet. During the waiting period, if a node receives another JOIN packet, it waits for another certain period which ensures that the eavesdropped JOIN packet has been broadcast in P. As long as it does not hear MERGE packet during the waiting period, it sends its JOIN packet. Otherwise, it extends the waiting time further until the merge process finishes. In this way, the JOIN packets are broadcast sequentially, and from perspective of the adversary, the safety level of P must be no less than θ before merge happens.

An adversary may launch DOS attacks by inserting fake LEAVE or JOIN packets. Such attacks can be prevented by simply letting nodes recalculate the population in their current cloaking box before each split or merge.

In the above algorithms, we assume the network is fully connected. However, in the case that the network is disconnected, the number of PLUS packets collected by a node may be less than the actual population of a partition. In the mobile ad hoc networks, the movement of mobile nodes may change the topology of the network. For example, suppose two disconnected groups of nodes are deployed in a same partition. Since they have no idea the existence of the other group of nodes, every node uses the number of nodes in its own group to compute the safety level of the partition, and thus the safety level is overestimated. When the two groups of nodes move towards each other and become connected, the adversary may find out the safety level of this cloaking box is lower than θ.

To prevent such threat caused by disconnection, we propose a simple strategy as follows. As we know, a node can updates the population of its cloaking box when it receives corresponding LEAVE and JOIN packets. Suppose the network is initially connected. After nodes start to move, if a node finds the population of its cloaking box suddenly changes a lot, which does not match the LEAVE and JOIN packets it received, it considers the network is disconnected. Then, the node can estimate the population of its cloaking box according to the number of nodes in the connected group it belongs to. For example, suppose at the time that network becomes disconnected, the node calculates the ratio r between the population before disconnection and the number of nodes in the connected group. Then, the node estimates the population of the cloaking box by timing r with the current population of the connected group. Actually, a node can adjust the ratio r as necessary. If a larger r is chosen, the node tends to overestimate the population, the cloaking boxes have high safety since the estimated population is larger than the actual population with a high probability. However, overestimate may lead to unnecessary partition merge. The resulting cloaking boxes with larger size deteriorate the communication efficiency.

4.1.3 Analysis

In this subsection, we propose an analytical model to estimate control overhead involved in the cloaking algorithms, which is measured by the number of control packets. Recall that in our algorithms all the control packets are sent by regional broadcast. Because nodes do not reveal their positions, there is no neighborhood information available to help packet forwarding.

Thus, we simply assume pure flooding, in which each sensor node has to forward data packet as long as it is inside the broadcasting region. Specifically, the cost of broadcasting a packet within a region is equal to the population of the region. We assume there are m wireless nodes uniformly distributed in the network domain of size $l \times w$, and we also assume $\frac{l \cdot w}{m} \geq 2\theta$. Because if $\frac{l \cdot w}{m} < 2\theta$, according to Algorithm 4, we know that all nodes will use the entire network domain as their cloaking box.

In the stationary ad hoc network, the cloaking boxes are calculated only once right after the network is deployed. When calculating the population of a partition P, every sensor node inside broadcasts a PLUS packet within P. The total number of packets is $N^2(P)$, and the cost for each node is $N(P)$. Under uniform distribution, the population of the partition is approximately proportional to its size, and the population of a partition at depth k in the BP-tree is $\frac{m}{2^{k-1}}$. Thus, the cost of a node when computing the population of a partition at depth k in BP-tree can be estimated as $C(k) = \frac{m}{2^{k-1}}$. In addition, as explained in Algorithm 4, a sensor node stops searching when it finds the safety level of a partition is less than 2θ. Under uniform distribution, the cloaking box b of any sensor node has the same size. It contains only the node itself, and it satisfies $\theta \leq \frac{A(b)}{N(b)} < 2\theta$. Thus, we can infer that b is the on the depth $d_{max} = \lfloor \log_2 \frac{l \cdot w}{\theta} \rfloor$ in the BP-three. Therefore, the overall number of packets sent by a sensor node in calculating its cloaking box can be estimated as $C_{init} = \sum_{i=1}^{d_{max}} C(i) = (2 - 2^{1-d_{max}})m$.

In the mobile ad hoc network, the initialization of cloaking boxes is very similar with the stationary ad hoc network. In the analysis, we focus on the cloaking box update and estimate average number of control packets sent by a mobile node per time unit. Suppose b is a cloaking box at the depth k in the BP-tree. When a node moves out of b, it broadcasts a LEAVE packet within b, and the cost is $C_{leave} = N(b)$. If the safety level of b becomes larger than $2\alpha\theta$, b is split, and every node inside b will recalculate its cloaking box by broadcasting a PLUS packet in b's child partitions. The cost C_{split} is bounded by $\frac{1}{2}N^2(b) \leq C_{split} \leq N^2(b)$. The lower bound denotes the cost when mobile nodes in b is uniformly distributed; the upper bound denotes the cost when one of b's child partition is empty. On the other hand, if a mobile node moves into b, a JOIN packet is broadcast inside b with cost $C_{join} = N(b)$. If the safety level of b downgrades lower than θ, every node in b broadcasts a MERGE packet within b's parent partition $P(b)$, the cost of which is $N(b)N(P(b))$. Then all nodes in $P(b)$ recalculate the safety level of $P(b)$ by broadcasting PLUS packets, the cost of which is $N^2(P(b))$. Thus, the total cost of the merging process is $C_{merge} = N(P(b))(N(P(b)) + (N(b)))$.

Suppose mobile nodes follow a random walk model, in which at every time unit, a mobile node moves with a randomly picked direction and speed. According to (64), the time duration that a randomly moving unit may stay in an area can be approximated as an exponential distribution and the mean staying time is $t = \frac{\pi A}{E[v]L}$, where A is the area, L is the perimeter of the area, and $E[v]$ is the average speed of the mobile unit. As discussed above, under uniformly distribution, the cloaking boxes of mobile nodes are at depth d_{max} in the BP-tree. Thus, the average time duration that a mobile host stays inside its cloaking box is $\bar{t} = \frac{\pi \cdot l \cdot w}{2^{d_{max}-1}E[v]L(d_{max})}$, where $L(k) = \frac{l}{2^{\lfloor k/2 \rfloor - 1}} + \frac{w}{2^{\lfloor (k-1)/2 \rfloor - 1}}$ is the perimeter of a partition at depth k in the BP-tree. Since the random walk does not change the uniform distribution, theoretically, neither split nor merge happens during the movement of mobile nodes, and the control overhead is composed of only LEAVE and JOIN messages. In reality, split and merge may happen but the frequency must be very low. Thus, in the analysis, we only count the LEAVE and JOIN messages. Since the cloaking boxes contain only one node, the cost of broadcasting LEAVE and JOIN is equal to 1. Therefore, during cloaking update, the average number of control packets sent by a mobile node per time unit can be estimated as $C_{update} = \frac{2}{t}$.

4.1.4 Performance study

To evaluate the performance of the proposed schemes, we have developed a detailed simulator. We implemented a mobile ad hoc network in which a number of mobile nodes are distributed in a rectangular domain. Our simulation consists of two phases: mobile nodes first initiate their cloaking boxes using Algorithm 4, and then move following a random walk model and update their cloaking boxes whenever necessary using Algorithm 5. We are mainly interested in two performance metrics:

- *Cloaking area*. This metric measures the potential impact of location cloaking on the applications that rely on node location information. We measure cloaking area as the average size of cloaking boxes used by each node in simulation. Since a node may use different cloaking boxes, we compute its cloaking area using time weighted average. Suppose during a simulation time period of $[0, T]$, a node has its cloaking box $\{a_0, a_1, \cdots, a_k\}$ at time ticks $\{0, t_1, \cdots, t_k\}$, respectively. The node's cloaking area is computed as $\frac{a_0 \cdot t_1 + \sum_{i=1}^{k-1} a_i \cdot (t_{i+1} - t_i) + a_k \cdot (T - t_k)}{T}$.

- *Control overhead*. This metric measures the communication cost incurred by location cloaking. We evaluate two communication costs, C_{init} and C_{update}, incurred in two simulation phases, respectively. C_{init} is the average number of control packets sent by each node in the initial domain partitioning. This cost measures the communication efficiency of Algorithm 4. On the other hand, C_{update} measures the communication cost incurred after nodes start to move, and therefore evaluates the efficiency of Algorithm 5 It is defined as the average control packets sent by each node per time unit.

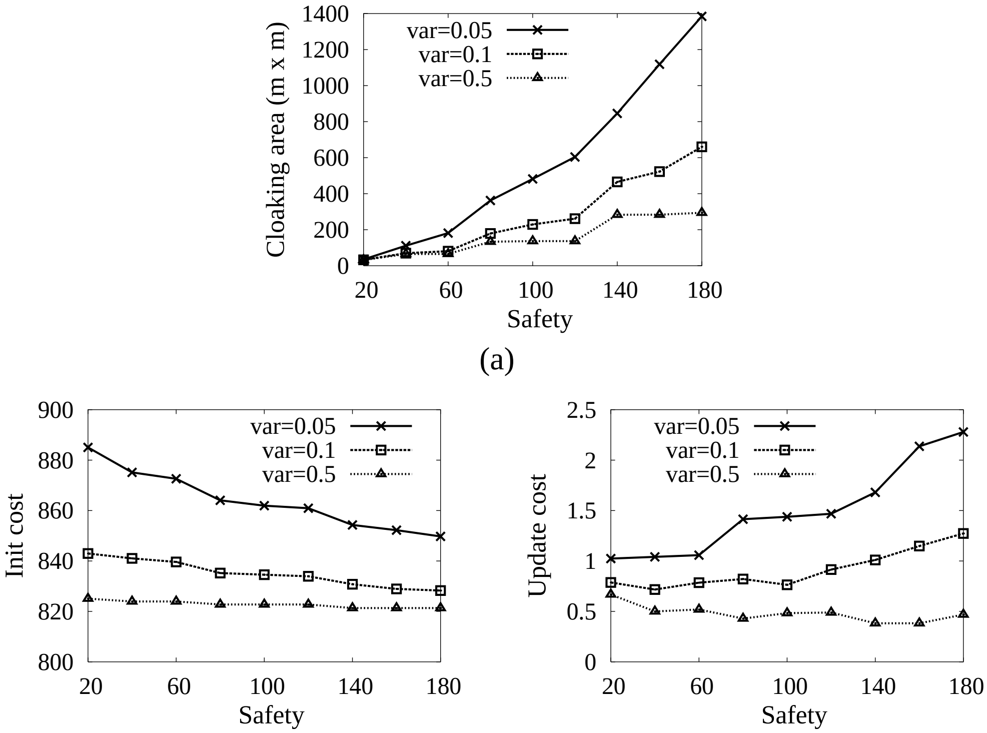
Table 4.1 summarizes the parameters used in our study. Unless otherwise specified, the default values are used. When the distribution of nodes is uniform, the performance of our techniques can be predicted with our analytical model. Thus, we focus on evaluate the performance of our techniques under a non-uniform distribution. In our simulation, the initial distribution of nodes follows a normal distribution. To ensure such a distribution, we partition the networks into many small grid cells, and then deploy different number of nodes in cells so that the node population in cells obeys a normal distribution approximately. The movement of nodes follows a random walk. As such, as nodes continue to move, their distribution eventually become uniformed. Our simulation stops when the change of average cloaking box sizes become less than 5%. In the next subsections, we report how the two performance metrics are affected by safety requirement θ, network density, and nodes' moving speed.

Table 4.1 Simulation parameters

parameter	range	default	unit
network domain	1000×1000	1000×1000	$meter^2$
node number	200 - 700	400	$unit$
transmission radius	50	50	$meter$
node speed	1 - 5	3	$meter/sec$
safety requirement θ	$20 - 180$	100	$unit$

4.1.4.1 Effect of safety level

In this study, we investigated the impact of safety level on the performance. We partitioned the network domain into a number of grid cells, each being $20 \times 20\ meter^2$, and deployed 400 nodes in the grid cells in a normal distribution with variance of 0.05, 0.1, and 0.5 respectively. Here the variance is normalized by dividing grid cell population with the total number of nodes. Thus, the larger the variance is, the more even the nodes are distributed. The value of safety requirement is varied from 20 to 180, and the performance results are plotted in Figure 4.5. Figure 4.5 (a) shows that the cloaking area is larger when the variance of the distribution is more skewed. This is due to the fact that under a more uneven distribution, more nodes are deployed closer in some small region, causing larger cloaking boxes for these nodes. The figure also shows that the cloaking area under all three variance settings increases as the safety requirement θ increases. Under the same distribution setting, a larger θ makes it more difficult to find a small partition that has sufficient safety level. Figure 4.5 (b) shows that C_{init} is higher when the distribution is more skew. This can be explained as follows. The cost that every node inside a partition broadcasts a PLUS message within it is equal to the square of its population. Thus, when a partition is split, the total broadcasting cost in two child partitions is less when the difference between their population is larger. Therefore, the more uneven distribution tends to have a higher init cost. The figure also shows that in all the three distributions, C_{init} decreases as θ increases. A smaller θ leads to smaller cloaking boxes, which have larger depths in the BP-tree. This means the domain partition procedure goes deeper in the BP-tree, thus incurring more control overhead.



(a)

(b)

(c)

Figure 4.5 Effect of safety requirement

Figure 4.5 (c) shows that as the safety level increases, C_{update} decreases under distribution with variance 0.5, but increases under distribution with variance 0.1 and 0.05. Under random walk, the uneven distribution will become more and more uniform as the simulation runs. For a distribution with a larger variance, the initial cloaking boxes are smaller. Partition splits or merges happen less frequently, and the overall update cost is mainly composed of the LEAVE and JOIN messages, the number of which is inversely proportional to the size of cloaking boxes. By contrast, when the distribution is more skewed, many initial cloaking boxes are very large. Thus, more partition splits take place after the nodes start to move, thus generating more control overhead.

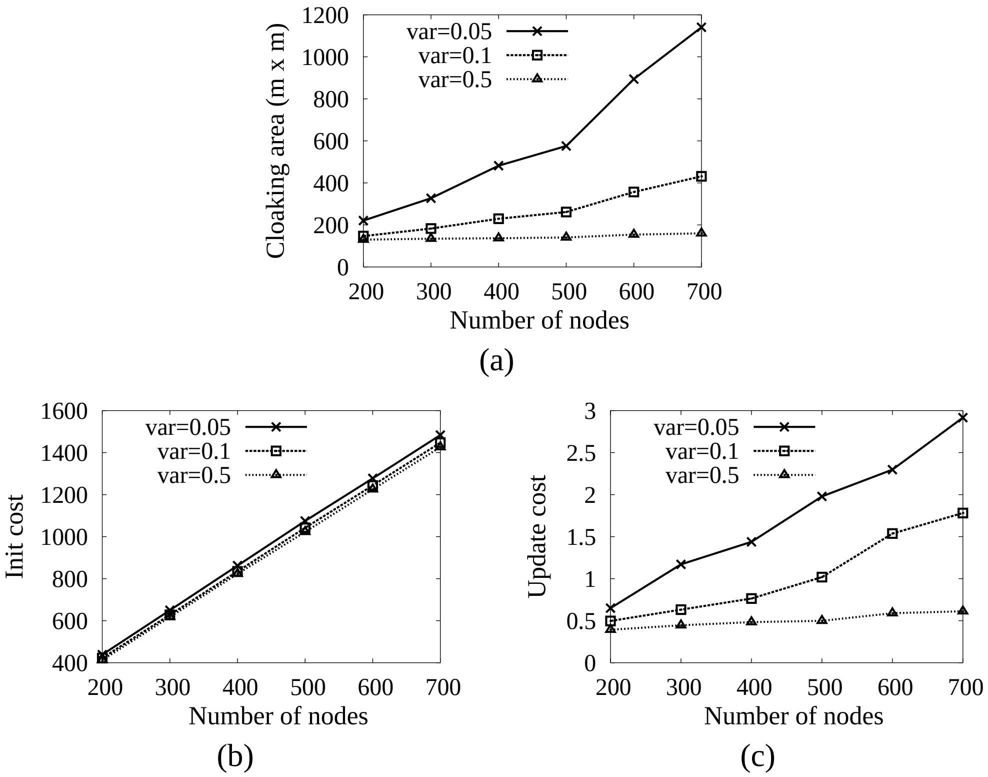


Figure 4.6 Effect of network density

4.1.4.2 Effect of network density

This study investigated the impact of network density on the performance of our cloaking algorithms. We set $\theta = 100$, and varied the node number from 200 to 700. The performance results are shown in Figure 4.6. As showed in Figure 4.6 (a), the cloaking area increases as the network becomes denser. In a non-uniform distribution, nodes are more densely deployed in some regions, and given a distribution with certain variance, the node density in these regions is proportional to the network population. Thus, the size of cloaking boxes increases as the network population increases. Figure 4.6 (b) shows that C_{init} increases linearly as the network density increases. Since the cost of broadcasting one PLUS packet to every node in a partition is equal to the population of the partition, given a distribution with certain variance, the population of a partition is proportional to the network population. Thus, the cost of broadcasting a PLUS packet in the partition is also proportional to the network population. Figure 4.6 (c) shows that as the network density increases, C_{update} for all three distributions increases, and the increment is sharper when the distribution variance is smaller. As explained in the previous study, when a distribution is more skew, more partition splits occur and thus generate more control overhead. In addition, the cost of splitting a partition is proportional to the square of the population of the partition.

4.1.4.3 Effect of node mobility

This study investigated the impact of node mobility on the performance of our cloaking algorithms. We deployed 400 nodes in the network domain, and varied the speed of mobile nodes from 1 m/s to 5 m/s. Under a random walk model, the distribution of mobile nodes will become more and more even as time elapses. In order to study the effect of node mobility on the distribution change, we ran all simulations within a same time interval. The performance results are shown in Figure 4.7.

Figure 4.7 (a) shows that as the moving speed increases, the cloaking area decreases. This is due to the fact that a higher mobility causes a skew distribution to become even faster. When the distribution becomes even, the size of cloaking boxes is smallest. Figure 4.7 (b) shows that the curves for C_{init} are flat. Since this cost is measured before nodes start to move, it is not affected by the node mobility. In contrast, as node mobility increases, C_{update} increases under all the three distributions, as showed in Figure 4.7 (c). When the mobility is higher, there are more events that nodes move out of their cloaking boxes. As the figure shows, the increment of C_{update} becomes smaller when the mobility increases. This can be explained as follows. When the average moving speed is higher, it takes less time for the node distribution to become even. When the distribution is even, the majority update cost comes from LEAVE and JOIN, the frequency of which will become stabilized.

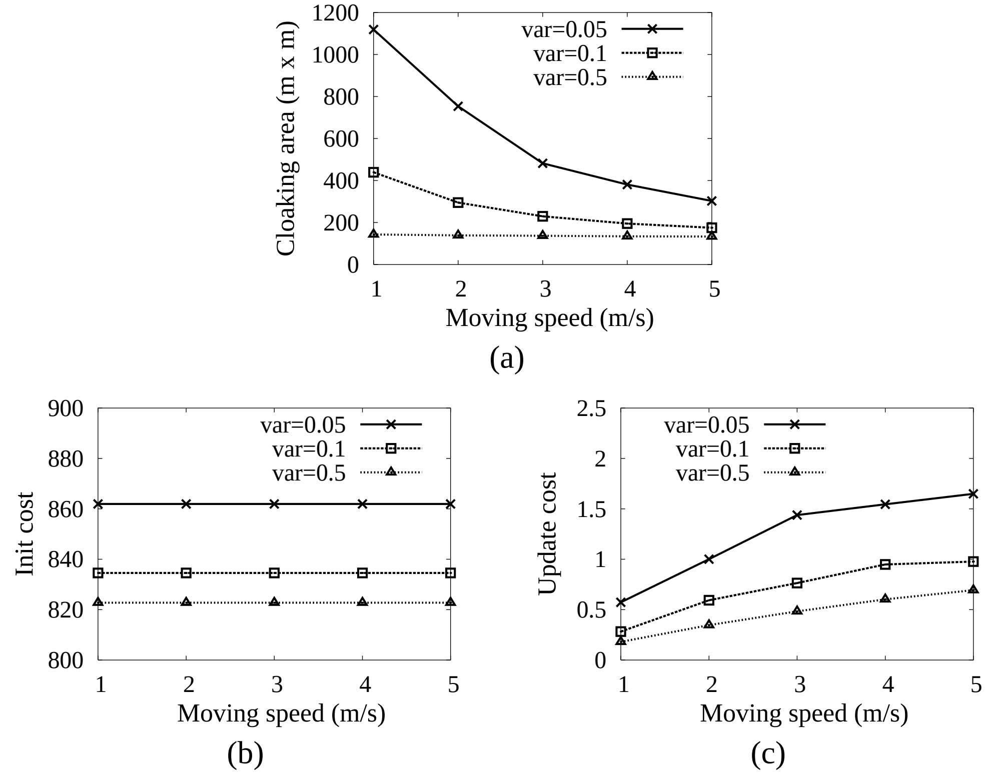


Figure 4.7 Effect of node mobility

4.2 Location secure routing

The above cloaking techniques show that reducing the location resolution is effective in protecting wireless nodes' location safety. However, it has a significant impact on the geographic routing protocols in ad hoc networks. First, existing protocols may suffer efficiency loss. For example, in order to deliver a packet to a destination node, we should broadcast the packet in the node's cloaking box, which incurs a lot of routing overhead. More importantly, the operations of these protocols such as packet forwarding may allow an adversary to refine a node's location resolution, thus reducing the required level of protection.

As an example, consider Figure 4.8. It shows three nodes and the corresponding cloaking regions that they disclose [1]. Suppose A sends a packet. If B forwards the packet, then A would know that B must be within its transmission range, thus allowing it to refine B's location. To prevent such location refinement, a node should avoid forwarding a packet unless its cloaking region is completely covered by the sender's transmission range. In this example, only C can forward a packet originating from A. The problem is how a receiving node can determine if this condition is satisfied. It may first appear that we can let the sender advertise its transmission radius. But doing this would place the sender in danger. If B knows A's transmission radius, it can refine A's location based on the signal strength it receives. Indeed, even if a node makes its transmission range known, such information cannot be trusted. This is due to the fact that the node may be compromised and falsify the information for location refinement attack.

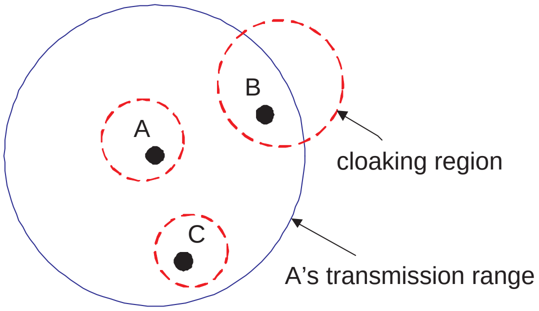


Figure 4.8 Location refinement attack

Our research in this section investigates the impact of location cloaking on geographic ad hoc routing protocols and introduce a new concept called *safe link*. A network link is said to be a safe link if the packet delivery through the link does not allow an adversary to refine the sender and receiver's location resolution. Assuming asymmetric communication links, where nodes keep their transmission radius in secret, we propose a solution that allows a node to determine whether or not a link is safe based on the received signal strength. With this technique in place, we propose a location secure routing protocol (LSR). Like existing protocols such as

[1] Throughout this chapter, we will use a dashed circle to denote a node's cloaking region and a solid one its transmission coverage.

GPSR (3), LSR tries to route a packet using nodes closer to the destination whenever possible, and if not, detours the packet along some faces in the network connectivity graph. However, LSR constructs a routing path using only safe links. Moreover, it can work with inaccurate location information. To our knowledge, LSR is the first ad hoc routing technique that is designed with built-in mechanisms to prevent routing activities from being used to refine nodes' location.

4.2.1 Safe link

We consider an ad hoc network deployed in a two-dimension space. Without loss of generality, we assume each location reported by nodes is a *cloaking circle*. As mentioned early, A node's cloaking circle needs to contain the its position and satisfy other conditions, depending on the protection type (privacy or safety) and the level of protection. Since the focus of this work is on the design of routing mechanism, we will not concern ourselves the details of computing a cloaking circle, but simply assume some existing technique is used.

We assume the adversary has access to the communications among the networking nodes (e.g., it can be one of these nodes) and know nodes' location information which they disclose in packet delivery. The adversary is interested in location refinement attack, which is to derive more accurate location than reported. As discussed in the introduction, the key to prevent such attack is to ensure that data packets are forwarded only through safe links. That is, when a node receives a data packet, it should not forward the packet unless the cloaking circle it discloses is completely covered by the sender's transmission range. The problem is how to verify this forwarding condition without knowing the sender's transmission radius. Here we present two approaches.

The first approach lets a node estimate if its cloaking circle is completely covered using only the signal strength it receives. According to the Free Space Model (65), given a pair of sender and receiver with a distance of d, the signal strength at the receiver can be computed as

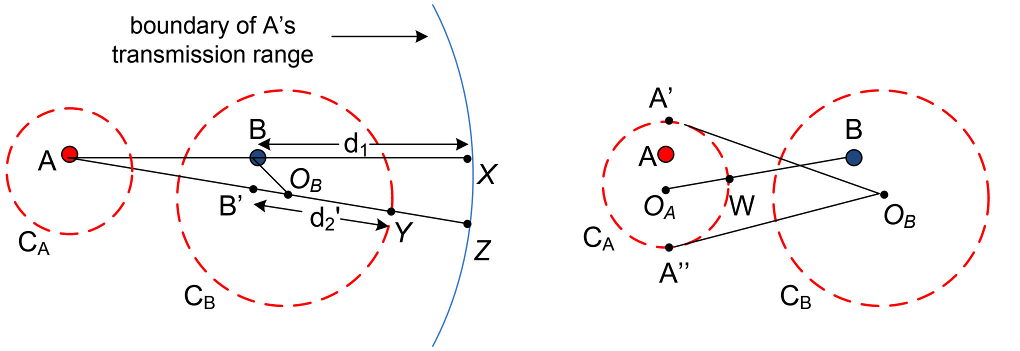
$$P_r = k\frac{P_t}{d^2}, \tag{4.1}$$

where P_t is the sender's transmitting power (i.e., the signal strength at the sender site) and k is a constant. Thus, given a received signal strength P_r, a node can estimate the distance that the signal can be further propagated as $d_1 = (k\frac{P_r}{P_l})^{\frac{1}{2}}$, where P_l is the minimum signal strength that is detectable to a node. Let d_2 be the maximum distance from the node's current position to the boundary of its cloaking circle. If $d_2 \leq d_1$, the receiving node can be assured that its cloaking circle is within the sender's transmission range.

The above scheme is simple to implement and guarantees zero false positive verification. It, however, is a pessimistic solution as it assumes the worst situation: the sender is right on the receiver's position using the minimum transmission power. In reality, the sender can be far away, which means that the actual transmission power is larger than the signal strength sensed by the receiver. Since we assume each node is willing to disclose its cloaking circle, and location verification techniques (e.g., (66; 67)) can be applied to verify the trustworthiness of such information, we can take advantage of the cloaking circle disclosed by the sender to estimate its transmission coverage.

Figure 4.9 (a) shows two nodes, A and B, and their corresponding cloaking circles C_A and C_B. Suppose B receives a data packet sent by A and the received signal strength is P_r. If B knows A's exact position, B can compute A's transmission power $P_t = \frac{P_r \cdot |AB|^2}{k}$, where $|AB|$ denotes the distance between A and B, and A's transmission radius $r = (k\frac{P_t}{P_l})^{\frac{1}{2}}$, where P_l is the minimum signal strength detectable to a node. B can then derive A's transmission coverage, which is the circle centered on A's position with a radius of r, and check if C_B is within the coverage. The problem is, B does not know A's exact position, but only knows that it is inside C_A. A simple solution is to compute A's transmission coverage for every position in C_A. If each of these possible coverages contains C_B, then B can forward the data packet. This approach, however, requires intensive computation.

To address the above problem, we have come up a more efficient approach. Let $d_1 = |BX|$ be the shortest distance from B to the boundary of A's transmission range. Let O_B denote the center of B's cloaking circle C_B, Y be the point that line $\overline{AO_B}$ intersects with C_B, and Z be the point that $\overline{AO_B}$ intersects with the boundary of A's transmission range. Let B' be the point on line $\overline{AO_B}$ such that $|AB| = |AB'|$, and d_2 denote the length of segment $B'Y$. These notations are illustrated in Figure 4.9 (a). Our solution is based on the following theorem.



(a) Verify if C$_B$ is covered by A's transmission range

(b) Estimate the value of d$_1$ and d$_2$'

Figure 4.9 Safe link verification

Theorem 1. *C_B is entirely covered by A's transmission range $\iff d_1 \geq d_2$.*

Proof. First of all, since $\overline{AO_B}$ overlaps with the diameter of C_B, we have the equivalence: C_B is entirely covered by A's transmission range $\iff |AZ| \geq |AY|$. In addition, since $|AX| = |AZ|$ which is the transmission radius, and $|AB| = |AB'|$, we have the equivalence: $d_1 \geq d_2 \iff |AX| \geq |AY| \iff |AZ| \geq |AY|$. Therefore, combining the two equivalences, we can infer that C_B is entirely covered by A's transmission range $\iff d_1 \geq d_2$. □

So the question now is how to compute d_1 and d_2. Let's first consider d_1. We know that $d_1 = r - |AB| = P_t^{\frac{1}{2}}[(\frac{k}{P_l})^{\frac{1}{2}} - (\frac{k}{P_r})^{\frac{1}{2}}]$. Since P_l, k, and P_r are all fixed, d_1 is determined by A's transmission power P_t, and a smaller P_t results in a smaller d_1. According to Equation 4.1, $P_t = \frac{|AB|^2 P_r}{k}$, P_t has a smaller value when A is closer to B. So P_t is smallest when A is at W (see Figure 4.9 (b)), the point where line $\overline{O_A B}$ intersects with the boundary of C_A. As such, B can use the distance $|WB|$ to compute the minimum value of P_t and then compute the minimum value of d_1. Clearly, as long as C_A does not cover B's position, the minimum value of P_t is larger than P_r, and therefore, d_1 will be larger than derived by the first approach.

Algorithm 6 Link safety verification

```
LinkStatus($p$) // executed by each node
$P_r \leftarrow$ receiving signal strength of $p$
$C_A \leftarrow$ cloaking circle of sender $A$
$C_B \leftarrow$ cloaking circle of receiver $B$
$\{$estimate minimum $d_1\}$
$W \leftarrow$ intersection of $\overline{BO_A}$ and $C_A$
$d_{min} \leftarrow |WB|$
$P_{min} = \frac{P_r d_{min}^2}{k}$
$d_1 \leftarrow P_{min}^{\frac{1}{2}}[(\frac{k}{P_l})^{\frac{1}{2}} - (\frac{k}{P_r})^{\frac{1}{2}}]$
$\{$estimate maximum $d_2\}$
compute two tangent lines $\overline{A'O_B}$ and $\overline{A''O_B}$
**if** $B$ is in between $\overline{A'O_B}$ and $\overline{A''O_B}$ **then**
    $d_2' \leftarrow |BO_B| + R$
**else**
    $d_2' = MAX\{|A'O_B| - |A'B|, |A''O_B| - |A''B|\} + R$
**end if**
**if** $d_1 \geq d_2'$ **then**
    return SAFE
**else**
    return UNSAFE
**end if**
```

We now consider d_2. As shown in Figure 4.9 (a), $d_2 = |AY| - |AB| = |AO_B| - |AB| + R$ where R is the radius of C_B. Since in the triangle $\triangle AO_BB$ the length of edge BO_B does not change no matter where A is, $|AO_B| - |AB|$ has a larger value when the angle $\angle ABO_B$ is larger and its maximum value is $|BO_B|$. Therefore, B can first calculate the two tangent lines of C_A which pass through O_B, denoted as $\overline{A'O_B}$ and $\overline{A''O_B}$, as illustrated in Figure 4.9 (b). If B is in between the two lines, $|AO_B| - |AB|$ has the maximum value when the angle $\angle ABO_B = \pi$ and in this case the maximum value of d_2 is equal to $|BO_B| + R$. Otherwise, B computes $|A'O_B| - |A'B| + R$ and $|A''O_B| - A''B| + R$ respectively and chooses the larger one as d_2's maximum value. Thus, if the maximum d_2 is no larger than the minimum d_1, B can be assured that it is within A's transmission range. We can see that this approach can compute a more accurate maximum value of d_2 as long as B is not in between the two tangent lines. Thus, it generates less false negative verification of safe links. The pseudo code of the second verification approach is shown in Algorithm 6.

4.2.2 LSR

We now consider how to construct a routing path with safe links and inaccurate location. Each data packet has the following fields in its header: *src_pos* (the location of the source node), *dst_pos*(the location of the destination node), *fwr_pos* (the location of the node which is forwarding this packet). When a source node sends a packet to a destination node, it fills in *src_pos* and *dst_pos*. The field *fwr_pos* is initialized with the sender's location. When a node forwards this packet, it updates the field with its own location.

Similar to other protocols like GPSR, the proposed LSR also works in two modes: 1) *greedy routing*, which is used whenever possible; 2) *face routing*, which is used where greedy routing does not work. We explain these two modes in the following subsections.

4.2.2.1 Greedy routing

When a node receives a packet, it computes whether the link is safe and whether it is closer to the destination than the sender. If any one of the two conditions is not satisfied, the node drops the packet. Otherwise, it waits for a certain time period. During the waiting time, if the node eavesdrops the same packet forwarded by some other node, it drops the packet. Otherwise, it forwards the packet. The length of the waiting period is set to be proportional to the distance between the node and the packet destination. As such, a node closer to the destination waits shorter and has a higher probability to forward. If a node forwards a packet, it also sends an acknowledgement packet back to the sender with a transmitting power which is ensured to cover the sender's cloaking circle. If the sender does not receive any acknowledgement, it means there is no safe link to any other node closer to the destination. When this happens, the packet forwarding switches to face routing mode, which we will discuss later.

The above strategy constructs packet routing path using only safe links. It is worth mentioning that this approach is also lightweight as it avoids proactive location advertising. In existing protocols like GPSR, a forwarding node needs to know all of its neighbors' position in order to choose the next hop that is closest to the destination. For this purpose, every node is required to periodically update its latest location to its neighbors. This not only incurs significant routing overhead, but also is subject to location refinement attack. When a node makes frequent location updates, the time-series sequence of the corresponding cloaking areas may be correlated to refine its location.

One problem of implementing the above routing scheme is how to compute the distance between two nodes. To make a forwarding decision, a node needs to compute the distance from the packet sender and itself to the destination node. Since no node reveals its exact position, we estimate the distance between two nodes by measuring the average distance between a pair of points in the two cloaking circles respectively, defined as follows:

$$D(C_1, C_2) = \frac{1}{A_1 A_2} \int_{C_1} \int_{C_2} Dist(p_1, p_2)\, dp_1\, dp_2, \tag{4.2}$$

where C_1 and C_2 are the two cloaking circles; $Dist(p_1, p_2)$ is the Euclid distance between a position p_1 in C_1 and a position p_2 in C_2; A_1 and A_2 are the area of C_1 and C_2 respectively.

Another problem is how to deliver the packet to the destination node without knowing its accurate location. It may first appear that we can simply apply the above routing scheme to forward the packet and when the packet reaches the cloaking circle of the destination node, a regional broadcast in the cloaking circle can be launched to accomplish the delivery. This strategy, however, does not guarantee the packet delivery. Since nodes use the average distance defined in Equation 4.2 to make a forwarding decision, the position where the packet enters the cloaking circle of the destination node may not be the one that is closest to its actual position. If the sub-network in the cloaking circle is not connected the packet may not be able to reach

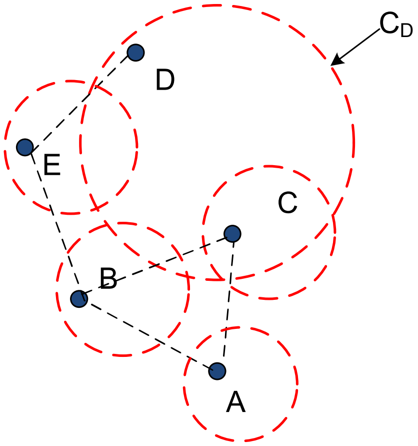


Figure 4.10 An example of delivery failure.

the destination node. In the example shown in Figure 4.10, the packet is forwarded to C in the cloaking circle C_D of destination node D. Since C cannot reach from D directly, the packet delivery will fail even with a regional broadcast of the packet within C_D. Yet, there actually exists an external path from C to D, which is $A \rightarrow B \rightarrow E \rightarrow D$.

To cope with this problem, we proposed to involve more nodes in routing when a packet reaches the cloaking circle of the destination node. If a node's transmission range overlaps the cloaking circle of the destination node, it will forward the packet as long as the link is safe, no matter whether it is closer to the destination than the sender, and whether it eavesdrops the packet forwarded by other nodes. In the same example as shown in Figure 4.10, the node B will also forward the packet from A even though the packet has been forwarded by C. As a result, the packet can reach the destination along the route $A \rightarrow B \rightarrow E \rightarrow D$. The pseudo code for our proposed greedy routing is given in Algorithm 7.

4.2.2.2 Face routing

When a node forwarding a packet does not have any neighbor that is closer to the destination and forms a safe link, the packet reaches a dead-end in greedy routing mode. When this happens, the packet forwarding switches to face routing mode, in which the packet delivery is detoured around the dead-end until a closer next-hop is found. The face routing in LSR consists of three components: 1) connectivity graph generation, 2) planarization, 3) packet forwarding. We discuss them as follows.

Algorithm 7 LSR greedy routing

```
Greedy($p$) // executed by each node
$s \leftarrow LinkStatus(p)$
if $s$ = UNSAFE then
    drop packet and return
else
    $C_D \leftarrow$ cloaking circle of the destination
    if node's transmission range overlaps with $C_D$ then
        forward packet and return
    else
        $d' \leftarrow$ distance from node to destination
        $d'' \leftarrow$ distance from sender to destination
        if $d' \geq d''$ then
            forwarding $\leftarrow$ false
        else
            forwarding $\leftarrow$ true
        end if
    end if
end if
if forwarding = true then
    wait $T$
    if packet is forwarded by others during waiting then
        drop packet
    else
        forward packet
    end if
else
    drop packet
end if
```

The first component constructs the network connectivity graph with safe links. Specifically, when a node A switches to the face routing, it locally broadcasts a query packet. When receiving such a query, each of its neighbors verifies its link safety and sends an acknowledge back if the verification result is positive. This allows A to figure out all its outgoing links in the connectivity graph. Figure 4.11 (a) shows an example where a subnetwork containing three nodes is mapped to the directed graph in Figure 4.11 (b). Note that since each node can have its own cloaking circle and transmission range, the safe links in the connectivity graph are asymmetric, and such asymmetric links can lead to routing failures. This will be explained later when we discuss the third component.

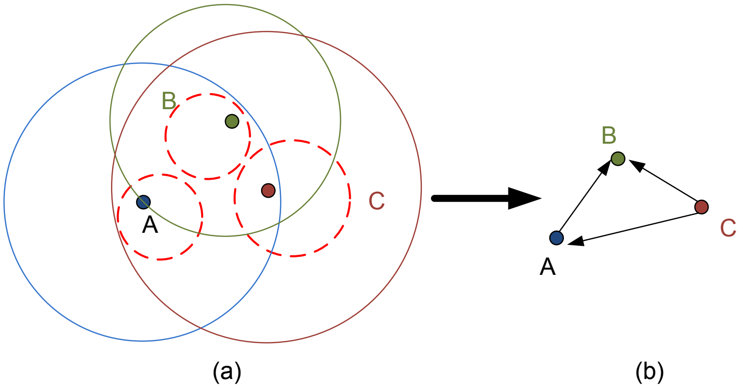


Figure 4.11 An example of connectivity graph generation.

Similar to GPSR, face routing in LSR forwards a packet using right-hand rule. The correct operation of right-hand rule requires the network connectivity graph be planar, which does not contain cross links. Therefore, the second components in LSR planarizes the network connectivity graph by removing the cross links. It is worth mentioning that in the presence of location inaccuracy, planarization may disconnect the network. This problem has been addressed in (68; 69). LSR just applies the techniques in these papers to ensure the network connectivity after planarization.

The third component applies the right-hand rule and forwards a packet around the dead-end until finding a closer next-hop to the destination. The major challenge is how to deal with routing failures brought by asymmetric links in the network. In a planar graph, the connecting line from the source to the destination must go through a number of open or close elementary faces [2]. If the connectivity graph is undirected, i.e., all links are symmetric, a packet will be routed around these faces one by one using the right-hand rule from the source to the destination. For example in Figure 4.12, a packet from S to D should be routed around faces SAB, ABF, $BEGF$, and FGD consequently. In LSR, however, the network connectivity

[2] An elementary face is a face that does not contain other faces

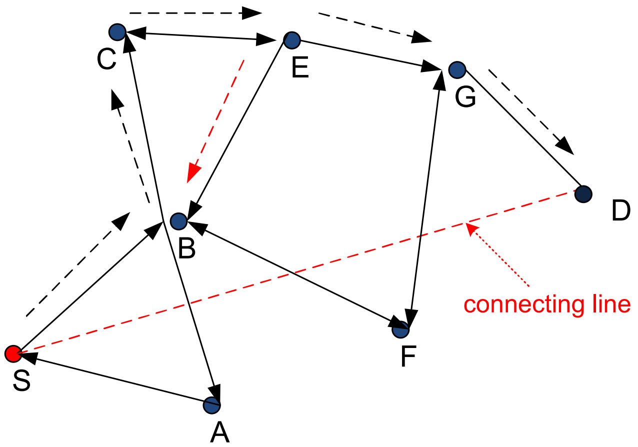


Figure 4.12   An example of face routing.

graph is directed. Simply applying the right-hand rule may not be able to route a packet correctly, because sometimes a packet cannot be routed around the boundary of a face in a directed graph. For example, as shown in Figure 4.12, when the packet reaches B, it cannot be forwarded around $BEGF$ since there is no link from B to E. Following the right-hand rule, the packet will be forwarded to C, and then it will loop around BCE until TTL is exhausted.

To solve this problem, we need to prevent the packet being routed in a face like BCE which is not crossed by the connecting line from the source to the destination. Our strategy is as follows. When a packet cannot be forwarded around a face **F** on the connecting line, we deactivate the one-way link, say l', which stops the packet forwarding (e.g., link $\overline{EB}$ in Figure 4.12), and then route the packet around the face $\mathbf{F}'$ which is composed of **F** and the elementary face on the other side of l'. Since l' is deactivated, $\mathbf{F}'$ must be an elementary face on the connecting line. In the example shown in Figure 4.12, LSR will route the packet around face $BCEGF$.

Algorithm 8 shows the pseudo code for a node to relay a packet p in face routing mode. We let every forwarding node include its identity in the routing header of the packet, and thus all the en-route nodes who have forwarded this packet form a *forwarding list* in the header. Then, when a new forwarding node selects a link using right-hand rule, it checks whether the node on the other side of the link already exists in the forwarding list. If not, the node forward the packet on this link. Otherwise, it means that the packet is being forwarded around a face not on the connecting line. Thus, the node deactivates this link and chooses the next available link using right-hand rule. In the same example shown in Figure 4.12, when the packet reaches E, nodes S, B and C has been included in the forwarding list. Following right-hand rule, E selects link $\overrightarrow{EB}$ first. But B exists in the forwarding list. Thus, E finds the next available link $\overrightarrow{EG}$. Since G is not in the forwarding list, E will forward the packet though this link. As a result, the packet will be routed around face $BCEGF$, and finally reaches destination D.

Algorithm 8 LSR face routing

```
Face($p$) // executed by each node
$l \leftarrow$ link where $p$ is sent from
$L \leftarrow$ forwarding list of $p$
while true do
    {find next link using right-hand rule}
    $l' \leftarrow RightHand(l)$
    if $l' = null$ then
        {no link available}
        drop packet and return
    else
        $N \leftarrow$ node on the other side of $l'$
        {check if this link should be deactivated}
        if $N \in L$ then
            continue
        else
            {add $N$ to forwarding list and forward packet}
            $L \leftarrow L + N$
            forward packet and return
        end if
    end if
end while
```

4.2.3 Performance evaluation

For performance evaluation, we have developed a detailed simulator. We implement two versions of LSR. The two schemes, which we will refer to as LSR-basic and LSR-adv, are different in the way of verify link safety. In LSR-basic, a node receiving a data packet uses only the received signal strength to determine if it is safe to forward the data packet, while in LSR-adv, a node would leverage both the signal strength and the location disclosed by the sender. For comparison purpose, we have also implemented an approach referred to as *Native*. This scheme lets a node forwards a received packet as long as it is closer to the destination than the sender. However, unlike LSR_{basic} and LSR_{adv}, this scheme lets a node forward a data packet without verifying link safety and is therefore subject to location refinement attack. We are mainly interested in two performance metrics:

- *Cloaking area*: The location disclosed by a node can be refined if the node participates in data forwarding without considering link safety. The metric of cloaking area is defined to be the size of a node's cloaking circle known to an adversary after location refinement. As such, this metric measures the degree that a node's location can be refined. We report the average cloaking area of all nodes.

- *Delivery rate*: This metric is defined to be the ratio between the number of data packets that are successfully delivered to their destination and the total number of data packets transmitted. The delivery rate measures how reduced location resolution and the forwarding conditions have impact on the routing performance.

Table 4.2 summarizes the parameters used in our study. Unless otherwise specified, the default values are used. We simulate an ad hoc network in which nodes are deployed in a rectangular domain. In each simulation, we generate a number of nodes and randomly place them in a 1000×1000 m^2 network domain. That is, each node's coordinates in X and Y axis are randomly chosen from $[0, 1000]$. After deployment, each node initializes its exposing location as a cloaking circle with a radius randomly chosen from [r_min, r_max]. In each simulation, we create a number of routing tasks, each containing a pair of source and destination nodes which are selected randomly. During routing, a forwarding node randomly chooses its transmission radius from [R_min, R_max]. In the next subsections, we focus on how the two performance metrics are affected by the initial size of nodes' cloaking circles, network density, and nodes' transmitting power.

Table 4.2 Parameters

parameter	range	default	unit
network domain	1000×1000	1000×1000	$meter^2$
node number	800 - 1200	1000	$unit$
r_min	20	20	$meter$
r_max	40 - 80	60	$meter$
R_min	60	60	$meter$
R_max	100 - 200	150	$meter$
routing tasks number	300	300	$meter$

4.2.3.1 Effect of initial cloaking circle

This study investigates the impact of nodes' cloaking circle size on the performance. We generate 1000 nodes and deploy them randomly in the network domain. The value of maximum radius of nodes' initial cloaking circle is varied from 40 to 80 meters, and the performance results are plotted in Figure 4.13. Figure 4.13 (a) shows that Native results in a much smaller cloaking area and thus fails to protect nodes' location privacy/safety. Since this scheme let a node forward a data packet as long as it is closer to the destination than the sender, it is prone to the location refinement attack. In contrast, the proposed LSR lets a node avoid forwarding a data packet whenever it determines that the link is not safe. As such, a node's location resolution known to an adversary is the same as that disclosed by the node. Note that both LSR-basic and LSR-adv have this feature, so they share the same line.

Figure 4.13 (b) shows that both LSR-basic and LSR-adv have a smaller delivery rate than Native, and the delivery rate decreases as the size of the cloaking circle increases. In LSR, a packet is forwarded only via safe links. Thus, sometimes there is not a safe path from the source to the destination even if the network is connected. When a cloaking circle increases, the chance of its being totally covered by a node's transmission range reduces. Thus, there

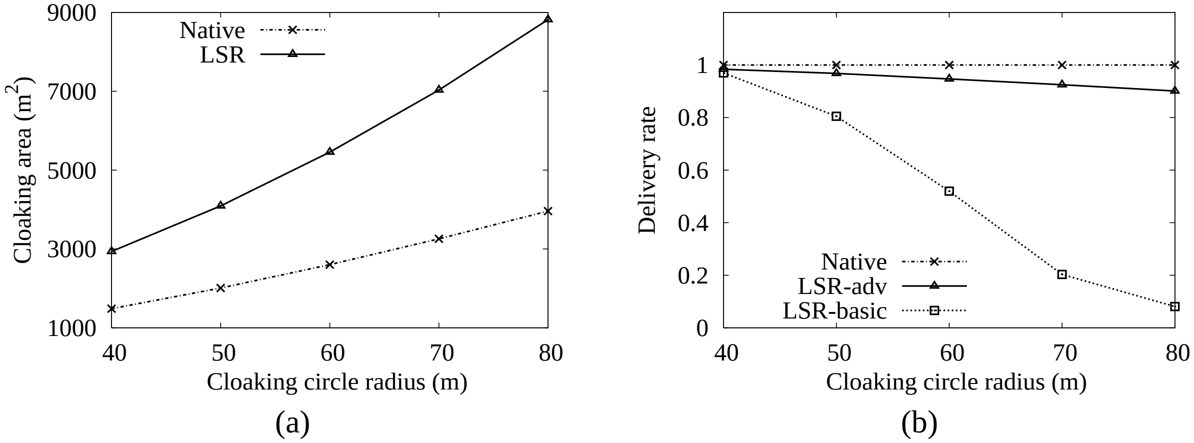
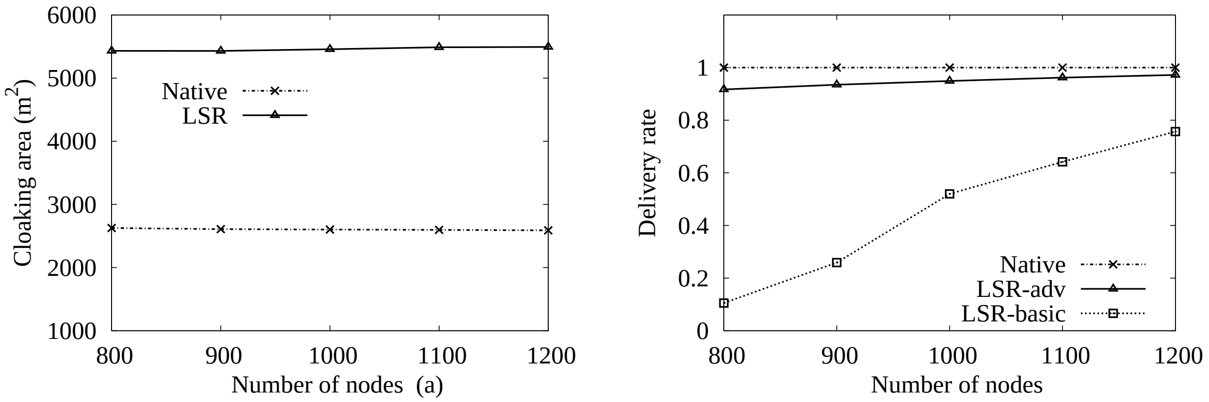


Figure 4.13 Effect of cloaking circle

are less safe links in the network when the size of cloaking circles increases, making it more difficult to find a safe route. Figure 4.13 (b) also shows that the delivery rate of LSR-basic decreases much faster than that of LSR-adv. This indicates that the basic scheme has a much higher false negative rate on safe link verification. When the size of cloaking circle is larger, the number of safe links become too small for an end-to-end packet delivery. On the other hand, as we can see the delivery rate of LSR is always more than 90%. It is worth mentioning that Native has 100% data delivery rate. This, however, is achieved at the expense that the location of nodes is known to an adversary more accurately.

4.2.3.2 Effect of network density

This study investigates the impact of network density on routing performance. We fix the maximum radius of initial cloaking circle as 60 meters, and vary the node number from 800 to 1200. The performance results are plotted in Figure 4.14. As showed in Figure 4.14 (a), the cloaking area of Native is always much lower than LSR, and it is not affected by network density. This is due to the fact that in Native the next hop is always selected as the closest one to the destination, and the distance between two consecutive forwarding nodes is not affected



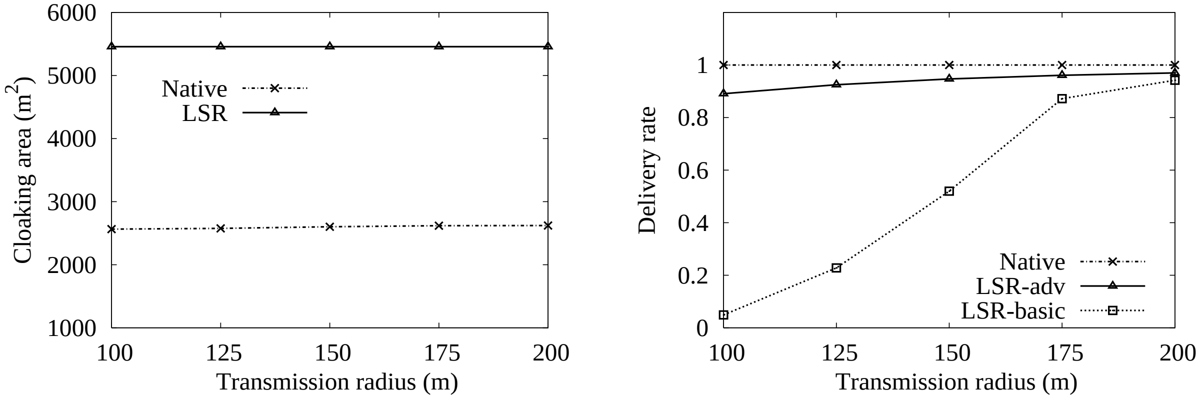
(a) (b)

Figure 4.14 Effect of network density

by the network density. Figure 4.14 (b) shows that the delivery rate of Native is always 100% which means the network is connected under all the above system settings. On the other hand, the delivery rate of both LSR-adv and LSR-basic increases as the network become denser. In a denser network, nodes are closer to each other and this results in more safe links. Thus, the chance of having a safe path from a source to a destination is higher. We can also see that the performance of LSR-basic is very sensitive to the network density. The the density is lower, the delivery rate deteriorates quickly. In contrast, the performance of LSR-adv is much stable.

4.2.3.3 Effect of transmitting power

This study investigates the impact of transmitting power on the performance of the routing protocols. We deploy 1000 nodes in the network domain, and vary nodes' maximum transmission radius from 100 m to 200 m. The performance results are shown in Figure 4.15. Figure 4.15 (a) shows that as the transmitting power increases, the cloaking area of Native has a very slight increment. This is due to the fact that only forwarding nodes' location may be refined and a larger transmission range decreases the hop number of the route between a pair of source and destination. Figure 4.15 (b) shows that the delivery rate of both LSR-basic and LSR-adv



(a) (b)

Figure 4.15 Effect of transmitting power

increase as the transmitting power increases. It is because under the same network density, a larger transmission range can cover more nodes' cloaking circle, and thus generates more safe links. More safe links in a network make it easier to find a safe route from a source to a destination. Figure 4.15 (b) also shows that LSR-basic is acceptable as a practical routing scheme only when the nodes' transmitting power is high, and comparatively LSR-adv is preferable in most scenarios. In the previous study, we have seen a similar result.

CHAPTER 5. Conclusion and Future Work

5.1 Summary of Contributions

This thesis includes two of our research subjects, location privacy protection and location safety protection. Their objectives are different but highly related. The major issue behind is how to let wireless users disclose their location information to enable many network applications, while preventing such location being used to compromise their privacy and safety. In summary, our contributions are:

Exploring historical location data for location privacy protection. Personal location data can be correlated with restricted spaces such as home and office addresses for subject re-identification. This is probably the most practical and economic way for an adversary to identify the anonymous users of LBSs. Existing location depersonalization techniques proposed to address this problem can support anonymous uses of LBSs, but not location privacy protection. We proposed to explore users' historical location data for location depersonalization. A cloaking region with different footprints means it has been visited by different people. Thus, if a user's location is disclosed as such a region, even though an adversary can identify all these visitors with restricted spaces, he will not know which of them was inside the area at the time of the service request.

Feeling-based privacy modeling. In order to get a certain level of privacy protection, a user needs to determine her privacy requirement. In this thesis, we address the challenge of modeling location privacy requirement. We first show that the traditional K-anonymity model is problematic, because privacy is about feeling, and it is difficult to scale one's feeling using a number. Then we propose our solution which circumvents this problem by allowing a user to identify her *public region*, a spatial region which she would feel comfortable that it is reported as her location should she request a service inside it. Compared to choosing a number of K, this *feeling-based* strategy provides a much more intuitive way for users to express their privacy requirement.

Distributed location cloaking for location safety protection. We differentiate location safety with location privacy. Here adversary in not interested in identifying a wireless node in a network, but locating and destroying it. We propose to protect location safety by reducing location resolution and we define *safety level* of a cloaking region. We identify three challenges of location cloaking and address these challenges by developing distributed cloaking algorithms for both static and mobile ad hoc networks.

Secure location-based routing with cloaked location. We discuss the impact of location cloaking on geographic routing protocols. Cloaked location downgrades the routing efficiency, and operations of routing will in turn jeopardize the safety protection provided by location cloaking. Our research address this issue with a novel location secure routing protocol called LSR. In LSR, the routing packets are forwarded only on *safe links* which ensure that adversaries cannot refine nodes' location resolution by analyzing the routing traffic.

5.2 Future Research

We envision extending this research along the following directions:

Modeling and thwarting new types of location privacy intrusion in LBS. Our current work considers only restricted space identification, but other types of attacks are likely. For example, a user may be observed at some time. A user under direct observation does not have location privacy, but the observed point may allow an adversary to learn her other visits. Orthogonal to such *observation* attack is *exclusiveness* attack. If a user is known to never visit a location, she cannot be the subject of any trajectory that contains the location. We will model and thwart these and other attacks that may be discovered during the course of this research.

Developing advanced safety cloaking algorithms. Our current design partitions the network domain into subdomains and each node take some subdomain as its cloaking box. Since the safety level of a subdomain can be up to 2θ, the cloaking box can be further refined to have a safety level as close as possible to, but no less than, θ. A possible solution is to apply secure multi-party computation (SMC) (22) that allows multiple nodes to jointly evaluate with their private values while ensuring that no one can learn additional information other than the evaluation results. In addition to such improvement, we will investigate differential safety cloaking, in which different nodes may need different levels of safety protection. The motivation is that instead of providing the same level of protection to all nodes, we can let less important nodes report more accurate location to improve network efficiency. The main research effort will be on preventing correlation attack. It would be interesting to see how such differential cloaking can complement the existing anonymous routing protocols (e.g., (38; 41; 43)) for safety protection.

BIBLIOGRAPHY

[1] Y. Ko and N. H. Vaidya. Location-Aided Routing (LAR) Mobile Ad Hoc Networks. In *Proc. of ACM International Conference on Mobile Computing and Networking MOBICOM'98*, pages 66–75, Dallas, TX, U.S.A, 1998.

[2] E. Royer and C. E. Perkins. Multicast Operation of the Ad-hoc On-Demand Distance Vector Routing Protocol. In *Proc. of ACM MOBICOM'99*, pages 207–218, Seattle, WA, U.S.A, August 1999.

[3] B. Karp and H. T. Kung. GPSR: Greedy Perimeters Stateless Routing for Wireless Network. In *Proc. of ACM MOBICOM'00*, pages 243 – 254, Boston, MA, U.S.A, 2000.

[4] L. Ackerman, J. Kempf, and T. Miki. Wireless Location Privacy: Law and Policy in the U.S., EU and Japan. http://www.isoc.org/briefings/015/.

[5] J. R. Cuellar, J. B. Morris, and D. K. Mulligan. Geopriv Requirements. In *Internet draft*,

[6] S. Duri, M. Gruteser, X. Liu, P. Moskowitz, R. Perez, M. Singh, and J. Tang. Framework for Security and Privacy in Automotive Telematics. In *Proceedings of the second International Workshop on Mobile Commerce*, pages 25–32. ACM Press, 2002.

[7] G. Myles, A. Friday, and N.Davies. Preserving Privacy in Environments with Location-based Applications. In *IEEE Pervasive Computing*, volume 02, pages 56–64, 2003.

[8] M. Langheinrich. A Privacy Awareness System for Ubiquitous Computing Environments. In *4th International Conference on Ubiquitous Computing*, volume 2498, pages 237–245, 2003. 1998.

[9] M. Gruteser and D. Grunwald. Anonymous Usage of Location-based Services through Spatial and Temporal Cloaking. In *ACM MobiSys'03*, pages 31–42, 2003.

[10] B. Gedik and L. Liu. A Customizable k-Anonymity Model for Protecting Location Privacy. In *ICDCS'05*, pages 620–629, 2005.

[11] C. Bettini, X. S. Wang, and S. Jajodia. Protecting Privacy Against Location-Based Personal Indentification. In *Proceedings of the 2nd VLDB Workshop on Secure Data Management*, 2005.

[12] A. Inan and Y. Saygin. Location Anonymity in Horizontally Partitioned Spatial-Temporal Data. In *Master Thesis, Sabanci University, Turkey*, 2006.

[13] R. Cheng, Y. Zhang, E. Bertino, and S. Prabhakar. Preserving User Location Privacy in Mobile Data Management Infrastructures. In *the 6th Workshop on Privacy Enhancing Technologies*, pages 393–412, 2006.

[14] M. F. Mokbel, C.-Y. Chow, and W. G. Aref. The New Casper: Query Processing for Location Services without Compromising Privacy. In *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB'06)*, pages 763–774, 2006

[15] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias. Preserving Anonymity in Location Based Services. In *Technical Report TRB6/06, Department of Computer Science, National University of Singapore*, 2006.

[16] C. Y. Chow, M. F. Mokbel, and X. Liu. A Peer-to-Peer Spatial Cloaking Algorithm for Anonymous Location-based Services. In *ACM GIS'06*, pages 171–178, November 2006.

[17] G. Ghinita, P. Kalnis, and S. Skiadopoulos. MobiHide: A Mobile Peer-to-Peer System for Anonymous Location-Based Queries. In *SSTD'07*, pages 221–238, 2007.

[18] G. Ghinita, P. Kalnis, and S. Skiadopoulos. PRIVE: Anonymous Location-based Queries in Distributed Mobile Systems. In *Proc. of the 16th international conference on World Wide Web*, pages 371–380, Banff, Alberta, Canada, 2007.

[19] C. Chow and M. F. Mokbel. Enabling Private Continuous Queries for Revealed User Locations. In *SSTD'07*, pages 258–275, 2007. http://www.ietf.org/internet-drafts/draft-ietf-geopriv-reqs-01.txt, 2002.

[20] H. Hu and J. Xu. Non-Exposure Location Anonymity. In *ICDE'09*, pages 1120–1131, 2009.

[21] I. Stoica, R. Morris, D. Karger, M. Kaashock, and H. Balakrishman. Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications. In *Proc. of ACM SIGCOMM*, pages 149–160, San Diego, CA, U.S.A, 2001.

[22] S. Goldwasser. Multiparty Computations: Past and Present. In *ACM Symposium on Principle of Distributed Computing*, 1997.

[23] U. Leonhardt and J. Magee. Security Considerations for a Distributed Location Services. *Journal of Networks and Systems Management*, 6(1):51–70, March

[24] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.-L. Tan. Private Queries in Location-Based Services: Anonymizers are Not Necessary. In *ACM SIGMOD 2008*, pages 121–132, 2008.

[25] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private Information Retrieval. In *IEEE Symposium on Foundations of Computer Science*, pages 41–50, 1995.

[26] D. Reid. An Algorithm for Tracking Multiple Targets. *IEEE Transactions on Automatic Control*, 24(6):843–854, December 1979.

[27] A. R. Beresford and F. Stajano. Location Privacy in Pervasive Computing. In *IEEE Security and Privacy*, volume 2, pages 46–55, 2003.

[28] B. Hoh and M. Gruteser. Protecting Location Privacy Through Path Confusion. In *IEEE/CreateNet Intl. Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm)*, pages 194–205, 2005.

[29] Manolis Terrovitis and Nikos Mamoulis. Privacy preservation in the publication of trajectories. In *The Ninth International Conference on Mobile Data Management (MDM'08)*, pages 65–72, Washington, DC, USA, 2008.

[30] Osman Abul, Francesco Bonchi, and Mirco Nanni. Never walk alone: Uncertainty for anonymity in moving objects databases. In *ICDE*, pages 376–385, 2008.

[31] Mehmet Ercan Nergiz, Maurizio Atzori, and Yucel Saygin. Towards trajectory anonymization: a generalization-based approach. In *SPRINGL*, pages 52–61. ACM GIS'08 International Workshop on Security and Privacy in GIS and LBS, 2008.

[32] B. C. M. Fung, K. Wang, L. Wang, and P. C. K. Hung. Privacy protection for rfid data. *Proceedings of the 2009 ACM Symposium on Applied Computing*, pages 1528–1535, 2009.

[33] T. Abdelzaher, Y. Anokwa, P. Boda, J. Burke, D. Estrin, L. Guibas, A. Kansal, S. Madden, and J. Reich. Mobiscopes for human spaces. In *IEEE Pervasive Computing*, volume 6(2), pages 20–29, 2007.

[34] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. K. Miu, E. Shih, H. Balakrishnan, and S. Madden. CarTel: A Distributed Mobile Sensor Computing System. In *4th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 125–138, 2006.

[35] O. Riva and C. Borcea. The Urbanet Revolution: Sensor Power to the People! In *IEEE Pervasive Computing*, volume 6(2), pages 41–49, 2007.

[36] A. Kapadia, N. Triandopoulos, C. Cornelius, D. Peebles, and D. Kotz. AnonySense: Opportunistic and Privacy-Preserving Context Collection. In *The Sixth International Conference on Pervasive Computing (PERVASIVE '08)*, volume 5013, pages 280–297, May 2008.

[37] B. Hoh, M. Gruteser, R. Herring, J. Ban, D. Work, J. Herrera, A. Bayen, M. Annavaram, and Q. Jacobson. Virtual Trip Lines for Distributed Privacy-Preserving Traffic Monitoring. In *ACM Mobisys'08*, pages 15–28, June 2008.

[38] S. Jiang, N. Vaidya, and W. Zhao. A Dynamic Mix Method for Wireless Ad Hoc Networks. In *IEEE Milcom'01*, pages 873–877, 2001.

[39] D. Chaum. Untraceable Electronic Mail, Return Addresses and Digital Pseudonyms. In *Communications of the A.C.M.*, volume 24(2), pages 84–88, 1981.

[40] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. *Mobile Computing*, pages 153–181, 1996.

[41] J. Kong and X. Hong. ANODR: Anonymous on demand routing with untraceable routes for mobile ad-hoc networks. In *ACM MobiHoc'03*, pages 291–302, 2003.

[42] M. Reed, P. Syverson, and D. Goldschlag. Anonymous Connections and Onion Routing. In *IEEE Journal on Selected Areas in Communications*, volume 16(4), pages 495–509, 1998.

[43] Y. Zhang, W. Liu, W. Lou, and Y. Fang. MASK: Anonymous On-Demand Routing in Mobile Ad Hoc Networks. In *IEEE Transactions on Wireless Communication*, volume 5(9), pages 2376–2385, 2006.

[44] C. Ozturk, Y. Zhang, and W. Trappe. Source-Location Privacy in Energy Constrained Sensor Network Routing. In *Proceedings of the 2nd ACM workshop on Security of Ad hoc and Sensor Networks (SASN)*, October 2004.

[45] P. Kamat, Y. Zhang, W. Trappe, and C. Ozturk. Enhancing Source Location Privacy in Sensor Network Routing. In *IEEE ICDCS'05*, pages 599–608, June 2005.

[46] Y. Ouyang, Z. Le, G. Chen, J. Ford, and F. Makedon. Entrapping Adversaries for Source Protection in Sensor Networks. In *Proceedings of the International Symposium on on World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 23–34, June 2006.

[47] A. Savvides, C. Han, and M. Srivastava. Dynamic Fine-grained Localization in Ad-hoc Networks of Sensors. In *Proceedings of ACM MOBICOM'01*, pages 166–179, 2001.

[48] A. Nasipuri and K. Li. A Directionality Based Location Discovery Scheme for Wireless Sensor Networks. In *Proceedings of ACM WSNA'02*, pages 105–111, September 2002.

[49] D. Niculescu and B. Nath. Ad Hoc Positioning System (APS) using AoA. In *Proceedings of IEEE INFOCOM'03*, 2003.

[50] Z. Zhi and Y. K. Choong. Anonymizing Geographic Ad Hoc Routing for Preserving Location Privacy. In *IEEE ICDCSW'05*, pages 646–651, 2005.

[51] X. Wu and B. Bhargava. AO2P: Ad Hoc On-Demand Position-based Private Routing Protocol. In *IEEE Transactions on Mobile Computing*, volume 4(4), pages 335–348, 2005.

[52] L. Sweeney. Achieving K-anonymity Privacy Protection Using Generalization and Suppression. In *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, volume 10(5), pages 571–588, 2002.

[53] P. Samarati. Protecting Respondents' Identities in Microdata Release. In *IEEE TKDE*, volume 13(6), pages 1010–1027, 2001.

[54] C. Shannon. The Mathematical Theory of Communication. In *Bell System Technical Journal*, volume 30, pages 50–64, 1948.

[55] Q. He, D. Wu, and P. Khosla. Personal Control over Mobile Location Privacy. In *IEEE Communications Magazine*, volume 42(5), 2004.

[56] K. Sha, Y. Xi, W. Shi, L. Schwiebert, and T. Zhang. Adaptive Privacy-Preserving Authentication in Vehicular Networks (Invited Paper). In *Proceedings of IEEE International Workshop on Vehicle Communication and Applications*, 2006.

[57] K. Ren, W. Lou, K. Kim, and R. Deng. A Novel Privacy Preserving Authentication and Access Control Scheme in Pervasive Computing Environments. In *IEEE Transactions on Vehicular Technology*, volume 55(4), 2006.

[58] T. Brinkhoff. A Framework for Generating Network-Based Moving Objects. In *GeoInformatica*, volume 6(2), 2002.

[59] TIGER/LINE CENSUS FILES. http://www.land.state.az.us/alris/doc/apendh.txt, 1990.

[60] Y. Cai and T. Xu. Design, Analysis, and Implementation of a Large-scale Real-time Location-based Information Sharing System. In *ACM MobiSys'08*, pages 106–117, Breckenridge, Colorado, June 2008.

[61] xda-developers. http://wiki.xda-developers.com/.

[62] C. E. Perkins and E. M. Royer. Ad Hoc On-demand Distance Vector Routing. In *Proc. of IEEE WMCSA*, pages 90–100, 1999.

[63] A. Perrig, R. Canetti, D. Tygar, and D. Song. The TESLA Broadcast Authentication Protocol. In *RSA CryptoBytes*, volume 5(2), pages 2–13, 2002.

[64] R. Thomas, H. Gilbert, and G. Mazziotto. Influence of the Moving of the Mobile Stations on the Performance of a Radio Cellular Network. In *Proc. of Third Nordic Seminar*, Copenhagen, Denmark, September 1988.

[65] T. S. Rappaport96. *Wireless communications, principles and practice*. Prentice Hall, 1996.

[66] Y. Wei, Z. Yu, and Y. Guan. Location Verification Algorithms forWireless Sensor Networks. In *ICDCS'07*, page 70, 2007.

[67] N. Sastry, U. Shankar, and D. Wagner. Secure Verification of Location Claims. In *ACM Workshop on Wireless Security (ACM WiSe)*, pages 1–10, 2003.

[68] K. Seada, A. Helmy, and R. Govindan. On the Effect of Localization Errors on Geographic Face Routing in Sensor Networks. In *IPSN'04*, pages 71–80, California, USA, April, 2004 2004.

[69] Y. Kim, R. Govindan, B. Karp, and S. Shenker. Geographic Routing Made Practical. In *NSDI'05*, pages 217–230, May, 2005 2005.