Privacy-aware Sharing of Location Information

Von der Fakultät Informatik, Elektrotechnik und
Informationstechnik der Universität Stuttgart
zur Erlangung der Würde eines Doktors der Naturwissenschaften
(Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von
Marius Alexander Wernke
aus Stuttgart

Hauptberichter: Prof. Dr. rer. nat. Dr. h.c. Kurt Rothermel
Mitberichter: Prof. Dr. Max Mühlhäuser

Tag der mündlichen Prüfung: 13.03.2015

Institut für Parallele und Verteilte Systeme (IPVS)
der Universität Stuttgart

2015

Acknowledgments

First of all, I would like to thank my supervisor Prof. Dr. Kurt Rothermel for giving me the opportunity to accomplish my research in his group. His guidance during the whole time of my research presented in this thesis was a key factor to its success. Special thanks also go to Dr. Frank Dürr for supporting and helping me during my research with many discussions and valuable feedback.

Furthermore, I want to thank Prof. Dr. Max Mühlhäuser for accepting the task of the second referee.

While working in the Distributed Systems department at the University of Stuttgart, many people supported me with discussions, comments, and valuable suggestions to advance my research. Here, I would like to thank especially Beate Ottenwälder, Dr. Andreas Grau, Dr. Ralph Lange, Patrick Baier, Damian Philipp, Gerald Georg Koch, Zohaib Riaz, Pavel Skvortsov, Lars Geiger, and Dr. Boris Koldehofe.

I also would like to thank the Deutsche Forschungsgemeinschaft (DFG) for their funding of the PriLoc project where the work presented in this thesis belongs to.

Finally, I would like to thank my parents, my sister, and Ina Friedmann for their continuous support while I was working on this thesis.

Contents

List of Figures

List of Tables

List of Abbreviations

AC	Adjacent *fragment* Correlation
ATM	Automated Teller Machine
BSP	Binary Space Partitioning
CPU	Central Processing Unit
DFG	Deutsche Forschungsgemeinschaft
FNR-tree	Fixed Network Range-tree
GLM	Geometric Location Model
GPS	Global Positioning System
HTC	High-Tech Computer
ID	Identifier
IP	Internet Protocol
LBAC	Location-Based Access Control
LBA	Location-Based Application
LS	Location Server
LTE	Long-Term Evolution
MO	Mobile Object
NL	Network Load
NSA	National Security Agency
NC	No *fragment* Correlation
PA	Position Adjustment
PC	Personal Computer
PIR	Private Information Retrieval
Probe	Privacy Preserving Obfuscation Environment
R-tree	Range-tree

RSG	Random Share Generation
SLM	Symbolic Location Model
SMI	Statistical Movement Information
STR-tree	Spatio-Temporal Range-tree
TB-tree	Trajectory-Bundle-tree
TCP	Transmission Control Protocol
TD	Temporal Delay
TFA	Trajectory Fragmentation Algorithm
TLS	Transport Layer Security
TTP	Trusted Third Party
UMTS	Universal Mobile Telecommunications System
UTM	Universal Transverse Mercator

Abstract

Location-based applications such as Foursquare, Glympse, or Waze attract millions of users by implementing points of interest finders, geosocial networking, trajectory sharing, or real-time traffic monitoring. An essential requirement for these applications is the knowledge of user location information, i.e., the user's position or his movement trajectory. Location-based applications typically act as clients to a location service, which manages mobile object location information in a scalable fashion and provides various clients with this information.

However, sharing location information raises user privacy concerns, especially if location service providers are not fully trustworthy and user location information can be exposed. For instance, an attacker successfully compromising a location service may misuse the revealed location information for stalking, mugging, or to derive personal user information like habits, preferences, or interests of the user. Driven by the increasing number of reported incidents where service providers did not succeed in protecting private user information adequately, user privacy concerns are further intensified.

Therefore, we present novel approaches protecting user location privacy when sharing location information without assuming location service providers to be fully trustworthy. To protect user position information, we present our position sharing concept. Position sharing allows to reveal only positions of decreased precision to different location services, while clients can query position shares from different location services to increase precision. To protect movement trajectories, we introduce our trajectory fragmentation approach and an approach protecting the speed information of movement trajectories.

Deutsche Zusammenfassung

Heutzutage sind Informationen über die Positionen mobiler Benutzer von wesentlicher Bedeutung für ortsbezogene Anwendungen. Weitläufig bekannte Anwendungen sind beispielsweise Suchdienste für interessante Orte, die Benutzer beim Auffinden von Restaurants, Tankstellen oder Geldautomaten in ihrer Umgebung unterstützen. Andere weit verbreitete Anwendungen sind beispielsweise Verkehrsinformationssysteme, die ihre Verkehrsflussinformationen aus den erfassten Bewegungstrajektorien der Benutzer ableiten. Des Weiteren werden Lokationsinformationen heutzutage in nahezu jedem sozialen Netzwerk verwendet, sodass Benutzer ihren Aufenthaltsort mit Freunden teilen können.

Ortsbezogene Anwendungen verwenden gewöhnlicherweise Lokationsdienste, um eine skalierbare Verwaltung der Lokationsinformationen der Benutzer zu gewährleisten. Der Lokationsdienst kann hierbei entweder ein integraler Bestandteil der Infrastruktur des Anbieters der ortsbezogenen Anwendung sein oder von einem externen Dienstanbieter bereitgestellt werden. Sobald ein Benutzer einem Lokationsdienst genaue Lokationsinformationen übermittelt, besteht allerdings die Gefahr, dass diese Informationen missbraucht werden. Dies kann durch den Anbieter des Lokationsdienstes selbst erfolgen oder durch einen Angreifer, der sich Zugang zu den Daten des Lokationsdienstes beschaffen konnte. Aufgrund dieser Tatsache und der zunehmenden Anzahl an Vorfällen, bei denen Dienstanbieter persönliche Daten ihrer Benutzer nicht vor dem unberechtigten Zugriff Dritter schützen konnten, sollten Dienstanbieter nicht als uneingeschränkt vertrauenswürdig betrachtet werden. Um die Privatheit eines Benutzers zu gewährleisten, sind Mechanismen zum Schutz von Lokationsinformationen notwendig, die nicht von der Vertrauenswürdigkeit des Lokationsdienstes abhängen.

In dieser Dissertation wird hierzu eine Klassifizierung bestehender Konzepte zum Schutz von Lokationsinformationen sowie möglicher Angriffe dargestellt. Anschließend wird ein neues *Position Sharing* Konzept zum Schutz von Positionsinformationen vorgestellt, welches nicht auf die Vertrauenswürdigkeit eines Lokationsdienstes angewiesen ist. Beim *Position Sharing* werden genaue Positionsinformationen in sogenannte *Shares* beschränkter Genauigkeit aufgeteilt und anschließend auf mehrere Lokationsdienste unterschiedlicher Betreiber verteilt. Jeder Lokationsdienst verwaltet somit nur Positionsinformationen beschränkter bzw. degradierter Genauigkeit und kann höchstens Positionsinformationen mit der zugehörigen Genauigkeit offenlegen. Durch die Zusammenführung mehrerer Shares ist es allerdings möglich, dass Positionen wohldefinierter Genauigkeit wiederhergestellt werden. Zur Realisierung des *Position Sharing* Konzeptes werden unterschiedliche Ansätze vorgestellt und hinsichtlich ihrer Sicherheit analysiert.

Neben dem Schutz von Positionsinformationen werden Mechanismen zum Schutz von Bewegungstrajektorien untersucht. Hierbei wird insbesondere ein neues Verfahren vorgestellt, welches die Bewegungstrajektorie eines Benutzers in kleinere, wohldefinierte Abschnitte aufteilt und diese auf unterschiedliche Lokationsdienste verteilt. Dieses Verfahren besitzt den Vorteil, dass ein Lokationsdienst nur einen bestimmten Teil anstelle der gesamten Bewegungstrajektorie des Benutzers offenlegen kann. Abschießend werden Mechanismen zum Schutz der Geschwindigkeitsinformation von Bewegungstrajektorien vorgestellt. Hierbei wird die Geschwindigkeit der an den Lokationsdienst übermittelten Bewegungstrajektorien an die jeweils erlaubte Höchstgeschwindigkeit angepasst. Die vorgestellten Mechanismen verhindern somit, dass Benutzern aufgrund einer unbeabsichtigten Geschwindigkeitsübertretung monetäre oder rechtliche Konsequenzen drohen, falls die vom Lokationsdienst gespeicherte Bewegungstrajektorie des Benutzers offengelegt wird. Der Schutz der Geschwindigkeitsinformation stellt somit einen wichtigen Bestandteil für die Akzeptanz von neuen ortsbezogenen Anwendungen dar.

1 Introduction

Nowadays, the number of available mobile devices such as smartphones, tablets, and portable PCs is increasing tremendously. In 2013, more than one billion smartphones were sold [Pre14]. Compared to 2012, the number of shipped smartphones increased by 38.4%, showing the increasing availability of mobile devices. Current state of the art smartphones like the iPhone [App14b] or Google Android phones [Goo14a] provide integrated positioning sensors, for instance, for the global positioning system (GPS). Based on these sensors, users can obtain their geographical position from their mobile device. In combination with cheap flat rates for Internet access and a widespread availability of powerful mobile communication technologies like UMTS or LTE, the widespread availability of positioning sensors on mobile devices paved the way for location-based applications (LBAs) entering people's daily life and attracting millions of users today.

Generally, LBAs rely on geographical location information and can be classified into three different classes [GL04]:

The first class refers to *position-aware* applications, for instance, navigation systems using the position of the user only locally on the user's mobile device.

The second class refers to LBAs focusing on *sporadic queries*, where users share their geographic position with a service provider in order to find, for instance, a certain point of interest next to the geographic position of the user. Typical examples of LBAs relying on sporadic queries include points of interest finders, friend finders, and geosocial networks. These LBAs have in common that they rely on the knowledge of a single user position to provide their service answering location-based queries. For instance, points of interest finders such as *Yelp* [Yel14] help users to find the next ATM, the next bus station, or the next

Chinese restaurant based on their current position. Friend finders such as *Find My Friends* [App14a] notify users about geographically close friends and keep friends informed about current activities of the user. Finally, LBAs implementing geosocial networking allow users for "checking-in" to different locations, e.g., to restaurants, bars, or certain points of interest, to document their presence and to share their current position with friends. Nowadays, one of the most prominent geosocial networks is Foursquare [Fou14b] having a community of more than 45 million users [Fou14c].

The third class of LBAs refers to *location tracking* applications, which require the knowledge of the user's movement trajectory defining the position of the user over time. Typical examples for this class of LBAs are applications for sharing movement trajectories and real-time traffic monitoring. LBAs for sharing hiking trails, jogging paths, and bike trips record the user's movement trajectory over time and allow for trajectory sharing with friends and to analyze the traveled path, the traveled distance, or the burned calories of the user [Map14]. Furthermore, applications for real-time traffic monitoring as provided by Google [Goo14b] or TomTom [Tom14] allow users to adapt their routes based on real-time traffic information. One of the most prominent LBA combining real-time traffic monitoring with trajectory sharing is Waze [Waz14], implementing a community-based traffic and navigation application where users share real-time road and traffic information. Users of Waze notify each other about car accidents, traffic jams, approaching police checkpoints, and further relevant information. The high attractiveness of Waze is reflected by the fact that more than 50 million users are currently using Waze [FR13]. Since Google acquired Waze in June 2013 for more than 1.1 billion dollars [Efr13], even more users are joining Waze.

As shown by these examples, LBAs relying on the knowledge of user positions and movement trajectories are getting huge attention and help users to facilitate their daily life. Since most LBAs today belong to the second or the third class of applications, we further focus on these two classes.

Often, LBAs make use of so-called location services, which manage mobile

object positions and allow for position sharing between the users of one or more applications. Mobile objects inform the corresponding location service about their current position, while clients of this service can query location information by means of position, range, or nearest neighbor queries. Location services provide access control mechanisms allowing users whose location information is managed by the location service to define who can access their location information with which granularity. An efficient and effective location service is a prerequisite for most of today's LBAs, which might be either a public location service as offered today in the Internet by Geoloqi [Ins14b], or an "LBA internal" location service as used in Google's geosocial network Google Plus [Goo14c] integrating the former public location service Google Latitude. Most of the provided access control mechanisms assume that the location service is fully trusted and hence will ensure that location information is only exposed to legitimate clients.

While sharing of location information is a highly desirable feature from an application's point of view, it gives rise to severe privacy concerns. For instance, Krumm [Kru07] showed that identifying user home locations is possible by analyzing movement trajectories. Referring to Golle and Partridge [GP09], even home and work locations of users can be reconstructed. Furthermore, the Webroot survey [Web10] showed that 55% of the 1.500 participants using LBAs were worried about losing their privacy. In case an attacker gets access to the position information or the movement trajectories a user provided to a location service, the attacker may derive in addition to the user's home and work location also other sensitive information of the user by analyzing the revealed data. For instance, checking-in to a bar or a night club late at night may reveal information about the user's personal preferences and habits [CCW+12]. By checking-in to different places, users may reveal to be not at home such that an attacker may use this information to find empty homes for burglaries [FRVM+10]. Even without explicitly checking-in to a certain location, movement trajectories leading to cardiology clinics, hospitals, churches, etc., may reveal information the user is not willing to share, such as information about his health situation or his political and religious affiliations [GKdPC10]. In [BE09], Blumberg and Eckersley

present further examples of sensitive personal information that can be derived from user movement trajectories.

These privacy concerns are further intensified by many reported cases from the past, where even service providers that were supposed to be trustworthy "lost" or leaked private user information. For instance, in 2004, an employee of a big online company has stolen 92 million personal records of company members and sold the records afterwards [KV04]. In 2008, a German telephone company lost a disk storing the personal information of 17 million customers [Say08]. In 2009, attackers broke into the credit card system of a leading payment processor in the United States and stole more than 130 million credit card numbers [Pil09, Aco09]. All these examples show that the number of security breaches where an attacker could get access to the infrastructure of a service provider and steal personal user data is increasing tremendously. Furthermore, service providers that were assumed to be trusted misused personal data of their customers. For example, the service provider of a popular music identification service from England sent until 2014 private user data secretly to different advertising portals [Hol14]. In addition to the presented examples, the revealed information about the Prism and Tempora projects of the National Security Agency (NSA) finally raises the question whether users can trust any service provider storing and managing personal user data. As a consequence, assuming location services to be fully trustworthy is at least questionable.

Driven by the justified doubts that location services are fully trustworthy, location privacy mechanisms are required to protect user position information and movement trajectories when sharing location information. Especially, privacy-aware sharing of position information must be possible even in the absence of trusted services providers. Therefore, we tackle the location privacy problem in this thesis to allow for privacy-aware sharing of location information in a non-trusted system environment of location services and clients.

In the next section, we present the contributions of this thesis before giving an overview of its remaining structure.

1.1 Contributions

In this thesis, we provide the following contributions to allow for privacy-aware sharing of location information:

Classification of Location Privacy Approaches: We systematically assess the robustness of existing privacy approaches protecting user location information. We compare the approaches based on the user's protection goal and provide a classification of existing location privacy approaches taking the identified goals and different attacks into account.

Position Sharing Concept: We propose a novel concept for sharing positions using non-trusted server infrastructures. The basic idea of our concept is to split up the precise position of the user on his mobile device into a set of *position shares* of strictly limited precision. These shares are distributed to *multiple* location services offered by different providers. Therefore, a compromised location service can only reveal information of limited precision. However, the precision of position information can be incrementally increased by combining shares. In fact, the obfuscation can be undone by combining all shares, for instance, the original precision as captured by the positioning system can be restored. By allowing the user to control the set of shares which are accessible by a particular client, different precision levels can be provided to different clients ranging from the lowest level up to the original precision. Another advantage of our position sharing concept is that it provides graceful degradation of privacy in the presence of compromised location services: The precision of the revealed position information only increases with the number of compromised location services.

Trajectory Fragmentation Algorithms: We propose a novel trajectory fragmentation concept protecting movement trajectories of users when sharing their movement traces. In particular, our trajectory fragmentation algorithms prevent that an individual location service can trace the move-

ment of the user over longer distances. Instead of providing the complete movement trajectory of a user to a single location service, we split up the trajectory into a set of *trajectory fragments* and store the fragments on different location services of different providers. By distributing fragments, we avoid a single point of failure with respect to privacy in case of compromised location services, while different clients can still reconstruct the movement trajectory of the user based on user-defined access rights.

Speed Protection Algorithms: Users sharing movement trajectories are typically aware that they reveal their position and the corresponding time information. However, many users are not aware that they share and reveal also their speed information, which may reveal violations of given speed limits. It is an important prerequisite for the acceptance of novel LBAs that such violations cannot be revealed. Otherwise, users could hesitate to share their movement trajectory, because they could fear negative impacts when revealing their speed information. Therefore, we introduce novel speed protection algorithms guaranteeing that shared movement trajectories do not reveal any speeding violation. The general idea of our approach is to slow down the shared speed information to the maximum speed the user is allowed to drive.

The research leading to these contributions was conducted in the PriLoc project [Pro14] at Institute for Parallel and Distributed Systems of University of Stuttgart. PriLoc focuses on novel concepts and algorithms for the secure management of private position information in non-trusted location server infrastructures. The major contributions of this thesis were published at different international conferences [WDR12, WDR13a, WDR13c, WDR14] and in international journals [WDR13b, WSDR14].

1.2 Structure of the Thesis

This thesis is organized based on the presented classification of different LBAs and the presented contributions:

In **Chapter 2**, we give an overview of the related work in the area of location privacy by presenting existing location privacy approaches and location privacy attacks. We classify related work based on the provided protection goals and the considered attacks. Based on our classification, we show where our work goes beyond related work and which protection goals our novel concepts provide that were not considered by existing privacy approaches before.

In **Chapter 3**, we address the problem of protecting user position information. In particular, we introduce our position sharing concept by presenting two different approaches: Our first approach is based on the concept of multi-secret sharing and supports geometric and symbolic location models. Our second approach is based on the concept of binary space partitioning and focuses on the efficiency of position sharing to reduce the number of required position shares that must be distributed to different location services for multiple position updates.

In **Chapter 4**, we address the problem of protecting movement trajectories. First, we present our novel trajectory fragmentation concept preventing that a location service can trace the movement of a user over longer distances. Second, we present our speed protection algorithms guaranteeing that shared trajectories do not reveal speeding violations.

Finally, we summarize the thesis in **Chapter 5** and give an outlook onto possible future research directions.

2 Related Work

In this chapter, we introduce existing LBAs and give an overview of different location privacy approaches that have been presented in the literature. Generally, location privacy approaches differ with respect to their protection goal and their robustness against different attacks. Therefore, we assess the applicability and robustness of the proposed approaches systematically and provide a classification of the approaches.

To be able to compare different privacy approaches, we introduce a common system model for the different approaches in Section 2.2. Then, we present in Section 2.3 how location services manage location information. In Section 2.4, we identify different protection goals from a user's point of view and show in Section 2.5 how existing privacy approaches protect these goals. Afterwards, we introduce in Section 2.6 different attacks that can undermine user privacy and classify in Section 2.7 the privacy approaches with respect to their protection goal and their ability to resist the introduced attacks. Finally, we point out in Section 2.8 where our contributions go beyond existing work on location privacy and conclude this chapter in Section 2.9.

2.1 Location-based Applications

First, we present different kinds of LBAs attracting millions of users today. These applications can be distinguished whether they rely on the knowledge of user position information or movement trajectories. We begin our description with applications relying on user position information and continue afterwards with applications relying on movement trajectories.

2.1.1 Applications Relying on User Positions

Widely known examples of LBAs relying on position information are *points of interest finders*, *friend finders*, applications for *location-based advertising*, and LBAs implementing *geotagging* and *geosocial networking*:

Points of interest finders such as *Yelp* [Yel14] answer queries like *"Where is the next supermarket?"* based on the geographical position of the user, or *"Find all hotels in downtown Stuttgart"* based on a specified geographical area. The requested information is either a single point of interest, e.g., the nearest supermarket, or a set of different points of interest, e.g., the set of all hotels in downtown Stuttgart. Typically, the query results are visualized on a map on the user's mobile device.

Friend finders such as *Find My Friends* [App14a] answer queries like *"Where is Bob?"* returning the known position of a certain user or friend, or *"Which of my friends are currently in Stuttgart?"* returning all friends within a certain geographical area. Furthermore, friend finders notify users about geographically close friends.

Location-based advertising applications such as *Deals Nearby You* [Dea14] notify users about special deals or promotions of businesses within a certain geographical area or in the vicinity of the user. For instance, users can benefit from local deals offered to customers of shops in a shopping mall.

Geotagging applications enable geographical content annotations by adding geographical position information, e.g., longitude and latitude values, to any kind of content like images, videos, messages, or news. For instance, the image and video sharing application Instagram [Ins14a] allows for searching and finding tagged images and videos based on their annotated geographical information. Another application using geotagged content is Twitter [Twi14], which can automatically annotate tweets (i.e., short text messages of limited length that are published by a certain user) with geographical information to show where tweets originate from. Since 2011,

more than three billion tweets have been tagged by geographical locations [Van13] showing the high popularity of geospatial content tagging.

Geosocial networking is nowadays implemented by many LBAs of different providers. One of the most prominent geosocial networks today is Foursquare [Fou14b], which started out in 2009 and has currently more than 45 million users [Fou14c]. In Foursquare, users can check-in to different locations and points of interest to share their current position with friends, or to document their presence at a certain location. Furthermore, users can earn badges by checking-in to different venues and get crowned as "mayor" by checking-in to a single location multiple times. Until now, Foursquare counted more than five billion user check-ins [Fou14c]. Driven by the success of geosocial networking, Facebook [Fac14] launched its own geosocial network called *Facebook Places* in 2010, which was afterwards integrated into the Facebook application. Nowadays, Facebook allows users to check-in to different locations and to share their location with friends. To improve its own capabilities in developing geosocial networking applications, Facebook acquired in 2011 the geosocial network *Gowalla*, which was the biggest geosocial network after Foursquare in 2011 [Seg11].

2.1.2 Applications Relying on Movement Trajectories

The movement trajectory of a mobile object is typically acquired as GPS trace recording the sensed positions plus the corresponding timestamps of the positions. Examples for LBAs relying on the knowledge of such trajectories are applications for *community-based mapping of collected GPS traces*, *trajectory sharing*, *real-time traffic monitoring*, *pay-as-you-drive insurances*, and *mobile target tracking*:

Community-based mapping of collected GPS traces is used, for instance, in the OpenStreetMap project [Ope14] where users upload their movement trajectories as GPS traces to improve quality and completeness of the project's public map information.

Trajectory sharing applications are getting more and more attention by users. Glympse [Gly14] is currently one of the most popular trajectory sharing applications, which has more than ten million users [Gly13]. Glympse allows users to share their current movement trajectories in real-time with friends for a predefined period of time. To this end, users send their trajectories as succeeding position fixes to Glympse showing precisely where they are located while sharing their movement traces. Further examples for LBAs sharing movement trajectories are applications for sharing jogging paths, hiking trails, and bike trips [Map14]. In addition to trajectory sharing, these applications allow users to derive further information from their traces such as traveled distance, average speed, or burned calories.

Real-time traffic monitoring applications as implemented by Google [Goo14b] and TomTom [Tom14] provide users real-time traffic information like traffic jams or expected delays on their current route. Another popular LBA implementing real-time traffic monitoring based on a community-driven approach is Waze [Waz14]. Users of Waze notify each other about relevant traffic information such as construction areas, road hazards, or traffic jams by sharing road reports with other users traveling in the same geographic area. In addition to this active participation in the Waze community, Waze also analyzes the current speed of its users to derive road traffic conditions which are afterwards shared among Waze users. Thus, users can adapt their routes in case of bad traffic conditions on their selected route to save time and fuel.

Pay-as-you-drive insurances adapt rates of their customers based on their individual driving behavior and their traveled distance. Safety-conscious drivers should receive better rates than other drivers. As shown by the *VPriv* system of Popa et al. [PBB09], pay-as-you-drive insurances can be implemented based on GPS-monitored cars. Typically, insurance companies like MetroMile [Met14] and National General Insurance [Nat14] insert additional hardware in the cars of their customers recording the traveled

distance, the traveled speed, geographical information, etc. However, since current state of the art smartphones provide all sensors to implement the required functionalities, insurance companies could also rely on the provided sensor information of their customers' smartphones for their service using real-time trajectory updates. For instance, Händel et al. [HOO+14] implement this idea in their framework combining real-time traffic monitoring with a usage-based insurance relying on smartphone measurements.

Mobile target tracking applications allow the tracking of mobile objects such as taxis, buses, or trains in real-time. Thus, managers and customers of the tracked objects can benefit from the known positions. An example for an LBA tracking taxis is Mytaxi [Int14], where users can see taxis in their vicinity and track the arrival of a previously booked taxi.

Despite the popularity of the presented applications, revealing private user information such as the user's position information or movement trajectory raises severe privacy concerns because of the introduced threats that an adversary may get access to the shared information and infer habits, interests, or information such as the user's religious affiliation [GKdPC10]. Furthermore, an attacker may misuse the revealed positions or movement trajectories for stalking, mugging, or to determine empty homes for a burglary [FRVM+10]. Therefore, mechanisms protecting user privacy are mandatory when using LBAs.

2.2 System Model

Before we discuss the details of protecting private user information, we introduce a common system model that is applicable to most privacy approaches described in the literature (cf. Figure 2.1). This model consists of three components, namely *mobile objects*, *location servers*, and *clients*.

The mobile object represents the mobile device of a user that is equipped with an integrated position sensor to determine the current position of the user. The device of a user is assumed to be trusted, and it is guaranteed that no mali-

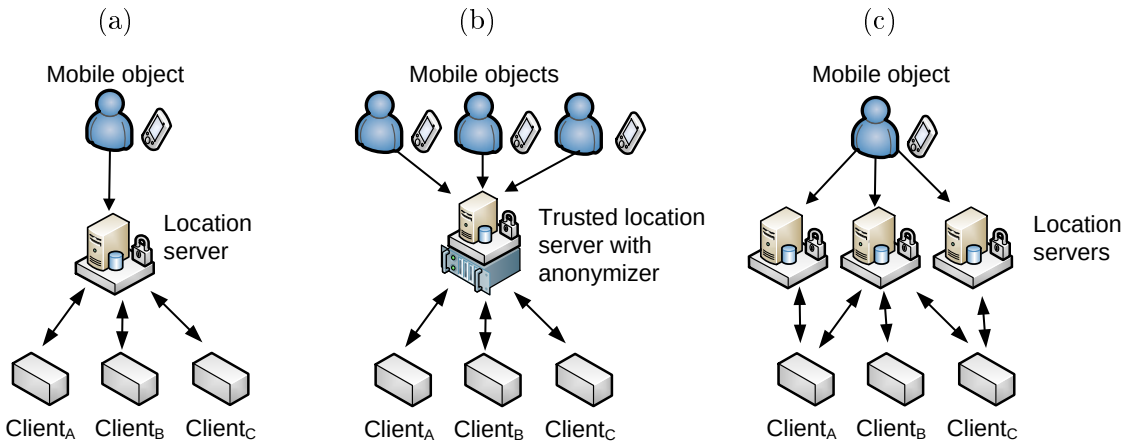


Figure 2.1: System model: (a) Without a trusted third party, (b) with trusted LS using an internal anonymizer, and (c) with multiple LSs

cious software component is running on the mobile device that has access to the position sensor. This can be assured by using a mobile trusted computing approach such as presented by Gilbert et al. [GCJW10] using, for instance, trusted hardware components. Otherwise, the location privacy approaches considered in the following are not effective since the malicious software component could transmit the precise position of the user to an adversary.

Mobile objects send their position information to a location server (LS), which stores and manages mobile object positions on behalf of the user. The communication between the mobile object and an LS must take place over a secure channel to avoid modification, sniffing, and message injection. Using an LS relieves mobile objects from sending their position individually to different clients and allows for the efficient and scalable management of mobile object positions. The position information of the mobile object has to be sent only once to the LS, while several clients can access the information stored on the LS. The communication between clients and the LS must also be protected by secure channels.

The LS can either be non-trusted (cf. Figure 2.1a) or trusted (cf. Figure 2.1b). In case of a trusted LS, the LS can perform trusted computations and act, for example, as anonymizer. For instance, a trusted LS

can use an internal anonymization algorithm to implement the concept of k-anonymity [GG03, KGMP07, NASG09] by using the positions of several mobile objects stored at the LS to make the mobile object indistinguishable from $k-1$ other objects. Furthermore, the anonymizer can calculate obfuscated positions covering the positions of several mobile objects. In addition to the system model relying on a single LS, approaches implementing the concept of position sharing distribute position information to multiple LSs (cf. Figure 2.1c) of different providers.

Clients, for instance different LBAs, query LSs for mobile object positions in order to implement a certain location-based service. The LSs grant clients access to the stored positions based on an access control mechanism. In practice, clients and LSs can also be integrated. However, we explicitly distinguish both components in our model.

LSs and clients can both be compromised, even if these entities are assumed to be trusted. For location privacy approaches relying on a trusted third party, this means that a successful attack undermines privacy. If an LS is compromised, the attacker is aware of all information different mobile objects provided to the LS. On the contrary, a compromised client does not necessarily have access to all information stored at the LSs but only a portion of it depending on its access rights.

2.3 Location Management

Since the focus of this thesis is on sharing location information, we present existing approaches for the management of location information on LSs. We start with approaches for the management of position information and continue afterwards with related work for the management of movement trajectories. Then, we present existing access control mechanisms managing the access of clients to location information stored at LSs.

2.3.1 Management of Position Information

LSs usually rely on spatial indexing mechanisms such as the *R-tree* [Gut84], the *R*-tree* [BKSS90], or the quadtree [FB74] to manage position information of mobile and static objects. Spatial indexing mechanisms allow LSs an efficient processing of spatial queries. In [IMI10], Ilarri et al. give an overview of existing indexing mechanisms supporting spatial query processing. Typically, LSs implement the functionality to answer *position queries*, *range queries*, and *nearest neighbor queries* in a scalable and efficient manner [BD05]:

Position queries return the positions of mobile or static objects like users, points of interest, or buildings. The identity of an object is usually defined by an identifier such as "taxi 345" or "user Bob". A typical example of a position query about a certain mobile object with a known identifier is *"Where is taxi 345?"*. The result of this query is the position information of the corresponding object that is stored at the LS. Depending whether a *geometric* or *symbolic* location model is considered, the position of an object is represented differently. For geometric location models, the position of an object can be represented as a tuple of geometric coordinates, such as the object's longitude and latitude values. For symbolic location models, symbolic locations are defined as abstract symbols representing, for instance, the name of a street or a building where the object is located.

Range queries return all objects such as users or points of interest that are located within a specified area. As an example, consider the query shown in Figure 2.2a specifying the area of interest as the dotted rectangle. The queried area covers the positions of the objects A and B representing the result for the range query. The objects C and D are not located in the queried area and are therefore not part of the query result.

Nearest neighbor queries return for a given value k and a position p_Q the k nearest objects to p_Q. To measure whether two objects are close to each other, different distance metrics can be applied. The most prominent

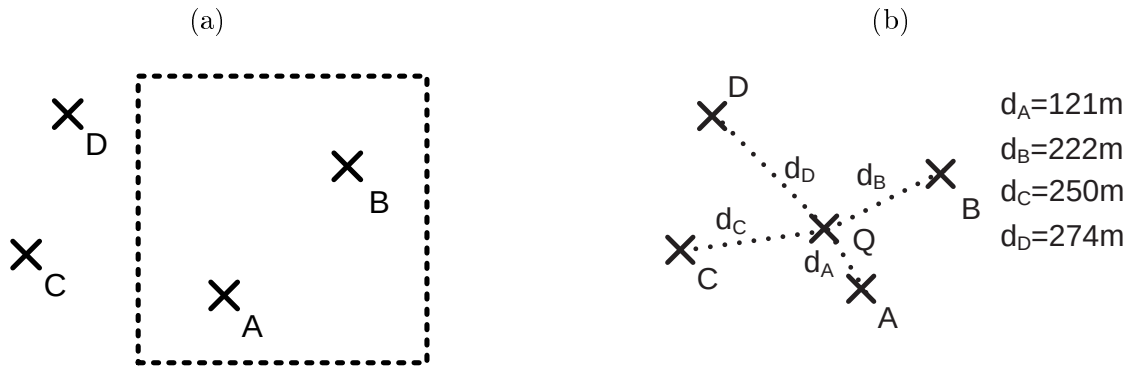


Figure 2.2: (a) Range query and (b) k-nearest neighbor query examples

distance metric is the Euclidean distance, which is typically applied in free space environments. Figure 2.2b shows an example for a nearest neighbor query where object Q located at position p_Q queries for the three ($k = 3$) nearest objects to its position. The calculated result is the set of objects A, B, and C having the smallest distance to Q, while object D is further apart from Q compared to the other objects. Another prominent distance metric is the length of the shortest path between two objects, which is often applied in restricted environments like a road network influencing the actual distance between two objects.

For a detailed description of the algorithms of an LS answering position, range and nearest neighbor queries using the Euclidean distance as distance measure, we refer to the work of Leonhardi and Rothermel [LR01]. For the algorithms considering the length of the shortest path as distance measure we refer to the work of Papadias et al. [PZMT03] and Jensen et al. [JKPT03].

As pointed out, exposing precise position information raises privacy concerns, especially if providers of LSs cannot be fully trusted. To prevent that a compromised LS reveals precise position information, spatial obfuscation approaches protect the position information of a mobile object without assuming a trusted third party. Instead of providing precise positions to an LS, mobile objects degrade the precision of their positions and provide only obfuscated positions to the

LS. As a consequence, LSs do not manage precise position information anymore. Instead, they manage obfuscated positions, which are typically represented as circles or rectangles defining the areas where the objects are located.

As shown by Chow et al. [CMA09], answering queries based on obfuscated positions can be classified into three different categories:

- *Private queries over public data*
- *Public queries over private data*
- *Private queries over private data*

Here, *public data* specifies the positions of mobile or static objects updating their precise position to the LS without protecting their position information. For instance, the positions of static objects such as restaurants, hospitals, ATMs, and cinemas are typically precisely known by an LS. Furthermore, mobile objects such as taxis, buses, or trains normally do not protect their position information. Instead, they provide their precise position to an LS for answering queries about their positions. *Private data*, on the other hand, describes personal information such as the protected position information of different users providing only positions of decreased precision to an LS.

Private queries over public data consider that public data of an LS is queried based on obfuscated position information. A typical example for such a query is *"Where is the next ATM to user Bob?"*, where the positions of the ATMs representing public data are known to the LS, while only the obfuscated position of the mobile object (e.g., user Bob) representing private data is available to the LS. Figure 2.3a shows an example for this kind of query, where the LS has to evaluate which ATM is the closest one to Bob. A detailed description of an algorithm evaluating private queries over public data is presented by Chow et al. [CMA09]. Another approach to answer private queries over public data is presented by Hu and Lee [HL06] focusing on nearest neighbor queries for obfuscated positions.

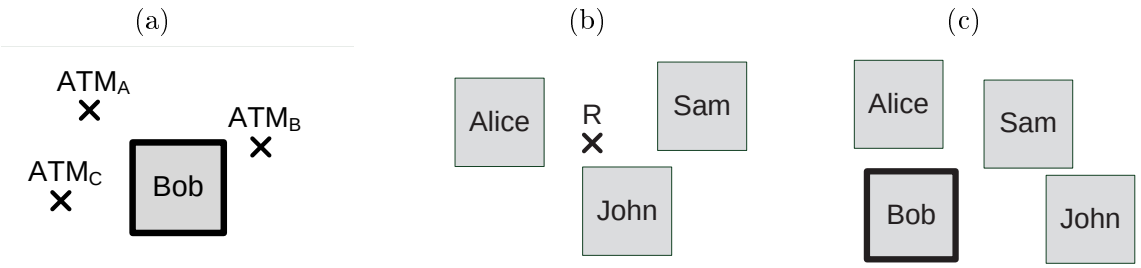


Figure 2.3: (a) Private query over public data, (b) public query over private data, and (c) private query over private data

Public queries over private data consider that the position of the query issuing object is not protected and, therefore, precisely known. The queried data on the other hand is protected such that only imprecise position information is available. An example for such a query is *"What is the nearest customer to my restaurant?"*, where the position of the restaurant is not protected, while the mobile objects, in this case, different customers, provide only positions of decreased precision to the LS to protect their privacy. As an example, consider Figure 2.3b, where the precise query position of the restaurant R is known to the LS, while the positions of the customers (Alice, John, and Sam) are obfuscated and not precisely known.

Private queries over private data consider that the position of the query issuing object and the positions of the queried data are protected by obfuscated positions. A typical example for such a query is that the obfuscated position of a user is used to query the positions of the nearest friends also providing only obfuscated positions to the LS. Figure 2.3c shows an example for this kind of query where Bob wants to know which friend (Alice, Sam, or John) is located next to him. To answer private queries over private data, Chow et al. [CMA09] present algorithms for nearest neighbor and range queries. In [CZBP06], Cheng et al. propose a solution to answer these queries based on probabilistic results taking the overlapping size of the query area and the obfuscated positions into account.

2.3.2 Management of Movement Trajectories

In the literature, different approaches for the management of movement trajectories on LSs have been proposed. According to Frenztos [Fre03], these approaches can be distinguished whether the movement of the mobile objects takes place in the whole of the two dimensional space, or whether their movement is limited to a constrained space by spatial restrictions. A typical example of a constrained space is a road network limiting the movement area of cars and pedestrians.

For LSs managing movement trajectories in a free space environment, Pfoser et al. [PJT00] propose the *Spatio-Temporal R-tree (STR-tree)* indexing mechanism, which is an extension of the R-tree, supporting an efficient query processing of spatio-temporal movement traces. The STR-tree indexes the updated positions of the mobile objects such that the movement trajectories of the objects are preserved in the index structure and queried efficiently based on the algorithms presented in [Gut84] and [PJT00]. Furthermore, the authors propose the *Trajectory-Bundle-tree (TB-tree)*, which preserves individual trajectories within a so-called trajectory bundle where the indexing structure manages the individual parts of the trajectory in the leaf nodes of the indexing tree.

For LSs managing movement trajectories in a constrained space like a road network, Güting et al. [GdAD06] present an expressive framework for modeling and querying mobile object positions. Furthermore, Wang and Zimmermann [WZ08] propose an approach to process spatial queries for mobile objects on road networks based on an R-tree managing the road network and a grid structure handling position updates of mobile objects. In [WZ11], the authors extend their approach to answer range queries and nearest neighbor queries for movement trajectories. Finally, Frenztos [Fre03] presents an indexing mechanism called *Fixed Network R-tree (FNR-tree)* answering spatio-temporal range queries based on a two dimensional R-tree indexing spatial information in combination with a one dimensional R-tree indexing time information.

2.3.3 Access Control Management

To manage the access of different clients to an LS, LSs rely on access control mechanisms as presented by Bonatti and Samarati [BS02] or Hengartner and Steenkiste [HS05]. That is, LSs grant clients access to their stored information if they provide the LS the required credentials that are specified in the access control rules of the LS. Otherwise, clients do not get access to the information stored on the LS. Based on the provided credentials, LSs can distinguish clients and manage which information is provided to a certain client. For instance, LSs can provide different clients position information of different precision levels as long as the LS knows the precise position of the mobile object. As presented by Leonhardt and Magee in [LM98] for a symbolic location model, LSs can regulate the access of different clients to the stored position information based on user-defined access policies and the current position of the user. For instance, clients could only receive access to the user's position information or movement trajectory stored at an LS as long as the user is traveling on a certain road or within a specified area. In addition to access control mechanisms defining access policies based on the user's position, *location-based access control (LBAC)* techniques as presented by Ardagna et al. [ACD+06] allow to take spatial conditions like the physical location of clients into account to define access policies.

2.4 Protection Goals

Before we discuss different approaches to protect user location privacy, we have to define the different protection goals which are considered by these approaches. The attributes to be protected are the user's identity, his spatial information (position), and temporal information (time). The information provided by a user can be defined as a tuple $\langle identity, position, time \rangle$. The protection goal of the user defines which attributes must be protected and which can be revealed. Prior to explaining the protection of the stated attributes in more detail, we present some examples of different protection goals and application scenarios.

As a first scenario, consider a user of an advanced navigation service providing real-time traffic information and points of interest information based on the current user position. Assume that the user is willing to provide anonymized position information to the navigation service provider. To this end, the user protects the identity attribute using an anonymization concept. However, as shown by Golle and Partridge [GP09], the user's identity can also be revealed from position information, for instance, based on the periodically visited home and work locations. Therefore, the position attribute also needs to be protected.

As a second scenario, assume that the user is willing to share his non-anonymous trajectory. However, he does not want to reveal that he is speeding on the motorway since revealing such information may have unforeseen consequences for the user if the service provider misuses the data and provides it to the police, his insurance company, etc. In this scenario, the position as well as the time attributes have to be protected to avoid the deduction of speeding violations.

In order to show that the protection of each attribute combination is relevant, we list further scenarios for each combination in Table 2.1. To achieve the stated protection goals, different location privacy approaches are required. As we will see later, no single location privacy approach is suited to protect all protection goals at the same time.

Next, we consider the protection of each attribute stated above in more detail.

2.4.1 Identity Information

One possible goal to ensure privacy is to hide the user's identity while anonymous positions are visible to clients. The identity of a user can be her name, a unique identifier, or any set of properties uniquely identifying the user. If a user publishes position information without personal information, an attacker can still try to derive the user's identity by analyzing the position information and additional context data such as the visited objects. In general, quasi-identifiers can be used to identify users as shown by Bettini et al. [BWJ05].

Attributes			Examples
ID	Pos.	Time	
✓	✓	✓	Protect the information where a user lives and make it impossible to infer it from several traces of the user.
✓	✓	×	Protect the user's identity and his precise position in a building while showing the security manager that someone is still in the building such that it cannot be locked. Protect that the user drove through a residents-only street.
✓	×	✓	Provide traces to the OpenStreetMap project to model new streets without revealing the user's identity or speed. Give feedback for a bar without revealing the user's identity.
✓	×	×	Use an advanced navigation system for real-time traffic monitoring without revealing the user's identity. Protect the user's identity when publishing jogging paths.
×	✓	✓	Protect a slight detour on a longer trip while keeping the general trip visible. Protect the speed information of the user's trajectory such that no speeding violation is revealed.
×	✓	×	Do not show that a user visits a bar while keeping friends informed of being in the inner city. Hide the fact that the user was in a hospital.
×	×	✓	Share the last hiking trail with friends without revealing to be currently not at home.

Table 2.1: Protection goal examples for protected and non-protected attributes. The symbol "✓" implies that the corresponding attribute must be protected, while the symbol "×" implies that the attribute can be revealed.

2.4.2 Spatial Information

Another protection goal is to provide position information of a user only with a given precision to clients. For instance, a user might want to provide precise position information to his friends, whereas only coarse positions with city-level precision should be provided to a location-based news feed service. In general, this goal is known as spatial obfuscation.

We also have to consider that user positions usually carry more information than only geometric information like longitude and latitude values. Often, the *semantic* of a position is defining the criticality of the information. For instance, a user might be willing to share his precise position as long as he does not enter a certain semantic location such as a hospital, since this could reveal further private information like the health status of the user. Therefore, a specific goal of protecting spatial information is the protection of semantic location information. In general, this is achieved by ensuring that a position is associated with several alternative locations of different semantics. For instance, as shown by Damiani et al. [DBS10], a semantic position might be protected if the user could be within a hospital *or* within one or more locations that are not hospitals.

2.4.3 Temporal Information

Temporal information defines the point in time or time period when the spatial information of the user is valid. In some scenarios, spatial information is only considered critical if it is associated with temporal information. For instance, a user might be willing to share with others where he is traveling, whereas he does not want to reveal that he is speeding. This means that real-time updates cannot be used in this case without raising privacy concerns, whereas temporally delayed updates could be used to reach the protection goal. In such scenarios, it must be considered that even if temporal information is not explicitly stated—for example, as a timestamp of the position update—, it can be implicitly derived. For instance, this is possible by knowing the time when the information was received by the LS and the update protocol that triggered the position update.

2.5 Location Privacy Approaches

After introducing possible protection goals in the previous section, we now give an overview of existing location privacy approaches to reach these goals. There are a number of works representing the state-of-the-art techniques to protect user location privacy [SDFMB08, Kru09, CM11]. Therefore, the goal of this section is to provide an overview of the fundamental concepts of these approaches. We distinguish the concepts *k-anonymity*, *mix zones*, *position dummies*, *spatial obfuscation*, *cryptography-based approaches*, and *position sharing*.

2.5.1 *k*-Anonymity

The idea of k-anonymity is to make a mobile object indistinguishable from $k-1$ other objects such that the probability to identify an individual object is $1/k$.

The concept of k-anonymity for location privacy was introduced by Gruteser and Grunwald [GG03]. The idea of their approach is that a user reports an obfuscation area to a client containing his position and the positions of $k-1$ other users. Here, the LS acts as trusted anonymizer calculating the set of k users, the used pseudonym of the k-anonymity set, and the corresponding obfuscation area based on the known positions of the users. As an example, consider that Alice is located at home and queries an LBA for the nearest cardiology clinic. Without using anonymization, this query could reveal to the client implementing the LBA that Alice has health problems. By using k-anonymity, Alice would be indistinguishable from at least $k-1$ other users and the client would not be able to link the request to Alice. Therefore, it is required that all k users of the calculated anonymization set share the same obfuscation area that was sent to the client such that the client cannot link the issued position to the home location of Alice.

Many other approaches make use of the k-anonymity concept to provide location privacy. Chow et al. [CMA09] calculate the obfuscation area of the k users in their *Casper* framework based on the user-defined values of k and an area value A_{min} indicating that the user wants to hide his location within an area size of

at least A_{min}. Kalnis et al. [KGMP07] and Ghinita et al. [GZPK10] calculate *anonymizing spatial regions* containing at least k users such that the calculated spatial region is identical for all k users. Thus, an attacker cannot exclude any of the k users from the set by analyzing their positions or the applied k-anonymity algorithm. In [WL09], Wang and Liu introduce their *XSTAR* approach realizing anonymous queries of mobile objects traveling on road networks.

Instead of considering individual positions of mobile objects, Gkoulalas-Divanis et al. [GDVM09] anonymize movement trajectories based on the underlying road network defining the restricted movement area where users can travel. Monreale et al. [MAA+10] focus on achieving trajectory anonymity for a published dataset of movement trajectories by applying spatial generalization to the k-anonymity concept. A similar idea is applied by Nergiz et al. [NASG09] preventing that the movement trajectory of a user reveals the user's identity when releasing the database containing the trajectory. Gedik and Liu [GL08] propose their *CliqueCloak* approach, which performs spatial and temporal cloaking to calculate the k-anonymity set. A user can define individual upper limits for both the obfuscation area size and time periods associated with positions in order to preserve an acceptable quality of service. The approach uses temporal cloaking by delaying updates such that the required k users are within the user-defined time interval and the maximum obfuscation area. The *ICliqueCloak* approach present by Pan et al. [PXM12] is similar to this approach. It focuses on the protection of the identity of a mobile object issuing multiple position updates based on a *clique* of k mobile objects sharing their obfuscation area and providing k-anonymity.

The basic concept of k-anonymity has been further extended by various approaches to increase user privacy. The most prominent extensions are *l-diversity*, *t-closeness*, *p-sensitivity*, *historical k-anonymity*, and *(k, δ)-anonymity*:

- The idea of location *l-diversity* presented by Bamba et al. [BLPW08] is that the location of the user must be unidentifiable from a set of l different locations such as restaurants, bars, clinics, churches, etc. To this end, their *PrivacyGrid* framework guarantees that the positions of the users

of the k-anonymity set are not just different, but are also located distant enough from each other. Otherwise, an attacker would know the target user location with low imprecision if all user positions belong to the same semantic location.

- The concept of *t-closeness* proposed by Li et al. [LLV07] extends the l-diversity concept. The authors consider the distance between an attribute's distribution within the selected set of k users and the same distribution over the total set of users. This distance should not be above a certain threshold value t.

- Xiao et al. [XXM08] improve k-anonymity guarantees by the concept of *p-sensitivity*. The idea of p-sensitivity is to guarantee that within a set of k users each group of sensitive attributes has at least p distinct values for each sensitive attribute within the same group. Otherwise, the attributes could be disclosed by the corresponding attributes of the group. As an example, consider that a group of friends meeting at a certain location wants to go to a bar. All friends query for the location of the bar and rely on a k-anonymity concept. However, since all friends are located close to each other and issue the same query, the probability that they are grouped to the same k-anonymity set sharing the same query for the bar is high. Thus, an attacker knows that the target user also queried for the bar.

- Mascetti et al. [MBW+09] and Bettini et al. [BMW+09] describe their *historical k-anonymity* approaches to provide k-anonymity guarantees for mobile objects by taking also the temporal component of the location information into account. The approaches consider historical information of multiple users to calculate spatial regions including the locations of at least k users. Furthermore, the approaches are designed to protect multiple queries *online* in real-time.

- Abul et al. [ABN08] present their (k, δ)-anonymity approach concentrating on the protection of a complete published user trajectory *offline*. To

this end, they apply an enhancement of k-anonymity for spatio-temporal cloaking. Their idea is to publish a cylindrical volume of radius δ that covers at least k trajectories of different users that are located next to each other instead of publishing precise movement traces.

2.5.2 Mix Zones

In [BS03], Beresford and Stajano introduce the idea of frequently changing pseudonyms to protect the identity of a user and to prevent that an attacker can derive user habits or interests. Since an attacker could link different pseudonyms together even if pseudonyms change frequently, Beresford and Stajano improve their idea by proposing the mix zone concept. A mix zone defines an area where all user positions must be hidden such that the user position is not known within the zone. This is achieved by suppressing position updates within a zone. If a user enters a mix zone, the user's identity is mixed with all other users in the zone by changing pseudonyms. Thus, an attacker cannot correlate different pseudonyms of the users even by tracing the entry and exit points of a mix zone.

The *MobiMix* approach proposed by Palanisamy and Liu [PL11] applies the mix zone concept to road networks. *MobiMix* takes various context information like geometrical and temporal constraints into account which an attacker could otherwise use to reconstruct detailed trajectories of users.

In [LZP+12], Liu et al. propose to use multiple mix zones in road networks to increase the provided privacy of the mix zones against attackers that are trying to identify individual users based on additional background knowledge.

2.5.3 Position Dummies

The goal of the position dummies approach presented by Kido et al. [KYS05] is to protect the user's real position by sending multiple false positions (so-called "dummies") in addition to the real position of the user to the LS. An essential advantage of this approach is that the user herself can generate dummies without any need for a trusted third party. However, it is challenging to create dummies

which cannot be distinguished from the real user position, in particular, if an adversary has additional context information such as a map and can trace the movement of the user for longer distances.

You et al. [YPL07] improve the generation of random dummies by generating them in a human-like movement fashion. That is, dummies are generated following the movement behavior of the user and his long-term movement pattern.

An advanced method to generate dummies is presented in the *SybilQuery* approach proposed by Shankar et al. [SGI09]. The approach assumes that the user has a database of historical traffic which allows him to create additional dummy positions that cannot be distinguished from the real user position.

2.5.4 Spatial Obfuscation

Spatial obfuscation approaches try to preserve user privacy by deliberately reducing the precision of position information sent from the mobile object to the LS and in turn to the client. Thus, an attacker can only retrieve coarse-grained position information of the user. A classic spatial obfuscation approach is presented by Ardagna et al. [ACDS11], where a user sends a circular area instead of his precise position to the LS. Instead of using geometric obfuscation shapes like circles, Duckham and Kulik [DK05] present a graph-based obfuscation model that can also be applied to obfuscate the position of the user on road networks. The advantage of spatial obfuscation approaches is that they typically provide location privacy without a trusted third party, since the user himself can define the obfuscation area. However, this advantage comes at the price that clients cannot retrieve precise user positions.

To answer k-nearest neighbor queries while protecting user privacy, Yiu et al. [YJML11] present their *SpaceTwist* framework. Instead of sending precise user positions to the LS, users send a so-called "anchor" representing a fake location to the LS. The anchor is used to iteratively request data points based on various distances to the anchor. The user then calculates the query results based on his precise position and the received data points. Thus, precise k-

nearest neighbor query results are provided to the user, while location privacy is achieved through higher query and communication costs.

One problem with many spatial obfuscation approaches is that the effective size of the intended obfuscation area can be reduced if an adversary applies background knowledge, in particular, map knowledge. For example, an attacker can increase precision by excluding non-reachable parts of the user's obfuscation area based on a map matching attack applying map knowledge. In order to resist such map matching attacks, Ardagna et al. [ACG09] proposed a *landscape-aware* obfuscation approach. The approach is based on a probability distribution function, which defines the probability that a user is located in certain areas of a map. For instance, the probability to be located next to a lake is typically higher than to be located on the lake. The obfuscation area is then selected considering the probability of the user to be located in a certain obfuscation area. Another advanced obfuscation approach is presented in the *Privacy Preserving Obfuscation Environment (Probe)* framework introduced by Damiani et al. [DSB11]. The framework applies a similar principle as Ardagna et al. [ACG09] to protect semantic locations such that a user position cannot be mapped with a high probability to a critical location like a hospital. Thus, the *Probe* framework expands the map-aware obfuscation area in a way that the probability of the user to be in a certain semantic location is below a given threshold value.

Beyond the obfuscation of spatial information, Gidofalvi et al. [GHP07] consider spatio-temporal obfuscation to protect movement trajectories. Besides decreasing the precision of positions, they also decrease the precision of the temporal information by using a grid-based framework to reduce the spatio-temporal resolution of the trajectory. Ghinita et al. [GDSB09] present a similar idea for their *spatio-temporal cloaking* approach. To improve the provided privacy of spatial cloaking, the authors consider the movement speed of an object as additional knowledge such that their approach can resist advanced attacks based on the known maximum speed of an object.

Yigitoglu et al. [YDAS12] extend the idea of the *Probe* framework by protecting sensitive information in movement trajectories. That is, the precise position

of the user is updated as long as the position is not related to a sensitive location. On the other hand, sensitive locations are protected by cloaked regions which consider that users typically move on streets and that different semantic locations are characterized by the time periods users stay at the locations.

Hoh et al. [HGXA10] introduce their uncertainty-aware path cloaking approach hiding location samples in a dataset stored on a trusted LS in order to provide *time-to-confusion* guarantees for all users sending their positions to the LS. The time-to-confusion guarantees consider the time users can be tracked by an attacker. Terrovitis and Mamoulis [TM08] apply the idea of suppressing positions violating user privacy constraints to prevent that movement trajectories can reveal the identity of a user. The idea of suppressing positions is also applied by Xue et al. [XZZ+13]. In their work, the authors remove positions from a movement trajectory that should not be published to protect against destination prediction. By deleting position information, the probability an attacker can assign to the destination of the trajectory can be decreased below a predefined threshold value.

In [HG05], Hoh and Gruteser protected movement trajectories by crossing paths of at least two mobile objects. Thus, whenever two paths meet, an attacker tracking an object is confused which path is the correct one to follow. A similar idea is used by Meyerowitz and Choudhury [MRC09] in their *CacheCloak* system providing anonymous location data to clients based on path confusion. Compared to [HG05], *CacheCloak* focuses on real-time anonymization by caching query results on the LS and by providing clients predicted paths that intersect each other.

The approach presented by Hoh et al. [HGH+08] allows for privacy-preserving traffic monitoring based on *virtual trip lines* representing geographic markers where users should update their position to the LS. The approach allows to avoid virtual trip lines at privacy critical locations and to cloak several positions based on the identifiers of the trip lines.

In [FRVM+10], Freni et al. address the problem of protecting the information that a user is absent at a certain point in time from a certain location such as his

home. To tackle this problem, the authors introduce their *minimal uncertainty region* concept representing spatio-temporal regions where no part of the region can be excluded by an attacker in order to derive further information about where the user is *not* located.

Finally, Chow and Mokbel [CM07] present a group-based approach to obfuscate the complete movement trace of a user in real-time.

2.5.5 Cryptography-based Approaches

Cryptography-based location privacy approaches use encryption to protect user positions. Mascetti et al. [MFB+11] propose an approach to notify users when friends are within their proximity without revealing the current position of the user to the LS. To this end, the authors assume that each user shares a secret with each of his friends and use symmetric encryption techniques.

Approaches as proposed by Ghinita et al. [GKK+08] make use of the *private information retrieval* (PIR) technique to provide user location privacy. By using PIR, an LS can answer queries without learning any information of the query. The used PIR technique relies on the quadratic residuosity assumption, which states that it is computationally hard to find the quadratic residues in modulo arithmetic of a large composite number for the product of two large primes.

In order to deal with the problem of non-trusted LSs, Marias et al. [MDKG05] propose an approach for the distributed management of position information based on the concept of secret sharing. The basic idea of this approach is to divide position information into shares, which are then distributed among a set of (non-trusted) LSs. In order to recover positions, clients need the shares from multiple servers. The advantage of this approach is that a compromised LS cannot reveal any position information since it does not have all the necessary shares. However, LSs cannot perform any computations on the shares, for instance, to perform range queries. In general, cryptographic approaches raise the question whether location-based queries like nearest neighbor queries or range queries can be performed efficiently over the encrypted data.

2.5.6 Position Sharing

To perform location-based queries such as nearest neighbor or range queries while protecting user location privacy, Dürr et al. [DSR11] proposed the concept of position sharing for the secure management of private position information in non-trusted systems. Position sharing splits up obfuscated position information into so-called position shares, where a share defines a position of strictly limited precision. These shares are distributed among a set of non-trusted LSs such that each LS only has a position of limited precision, which can also be used to perform calculations on these shares. Through share combination algorithms, multiple shares can be fused into positions of higher precision such that clients can be provided with position information of different precision levels depending on the number of accessible shares. Since an LS only has information of limited precision, the approach has a graceful degradation property, where the precision of position revealed by an attacker gradually increases with the number of compromised LSs. In [SDR12], the authors extended their work by taking map knowledge into account to prevent attackers from increasing precision.

The publications of Dürr et al. [DSR11] and Skvortsov et al. [SDR12] introducing position sharing based on random geometric transformations were developed in the same project and at the same time as the position sharing approaches presented in this thesis.

2.6 Location Privacy Attacks

In this section, we classify different attackers according to the knowledge they exploit and the methods they use to derive private user information.

2.6.1 Attacker Knowledge

We classify attacker knowledge according to two dimensions, namely *historic information* and *context information*. These two dimensions represent the horizontal and the vertical axes of our classification of location privacy attacks,

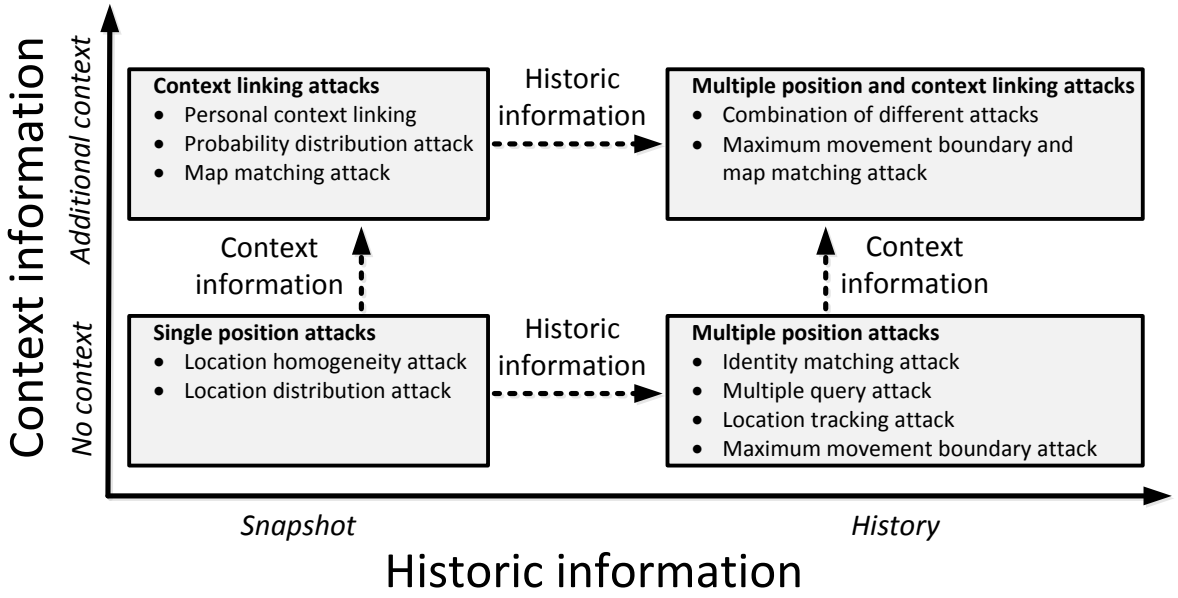


Figure 2.4: Classification of attacker knowledge and location privacy attacks

which is shown in Figure 2.4.

In the *historic information* dimension, we consider whether an attacker has access only to a single user position or whether the attacker can access historical information. In the first case, the attacker knows only a single snapshot of a user position. This is a common assumption for many location privacy approaches. In the second case, the attacker knows a set of multiple positions collected over time or even the whole movement trajectory of a user. Such information could be revealed, for instance, by a compromised LS or a compromised client. In particular, if an LS gets compromised, the attacker might also get historical position information of several users.

In the *context information* dimension, we distinguish whether the attacker has additional context knowledge beyond spatio-temporal information. For instance, an advanced attacker might have additional context information provided by a phone book, statistical data, a map, etc. The attacker can use this information in addition to the known user positions. For instance, an attacker could decrease the effective area size of an obfuscated position of a user based on map knowl-

edge to determine where users can actually be located, or use a phone book to determine the home address of a user.

2.6.2 Classification of Location Privacy Attacks

We classify different attacks based on the presented knowledge of an attacker. Each combination of context information and historic information leads to a different class of attacks. Our classification of attacks is shown in Figure 2.4 and structured according to the introduced attacker knowledge. We distinguish between the classes of *single position attacks*, *context linking attacks*, *multiple position attacks*, and attacks combining *multiple positions* and *context linking attacks*. The concrete methods of the attacker are shown in Figure 2.4 below the different classes of attacks. In addition to these four classes of attacks, we consider also the attack of *compromising a trusted third party*, which can be applied in addition to all presented attacks.

Single Position Attacks

The general idea of a single position attack is that the attacker analyzes a single updated user position to infer more information about the position or the identity that the user intended to hide.

- A **location homogeneity attack** [MGKV06] can be used against simple k-anonymity approaches. The attacker analyzes the positions of all members of the k-anonymity set. If their positions are almost identical (cf. Figure 2.5a), the position information of each member of the set is revealed. If the members are distributed over a larger area, their position information is protected (cf. Figure 2.5b). An advanced location homogeneity attack can utilize map knowledge to reduce the effective area size where users can be located. For instance, the area can be restricted to a single building (cf. Figure 2.5c). Here, the attacker analyzes the semantic location information of the members of the k-anonymity set and

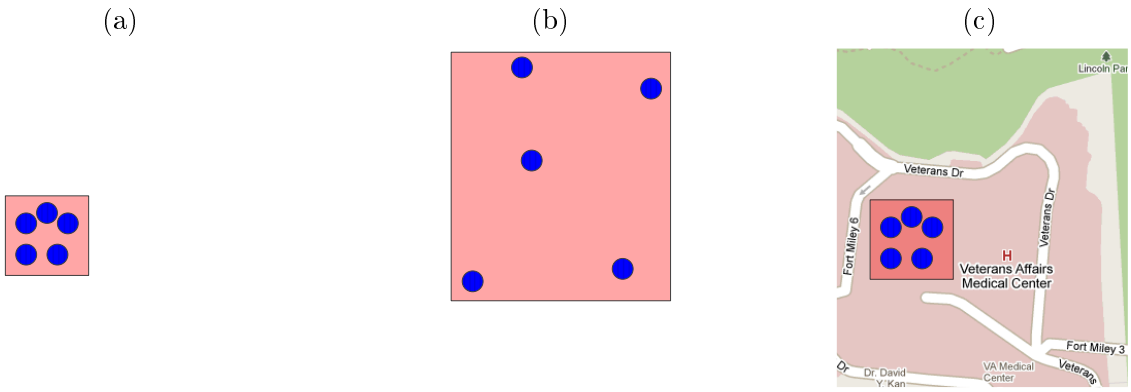


Figure 2.5: Location homogeneity attack: (a) Almost identical positions, (b) distributed positions, and (c) positions belonging to a single building

determines the diversity of the different positions. Only diverse positions provide location privacy while homogeneous positions do not.

- A **location distribution attack** [CM07] is based on the observation that users are often not distributed homogeneously in space. This can be utilized to attack some k-anonymity approaches. Consider a calculated set of k users whose members cover a densely and sparsely populated area as depicted in Figure 2.6. Here, the dark square area covering the users A, B, C, and D defines the calculated area of the set of k users. In such a set, the protected user is most likely the single user A located in the sparsely populated area further away from the other users. This is based on the fact that only in that case the obfuscation area has to be extended into the dense area to cover the requested number of k users. If B were the protected user, a different set would be the result covering the users B, C, D, and E in the bright square area in Figure 2.6.

Context Linking Attacks

A context linking attack generally exploits context information in addition to spatio-temporal information. An attacker can use personal context knowledge about a user as well as external background knowledge such as an office plan, an

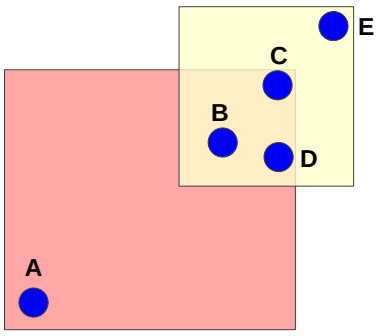


Figure 2.6: Location distribution attack

address book, or a map to decrease user privacy. For the context linking attack, we distinguish between three different kinds of attacks: the *personal context linking attack*, the *probability distribution attack*, and the *map matching attack*:

- A **personal context linking attack** [GG03] is based on personal context knowledge about individual users such as user preferences or interests. For instance, assume it is known that a user visits a pub on a regular basis at a certain point in time and that he uses a simple obfuscation mechanism to protect his position information. Then, an attacker can increase his known precision of an obfuscated position by decreasing the obfuscation area to locations of pubs in the obfuscation area.

  A special method of the personal context linking attack is *spatial observation* [MYYR10], where the attacker has user knowledge gathered through observation. For instance, if a user protects his identity using pseudonyms and the attacker can see the user, then the attacker can retrace all prior locations of the user for the same pseudonym by a single correlation.

- A **probability distribution attack** [STLBH11] is based on gathered traffic statistics and environmental context information. Here, the attacker tries to derive a probability distribution function of the user position over the obfuscation area. If the probability is not uniformly distributed, an attacker can identify areas where the user is located with high probability.

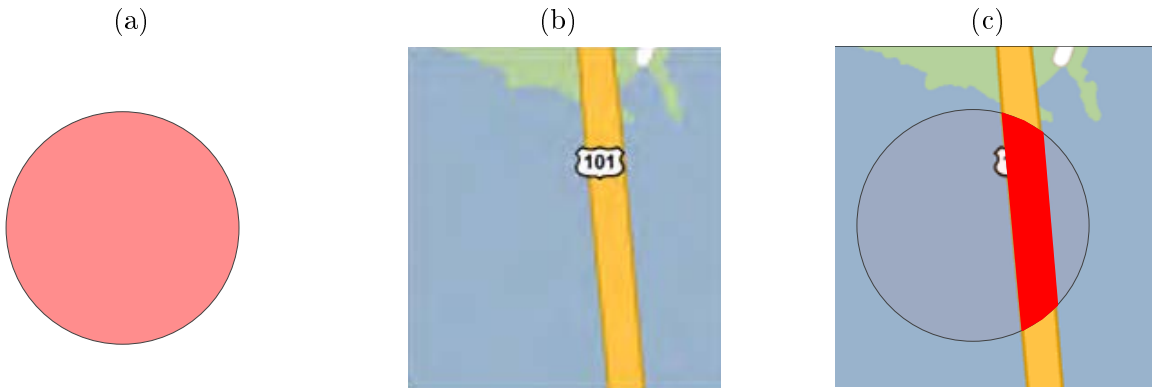


Figure 2.7: (a) Obfuscated position, (b) map knowledge, and (c) map matching attack refining the position to a road bridge using map knowledge

- A **map matching attack** [ACG09] can restrict an obfuscation area to certain locations where users can be located by removing irrelevant areas. For instance, a map could be used to remove areas like lakes from the obfuscation area, which effectively shrinks the obfuscation area size below the intended size (cf. Figure 2.7). The attacker can also use semantic information provided by maps like points of interest or the type of buildings (e.g., bars, hospitals, residential buildings, etc.) to further restrict the size of the effective obfuscation area.

Multiple Position Attacks

The general idea of a multiple position attack is that an attacker tracks and correlates several user positions to decrease user privacy. For this class of attacks, we distinguish between the *identity matching attack*, the *multiple query attack*, the *location tracking attack*, and the *maximum movement boundary attack*.

- The **identity matching attack** [GKdPC10] can be applied to correlate several pseudonyms of a user. Here, the attacker links several pseudonyms based on equal or correlating attributes to the same identity such that the provided privacy of the changed pseudonyms is broken.

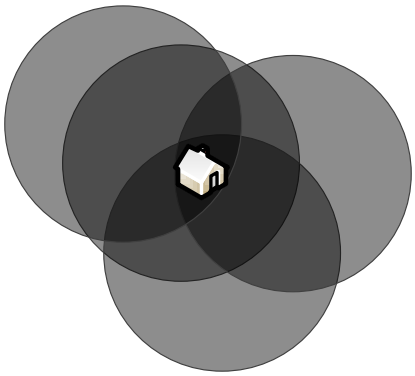


Figure 2.8: Region intersection attack refining obfuscated positions

- For a **multiple query attack** [TA10], the attacker analyses several positions and uses the *shrink region* or *region intersection* method:
  1. The *shrink region* method can reveal the identity and the position of a user. To this end, the attacker monitors multiple positions and the corresponding members of the k-anonymity set. If the members of the set change, an attacker can infer which user sent the initial position. As an example, consider three users A, B, and C located at different positions. User A issues two different queries based on his position to the same client. The simple k-anonymity approach used by user A once generates the k-anonymity set (A, B) for the first query and the anonymity set (A, C) for the second query. If the client can now correlate both queries, the client can infer that user A originally issued both queries.
  2. The *region intersection* method can be used against location obfuscation approaches to increase the precision of obfuscated positions. To this end, the attacker uses several obfuscated position updates from a user to calculate their intersection. From the intersections, the attacker can infer where privacy-sensitive regions of the user are, or where the user is located. As example, consider a random obfuscation mechanism generating different obfuscation areas whenever the

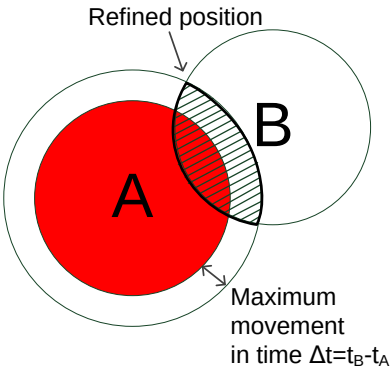


Figure 2.9: Maximum movement boundary attack

user reaches his home (cf. Figure 2.8). Then, the intersection of the obfuscation areas can be used to decrease user privacy.

- A **location tracking attack** [GG03] analyses several user positions that are known to the attacker. For example, this attack can be used against randomly changing pseudonyms while mix zones are not used. Here, the attacker can correlate succeeding pseudonyms by linking spatial and temporal information of succeeding positions, even if pseudonyms are changed. For instance, the attacker can try to reconstruct the movement of a user based on the provided positions of several pseudonyms.

- In a **maximum movement boundary attack** [GDSB09] the attacker calculates the maximum movement boundary area that is reachable for a user between two position updates. As shown in Figure 2.9, the position of area A of the first update performed at time t_A helps the attacker to increase the precision of the second update (area B) sent at time t_B. In this example, only the striped part of area B is reachable within the maximum movement boundary of the user, while the remaining part of area B cannot be reached from area A in time $\Delta t = t_B - t_A$ for a known maximum velocity v_{max} of the user. Therefore, the remaining part of area B can be excluded from the obfuscated position by the attacker.

Multiple Positions and Context Linking Attacks

Instead of using only one single attack presented so far, an attacker can also combine several of the proposed attacks or use them in sequence to undermine user privacy. For instance, an attacker can combine the knowledge of map restrictions gathered by the map matching attack and the restrictions of the maximum movement boundary attack to determine where the user is moving.

Compromised Trusted Third Party

The attack of compromising a trusted third party describes the fact that an attacker could get access to the data stored at a service provider that is assumed to be trusted. For instance, an attacker could compromise a trusted LS and get access to the stored user data. This attack is not considered in approaches that rely on a trusted third party, as these approaches are insecure if providers are not trustworthy. However, as shown by the rapidly increasing number of reported incidents and successful attacks on different providers managing private user information [Cle14, Fou14a], this kind of attack is realistic and not negligible. Therefore, it is at least questionable whether the assumption of a trusted third party is realistic.

2.7 Classification of Location Privacy Approaches

In this section, we present our classification of location privacy approaches based on the analysis of the protection goals they fulfill under different attacks (cf. Table 2.2). Although different classifications of location privacy approaches exist [SDFMB08, BAB+09, KS10], they do not compare the robustness of different approaches under different attacker models.

As shown in Section 2.4, we distinguish between different protection goals of the user, which are defined by the attributes *identity*, *position*, and *time*. These protection goals are presented in Table 2.2 on the vertical axis on the left. For each protection goal, we depicted the basic approaches presented in Section 2.5

Goals			Approaches	Attacks						
ID	Pos.	Time		Location homog. attack	Map match. attack	Pers. context linking	Prob. distr. attack	Mult. query attack	Max. mov. bound.	Map m. & max. mov. b.
✓	✓	✓	Historical anonymity [BMW+09]TTP	✓			✓	✓	✓	
			ICliqueCloak [PXM12]TTP	✓			✓	✓	✓	
			Traffic-aw. mix zones [LZP+12]TTP	✓	✓		✓	✓	✓	✓
			SybilQuery [SGI09]	✓	✓		✓	✓	✓	✓
✓	✓	✗	(k, δ)-anonymity [ABN08]TTP	✓				✓		
			k-an.+A_{min} (*Casper*) [CMA09]TTP	✓			✓	✓		
			l-div. (*PrivacyGrid*) [BLPW08]TTP	✓	✓		✓			
			p-sensitive k-anon. [XXM08]TTP	✓	✓		✓			
✓	✗	✓	Time-t. (*CliqueCloak*) [GL08]TTP	✓					✓	
			Mix zones (*MobiMix*) [PL11]TTP	✓	✓			✓	✓	✓
✓	✗	✗	k-anonymity [GG03]TTP				✓			
			Chang. Pseudonyms [BS03]		✓			✓	✓	✓
			Private inf. retrieval [GKK+08]	✓	✓	✓	✓	✓	✓	✓
✗	✓	✓	Spat.-temp. cloak. [GHP07]	✓				✓		
			Max. velocity prot. [GDSB09]	✓				✓	✓	
			Path cloaking [HGXA10]TTP	✓	✓		✓	✓		
			Semantic map obf. [YDAS12]	✓	✓		✓	✓	✓	✓
✗	✓	✗	Spatial obfuscation [ACDS11]	✓						
			Map-aw. pos. sharing [SDR12]	✓	✓					
			Position dummies [KYS05]	✓	✓					
			Map-aw. obf. (*Probe*) [DSB11]TTP	✓	✓		✓			
✗	✗	✓	Temp.-generalization [FRVM+10]TTP	✓	✓		✓	✓	✓	✓

Table 2.2: Classification of location privacy approaches. Each protection goal is defined by whether the attribute identity, position, and time should be protected ("✓") or not ("✗"). The stated approaches provide the corresponding protection goal assuming certain attacker knowledge. If the technique can resist an attacker with a certain attack, this is denoted by the symbol "✓" in the main part of the table, whereas an empty cell denotes that the attack can be successful against the stated approach.

providing the corresponding goal. For each approach, we marked whether it needs a trusted third party (denoted as TTP) or not.

The horizontal axis on top of Table 2.2 represents possible location privacy attacks as presented in Section 2.6. For a clearer representation, we omitted the location distribution attack, the identity-matching attack, and the location tracking attack, which are only applicable to a small set of approaches in the area of k-anonymity and pseudonyms. In the main part of Table 2.2, we use the symbol "✓" to show for each combination of location privacy approach and location privacy attack that the corresponding protection goal can be achieved. An empty cell shows that the attack can successfully undermine the privacy approach such that the protection goal cannot be achieved.

Next, we summarize the observations that can be derived from Table 2.2 and our analysis of the different approaches:

1. Most approaches providing user privacy require a trusted third party. Especially, approaches protecting the identity of a user require a trusted anonymizer, while only few approaches protect the user's identity without trusted third party.

2. The most popular technique to protect user position information is spatial obfuscation decreasing the precision of the shared position information of the user.

3. Most approaches protecting the attribute position or the attributes position and time focus on single positions, while only few approaches consider multiple positions. However, only approaches considering multiple positions can resist multiple query attacks or maximum movement boundary attacks.

4. Many approaches assume a free space environment of unrestricted movement. However, the movement of users is typically restricted by spatial boundaries such as walls or by the road network where users travel. An

attacker can retrieve this map-related knowledge from existing map services to restrict the movement area of the user by using a map matching attack.

5. Many approaches protecting the attributes position and time consider only the maximum movement boundary attack or already published trajectories. However, a challenging problem remains to have privacy mechanisms protecting movement trajectories in real-time as used for online tracking.

6. The combination of the map matching attack and the maximum movement boundary attack is only rarely considered by approaches protecting the attribute position or the attributes position and time.

7. Only the approach presented in [GKK+08] resists a personal context linking attack. Most approaches do not consider that an attacker may have knowledge about user habits, regular user behavior, user interests, etc.

2.8 Our Approaches: Considered Attacks and Protection Goals

In this section, we give an overview of how the approaches we introduce in this thesis improve existing location privacy approaches based on the stated observations of our classification. We show which protection goals our approaches achieve and which attacks we consider. The classification of our approaches is shown in Table 2.3.

Generally, our goal is to protect user location privacy without relying on a trusted third party (cf. Observation 1). Existing spatial obfuscation approaches protecting the position attribute of the user have the drawback that clients can only retrieve positions of decreased precision instead of precise user positions (cf. Observation 2). To overcome this drawback, we propose the concept of position sharing. Position sharing allows to provide different precision levels to different

Goals			Approaches	Attacks						
ID	Pos.	Time		Location homog. attack	Map match. attack	Pers. context linking	Prob. distr. attack	Mult. query attack	Max. mov. bound.	Map m. & max. mov. b.
✗	✓	✓	Trajectory fragment. (Section 4.1)	✓	✓		✓	✓	✓	✓
			Speed protection (Section 4.2)	✓	✓		✓	✓	✓	✓
✗	✓	✗	*PShare-GLM/SLM* (Section 3.1)	✓	✓		✓	✓	✓	
			PShare-BSP (Section 3.2)	✓			✓	✓	✓	

Table 2.3: Classification of the concepts presented in this thesis

clients without limiting the maximum precision of a client due to lack of trustworthiness of the LS. Based on this idea, we present two novel position sharing approaches protecting user position information. Our first position sharing approach presented in Section 3.1 is based on the concept of multi-secret sharing and is denoted as *PShare-GLM/SLM* in Table 2.3. Our second position sharing approach presented in Section 3.2 is based on the concept of binary space partitioning and denoted as *PShare-BSP* in Table 2.3.

Since most location privacy approaches protecting the attribute position or the attributes position and time focus on single user positions (cf. Observation 3), they typically do not resist a multiple query attack or a maximum movement boundary attack. Our position sharing approaches improve the provided privacy of these approaches by resisting the stated attacks. Furthermore, our position sharing approaches resist probability distribution attacks by using a deterministic obfuscation technique.

To resist map matching attacks (cf. Observation 4), *PShare-GLM/SLM* takes map knowledge into account and guarantees that an attacker cannot decrease the effective area size of an obfuscated position below a user-defined threshold value.

Regarding that mechanism protecting multiple positions in real-time are required (cf. Observation 5), we present our trajectory fragmentation algorithms in Section 4.1. Our trajectory fragmentation algorithms focus on the protection of the position and time attribute, i.e., the protection of the user's movement

trajectory. Our algorithms consider that the movement of a user is restricted by a road network and that an attacker has map knowledge. To resist the maximum movement boundary attacks and the map matching attack (cf. Observation 6), our trajectory fragmentation algorithms consider different movement alternatives of the user based on the given road network.

An attacker can combine the map matching attack and the maximum movement boundary attack to reveal speeding violations of the user based on known speed limits and the user's maximum reachable position which does not violate speed limits. To avoid revealing speeding violations, we present our speed protection algorithms in Section 4.2.

Personal context linking attacks may currently decrease the provided privacy of our proposed approaches (cf. Observation 7). For instance, the knowledge of frequently visited locations can provide an attacker additional information to increase the precision of the user's position information. Furthermore, protecting movement trajectories is challenging if an attacker knows the destination of the user based on personal context information. Finally, an attacker could use the combination of the map matching attack and the maximum movement boundary attack against our position sharing approaches to increase precision. For the different attacks that are currently not considered in our approaches (cf. Table 2.3), we point out possible extensions to resist these attacks when presenting the approaches in detail.

2.9 Conclusion

The literature describes many different concepts and approaches to protect user location privacy, which differ in terms of the protected information and their robustness against different attacks. Therefore, we presented existing location privacy approaches and provided a classification for them based on their protection goals and their ability to resist attacks. Based on our classification, we described how the approaches introduced in this thesis improve existing privacy approaches.

3 Protecting Position Information

In this chapter, we address the problem of storing position information on non-trusted location servers. To allow for privacy-aware sharing of position information without assuming that location servers are fully trustworthy, we present two novel position sharing approaches for protecting user position information. Our first position sharing approach presented in Section 3.1 is based on the concept of multi-secret sharing and supports geometric and symbolic location models. Our second position sharing approach presented in Section 3.2 is based on the concept of binary space partitioning and focuses on the efficiency of position sharing. Finally, we show in Section 3.3 how our approaches differ from existing location privacy approaches and conclude this chapter in Section 3.4.

3.1 Position Sharing based on Multi-Secret Sharing

The general idea of the position sharing approach presented in this section is to split up the precise position of the mobile object into a set of *position shares* and to distribute the generated shares to *different* LSs of different providers. Each share represents an imprecise position. Therefore, LSs manage only position information of decreased precision. Nevertheless, clients can combine several shares from different LSs to increase precision of the mobile object's position. Thus, position sharing allows to provide positions of different precision levels to different clients without storing precise position information at any single LS.

Our position sharing approach has the following properties:

1. Provides location privacy without relying on a trusted third party.

2. Supports geometric and symbolic location models.
3. Keeps the mobile object in control of its shared position information that is provided to different LSs.
4. Resists map matching attacks by taking map knowledge into account.
5. Does not limit the maximum precision that can be provided to a client by the trustworthiness of an LS.
6. Allows to provide different precision levels to different clients.
7. Provides a graceful degradation of privacy in case of compromised LSs.

First, we introduce our extended system model and describe two variants of our scheme: one for geometric locations (*PShare-GLM*), and one for symbolic locations (*PShare-SLM*). Afterwards, we analyze the robustness of our approaches to resist different attacks and show the applicability of our approaches.

3.1.1 Extended System Model

Our extended system model (cf. Figure 3.1) is based on the system model presented in Section 2.2 consisting of mobile objects, location servers, and clients. Below, we present the functionality and the interaction between the different components in more details.

The mobile object (MO) uses an integrated positioning system, such as GPS, to determine the MO's precise current position π. We assume that the MO is not compromised and no malicious software component can access π. The only component allowed to directly access the integrated positioning system is our share generation component (see below). All other components on the MO query the MO's position as shown below. In order to ensure that no other component than the share generator can directly access the integrated positioning system, trusted computing approaches can be used, for instance, based on trusted module hardware as presented by Gilbert et al. [GCJW10]. The MO executes a local

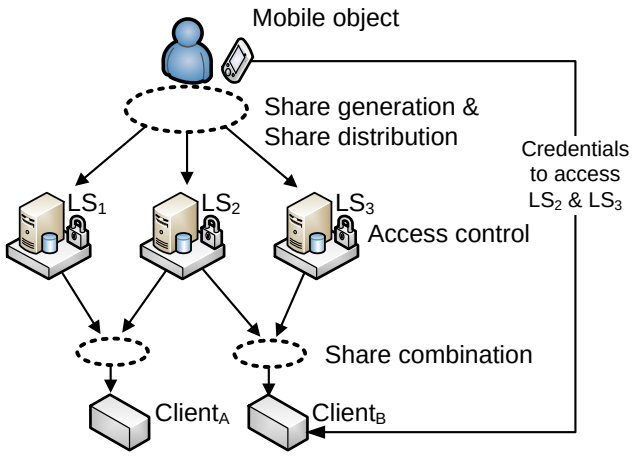


Figure 3.1: System components

software component for share generation that splits up π into a *master share* m_π, denoted as m-share, and set $S_\pi = \{r_{\pi,1}, \ldots, r_{\pi,n}\}$ of n *refinement shares*, denoted as r-shares, by calculating

$$generate(\pi, l_{max}, n) = (m_\pi, S_\pi).$$

Parameter l_{max} defines the number of different *precision levels*, i.e., positions of different well-defined precisions that can be offered to clients. We use the notation $p(\pi, l)$ to denote a position on precision level l derived from the precise position π. $p(\pi, 0)$ represents the least precise position on level zero, and $p(\pi, l_{max})$ the position of highest precision on level l_{max}. The concrete definition of precision is dependent on the type of location model (geometric or symbolic), and is introduced later for each model.

The m-share m_π consists of position $p(\pi, 0)$ and additional information required to reconstruct positions of precision levels greater than zero. m_π is public, i.e., everyone can know the least precise position $p(\pi, 0)$. r-shares contain further secret information to refine $p(\pi, 0)$ to more precise positions of higher levels. After share generation, the r-shares are distributed to different location servers such that each server receives one r-share.

Location servers (LSs) store and manage r-shares. Each LS implements an access control mechanism as presented in Section 2.3.3 to manage the access of different clients to shares. The access rights are defined by the MO and provided to the clients of the LS, for instance, as credentials to access a certain number of r-shares, where the number of accessible shares defines the intended precision offered to a client.

Clients receive permissions to access a well-defined set $S'_\pi \subseteq S_\pi$ of r-shares from the MO and perform the *share combination* on the public m-share and these r-shares:

$$combine(m_\pi, S'_\pi) = p(\pi, l)$$

Combining the m-share m_π and the set S'_π of r-shares yields position $p(\pi, l)$ of precision level l. Local clients running on the mobile device directly access the set $S'_\pi \subseteq S_\pi$ of r-shares provided by the share generation component to reconstruct $p(\pi, l)$. The concepts for share combination are identical to the case where shares are queried from remote LSs. Thus, we focus the following descriptions on the more general case where clients access different r-shares from a set of remote LSs.

3.1.2 Problem Statement

The goal is to design *secure* share generation and combination algorithms such that an attacker—either (malicious) client or LS—knowing a set $S'_\pi \subseteq S_\pi$ of shares refining $p(\pi, l)$ of precision level l cannot derive a position of higher precision than $p(\pi, l)$. This is the essential requirement for our approach. Otherwise, the MO could not control the precision offered to LSs and clients by limiting access to a subset of shares.

Note that the basic assumption of position sharing is that the unauthorized access to shares cannot be perfectly prevented. Thus, a malicious client or LS could get a position of the precision defined by the accessible shares. However, using secure shares, we can limit this risk by limiting the precision that can be derived from a certain number of (compromised) shares.

3.1.3 Geometric Position Sharing

We start the description of our position sharing approaches with *PShare-GLM*, the approach for geometric location models. First, we introduce our geometric location model, which is used to define positions on different precision levels, and give an overview on how to apply multi-secret sharing to position sharing. Then, we describe the algorithmic details of share generation and combination. Afterwards, we consider multiple position updates of the MO and present an extension of *PShare-GLM* taking map knowledge into account.

Geometric Location Model

In *PShare-GLM*, the MO's precise position π and the obfuscated positions $p(\pi, l)$ are defined as geometric locations based on a Cartesian coordinate system. We use a common map projection, e.g., Universal Transverse Mercator (UTM) projection, to map ellipsoidal coordinates (longitude, latitude) to Cartesian coordinates. The UTM projection divides the Earth into sixty zones, each representing a six degree band of longitude. For a MO traveling from one zone to another, the zone is changed as soon as the area of the new check-in location is completely covered by the new zone. Position π is a point coordinate. A position $p(\pi, l)$ of precision level l is defined as square area $p(\pi, l) = ((x_l, y_l), b^{l_{max}-l})$, where (x_l, y_l) defines the coordinates of the south-west corner of the square and $b^{l_{max}-l}$ defines the side length. Hence, the *precision* corresponds to the side length of the square. We assume that a maximum precision of one meter, which is well below the precision of common positioning systems such as GPS, is sufficient for every practical application. Therefore, we set the precision of the position $p(\pi, l_{max})$ of the highest precision level l_{max} to one meter. Parameter b defines the granularity of the precision levels, where an increase of the precision level by one increases the precision by a factor of b and partitions the area of $p(\pi, l)$ into b^2 squares. For $b = 2$, the result is a quadtree as depicted in Figure 3.2, where each position of level l is refined into four positions of level $l + 1$.

To encode a position on level l, we specify the x and y coordinates of π as

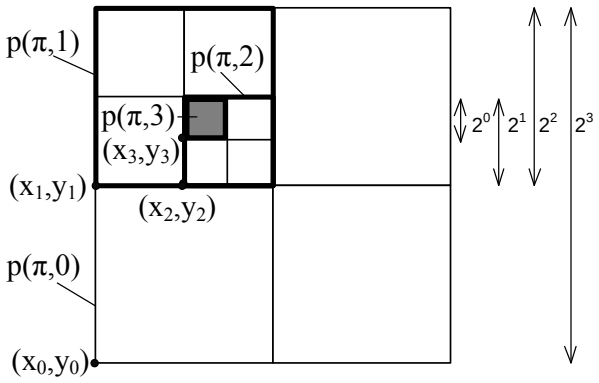


Figure 3.2: Geometric area of $p(\pi, l)$ for $b = 2$ and $l_{max} = 3$

N digits with base b, where N is selected such that each position in the MO's movement area can be expressed by the corresponding Cartesian coordinates:

$$\pi.x = \sum_{k=0}^{N-1} \alpha_k b^k = (\alpha_{N-1} \cdots \alpha_1 \alpha_0)_b$$
$$\pi.y = \sum_{k=0}^{N-1} \beta_k b^k = (\beta_{N-1} \cdots \beta_1 \beta_0)_b$$

Position π is degraded to $p(\pi, l)$ by setting the $l_{max} - l$ least significant digits to 0, meaning the actual digit values are unknown. For instance, for $b = 2$ and $l_{max} = 3$, $p(\pi, 0)$ can be written as follows, where underlined digits are unknown:

$$p(\pi, 0).x = 000101010111011100111\underline{000}$$
$$p(\pi, 0).y = 111001001100010001011\underline{000}$$

Based on the selected value of l_{max}, the MO's movement area is partitioned deterministically into a grid of cells with a side length of $2^{l_{max}}$ meter (cf. Figure 3.3) representing different positions of the MO on precision level zero.

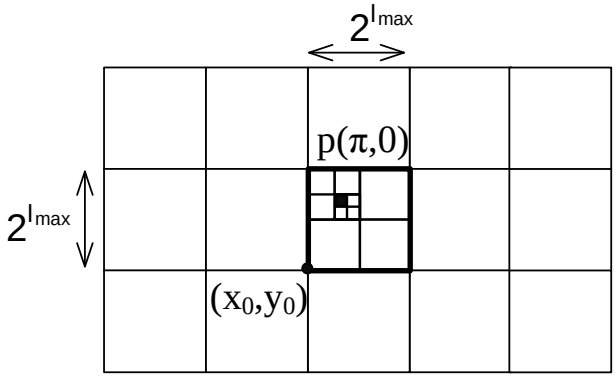


Figure 3.3: Deterministic grid partitioning based on l_{max}

Overview of Multi-Secret Sharing Algorithm

PShare-GLM utilizes multi-secret sharing algorithms for share generation and combination. Therefore, we first give a brief introduction to multi-secret sharing, before we describe the basic relation between multi-secret and position sharing.

Multi-secret sharing is an extension of *secret sharing*. A widely known secret sharing scheme is Shamir's (τ, n)-threshold scheme [Sha79]. The general idea of this scheme is to split up a secret, say K, into a set of n *shares* that can be distributed to different participants. The so-called *dealer*, which initiates secret sharing, defines a threshold value τ, which defines the required number of shares to reconstruct K, and distributes the shares to the participants, where each participant owns one share. Any τ out of the n participants putting their shares together can reconstruct secret K. If less than τ shares are available, K cannot be reconstructed.

The general idea of *multi-secret sharing* is that a dealer splits up m secrets $K_1, \ldots, K_m$ into a set of n shares so that each secret K_i can be reconstructed by any set of at least $\tau_i \leq n$ shares. The number τ_i of required shares to reconstruct each secret K_i is again defined by the dealer. For less than τ_i shares, no information about K_i is exposed.

We apply the idea of multi-secret sharing as follows to our position sharing approach *PShare-GLM*. We use the required information to refine position

$p(\pi, 0)$ to position $p(\pi, 1), \ldots, p(\pi, l_{max})$ as secrets $K_1, \ldots, K_{l_{max}}$ of the multi-secret sharing scheme. The MO corresponds to the "dealer", which creates n r-shares using function $generate(\pi, l_{max}, n)$ as presented below in detail. We assign each precision level l the threshold value $\tau_l = l$, i.e., l shares are required to reveal $p(\pi, l)$. However, our approach provides the flexibility to use any number τ_l for level l, where greater values increase robustness at the price of a greater overhead, as discussed later.

The r-shares are then distributed among n LSs by the MO. The role of the "participants" is split up between LSs and clients. Whereas participants of the original multi-secret sharing scheme manage *and* combine shares, LSs only manage at most one share per position, and clients combine multiple shares queried from different LSs. This role split allows for providing different precision levels to different clients, and limits the precision known by a single LS.

The m-share contains public data necessary for share combination (see below) similar to traditional multi-secret sharing. However, in contrast to multi-secret sharing, the m-share additionally contains coarse-grained position information serving as origin for the refinements.

Share Generation

The following description of share generation is based on the multi-secret sharing approach of Chan and Chang [CC05]. However, also other multi-secret sharing approaches could be applied.

Figure 3.4 visualizes the whole process of share generation and combination. Algorithm 1 defines the process of share generation, which is entirely performed by the MO. After sensing π, the MO first calculates $p(\pi, 0)$ (position of minimal precision) by simply setting the least l_{max} significant digits to zero using function $floorDigits(\pi, l_{max}, b)$, as described previously. $p(\pi, 0)$ is part of the public m-share m_π.

Then, the MO calculates the r-shares for each precision level greater than zero. As mentioned, the basic idea is to create one secret of the multi-secret scheme for each position $p(\pi, l)$ with $l > 0$. To this end, we first have to translate each

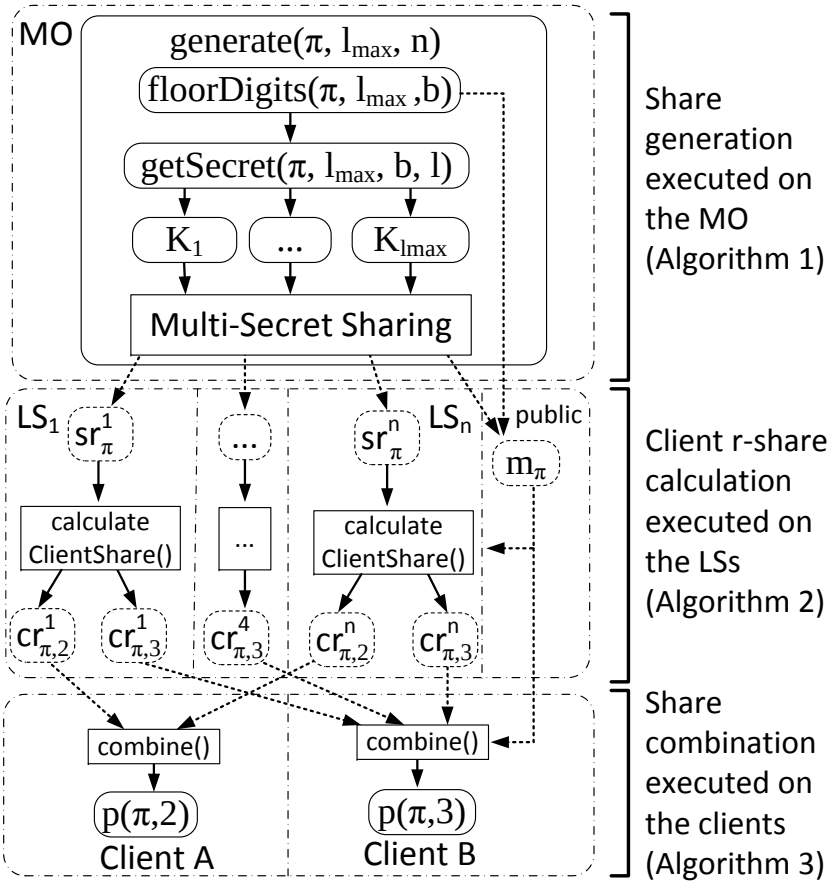


Figure 3.4: *PShare-GLM* process overview

position $p(\pi, l)$ into secret K_l using function $getSecret(\pi, l_{max}, b, l)$. Since K_l is a single integer number, x and y values are to be encoded as a single number. This is done by interleaving the digits of x and y values. In more detail, function $getSecret(\pi, l_{max}, b, l)$ calculates each secret K_l as difference of the interleaved digit values of $p(\pi, l_{max})$ and $p(\pi, 0)$ with the $2 \cdot (l_{max} - l)$ least significant digits set to zero.

After the translation of $p(\pi, l)$ to secret K_l, Chan and Chang's multi-secret sharing scheme is applied to the calculated secrets $K_1, \ldots, K_{l_{max}}$. To protect a secret, a secret polynomial $f_l(X)$ of degree $\tau_l - 1$ is calculated by the MO for each secret K_l:

Algorithm 1 *PShare-GLM*: Share generation executed on the MO

Function: $generate(\pi, l_{max}, n)$

1: $p(\pi, 0) \leftarrow floorDigits(\pi, l_{max}, b)$ ▷ Calculate position of minimal precision
2: **for** $l = 1$ to l_{max} **do**
3: $\quad K_l \leftarrow getSecret(\pi, l_{max}, b, l)$ ▷ Define secrets
4: **end for**
5: $P, f(X) \leftarrow calculatePolynomial(K)$ ▷ Apply multi-secret sharing
6: $X \leftarrow n$ distinct integer
7: **for** $i = 1$ to n **do**
8: $\quad y'_i \leftarrow f(x_i)$ ▷ $x_i \in X$
9: $\quad sr^i_\pi \leftarrow (x_i, y'_i)$ ▷ Define server r-share
10: **end for**
11: $m_\pi \leftarrow P, p(\pi, 0)$ ▷ Define m-share
12: **return** $m_\pi, \{sr^1_\pi, \ldots, sr^n_\pi\}$

$$f_l(X) = a'_0 + a'_1 X + \cdots + a'_{\tau_l - 1} X^{\tau_l - 1}$$

The constant term a'_0 corresponds to the protected secret K_l. The polynomial and therefore the secret can be reconstructed by polynomial interpolation using modular arithmetic if τ_l distinct points of the polynomial are known. Thus, each r-share contains information to determine a single distinct point (x, y) of the secret polynomial as shown below.

According to the introduced multi-secret sharing scheme, the secret polynomials $f_1(X), \ldots, f_{l_{max}}(X)$ of all secrets are packed together using the *Chinese Remainder Theorem* into one single secret polynomial $f(X)$:

$$f(X) = a_0 + a_1 X + \cdots + a_{\tau_{l_{max}} - 1} X^{\tau_{l_{max}} - 1}$$

The constant term a_0 of $f(X)$ is the solution of the following simultaneous congruences for l_{max} randomly selected distinct primes p_l with $p_1 < \ldots < p_{l_{max}}$

and $K_l < p_1$ for $l = 1, \ldots, l_{max}$:

$$\begin{aligned} a_0 &\equiv K_1 \bmod p_1, \\ &\vdots \\ a_0 &\equiv K_{l_{max}} \bmod p_{l_{max}}. \end{aligned}$$

The terms $a_1, \ldots, a_{\tau_1 - 1} \in \mathbb{Z}_M$ are randomly selected for $M = \prod_{l=1}^{l_{max}} p_l$. The terms $a_{\tau_{l-1}}, \ldots, a_{\tau_l - 1}$ are computed for $l = 2, \ldots, l_{max}$ as follows:

For each $j = \tau_{l-1}, \ldots, \tau_l - 1$ the term a_j is calculated:

$$a_j \equiv b_j \cdot r_j \cdot \prod_{k=1}^{l-1} p_k \bmod M.$$

The term b_j is randomly selected from $\{0, \ldots, p_l - 1\}$ and r_j is a random integer. Thus, it holds that $a_j \equiv 0 \bmod p_k$ for all coefficients a_j with $j \geq \tau_l$ and $k = 1, \ldots, l - 1$.

By using this calculation, the secret polynomial $f(X)$ is defined such that $f_l(X) \equiv f(X) \bmod p_l$. That is, we can calculate $f_l(X)$ by calculating $f(X)$ modulo p_l for prime p_l defining the field $\mathbb{Z}_{p_l}[X]$ of $f_l(X)$. $f_l(X)$ is a uniquely defined polynomial of degree less than τ_l over $\mathbb{Z}_{p_l}[X]$. The set of primes $P = \{p_1, \ldots, p_{l_{max}}\}$, which is required together with the r-shares to reconstruct the secrets, is part of the m-share.

This leaves the question of the detailed content of an r-share. As pointed out before, each r-share should contain information about a single distinct point (x, y) of a certain polynomial $f_l(X)$. Using multi-secret sharing, we actually have to distinguish between the information of the r-shares generated by the MO, which is sent to and stored by the LSs (called *server r-share* sr_{π}^{LS}), and the information sent from the LSs to the clients (called *client r-share* $cr_{\pi,l}^{LS}$). Each server r-share contains a distinct point (x, y') of the secret polynomial $f(X)$. Each client r-share contains a distinct point (x, y) of $f_l(X)$, which is required for share combination. Upon a request of the client for $cr_{\pi,l}^{LS}$, $cr_{\pi,l}^{LS}$

Algorithm 2 *PShare-GLM*: Client r-share calculation executed on the LSs

Function: *calculateClientShare*(l)

1: $cr_{\pi,l}^{LS}.x \leftarrow sr_{\pi}^{LS}.x$
2: $cr_{\pi,l}^{LS}.y \leftarrow sr_{\pi}^{LS}.y' \bmod m_{\pi}.p_l$ ▷ Retrieve information from server r-share
3: **return** $cr_{\pi,l}^{LS}$

is calculated by the LS from sr_{π}^{LS} as $y = y' \bmod p_l$ (cf. Algorithm 2). Note that different client r-shares of different levels can be calculated from one server r-share using the specific prime of level l.

Share Combination

In order to calculate $p(\pi, l)$, a client retrieves the publicly available m-share m_π and τ_l client r-shares $cr_{\pi,l}^{LS_1}, \ldots, cr_{\pi,l}^{LS_{\tau_l}}$ from τ_l different LSs. As described previously, m_π contains the set of prime numbers (P) and the least precise position $p(\pi, 0)$ of the MO. Each client r-share $cr_{\pi,l}^{LS_i}$ defines a distinct point (x_i, y_i) in $f_l(X)$.

Share combination (cf. Algorithm 3) uses the *Lagrange interpolation* over the field $\mathbb{Z}_{p_l}[X]$:

$$f_l(X) = \sum_{i=0}^{\tau_l - 1} y_i \prod_{\substack{0 \le j \le \tau_l - 1 \\ j \neq i}} \frac{X - x_j}{x_i - x_j} \in \mathbb{Z}_{p_l}[X].$$

This reconstructs polynomial $f_l(X)$ by interpolating the τ_l distinct points of the client r-shares, which uniquely define $f_l(X)$. Secret K_l is the constant term of $f_l(X)$ and is calculated as $f_l(0) = K_l \bmod p_l$. The position $p(\pi, l)$ is calculated by adding the reconstructed secret K_l to the interleaved representation of $p(\pi, 0)$ and splitting up the sum into the x and y values of $p(\pi, l)$.

Each polynomial $f_l(X) \in \mathbb{Z}_{p_l}[X]$ has a degree of at most $\tau_l - 1$ and fulfills the condition $f_l(x_i) = y_i$. Because at least τ_l distinct points are required to

Algorithm 3 *PShare-GLM*: Share combination executed on the clients

Function: $combine(m_\pi, \{cr_{\pi,l}^{LS_1}, \ldots, cr_{\pi,l}^{LS_{\tau_l}}\}, l)$

1: $f_l(X) \leftarrow lagrange(\{cr_{\pi,l}^{LS_1}, \ldots, cr_{\pi,l}^{LS_{\tau_l}}\}, m_\pi.p_l)$ ▷ Polynomial interpolation
2: $K_l \leftarrow f_l(0)$ ▷ Calculate secret
3: $p(\pi, l) \leftarrow split(interleave(m_\pi.p(\pi, 0)) + K_l)$ ▷ Refine position
4: **return** $p(\pi, l)$

interpolate a polynomial of degree $\tau_l - 1$, it is guaranteed that $p(\pi, l)$ cannot be reconstructed with less than τ_l client r-shares.

Correctness

The correctness of our position sharing approach is based on the property that the secrets provided to the multi-secret sharing scheme of Chan and Chang [CC05] can be reconstructed by share combination. To this end, the multi-secret sharing scheme [CC05] relies on the presented modulo calculation when solving the simultaneous congruences and Shamir's well-known secret sharing scheme [Sha79]. What remains to be shown is that for each level l the secret K_l calculated in our share generation refines position $p(\pi, 0)$ of the m-share to position $p(\pi, l)$. Thus, the share combination implements the inverse function of the share generation splitting up the provided integer value of K_l into the digits of the MO's coordinates refining position $p(\pi, 0)$ to $p(\pi, l)$.

Multiple Position Updates

Up to now, we only considered share generation for single position updates. However, in the worst case, a compromised LS or client could reveal a complete history of positions for a certain precision level (either precision level one for an LS with access to a single server r-share, or a certain level l in case of a client that was granted access to the client r-shares of level l). As shown by the location privacy attacks presented in Section 2.6.2, the knowledge of multiple obfuscated

positions might enable an attacker to further refine the precision beyond the intended precision. To avoid this, we extend our basic algorithm as follows.

We assume that the MO has a known maximum velocity v_{max}, which is also known by an attacker. Moreover, we consider that in the worst case an attacker knows the complete history $U = \{(p(\pi_{first}, l), t_{first}), \ldots, (p(\pi_{last}, l), t_{last})\}$ of position updates for a certain precision level l. Here, t_{first} denotes the time of the first update, and t_{last} the time of the last update up to the present time. Level l depends on the available shares accessible by the attacker. Then, we have to guarantee that for all $t \in [t_{first}, t_{last}]$ the attacker cannot derive a position of higher precision than the precision of $p(\pi_t, l)$.

Before describing our counter measure, we have to consider the maximum movement boundary attack introduced in Section 2.6.2 in more detail. As shown by Ghinita et al. [GDSB09], a sequence of position updates resists a maximum movement boundary attack if each pair of succeeding updates resists this attack. Therefore, it is sufficient to only consider two directly succeeding positions. For this attack, the attacker considers two succeeding positions $(p(\pi_i, l), t_i)$ and $(p(\pi_{i+1}, l), t_{i+1})$ with the obfuscated areas $A = p(\pi_i, l)$ and $B = p(\pi_{i+1}, l)$ at times $t_A = t_i$ and $t_B = t_{i+1}$. With this information, the attacker tries to remove areas from the obfuscated area B that are not reachable from the obfuscated area A in time $\Delta t = |t_B - t_A|$ for a MO traveling with a maximum speed v_{max}. Figure 3.5 shows an example where an attacker can remove the dark part of area B, which is not reachable from area A in time Δt. Furthermore, the attacker tries to remove areas from area A without reachable point in area B considering Δt and v_{max}. By removing unreachable parts of area A or area B, the obfuscation area is decreased and the precision of the attacker is increased.

To prevent such an attack, we first only consider position updates for level zero. Later we will show that protecting the position updates of level zero against maximum movement boundary attacks also protects the position updates for every level $0 \leq l \leq l_{max}$. Let $d_{pp}(p(\pi_i, 0), p(\pi_{i+1}, 0))$ be the point pairwise distance of two succeeding MO positions $p(\pi_i, 0)$ and $p(\pi_{i+1}, 0)$. The point pairwise distance of two rectangular areas is defined as the maximum Euclidean distance

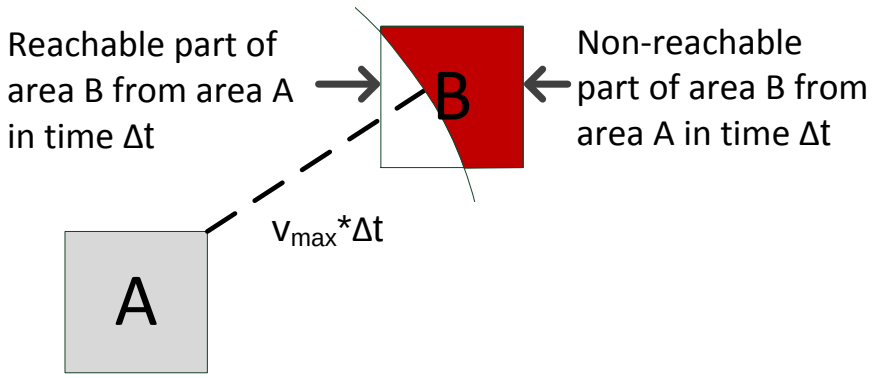


Figure 3.5: Maximum movement boundary attack

between any point in the first area to any point in the second area. Furthermore, let $\Delta t = |t_{i+1} - t_i|$ be the time between the two updates. Then, the MO only sends an update for $p(\pi_{i+1}, 0)$ at time t_{i+1} if the following condition is fulfilled:

$$\Delta t \geq \frac{d_{pp}(p(\pi_i, 0), p(\pi_{i+1}, 0))}{v_{max}}. \tag{3.1}$$

If this condition is fulfilled, every point in $p(\pi_{i+1}, 0)$ is reachable from every point in $p(\pi_i, 0)$ in time Δt. Otherwise, $p(\pi_{i+1}, 0)$ has to be delayed until this condition is fulfilled.

For precision levels greater than zero, the point pairwise distance is always smaller than or equal to the point pairwise distance of level zero. Thus, if Equation 3.1 is fulfilled for level zero, it is also fulfilled for levels greater than zero.

The *maximum delay time* δt between two updates depends on the values of b and l_{max} and is defined as:

$$\delta t = \frac{2 * (\sqrt{2} * b^{l_{max}})}{v_{max}}. \tag{3.2}$$

Intuitively, δt describes the maximum time that is required to travel a distance of two times the diagonal of the area of $p(\pi, 0)$ with v_{max}. Therefore, the MO

can trade-off the maximum delay time against the minimal revealed precision ($p(\pi, 0)$) by adjusting the size of $p(\pi, 0)$.

We analyzed the check-in behavior of users from different LBAs in our real world evaluation presented in Section 3.1.6 and show that typically only few check-ins have to be delayed. For applications requiring higher update rates than used for sporadic check-ins, the *maximum precision decrease* δs introduced by the maximum delay time δt is defined by the distance of two times the diagonal of the area of $p(\pi, 0)$:

$$\delta s = 2 * \sqrt{2} * b^{l_{max}}. \tag{3.3}$$

Geometric Position Sharing Using Map Knowledge

Until now, we assumed a free space environment for *PShare-GLM* where MOs can be located at each position within $p(\pi, l)$. However, in many scenarios, the position of the MO is restricted to streets, places, buildings, etc. Therefore, an attacker could use map knowledge for the map matching attack presented in Section 2.6.2 to decrease the effective area size of the obfuscation area $p(\pi, l)$. The *effective area* of a MO's position $p(\pi, l)$ is defined as the area within $p(\pi, l)$ where the MO can actually be located. An example for a map matching attack is shown in Figure 3.6, where the effective area size of the obfuscation area $p(\pi, 0)$ is decreased by considering only positions on roads and buildings.

To overcome this problem, we present *PShare-GLM*$_{map}$, which takes map knowledge into account. The idea of our extension to *PShare-GLM* is to adapt MO positions in a first step to map knowledge such that each position $p(\pi, l'_i)$ covers at least an area with an effective area size of a MO-defined value A_i. In a second step, we use the calculated positions for the share generation and apply the multi-secret sharing scheme as presented for *PShare-GLM*.

For the first step, the MO specifies its privacy requirements by defining different area values $A_i \in A_0, \ldots, A_m$ each representing the required minimum effective area size of precision level l'_i. The goal is now to find the smallest position $p(\pi, l'_i)$ for each value A_i, such that the effective area size of $p(\pi, l'_i)$ is equal

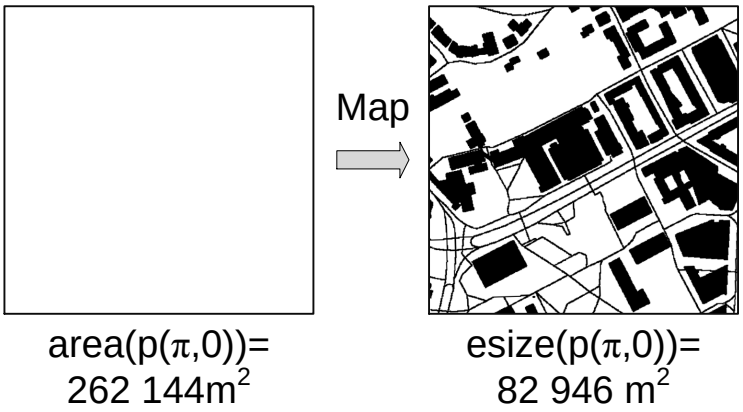


Figure 3.6: Map knowledge reducing the effective area size ($b = 2, l_{max} = 9$)

to or greater than A_i. To calculate the effective area size of position $p(\pi, l)$, we use function $esize(p(\pi, l))$. Now, our goal is to find the maximum level l_i' such that $esize(p(\pi, l_i')) \geq A_i$ for each $A_i \in A_0, \ldots, A_m$. For the sake of simplicity, we restrict our explanations to parameter $b = 2$ defining the granularity of the area of the MO's position. Nevertheless, our approach can also be applied to other parameter values of b.

We analyze map information in a preprocessing step to adapt positions to map knowledge. Thus, we calculate for each position whether or not it is a possible MO position, for instance, located on a street or in a building. To allow for an efficient search of $p(\pi, l_i')$ for each A_i, we store all possible MO positions in a quadtree aggregating the effective area sizes for each level in a bottom-up approach (cf. Figure 3.7). The quadtree has a depth of l_{max} and its root is defined by position $p(\pi, 0)$.

To calculate the maximum level l_i' fulfilling $esize(p(\pi, l_i')) \geq A_i$ for a given position π and a required effective area size A_i without raising privacy concerns, we traverse the corresponding quadtree of π top down by evaluating the effective area size of position $p(\pi, l)$ for each level l and the four child nodes of $p(\pi, l)$ on level $l + 1$. If the described nodes of the quadtree cover an effective area of size equal to or greater than A_i, the quadtree is further traversed on the next level

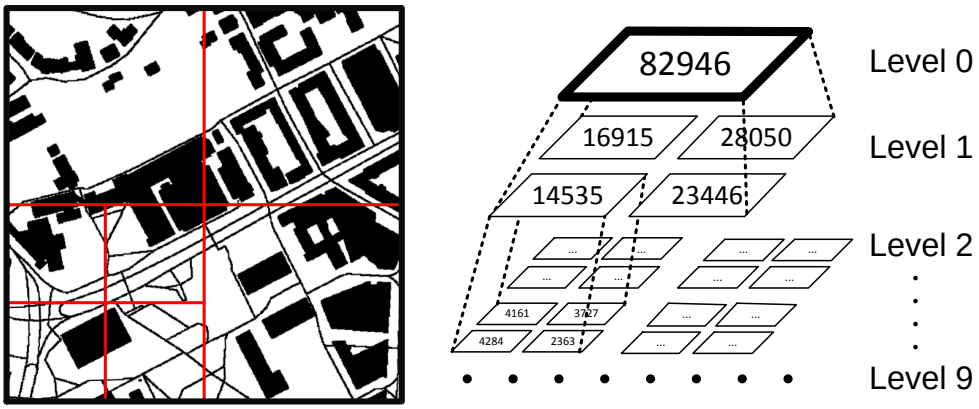


Figure 3.7: Quadtree and effective area example ($b = 2, l_{max} = 9$)

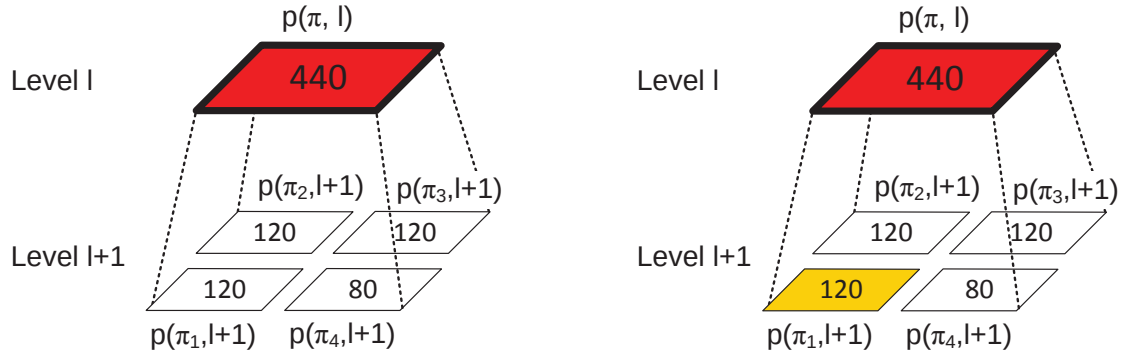
$l + 1$. Otherwise, level $l'_i = l$ is found.

To explain why it is required to consider $esize(p(\pi, l))$ for level l and also for the four child nodes of $p(\pi, l)$ on level $l+1$ within each step when traversing the quadtree, consider the following example as depicted in Figure 3.8:

Assume position $p(\pi, l)$ of an arbitrary level l covers the areas $p(\pi_1, l + 1)$, $p(\pi_2, l+1)$, and $p(\pi_3, l+1)$, all having an effective area size of 120. Area $p(\pi_4, l+1)$ covered by $p(\pi, l)$ has an effective area size of 80. For a given threshold $A_i = 100$, the dark area in Figure 3.8a of level l would be calculated as obfuscation area for each position $\pi \in p(\pi, l)$.

When not taking the child nodes of $p(\pi, l)$ into account when determining l'_i, an attacker could increase his precision for an update in an area with an effective area size smaller than A_i based on the returned obfuscation area and the knowledge of the share generation algorithm. For instance, each position $\pi \in p(\pi_1, l+1)$ would lead to an obfuscation area of level $l+1$ marked in a light color in Figure 3.8b, whereas $\pi \in p(\pi_4, l+1)$ would lead to an obfuscation area of level l marked dark. Thus, an attacker could increase his precision based on the returned obfuscation area by excluding $p(\pi_1, l+1)$ from a calculated obfuscation area of $p(\pi, l)$ for $A_i = 100$.

In the second step, we calculate the secrets $K_1, \ldots, K_m$ that are used as input



(a) Obfuscation area for each position $\pi \in p(\pi, l)$ considering effective area sizes and child nodes

(b) Different obfuscation areas considering effective area sizes without considering child nodes

Figure 3.8: Example showing impact of child node consideration (A_i=100, b=2)

for the multi-secret sharing scheme based on the calculated positions $p(\pi, l'_i)$ for each $A_i \in A_0, \ldots, A_m$. The number m of generated secrets is defined by the $m+1$ MO-defined area threshold values. Each secret K_i is calculated as the interleaved digits refining position $p(\pi, l'_0)$ of A_0 to position $p(\pi, l'_i)$ of A_i for $1 \leq i \leq m$.

The share generation of *PShare-GLM*$_{map}$ extends the share generation algorithm presented in Algorithm 1 for *PShare-GLM*. We apply the area adaptation to calculate the secrets by changing line 1 to 4 of Algorithm 1 and, therefore, the way how the secrets are generated. Furthermore, we change the position provided within the m-share from $p(\pi, 0)$ to $p(\pi, l'_0)$.

Details of the share generation are presented in Algorithm 4. First, we calculate the obfuscation area $p(\pi, l'_i)$ for each MO-defined threshold value $A_i \in A_0, \ldots, A_m$ by using function *getMaxLevelConsideringChilds*(π, A_i, q) traversing quadtree q top-down by checking the previously described requirements for the child nodes and $esize(p(\pi, l)) \geq A_i$ for each position $p(\pi, l)$. Then, the obfuscation areas are used to generate the secrets K_i by interleaving the refining digits of position $p(\pi, l'_0)$ of A_0 to position $p(\pi, l'_i)$ of A_i. The calculated secrets are then used as input for the multi-secret sharing scheme as presented in Algo-

Algorithm 4 *PShare-GLM$_{map}$*: Map-based share generation of the MO

Function: $generate(\pi, l_{max}, n, \{A_0, \ldots A_m\})$

1: $q \leftarrow getQuadtree(\pi, l_{max})$ ▷ Calculate quadtree
2: **for** $i = 0$ to m **do**
3: $\quad p(\pi, l'_i) \leftarrow getMaxLevelConsideringChilds(\pi, A_i, q)$
4: **end for**
5: **for** $i = 1$ to m **do**
6: $\quad K_i \leftarrow interleave(refinement(p(\pi, l'_0), p(\pi, l'_i)))$ ▷ Define secrets
7: **end for**
8: $P, f(X) \leftarrow calculatePolynomial(K)$ ▷ Apply multi-secret sharing
9: $X \leftarrow n$ distinct integer
10: **for** $i = 1$ to n **do**
11: $\quad y'_i \leftarrow f(x_i)$ ▷ $x_i \in X$
12: $\quad sr^i_\pi \leftarrow (x_i, y'_i)$ ▷ Define server r-share
13: **end for**
14: $m_\pi \leftarrow P, p(\pi, l'_0)$ ▷ Define m-share
15: **return** $m_\pi, \{sr^1_\pi, \ldots, sr^n_\pi\}$

rithm 1 for *PShare-GLM*. Finally, position $p(\pi, l'_0)$ is stored within the m-share fulfilling $esize(p(\pi, l'_0)) \geq A_0$.

The share combination of *PShare-GLM$_{map}$* is basically calculated as presented in Algorithm 3 for *PShare-GLM*. The only difference is that position $p(\pi, l)$ in line 3 is now defined as $p(\pi, l'_i)$ and is calculated by splitting up K_i into the x and y part refining $p(\pi, l'_0)$ of the m-share to $p(\pi, l'_i)$ by substituting the corresponding digits of $p(\pi, l'_0)$. Furthermore, the value of l'_i is calculated based on the length of the reconstructed secret K_i.

3.1.4 Symbolic Position Sharing

Next, we present *PShare-SLM*, the symbolic counterpart to the position sharing algorithm *PShare-GLM*. The general idea of *PShare-SLM* is similar to *PShare-GLM*: We apply the multi-secret sharing scheme presented by Chan

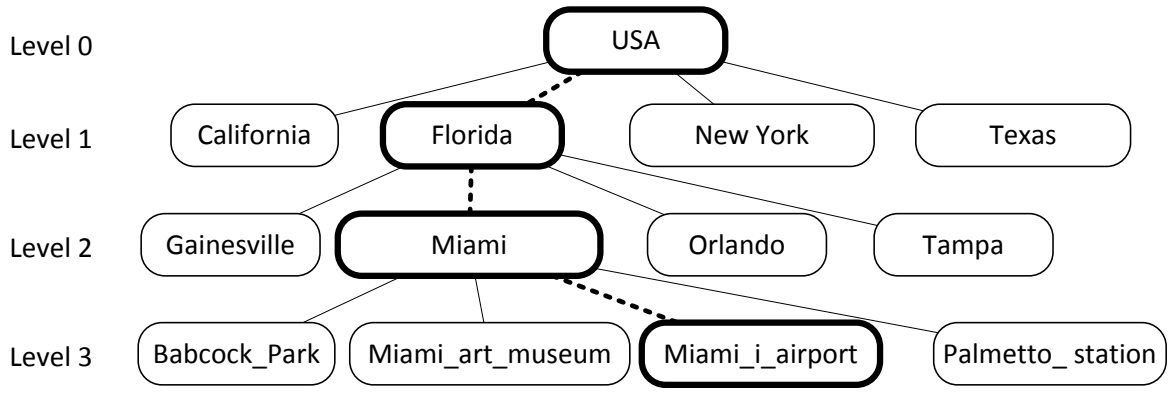


Figure 3.9: Simplified location hierarchy example

and Chang [CC05] to share the MO's position in different precision levels with different clients. Since the symbolic location definition differs from the geometric definition, we start with an explanation of our symbolic location model, before we present the specific share generation and combination algorithms.

Symbolic Location Model

Our symbolic location model consists of a location hierarchy based on the spatial containment relationship. Each location is represented as a vertex v_j in the hierarchy and has a level l defining the length of the path from the root to v_j (cf. Figure 3.9). The root location is on level zero, and we assume that all leaf vertices have the same level l_{max}. Each location has a unique location name in the context of its parent location, for instance, "Florida" for the location representing the State of Florida as child of the location representing the country USA. The concatenation of names on the path from the root to a location defines a unique label for each location, such as `usa/florida/miami/miami_i_airport` for the location representing the Miami International Airport. Each location label can be mapped to a unique identifier represented as an integer, which serves as input to the secret sharing scheme as presented below.

Using a hierarchical model makes it easy to define positions of different pre-

Algorithm 5 *PShare-SLM*: Share generation executed on the MO

Function: $generate(\pi, l_{max}, n)$

1: $p(\pi,0),\ldots,p(\pi,l_{max}) \leftarrow ancestors(p(\pi,l_{max}))$ ▷ Calculate ancestor vertices
2: **for** $i = 1$ to l_{max} **do**
3: $\quad K_i \leftarrow p(\pi,i).id$ ▷ Define secrets
4: **end for**
5: $P, f(X) \leftarrow calculatePolynomial(K)$ ▷ Apply multi-secret sharing
6: $X \leftarrow n$ distinct integer
7: **for** $i = 1$ to n **do**
8: $\quad y'_i \leftarrow f(x_i)$ ▷ $x_i \in X$
9: $\quad sr^i_\pi \leftarrow (x_i, y'_i)$ ▷ Define server r-share
10: **end for**
11: $m_\pi \leftarrow P, p(\pi,0).id$ ▷ Define m-share
12: **return** $m_\pi, \{sr^1_\pi,\ldots,sr^n_\pi\}$

cisions. Each hierarchy level defines a precision level, where level l_{max} defines the highest precision where the MO is located. Again, $p(\pi,l)$ denotes a position of precision level l similar to the geometric model. However, $p(\pi,l)$ is now represented as symbolic location identifier rather than geometric coordinates. The sequence of ancestor vertices of a position $p(\pi,l)$ is denoted as

$$ancestors(p(\pi,l)) = (p(\pi,0),\ldots,p(\pi,l)).$$

Share Generation and Share Combination

We now apply the idea of multi-secret sharing to our symbolic position sharing approach *PShare-SLM*. The share generation executed by the MO is shown in Algorithm 5.

First, the MO evaluates function $ancestors(p(\pi,l_{max}))$ to determine the set of positions $(p(\pi,0),p(\pi,1),\ldots,p(\pi,l_{max}))$ (line 1). The identifier of $p(\pi,0)$ defines the root of the hierarchy and is stored in the m-share. The identifiers of $p(\pi,1)$ to $p(\pi,l_{max})$ are used as secrets $K_1,\ldots,K_{l_{max}}$ for the multi-secret sharing scheme.

Algorithm 6 *PShare-SLM*: Share combination executed on the clients

Function: $combine(m_\pi, \{cr_{\pi,l}^{LS_1}, \ldots, cr_{\pi,l}^{LS_{\tau_l}}\}, l)$

1: $f_l(X) \leftarrow lagrange(\{cr_{\pi,l}^{LS_1}, \ldots, cr_{\pi,l}^{LS_{\tau_l}}\}, m_\pi.p_l)$ ▷ Polynomial interpolation
2: $p(\pi, l).id \leftarrow f_l(0)$ ▷ Reconstruct secret as location identifier
3: **return** $p(\pi, l)$

The remaining part of the algorithm is similar to the share generation of *PShare-GLM*. That is, we apply the multi-secret sharing algorithm by calculating the server r-shares, and distribute them to the LSs.

Similarly, the share combination algorithm as depicted in Algorithm 6 again uses the *Lagrange interpolation* over the field $\mathbb{Z}_{p_l}[X]$ to reconstruct the secret polynomial $f_l(X)$ for a given level l. The constant term of $f_l(X)$ is the identifier of $p(\pi, l)$, which can be mapped to the label of $p(\pi, l)$.

Multiple Position Updates

Next, we analyze *PShare-SLM* with regard to multiple symbolic position updates. As pointed out, an attacker knowing the complete position history $U = \{(p(\pi_{first}, l), t_{first}), \ldots, (p(\pi_{last}, l), t_{last})\}$ for a certain level l could try to use a maximum movement boundary attack to increase precision. Note that although the location hierarchy itself does not define the distance information necessary for such attacks, an attacker can determine this information by matching symbolic locations to available topographic maps.

The basic idea to counter such attacks is similar to the geometric case: A new update $p(\pi_{i+1}, l)$ is only permitted if any position within $p(\pi_{i+1}, 0)$ is reachable from any position within $p(\pi_i, 0)$ considering $\Delta t = |t_{i+1} - t_i|$ and the MO's maximum velocity v_{max}. Although theoretically this would be an effective countermeasure, it impacts the minimal update time between two succeeding updates as specified in Equation 3.1. In contrast to the geometric case, where the precision of $p(\pi_i, 0)$ can be specified by the MO, level zero is now defined by the

root location of the given symbolic location model. Therefore, it is worthwhile to have a closer look at the influence on the minimal update time.

Assume a model where the root location covers a whole country like Germany. In this case, the maximum distance between two positions within the hierarchy would be about 1 000 km. For a MO walking with $v_{max} = 6\,km/h$, this results in a maximum delay of 6.94 days, whereas a maximum velocity of $v_{max} = 200\,km/h$ of a car decreases the minimum time between two updates to five hours. For an inner city scenario with a maximum distance of 10 km and a MO walking with at most $v_{max} = 6\,km/h$, the minimum delay between two updates would be 1.66 hours.

These examples show that there are scenarios where the minimal update time between two succeeding updates would be hours rather than days. This would be sufficient for many "check-in" applications, as our evaluations based on real world position information show in Section 3.1.6. For applications with shorter update intervals, the geometric approach would be better suited.

3.1.5 Privacy Analysis

Next, we present the privacy analysis for *PShare-GLM* and *PShare-SLM*. We start with a description of the attacker model and present afterwards the analyzed attacks in detail.

Attacker Model

As attackers we consider malicious LSs and malicious clients. Each attacker has access to the public m-shares. Each malicious LS additionally knows one server r-share for each position. Each malicious client with access to a position of precision level l additionally knows l client r-shares. We both consider single attackers (a single malicious LS or client), as well as colluding attackers (multiple malicious LSs or clients). We structure the following analysis according to different attacks. First, we consider *single attackers* who analyze a single (current) position, or even the complete history of positions based on the differ-

ent attacks presented in our attacker classification in Section 2.6.2. Second, we analyze the effect of *colluding attackers* who combine their compromised shares. Since *PShare-GLM* and *PShare-SLM* are based on the same multi-secret sharing scheme, we do not distinguish between them unless the difference is relevant.

Single Attacker

First, we consider a malicious client having access to l client r-shares of a single position that can be used to reconstruct $p(\pi, l)$. Thus, the client knows secret K_l refining $p(\pi, 0)$ to $p(\pi, l)$ from these shares. As shown by Chan and Chang [CC05], their multi-secret sharing scheme ensures that different secrets are independently protected by different polynomials. Thus, the information from K_l cannot be used to reconstruct other secrets and positions of levels greater than l.

A single malicious LS has access to the m-share and one server r-share, i.e., it knows one distinct point of the secret polynomial $f(X)$. Therefore, the malicious LS can calculate for each precision level l with $0 < l \leq l_{max}$ exactly one point of the polynomial $f_l(X)$. Thus, the malicious LS can reconstruct the MO's position $p(\pi, 1)$ of level one, while the positions of all levels greater than one, which require at least two r-shares, cannot be reconstructed.

Our proposed extension against map matching attacks (cf. Section 3.1.3) ensures that an attacker cannot reduce the effective area size of a known position $p(\pi, l'_i)$ below a MO-defined value A_i by using additional map knowledge.

Next, we consider attackers knowing for a certain level l the complete position history $U = \{(p(\pi_{first}, l), t_{first}), \ldots, (p(\pi_{last}, l), t_{last})\}$ instead of only a single position of the MO. Our algorithms create position shares of different positions independently from each other. Therefore, shares generated for $(p(\pi_i, l), t_i)$ cannot be combined with shares for $(p(\pi_{i+1}, l), t_{i+1})$. Nevertheless, the reconstructed positions $p(\pi_{first}, l), \ldots, p(\pi_{last}, l)$ could be used by an attacker for a region intersection attack or a maximum movement boundary attack as presented in Section 2.6.2. However, our deterministic obfuscation approach guarantees that two calculated positions of a certain precision level l are always either identical

or do not intersect each other. Therefore, an attacker cannot increase his known precision based on a region intersection attack. Since we use delayed updates as countermeasure against maximum movement boundary attacks, these attacks are also futile.

To prevent that an attacker can increase the precision of an obfuscated position by combining the map matching attack and the maximum movement boundary attack, our approaches could further be extended based on the idea presented by Yigitoglu et al. [YDAS12] to calculate the delay time between two position updates according to the map-based reachability of two obfuscated positions instead of the point-pairwise distance.

Colluding Attackers

Assume, for instance, three malicious clients $Client_A$, $Client_B$, and $Client_C$. Assume that $Client_A$ and $Client_B$ own l client r-shares of the same precision level l so that both can calculate $p(\pi, l)$. $Client_C$ owns $l+1$ client r-shares of the next precision level $l+1$ to reconstruct $p(\pi, l+1)$. Then, the collusion of $Client_A$ and $Client_B$ does not reveal anything new to $Client_A$ and $Client_B$, as they were both already allowed to reconstruct $p(\pi, l)$ by calculating polynomial $f_l(X)$ using their l client r-shares each representing a distinct point of $f_l(X)$. Because polynomial $f_l(X)$ is uniquely defined by l distinct points, even using more than l client r-shares of the same precision level leads to the same polynomial $f_l(X)$ and therefore $p(\pi, l)$. Furthermore, the client r-shares of precision level l reconstructing $f_l(X)$ reveal nothing about the polynomial $f_{l+1}(X)$ and therefore about $p(\pi, l+1)$. This is based on the fact that the polynomials of different precision levels are generated independently from each other in the multi-secret sharing scheme [CC05]. Thus, even the collusion of $Client_A$ and $Client_C$ with client r-shares of different precision levels does not reveal any new information. $Client_C$ can reconstruct $p(\pi, l+1)$ even without collusion and the additional client r-shares of $Client_A$ carry no information about polynomial $f_{l+2}(X)$ of the next precision level $l+2$.

Second, we consider multiple malicious LSs. Let M be the number of colluding

LSs. These LSs can use their stored server r-shares to calculate M different client r-shares for each level l. Since we defined the threshold values as $\tau_l = l$, l client r-shares are required to get a position $p(\pi, l)$ of precision level l. Therefore, M colluding LSs can reveal positions up to level $l = M$. This shows the desired graceful degradation of privacy property of our approach. The revealed precision increases with the number of compromised LSs. We could even increase the robustness by setting the threshold values τ_l to values greater than l. Then, more LSs are required to calculate a position of a certain level, which increases the overhead on the one hand, but on the other hand increases the robustness of our approach. Thus, our scheme allows for trading off overhead against robustness.

The collusion of malicious LSs and malicious clients is a special case of the collusion of malicious LSs. In this case, either the client with the highest precision level or the number of colluding LSs defines the revealed precision level.

3.1.6 Evaluation

Next, we present our evaluation results. We begin with our evaluation based on real world positions and present our runtime performance evaluation afterwards.

Real World Position Evaluation

As defined in Equation 3.1, the minimum time between two updates is restricted by the MO's maximum speed and the size of the level zero position to guarantee protection against maximum movement boundary attacks. To analyze the practical impact of this restriction, we analyzed real datasets of position check-ins from existing LBAs to see how they comply with this restriction. If many updates were violating the restriction, this would indicate that our approach is not applicable to these applications as users would be unable to perform many desired updates.

The analyzed dataset, which was collected by Cheng et al. [CCLS11] between September 2010 and January 2011, contains 22 387 922 user position check-ins of 224 803 users from different LBAs, such as Foursquare [Fou14b], all over the

Category	Pedestrian	Ground vehicle	Plane
Distance (d) and average speed (v)	$d \leq 10\,\text{km}$ and $v \leq 6\,\frac{\text{km}}{\text{h}}$	($d \leq 10\,\text{km}$ and $6\,\frac{\text{km}}{\text{h}} < v \leq 200\,\frac{\text{km}}{\text{h}}$) or ($10\,\text{km} < d \leq 10\,000\,\text{km}$ and $v \leq 200\,\frac{\text{km}}{\text{h}}$)	$d > 10\,000\,\text{km}$ or $v > 200\,\frac{\text{km}}{\text{h}}$
$\mathbf{v_{max}}$	$6\,\frac{\text{km}}{\text{h}}$	$200\,\frac{\text{km}}{\text{h}}$	$1\,000\,\frac{\text{km}}{\text{h}}$
#updates	15 895 691	6 079 316	412 915

Table 3.1: Position update classification

world. Since the dataset only contains geometric coordinates, this evaluation focuses on *PShare-GLM*. For our purpose, we processed the dataset by classifying each position update as pedestrian, ground vehicle, or plane based on the traveled distance and the average speed between two succeeding updates, as shown in Table 3.1. The categories pedestrian and ground vehicle represent typical user movement behavior, while category plane considers all updates that do not match the first two categories.

Table 3.1 also shows the assumed maximum speed for *PShare-GLM* and the resulting number of updates per category. For each category, we calculated the percentage of updates that can be performed without violating the restriction presented in Equation 3.1. Figure 3.10 depicts the results for the categories and different sizes of level zero positions ranging from 1 m (2^0) square side length to 32 768 m (2^{15}). As we can see, in the worst case, for an obfuscation area with a side length of 32 768 m, 55.19% of the pedestrian updates are possible. For a side length of 1 024 m, which provides sufficient privacy for pedestrians in an inner city scenario for example, 88.09% of the updates are possible. For ground vehicles, 83.59% of the updates are possible using even the coarsest obfuscation of 32 768 m. Figure 3.11 depicts the results of all position updates together from all categories. As it can be seen, for a level zero size of 1 024 m, 90.08% of all updates can be published. Thus, we can state that the minimum update time restriction only affects a small number of updates.

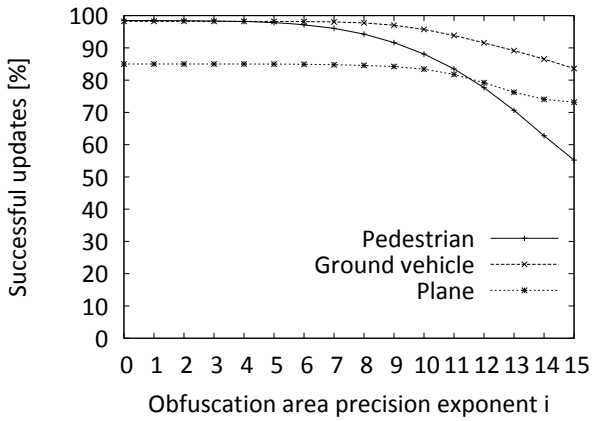


Figure 3.10: Successful updates for obfuscation areas of precision 2^i[m]

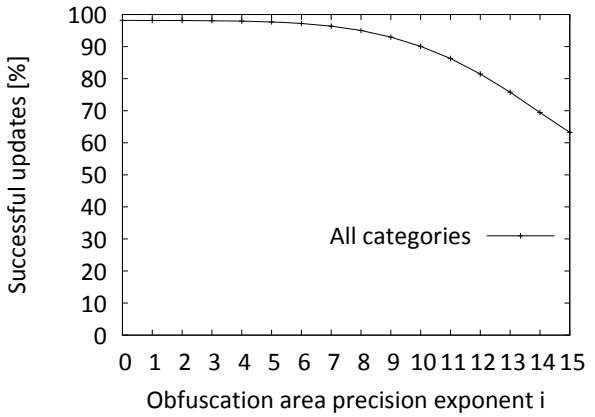


Figure 3.11: Successful updates for obfuscation areas of precision 2^i[m]

Runtime Performance Evaluation

Besides privacy, the efficiency of share generation is important for the practical application of our approaches. Share generation is performed on the MO's mobile device, which typically has low performance in terms of computational speed. Also on such resource-poor devices, share generation must be possible in short time. An efficient share generation also leads to small overhead in terms of energy, which is desirable for battery-operated mobile devices.

We measured the overall time for share generation of *PShare-GLM*, *PShare-*

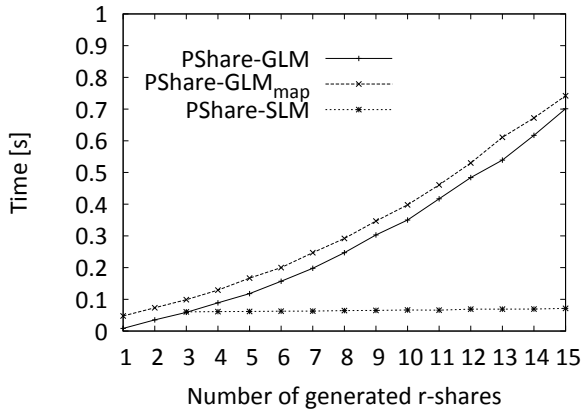


Figure 3.12: Performance evaluation of share generation ($n = l_{max}$ shares, $b = 2$)

GLM$_{map}$, and *PShare-SLM* on a mobile device (HTC Desire HD, 1 GHz Qualcomm Snapdragon S2, 768 MB RAM). For *PShare-SLM*, we used a hierarchy with a maximum level of $l_{max} = 3$ and measured the time to generate one m-share and three to fifteen r-shares. The shares are all generated for the same number of $l_{max} = 3$ secrets so that three r-shares are required to reconstruct the precise position of the MO on level three. For *PShare-GLM*, we measured the time to create one m-share and a varying number of $n = l_{max}$ r-shares. By increasing n, we also increased the number of secrets used. To evaluate *PShare-GLM$_{map}$*, we used road network and building data provided by OpenStreetMap [Ope14] for the inner city of Stuttgart. We defined the obfuscation area of $p(\pi, 0)$ by setting $l_{max} = 13$, such that $p(\pi, 0)$ covers an area of 67.1 km^2 and has an effective area size of 10.5 km^2. We increased the number of generated r-shares from one to fifteen by increasing the number of area thresholds used. We varied each area threshold between 10 km^2 and 10 m^2. The results are shown in Figure 3.12. The depicted results of *PShare-GLM* are measured for $b = 2$ only since other b values led to almost identical results. The plotted values are the average over several runs per share number.

As it can be seen, the runtime of *PShare-SLM* is almost constant for the different numbers of generated r-shares. This is based on the fact that the

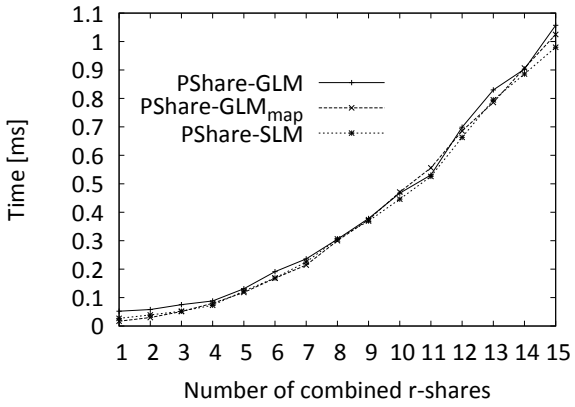


Figure 3.13: Performance evaluation of share combination

number of used secrets is defined by the maximum level three of the hierarchy. For *PShare-GLM* and *PShare-GLM*$_{map}$, the number of secrets was increased by increasing the number of generated shares. Nevertheless, the runtime of all three approaches stays well below one second even for larger numbers of shares. Therefore, we can state that the share generation algorithms are efficient and suitable even for resource-poor devices.

Next, we show the performance evaluation for the share combination, which is done by clients typically running in an infrastructure without energy restrictions and high computational power. For *PShare-GLM*, *PShare-GLM*$_{map}$, and *PShare-SLM*, we measured the time to combine the m-share with up to 15 r-shares on a personal computer (Intel Core 2 Duo, 2.53 GHz, 3 GB RAM). The results are shown in Figure 3.13. As it can be seen, the time for share combination of all three approaches is nearly the same and stays below 1.1 milliseconds even for a larger number of combined shares.

3.2 Position Sharing based on Binary Space Partitioning

In this section, we present our second position sharing approach *PShare-BSP*, which is based on binary space partitioning instead of multi-secret sharing. *PShare-BSP* focuses on the efficiency of position sharing by implementing an optimized share update protocol reducing the communication costs for position updates.

First, we describe our extended system model and our problem statement for *PShare-BSP*. Afterwards, we introduce the details of *PShare-BSP* and analyze the level of privacy it provides. Finally, we present our evaluation results.

3.2.1 Extended System Model

The basic functionality of the mobile objects, location servers, and clients in our extended system model corresponds to their functionality presented for *PShare-GLM* in Section 3.1.1. Compared to *PShare-GLM*, *PShare-BSP* is based on a different technique to implement position sharing such that the definition of the m-share and the r-shares differ with respect to their contained information as further introduce below. Nevertheless, the general definition of positions on different precision levels is identical. The share generation executed on the MO splits up the MO's precise position π into the m-share m_π and the set of r-shares $S_\pi = \{r_{\pi,1}, \ldots, r_{\pi,l_{max}}\}$ by using function

$$generate(\pi, l_{max}) = (m_\pi, S_\pi).$$

The MO-defined parameter l_{max} defines the number of $l_{max} + 1$ different precision levels offered to clients, and the number of generated r-shares. Due to the share combination technique presented below, the number of generated r-shares in *PShare-BSP* corresponds to the number of different precision levels provided to clients. In *PShare-GLM*, the number n of generated r-shares was independent from the number of supported precision levels. Thus, even more than l_{max}

r-shares could be generated. The definition of the MO's position of precision level l with $0 \leq l \leq l_{max}$ is again $p(\pi, l)$. The m-share m_π defines position $p(\pi, 0)$ with a precision which is low enough to be published without raising privacy concerns. The r-shares increase the precision of the m-share ($p(\pi, 0)$) by providing refinement information stored in the r-shares.

LSs store position shares and answer queries from different clients by returning the corresponding shares based on the implemented access control mechanism.

Clients query shares from different LSs and use the *share combination* function

$$combine(m_\pi, \{r_{\pi,1}, \ldots, r_{\pi,l}\}) = p(\pi, l)$$

to increase the provided precision of $p(\pi, 0)$ to $p(\pi, l)$ with $0 < l \leq l_{max}$ by combining the m-share m_π and l r-shares from different LSs. Each r-share $r_{\pi,i}$ has an index i defining the sequence of the l_{max} r-shares. Compared to *PShare-GLM*, where any l out of the n LSs could be queried for their r-shares to increase precision from level zero to level l, the set of r-shares which a client has to query in *PShare-BSP* is the set of the LSs storing the sequence of the first l r-shares.

3.2.2 Problem Statement

As introduced in Section 3.1.2, an ideal position sharing approach will not allow an attacker knowing position $p(\pi, l)$ to derive a position with a precision beyond the precision of $p(\pi, l)$. Otherwise, the precision offered to LSs and clients could not be controlled by providing access to a certain number of shares. Therefore, we consider the same problem statement as presented for *PShare-GLM*, but rely on a different technique to solve this problem, namely, binary space partitioning instead of multi-secret sharing. Additionally, our goal is to allow for an efficient update of position shares to reduce the overhead of position sharing.

3.2.3 Position Sharing Approach

In this section, we present our position sharing approach *PShare-BSP* implementing the functions for share generation and share combination introduced above. Then, we introduce an optimization reducing the number of required share updates significantly.

Basic Approach

Both share generation and combination depend on the concept of how to translate the MO's precise position π into an obfuscated position $p(\pi, l)$ of a certain precision level l. As pointed out, *PShare-BSP* focuses on geometric locations and relies on the same location model as presented for *PShare-GLM* in Section 3.1.3. We assume a granularity parameter of $b = 2$ such that an obfuscated position $p(\pi, l)$ of a given precision level l is defined as $p(\pi, l) = ((x_l, y_l), 2^{l_{max}-l})$ representing the square area that contains the MO's precise position π. Coordinates (x_l, y_l) of $p(\pi, l)$ are defined as coordinates of $\pi.x$ and $\pi.y$ with the $l_{max} - l$ least significant bits set to zero. These bits are the undefined bits of $p(\pi, l)$. The position of the i-th undefined bit of a coordinate is equal to its $(l_{max} + 1) - i$ least significant bit. The precision value of $p(\pi, l)$ is set to $2^{l_{max}-l}$ defining the range of the $l_{max} - l$ undefined bits. The less undefined bits exist, the higher is the precision of a position. As an example, consider Figure 3.14 where the undefined bits of $p(\pi, l)$ that were set to zero are underlined for each level l.

The m-share m_π represents the coarsest obfuscation area $p(\pi, 0)$ with the coordinates (x_0, y_0) and the precision value l_{max}. To calculate $p(\pi, 0)$, the l_{max} least significant bits of $\pi.x$ and $\pi.y$ are replaced by zero. l_{max} again defines the deterministic partitioning of the movement area into the grid of cells with a side length of $2^{l_{max}}$ meter.

To increase the precision of the m-share m_π, the undefined bits of $p(\pi, 0)$ have to be defined. To this effect, we use a set $S'_\pi = \{r_{\pi,1}, \ldots, r_{\pi,l}\}$ of $l \leq l_{max}$ r-shares, where each r-share $r_{\pi,i}$ defines the i-th undefined bit of the x and y value of $p(\pi, 0)$. At the same time the precision is improved by decreasing the

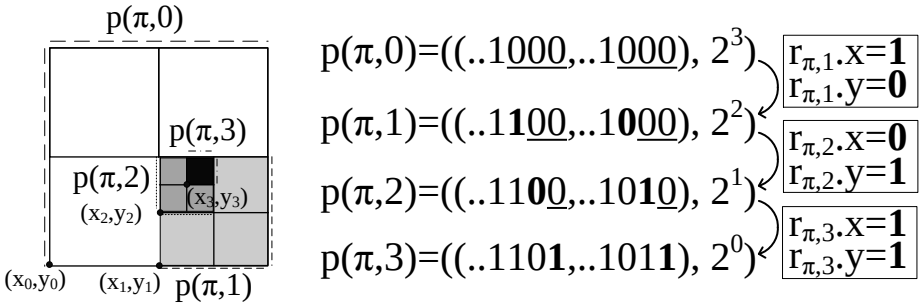


Figure 3.14: Refinement example for $p(\pi, 0)$ based on $l_{max} = 3$

side length value of $p(\pi, 0)$. Thus, position $p(\pi, 0)$ can be refined up to $p(\pi, l)$ by incrementally substituting the undefined bits of $p(\pi, 0)$ by the corresponding bits of the r-shares. As mentioned, the number of generated r-shares for each position update is defined by the value l_{max} such that the precision of $p(\pi, 0)$ can be increased up to l_{max} different precision levels from $p(\pi, 1)$ to $p(\pi, l_{max})$. As shown in Figure 3.14, the refinement information of the r-shares $r_{\pi,1}$, $r_{\pi,2}$, and $r_{\pi,3}$ defines stepwise the undefined bits of $p(\pi, 0)$ increasing the provided precision up to the precise position without any undefined bits. As we can see, a partial refinement up to a certain precision level l is also possible where certain bits remain undefined.

The share generation uses function $generate(\pi, l_{max})$ presented in Algorithm 7 to calculate the m-share and the r-shares. The m-share is calculated in function $floorBits(\pi, l_{max})$ by setting the l_{max} least significant bits of $\pi.x$ and $\pi.y$ to zero. Each r-share $r_{\pi,i}$ with $1 \leq i \leq l_{max}$ defines the two bits of $\pi.x$ and $\pi.y$ corresponding to the bits at the position of the i-th undefined bit in $p(\pi, 0)$. These bits are returned by function $getBits(\pi, i)$.

The share combination function $combine(m_\pi, \{r_{\pi,1}, \ldots, r_{\pi,l}\})$ is implemented as shown in Algorithm 8. It takes as input the m-share m_π representing the obfuscation area $p(\pi, 0)$ and a sequence of l r-shares $r_{\pi,1}, \ldots, r_{\pi,l}$. In order to refine the position up to a certain precision level l, the sequence of r-shares must contain all r-shares for level one to l. The refined position $p(\pi, l)$ is calculated

Algorithm 7 *PShare-BSP*: Share generation executed on the MO

Function: $generate(\pi, l_{max})$

1: $m_\pi.p(\pi, 0) \leftarrow floorBits(\pi, l_{max})$ ▷ Generate m-share
2: **for** $i = 1$ to l_{max} **do**
3: $\quad r_{\pi,i} \leftarrow getBits(\pi, i)$ ▷ Define r-shares
4: **end for**
5: $S_\pi \leftarrow \{r_{\pi,1}, \ldots, r_{\pi,l_{max}}\}$ ▷ Define set of l_{max} r-shares
6: **return** m_π, S

Algorithm 8 *PShare-BSP*: Share combination executed on the clients

Function: $combine(m_\pi, \{r_{\pi,1}, \ldots, r_{\pi,l}\})$

1: $p(\pi, 0) \leftarrow m_\pi.p(\pi, 0)$
2: **for** $i = 1$ to l **do**
3: $\quad p(\pi, i) \leftarrow setBits(p(\pi, i-1), r_{\pi,i})$ ▷ Define undefined bits
4: **end for**
5: **return** $p(\pi, l)$

by replacing the i-th undefined bit of the x and y values of $p(\pi, 0)$ by the bits of r-share $r_{\pi,i}.x$ and $r_{\pi,i}.y$ for $1 \leq i \leq l$. This step is implemented in function $setBits(p(\pi, i-1), r_{\pi,i})$.

To protect multiple position updates from revealing additional information to an attacker using the maximum movement boundary attack, we rely on the concept of delayed position updates as counter measure, which we introduced and analyzed for *PShare-GLM* in Section 3.1.6. Since *PShare-GLM* and *PShare-BSP* both rely on the same representation of an obfuscated position $p(\pi, l)$ for a certain precision level l, the analysis and the results for the temporal delayed position updates of *PShare-GLM* can be applied to *PShare-BSP*.

Update Optimization

A drawback of position sharing is that multiple shares have to be updated per position fix instead of one. To alleviate this problem, we present an optimization of our basic approach *PShare-BSP*, called *PShare-BSP*^{opt}, reducing the number of messages required for position updates significantly.

Existing position sharing approaches [DSR11, SDR12], *PShare-GLM/SLM*, and the basic approach of *PShare-BSP* need to update the m-share and all r-shares for each new position. The general idea of *PShare-BSP*^{opt} is that, initially, the m-share and all r-shares are updated once. For the following $k-1$ position updates, we re-use the r-shares to refine the positions of several m-shares and update only the m-share for every new position. To this end, a m-share contains a partially encrypted position that can be decrypted by the bits of the corresponding key that is split up and stored within different r-shares. Therefore, only one share has to be updated per new position rather than $1 + l_{max}$ (one m-share and l_{max} r-shares). In order to allow for the re-use of r-shares for the next $k-1$ updates, we have to include additional information for each r-share. The value of k can be defined by the MO and determines the number of times the r-shares can be re-used before they must be updated. For simplicity, we limit our explanations to the x coordinate of the MO's position. The y coordinate is handled analogously.

We use a one-time pad encryption to protect the l_{max} least significant bits of $\pi.x$ for each position update. Generally, a one-time pad encryption can be used to protect a secret by XORing it with a random set of bits defining the key of the encryption. The result of the encryption is a cipher that does not provide any information about the secret without the key. We define the secret s_x as the l_{max} least significant bits of $\pi.x$. The corresponding key is of length l_{max} and is called *refinement-key* (r-key). The cipher c_x is calculated by bitwise XORing s_x with the corresponding r-key as

$$c_x = s_x \text{ XOR } r\text{-key}_x. \tag{3.4}$$

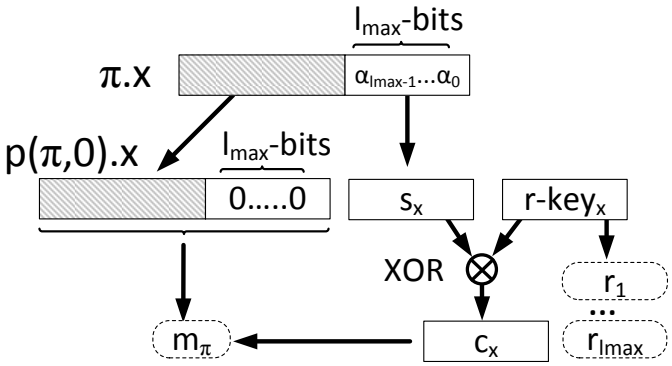


Figure 3.15: Share generation overview for *PShare-BSP*^{opt}

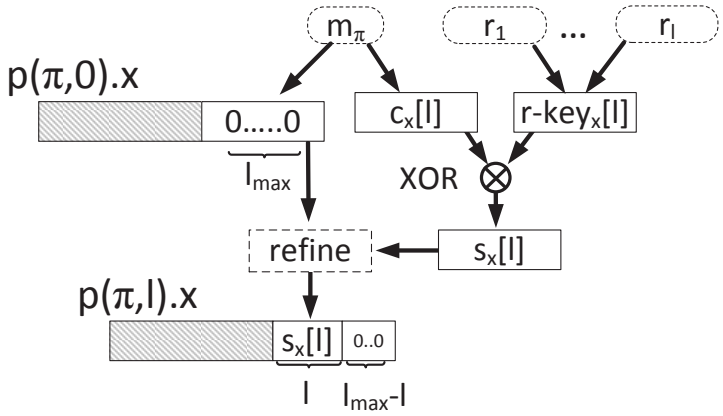


Figure 3.16: Share combination overview for *PShare-BSP*^{opt}

Cipher c_x is then stored within the m-share m_π in addition to $p(\pi, 0)$ as shown in Figure 3.15. The idea is now to split up r-key$_x$ into its l_{max} bits and distribute them as part of the r-shares to different LSs. The refinement of $p(\pi, 0)$ to $p(\pi, l)$ for a certain precision level l requires the l most significant bits of the r-key$_x$, denoted as r-key$_x[l]$. The refinement is done by decrypting c_x with the combined r-key$_x[l]$ of the r-shares as shown in Figure 3.16. The result of the decryption defines the l most significant bits of s_x, denoted as $s_x[l]$. Finally, $p(\pi, 0)$ is refined to $p(\pi, l)$ by substituting the l most significant undefined bits of $p(\pi, 0)$ by $s_x[l]$. The $l_{max} - l$ undefined bits remain zero.

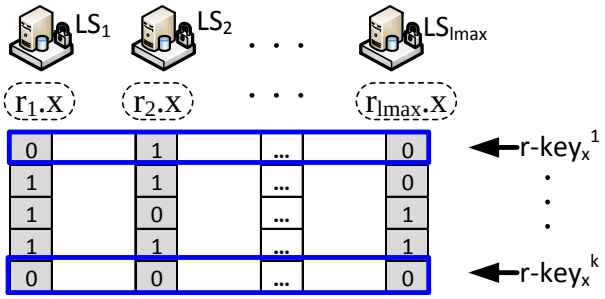


Figure 3.17: Correlation of r-shares and r-keys in *PShare-BSP*^{opt}

To fulfill the optimization goal, we provide within each r-share k bits that can reconstruct the r-keys of k updates. The r-share r_i stores the i-th bit of any of the k generated r-keys. The different r-keys are referenced by an identifier, denoted as r-key^{id}. The correlation of r-shares and r-keys is shown in Figure 3.17, where LS_i stores the r-share r_i representing a secure random set of k bits.

Next, we describe the algorithms for share generation and share combination of *PShare-BSP*^{opt}. The share generation presented in Algorithm 9 checks whether a distributed r-key can be used for the next update. If no unused r-key is available, a set of k new r-keys and l_{max} new r-shares is generated. Then, the id of the r-key to use is set. Finally, Algorithm 10 generates the m-share m_π including cipher c_x. After generation, the shares are distributed to the LSs.

The share combination presented in Algorithm 11 reconstructs position $p(\pi, l)$ of precision level l by combining the m-share m_π with l r-shares $r_1, \ldots, r_l$. The l r-shares compose the corresponding $r\text{-}key_x^{id}[l]$ of length l. Then, cipher c_x of m_π is XORed with $r\text{-}key_x^{id}[l]$. The reconstructed refinement bits then substitute the corresponding undefined bits. The remaining $l_{max} - l$ bits are still undefined. Without knowing the missing $l_{max} - l$ r-shares and thus the corresponding parts of the r-*key*, it is not possible to further increase the precision of $p(\pi, l)$.

To guarantee the provided privacy of *PShare-BSP*^{opt}, it is essential that each r-key is only used once. Otherwise, the encryption can be broken. Thus, we use each r-key only once and renew the r-shares after all k r-keys have been used.

Algorithm 9 *PShare-BSP^{opt}*: Share generation executed on the MO

Function: $generateSharesOPT(\pi, l_{max})$

1: **if** $noUnusedRKeyAvailable()$ **then**
2: **for** $id = 1$ to k **do**
3: $r\text{-}key_x^{id} \leftarrow getRandBits(l_{max})$ ▷ Generate random bits
4: **end for**
5: **for** $i = 1$ to l_{max} **do**
6: $r_i.x \leftarrow getBitsFromRKeys(i)$ ▷ Define r-shares
7: **end for**
8: $S \leftarrow \{r_1, \ldots, r_{l_{max}}\}$
9: **end if**
10: $id \leftarrow getUnusedRKeyID()$ ▷ Select r-key id
11: $m_\pi \leftarrow generateMShareOPT(\pi, l_{max}, r\text{-}key_x^{id}, id)$ ▷ Calculate m-share
12: **return** m_π, S

Algorithm 10 *PShare-BSP^{opt}*: m-share generation executed on the MO

Function: $generateMShareOPT(\pi, l_{max}, r\text{-}key_x^{id}, id)$

1: $m_\pi.p(\pi, 0) \leftarrow floorBits(\pi, l_{max})$ ▷ Calculate $p(\pi, 0)$
2: $s_x \leftarrow getXRefinement(\pi.x, l_{max})$ ▷ Define secret
3: $m_\pi.c_x \leftarrow s_x$ XOR $r\text{-}key_x^{id}$ ▷ Calculate cipher of m-share
4: $m_\pi.rKeyID \leftarrow id$ ▷ Set r-key id of m-share
5: **return** m_π

Algorithm 11 *PShare-BSP^{opt}*: Share combination executed on the clients

Function: $combineOPT(m_\pi, \{r_1, \ldots, r_l\})$

1: $r\text{-}key_x^{id}[l] \leftarrow getRKey(m_\pi.rKeyID, \{r_1.x, \ldots, r_l.x\})$ ▷ Calculate r-key
2: $p(\pi, l).x \leftarrow setXBits(m_\pi.p(\pi, 0).x, m_\pi.c_x[l]$ XOR $r\text{-}key_x^{id}[l])$ ▷ Define bits
3: **return** $p(\pi, l)$

3.2.4 Privacy Analysis

In this section, we present our attacker model and analyze the robustness of our approaches against different location privacy attacks.

Attacker Model

We assume malicious clients and malicious LSs that could be compromised as possible attackers. Malicious clients are a special sub-case of malicious LSs. We consider the case that the LSs storing the m-share and l r-shares are compromised and collude together such that the attacker knows the MO's position $p(\pi, l)$ of precision level l. As presented in our problem statement, an ideal position sharing approach will not allow an attacker knowing $p(\pi, l)$ to derive a position with a precision beyond the precision of $p(\pi, l)$.

Attacks on *PShare-BSP* and *PShare-BSP*^{opt}

PShare-BSP and *PShare-BSP*^{opt} both deterministically transform the MO's position π for a given value of l_{max} to position $p(\pi, 0)$. For each position $\pi' \in p(\pi, 0)$ it holds that the same position $p(\pi, 0)$ is calculated. This is guaranteed by mapping the l_{max} least significant bits of the x and y coordinate to zero, independent whether the bit was zero or one before. An attacker knowing $p(\pi, l)$ trying to increase his precision to level $l + 1$ would have to determine the first unknown bit of the x and the y coordinate of $p(\pi, l)$. Thus, the attacker could analyze the undefined bits of $p(\pi, l)$ and try to calculate the inverse function of the mapping to zero values. However, as values of zero and one are both mapped to zero, no further information is revealed to the attacker without knowing the corresponding r-share r_{l+1}. The same holds for an attacker analyzing the four possible refinement areas of $p(\pi, l)$ on level $l + 1$. All four areas are of the same size and share therefore the same probability to cover π. Thus, the probability of selecting the correct area on level $l + 1$ is equal to randomly guessing a certain area on level $l + 1$.

An attacker knowing m_π in *PShare-BSP*^{opt} also knows ciphers c_x and c_y. As

proven by Shannon [Sha49], the cipher of a one-time pad encryption provides no information about the protected secret even if the attacker has infinite computational power. Thus, it is not possible for an attacker to increase precision from $p(\pi, l)$ to $p(\pi, l+1)$ without knowing r-share r_{l+1} defining the corresponding parts of the r-keys to decrypt c_x and c_y.

In addition to analyze the undefined bits, an attacker can try to analyze the generated positions and apply the attacks presented in Section 2.6.2. Therefore, we analyze how *PShare-BSP* and *PShare-BSP*^{opt} resist these attacks.

An attacker can try to use the *region intersection attack*, which is a special method of the *multiple query attack* that can be successful on obfuscation-based approaches, if different obfuscation areas are generated for the same position π. However, we always deterministically generate the same obfuscation area for each position π and each level l. For the MO's value of l_{max} and two different positions π' and π'', either the obfuscation areas of level l are equal, i.e., $p(\pi', l) = p(\pi'', l)$, or the areas do not intersect each other, i.e., $p(\pi', l) \cap p(\pi'', l) = \emptyset$. In both cases, an attacker cannot refine $p(\pi, l)$ based on the region intersection attack.

The *probability distribution attack* calculates the probability that the MO is located in certain areas. It is most beneficial for probabilistic privacy algorithms that might lead to an uneven distribution of (possible) MO positions within the obfuscation area. For instance, the algorithm presented in [DSR11] leads to a concentration in the center of the obfuscation area. We try to avoid such concentrations and strive for a uniform distribution within the obfuscation area. Our deterministic share generation algorithm guarantees that each position $\pi' \in p(\pi, l)$ leads to the obfuscation area $p(\pi, l)$ with the same probability for a certain precision level l. Thus, running a Monte Carlo simulation for the deterministic obfuscation and share generation algorithm over $\pi' \in p(\pi, l)$ leads to a uniform distribution over $p(\pi, l)$. Therefore, probability distribution attacks do not provide any additional information to the attacker.

By design, we only generate and update new position shares if they are not vulnerable to the *maximum movement boundary attack* by using delayed position updates as a countermeasure. Thus, an attacker cannot increase his precision

by analyzing multiple positions using a maximum movement boundary attack.

A possible extension of *PShare-BSP* to resist *map matching attacks* could be based on the introduced idea of *PShare-GLM* taking map knowledge into account. Then, the refinement shares of *PShare-BSP* would increase precision to precision levels of different effective area sizes as presented for *PShare-GLM*.

3.2.5 Evaluation

Next, we analyze the computational efficiency of our approaches by measuring the runtime performance of share generation and combination using a prototype of our system. Then, we analyze the bandwidth efficiency of our approaches.

Runtime Performance Evaluation

Generally, the share generation is performed on a resource-constrained mobile device with limited CPU power and battery capacity. Even on such resource-poor devices, share generation must be possible in short time, which results in a small overhead in terms of energy. To show the performance of our approaches, we measured the overall time for share generation on a mobile device (HTC Desire HD, 1 GHz Qualcomm Snapdragon S2, 768 MB RAM). We measured the time to create one m-share and one to 15 r-shares and plotted the overall time over the number of r-shares in Figure 3.18. For reference values, we used the results presented in [DSR11] for the random share generation algorithm (denoted as *RSG*) and the results presented in Section 3.1.6 for *PShare-GLM*. We limited the number of generated r-shares to 15, because a precision of 32 768 m should be sufficiently coarse to provide location privacy for the MO. As we can see, the share generation of both *PShare-BSP* and *PShare-BSP*^{opt} stays well below one millisecond even when providing for *PShare-BSP*^{opt} $k = 184$ different r-keys within the r-shares. Thus, we can state that the share generation is highly efficient and suitable even for resource-poor devices. *PShare-BSP* decreases the computational complexity by one order of magnitude compared to [DSR11], and by three orders of magnitude compared to *PShare-GLM*.

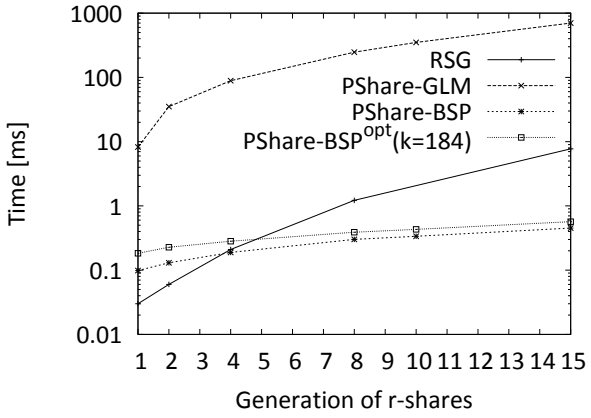


Figure 3.18: Performance evaluation of share generation

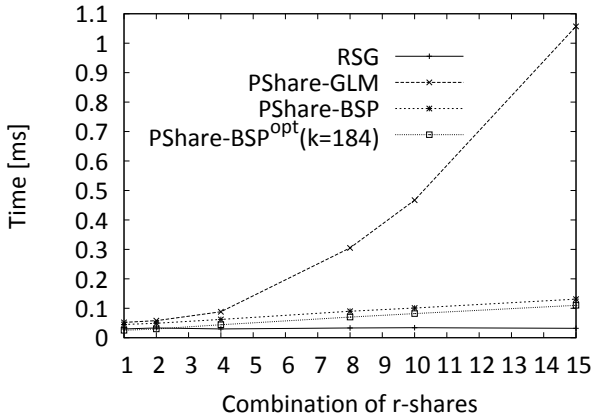


Figure 3.19: Performance evaluation of share combination

In contrast to share generation, the share combination is done by clients that typically run in the infrastructure with no energy restriction and high computational power. We measured the time required to combine one m-share with up to 15 r-shares on a personal computer (Intel Core 2 Duo, 2.53 GHz, 3 GB RAM). As we can see in Figure 3.19, share combination is calculated very efficiently in less than 150 microseconds even for a larger number of shares.

Bandwidth Efficiency

To analyze the efficiency of our approaches in terms of communication overhead, we compare the number of required update messages for *PShare-BSP* and *PShare-BSP*^{opt}. We assume that the MO performs a number of $k' \leq k$ position updates using $l_{max} + 1$ LSs to store the generated m-share and the l_{max} r-shares. Then, for *PShare-BSP*, the MO has to send a total of $k' \cdot (1 + l_{max})$ update messages, where each of the k' position updates triggers an update of the m-share and all l_{max} r-shares. For *PShare-BSP*^{opt}, we once update the m-share and all l_{max} r-shares, while for the next $k - 1$ updates only the m-share has to be updated. This results in a total number of $k' + l_{max}$ update messages. The r-shares are updated after $k' = k$ position updates were sent.

Next, we analyze the generated network load by analyzing the different share sizes and the overhead of lower level communication protocols. Each share has a share-ID (32 bits), a user-ID (32 bits), and a type definition (8 bits). In *PShare-BSP*, the m-share m_π adds position information (112 bits) and a list of IDs defining the r-shares of m_π ($l_{max} \cdot 32$ bits). An r-share r_π adds 2 bits for its refinement property. In *PShare-BSP*^{opt}, the m-share m_π^{opt} adds cipher c_x (l_{max} bits), cipher c_y (l_{max} bits), and the used r-key id (32 bits) to m_π. The additional payload of m_π^{opt} compared to m_π is denoted as Δm_π^{opt}. Each r-share r^{opt} consists of k (32 bits) and $2 \cdot k$ bits to reconstruct r-key$_x$ and r-key$_y$.

Thus, the network load of *PShare-BSP* for k' position updates is

$$\mathrm{NL}_{basic} = k' \cdot ((size(m_\pi) + o) + l_{max} \cdot (size(r_\pi) + o)) \text{ bits}, \tag{3.5}$$

where o defines the protocol overhead introduced by lower level protocols for each message. The network load of *PShare-BSP*^{opt} is

$$\mathrm{NL}_{opt} = k' \cdot (size(m_\pi^{opt}) + o) + l_{max} \cdot (size(r^{opt}) + o) \text{ bits}. \tag{3.6}$$

Comparing NL_{basic} and NL_{opt} leads to

$$k' \geq \frac{l_{max} \cdot (size(r^{opt}) + o)}{l_{max} \cdot (size(r_\pi) + o) - size(\Delta m_\pi^{opt})} \tag{3.7}$$

denoting the number of k' updates that have to be sent until *PShare-BSP*^{opt} outperforms *PShare-BSP* and $\mathrm{NL}_{opt} \leq \mathrm{NL}_{basic}$ holds. The size of the message overhead measured for sending a share over TCP/IP is $o = 320$ bits. For $l_{max} = 16$ generated r-shares and, for example, $k = 128$, this results in a value of $k' \geq 1.72$. This means that *PShare-BSP*^{opt} outperforms *PShare-BSP* as soon as the second position of the MO is updated. By using Equation 3.7, we can calculate that *PShare-BSP*^{opt} always outperforms *PShare-BSP* with the second update, as long as $k \leq 184$. For sending $k' = k = 184$ position updates, *PShare-BSP* generates a total network load of NL_{basic}= 172 040 bytes when also taking the TCP/IP overhead into account. *PShare-BSP*^{opt} at the same time only generates a load of NL_{opt}= 27 896 bytes. This results in a reduction of 83.8% of the generated network load. By considering the additional overhead required to provide secure channels by using, for instance, Transport Layer Security (TLS), the overhead of each message is further increased. Thus, reducing the number of messages by *PShare-BSP*^{opt} further increases its efficiency.

In addition to optimizing the updates of shares, i.e., the communication between MO and LSs, *PShare-BSP*^{opt} also optimizes the communication between the LSs and the clients. A client has to query all accessible r-shares only once within k updates instead of querying the r-shares every time a new position of the MO is updated.

3.3 Related Work

The most prominent location privacy concept is *k-anonymity* [KGMP07], which protects the user's identity. However, *k-anonymity* approaches and extensions such as *l-diversity* [BLPW08] or *t-closeness* [LLV07] usually require a trusted third party anonymizer, although the presented doubts exist that LSs can be

assumed to be fully trusted. In contrast, our position sharing approaches protect user privacy without the use of a trusted third party. Furthermore, our approaches protect the user's position information instead of the user's identity.

A simple approach to protect user position information is to store encrypted positions on the LS. This approach does not rely on any security mechanism of the LS. However, the LS cannot perform essential computations like nearest neighbor queries or range queries on the server side. In contrast, our position sharing approaches provide obfuscated position information to the LSs that can be used to answer position, range, and nearest neighbor queries.

Dummy approaches [KYS05] send the real user position together with several false positions to the LS. Advanced dummy approaches as presented by Shankar et al. [SGI09] make dummy identification more difficult using databases of collected position information. However, this leads to the problem of collecting user position information without raising privacy concerns, and to the problem of operating such a database without a trusted third party such that it cannot be manipulated. Furthermore, the provided privacy of these approaches is reduced if dummies can be identified. As shown by Peddinti and Saxena [PS11], this is possible even if dummies are generated by sophisticated algorithms.

Spatial obfuscation approaches [DK05, GDSB09, DSB11] provide user privacy by sending positions of degraded precision to the LS. Spatial obfuscation approaches generally do not require a trusted third party. Unfortunately, they limit the maximum allowed precision of mobile object positions that can be provided to clients of an LS by the trustworthiness of the LS. Consequently, clients cannot be provided with more precise positions than stored at any LS, no matter how trustworthy the clients are. This might have severe impact on clients, since usually the quality of an application degrades with the quality of the position information. Furthermore, an incremental precision increase for different clients with different quality of service demands and trust levels, as provided by our position sharing approaches, is not supported.

The *position sharing approach* introduced by Dürr et al. [DSR11] and its extension to maps presented by Skvortsov et al. [SDR12] do not require a trusted

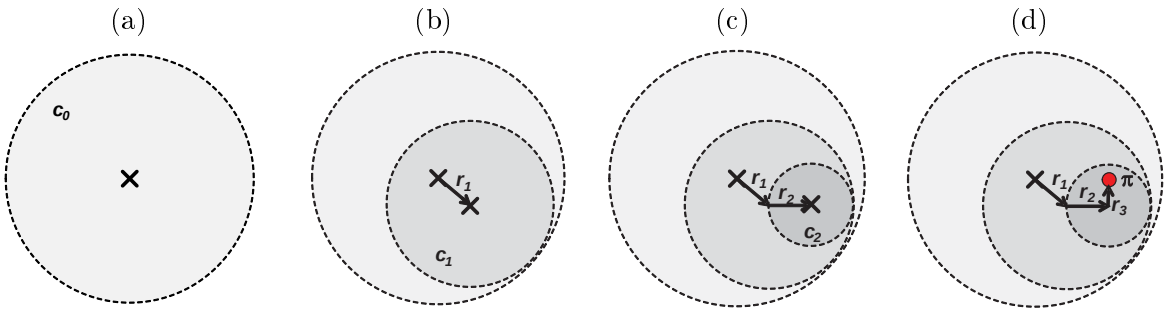


Figure 3.20: Position sharing based on vector addition

third party. Furthermore, they can provide different precision levels to different clients by combining position shares from different LSs based on random geometric transformations. In [DSR11], each LS stores the imprecise position of a mobile object, which is represented as a circle with a well-defined radius (cf. circle c_0 in Figure 3.20a). Additionally, each LS stores an individual shift vector that can increase the precision of the mobile object's position by shifting the center of the circle along the vector. At the same time, the radius of the circle is decreased. An example for the different refinement steps is shown in Figure 3.20b, Figure 3.20c, and Figure 3.20d, where the precision of the most imprecise position of circle c_0 is increased by using three refinement shares (denoted as r_1, r_2, and r_3). These refinement shares represent the shift vectors from different LSs that were queried by a client to obtain the precise position of the mobile object. In [SDR12], map information is considered to define possible mobile object positions instead of assuming a uniform probability distribution of the mobile object's position within the circles. Then, the circles and shift vectors are calculated according to the map information such that an attacker cannot use map matching attacks to increase precision.

Although the two approaches [DSR11] and [SDR12] provide sufficient privacy in many scenarios, an attacker can apply the location distribution attack presented in Section 2.6.2 by using Monte Carlo simulations to derive positions of higher precision than intended with a certain probability. In contrast, our position sharing approaches make such an attack impossible because of the de-

terministic obfuscation approach calculating obfuscated positions on different precision levels. This improves the robustness of our approaches significantly. Furthermore, our position sharing approaches do not just consider isolated position check-ins, but also succeeding position fixes of the same mobile object, which might unintentionally increase precision if updated in an uncontrolled manner.

While the position sharing approaches [DSR11] and [SDR12] apply probabilistic geometric transformations for obfuscation, our first position sharing approach presented in Section 3.1 is based on the concept of multi-secret sharing [CC05]. Using multi-secret sharing for position sharing allows to apply our approach not only to geometric positions (longitude, latitude values) but also to symbolic locations, such as cities, buildings, or restaurants, which are important for a wide range of applications.

Our second position sharing approach based on binary space partitioning improves the existing position sharing approaches by significantly increasing the communication efficiency. To this end, our position sharing approach implements an optimized share update protocol reducing the communication costs for position updates. The approaches [DSR11] and [SDR12] would have to update all shares for each position update, while our approach has to update all shares once. Afterwards, only a single share has to be updated.

The approach based on *(single) secret sharing* presented by Marias et al. [MDKG05] divides the position of the mobile object into several shares so that only a predefined number of shares can reconstruct the mobile object's position. This approach provides location privacy, however, it reveals the precise position of the mobile object to each client. In contrast, our position sharing approaches ensure *graceful degradation of privacy* instead of implementing an "all-or-nothing" approach. Therefore, clients can be assigned to different precision levels without revealing the precise position of the mobile object.

3.4 Conclusion

In this chapter, we focused on the protection of user positions when sharing location information with location services and clients. To protect user privacy, we presented two novel position sharing approaches protecting user position information against advanced attackers without assuming a trusted third party. Our first position sharing approach *PShare-GLM/SLM* makes use of the concept of multi-secret sharing to calculate position shares. Our second position sharing approach *PShare-BSP* is based on binary space partitioning. The basic idea of our position sharing approaches is to split up precise position information into position shares of limited precision, which are distributed to multiple location servers of different providers. Clients query position shares from different location servers to increase precision. Thus, the individual quality requirements of different clients can be satisfied.

Finally, we compare our position sharing approaches against each other. Especially, we consider the *flexibility*, the *robustness*, the computational *complexity*, and the *overhead* of each position sharing approach.

Flexibility: *PShare-GLM/SLM* supports geometric and symbolic location models, which demonstrates the versatility of the approach, while *PShare-BSP* supports only geometric locations. Furthermore, the number of shares that can be generated in *PShare-GLM/SLM* can be selected equal to or above the number of different precision levels that should be provided to clients, while the number of generated shares in *PShare-BSP* is predefined and limited based on the number of different precision levels. Additionally, *PShare-BSP* requires that clients query shares in a well-defined sequence from the corresponding LSs, while *PShare-GLM/SLM* provides the flexibility that clients can query the required number of shares from any set of available LSs storing the corresponding shares. Thus, we can state that *PShare-GLM/SLM* is more flexible than *PShare-BSP*.

Robustness: *PShare-GLM/SLM* and *PShare-BSP* both resist attackers using

probability distribution attacks, multiple query attacks, and maximum movement boundary attacks. Furthermore, both approaches do not require a trusted third party. Since *PShare-GLM/SLM* takes also map knowledge into account to resist map matching attacks, *PShare-GLM/SLM* provides a higher robustness compared to *PShare-BSP*. In addition, *PShare-GLM/SLM* allows to individually specify the number of shares that are required to reconstruct a position of a given precision level to increase robustness, while each share in *PShare-BSP* increases the precision by one level. Based on these properties, we can state that *PShare-GLM/SLM* provides a higher robustness than *PShare-BSP*

Complexity: Even if *PShare-GLM/SLM* and *PShare-BSP* can both generate position shares very fast, the complexity of *PShare-GLM/SLM* is higher than the complexity of *PShare-BSP* because of the multi-secret sharing scheme that has to be applied to generate the position shares. As shown in our performance evaluation in Section 3.2.5, the share generation and share combination of *PShare-BSP* outperform the share generation and share combination of *PShare-GLM/SLM*.

Overhead: Generally, position sharing approaches have the overhead of distributing position shares to multiple LSs for each position update. *PShare-GLM/SLM* requires to update the position shares for all LSs for each position update. As shown in our evaluation in Section 3.2.5, the update optimization of *PShare-BSP* significantly reduces the number of required update messages compared to an non-optimized position sharing approach like *PShare-GLM/SLM*. To this end, *PShare-BSP* initially updates the shares of all LSs, while afterwards only the share of one LS has to be updated. Therefore, *PShare-BSP* outperforms *PShare-GLM/SLM* with respect to the message overhead of the position sharing approaches.

As summarized in Table 3.2, *PShare-GLM/SLM* outperforms *PShare-BSP* considering the flexibility and the robustness of the approaches, while *PShare-BSP* has its advantages considering the complexity and the introduced overhead.

Property	*PShare-GLM/SLM*	*PShare-BSP*
Flexibility	✓	
Robustness	✓	
Complexity		✓
Overhead		✓

Table 3.2: Comparison of position sharing approaches

Therefore, users can apply the corresponding position sharing approach depending on whether a higher privacy or a higher efficiency is required.

4 Protecting Movement Trajectories

In this chapter, we address the problem of storing movement trajectories on non-trusted location servers. To protect movement trajectories, we introduce in Section 4.1 our novel trajectory fragmentation approach sharing the user's movement trajectory among multiple LSs of different providers. To this end, the user's movement trajectory is split up into a set of smaller fragments that are distributed to the LSs. An attacker successfully compromising an LS therefore gets only limited knowledge about the user's movement instead of his complete movement trace.

Our second approach focuses on the protection of private information that can be derived from movement trajectories, in more detail, the protection of *speed information*. Although LBAs are attracting a tremendous amount of users today, many users may still hesitate to use LBAs sharing movement trajectories as they are not willing to reveal, for instance, their driving behavior (e.g., whether the user is a safety conscious or an aggressive driver) or the occurrence of a speeding violation. Therefore, we introduce in Section 4.2 our novel speed protection algorithms protecting users from revealing violations of given speed limits when using LBAs. Afterwards, we conclude this chapter in Section 4.3.

4.1 Trajectory Fragmentation

Instead of providing the complete movement trajectory of a mobile object to a single LS, the idea of our trajectory fragmentation approach is to split up

the trajectory of the mobile object into a set of smaller fragments and to distribute the fragments among LSs of *different* providers. Thus, each LS stores and manages only a certain part of the trajectory and no LS knows the complete movement trace of the mobile object. Thus, an attacker compromising an LS has only limited knowledge about the mobile object's movement. To prevent an attacker compromising multiple LSs from correlating the retrieved fragments, our approach uses different pseudonyms for the calculated fragments. Different clients of the LSs, for instance, different LBAs access the trajectory of the mobile object by querying fragments from multiple LSs based on the known pseudonym used for the corresponding LS.

Our trajectory fragmentation approach provides the following contributions:

1. A concept for distributing trajectory fragments among a set of LSs to avoid that a single LS can reveal the complete movement trace of a mobile object.

2. An optimal algorithm that calculates a minimum number of fragments under the constraint of fulfilling a given level of privacy.

3. A privacy evaluation with attackers of different strength showing the robustness of our approach.

First, we introduce our extended system model, formalize our problem statement, and present our trajectory fragmentation algorithm in detail. Afterwards, we present our privacy analysis and our evaluation results before we compare our approach to related work.

4.1.1 Extended System Model

The components of our extended system model are depicted in Figure 4.1. The mobile object executes a local software component providing the MO's current position π to a set of different location servers, which store and manage trajectories of several MOs. LSs provide an access control mechanism to manage

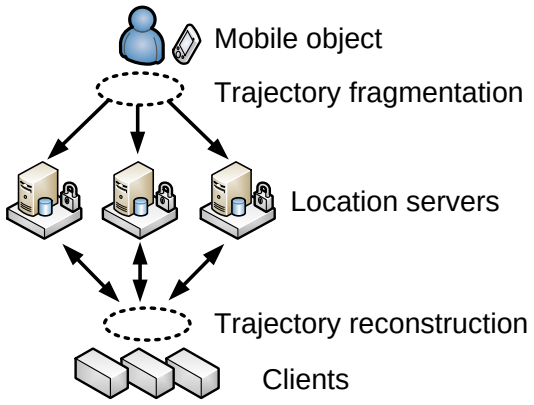


Figure 4.1: System components

access to the stored trajectories. Different clients get access to the trajectories on different LSs based on their provided access rights.

The MO's position π is defined by its longitude and latitude values. Since MOs usually travel on streets, we map π to a graph representing the road network by using existing map matching approaches. For an overview of existing map matching algorithms, we refer to the work of Quddus et al. [QON07]. As shown in Figure 4.2, the road network can be modeled as graph $G = (V, E)$ consisting of a set of nodes V and a set of edges E. Each node $v_i \in V$ represents a junction or an intermediate node that models the shape of the road. Each edge $e_j \in E$ represents a road segment between two nodes. The MO's movement trajectory $T = \{(\pi_{start}, t_{start}), \ldots, (\pi_{end}, t_{end})\}$ represents a set of succeeding position fixes π_i where the MO is located at time t_i. An example for trajectory T is shown in Figure 4.2. We assume that the destination π_{end} of trajectory T is already known at time t_{start} because the MO typically knows its movement destination.

4.1.2 Problem Statement

Since we have to consider that LSs are non-trusted, an attacker can successfully compromise an LS with a certain probability greater than zero. In this case, the part of the MO's movement trajectory that was provided to the LS is revealed.

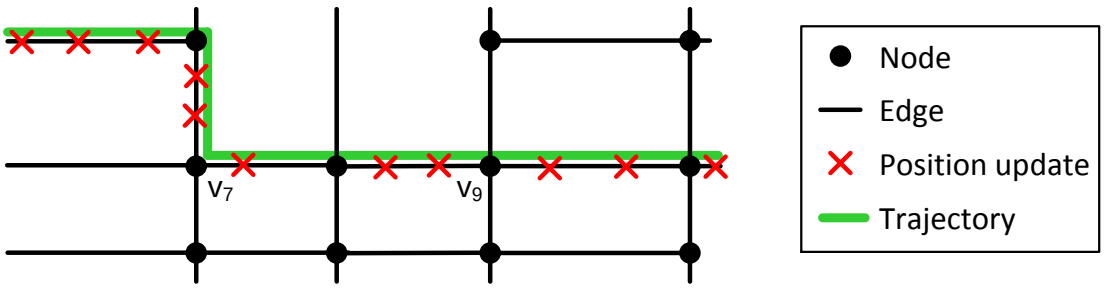


Figure 4.2: Road network graph and trajectory example

Therefore, our goal is to protect the MO's movement trajectory by minimizing the revealed information of an attack on an LS.

We use a distributed approach to store and manage the MO's movement trajectory T. Our trajectory fragmentation algorithm splits up T into a set $F = \{f_1, \ldots, f_n\}$ of n trajectory fragments (or *fragments* for short). The number n of generated fragments corresponds to the number of used LSs to store the fragments and is either predefined by the MO or calculated by the fragmentation algorithm. While traveling, the MO uses the fragmentation algorithm to update its position over time on different LSs. Each position is provided to exactly one LS, and each LS receives only the positions of a single fragment.

To measure the information exposed to an LS, we use weight function $\Phi(f_j)$ assigning each fragment f_j the value of its exposed information. More precisely, we measure the length of the trajectory that is revealed to an attacker, called the *revealed trace length* $\Phi(f_j)$. Nevertheless, also other information, for instance, the traveled time or the number of visited road segments of the MO could be used to measure the exposed information of an LS. We selected the measure of the revealed trace length $\Phi(f_j)$ since a MO is typically concerned about how long an LS can trace its movement. Furthermore, the MO may be concerned to reveal information where it came from or where it is going to. This information is also taken into account by the revealed trace length $\Phi(f_j)$.

Before presenting the details of how to calculate $\Phi(f_j)$, we define the length

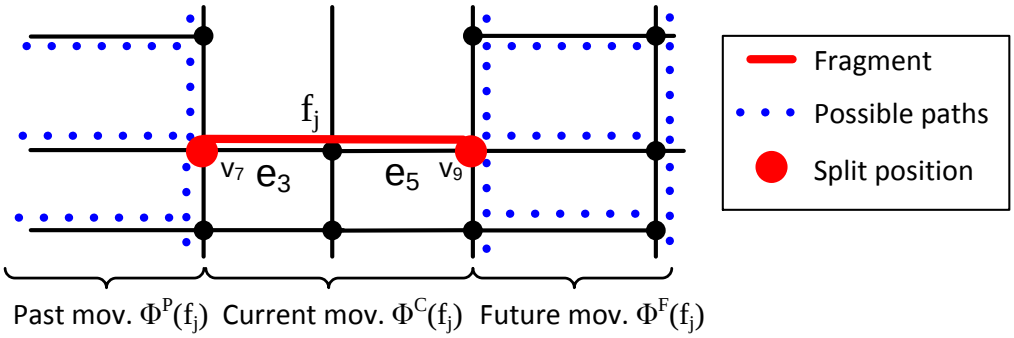


Figure 4.3: Different information parts

of fragment f_j as the length of the road segments where the MO is traveling while updating its position to the LS storing fragment f_j. Thus, the length of fragment f_j is the length of all edges $e_i \in f_j$ on the MO's movement trajectory T between two junctions splitting up trajectory T into different fragments.

The revealed trace length $\Phi(f_j)$ is defined as the distance an attacker compromising fragment f_j can trace the MO. The value of

$$\Phi(f_j) = \Phi^C(f_j) + \Phi^F(f_j) + \Phi^P(f_j)$$

is calculated by the MO as the sum of the following three parts (cf. Figure 4.3):

- $\Phi^C(f_j)$ is the length of the current fragment f_j.
- $\Phi^F(f_j)$ is the length of the predicted future movement. Since multiple possible future paths exist where the MO can continue traveling after fragment f_j, each road segment gets a probability assigned that the MO will continue traveling on the corresponding road segment. The probability that the MO travels along an edge e_i after traveling on f_j is $pr(e_i|f_j)$. The MO's future movement trajectory is predicted as the fastest path from fragment f_j to the MO's destination π_{end}. Then, $\Phi^F(f_j)$ is calculated as the length of all edges e_i on the MO's predicted future movement trajectory weighted by the probability $pr(e_i|f_j)$. We assume that the probability $pr(e_i|f_j)$ is

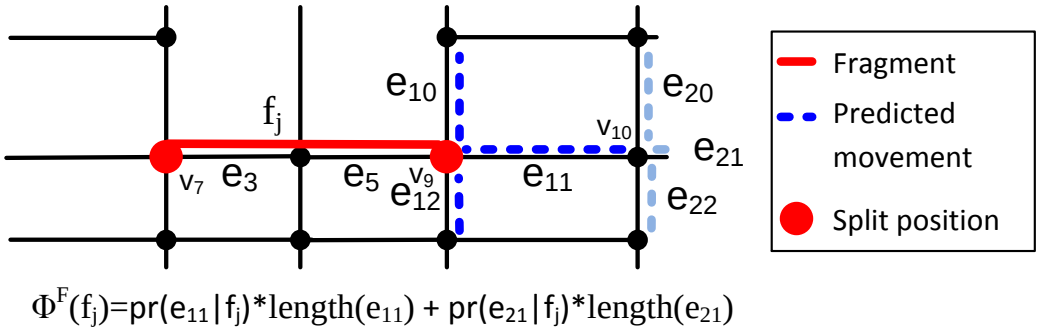


Figure 4.4: Calculation of $\Phi^F(f_j)$

given for each edge e_i, for instance, by a statistical movement analysis, or by using map knowledge assuming a uniform distribution for all edges. The uniform distribution is used, for instance, by Krumm [Kru08] to predict the MO's future movement. An example to calculate $\Phi^F(f_j)$ based on map knowledge is shown in Figure 4.4. Here, the probability $pr(e_{11}|f_j)$ is 33.33%, since three alternatives exist to continue traveling after f_j. The probability $pr(e_{21}|f_j)$ in Figure 4.4 is 11.11% taking also $pr(e_{11}|f_j)$ into account.

- $\Phi^P(f_j)$ is the length of the reconstructed past movement. The past movement describes where the MO came from when reaching fragment f_j. It is calculated as the fastest path from the MO's starting position π_{start} to fragment f_j. $\Phi^P(f_j)$ is calculated analogously to $\Phi^F(f_j)$ by weighting the length of each road segment of the MO's past movement by the probability that the MO travels on the corresponding road segment before reaching fragment f_j.

In order to find a fragmentation of trajectory T such that each fragment has a minimum revealed trace length $\Phi(f_j)$, we have to minimize the maximum value of $\Phi(f_j)$. This means, no matter which LS is compromised, only the maximum trace length $\Phi(f_j)$ can be revealed. Formally, we have to solve the following

optimization problem:

$$\begin{aligned} &\text{minimize} && \max_{j\in[1;n]}(\Phi(f_j)) \\ &\text{subject to} && \bigcup_{j\in[1;n]} f_j = \mathrm{T} \end{aligned}$$

Additionally, we want to answer the question of how many LSs are required to protect trajectory T from revealing a larger trace length than specified by a MO-defined maximum trace length Φ_{max}^{MO}. This means, we consider the problem of minimizing the number n of required LSs for Φ_{max}^{MO}. Formally, this optimization problem is defined as

$$\begin{aligned} &\text{minimize} && n \\ &\text{subject to} && \max_{j\in[1;n]}(\Phi(f_j)) \leq \Phi_{max}^{MO} \\ & && \bigcup_{j\in[1;n]} f_j = \mathrm{T} \end{aligned}$$

4.1.3 Trajectory Fragmentation Algorithms

In this section, we present our trajectory fragmentation algorithms (*TFA* for short). We start with an overview of our approach and present the individual steps of *TFA* afterwards.

Overview

The concept of *TFA* consists of two parts:

1. The *trajectory fragmentation* performed by the MO.
2. The *trajectory reconstruction* performed by clients.

The trajectory fragmentation first calculates the MO's predicted trajectory T based on the given destination. Secondly, it splits up T into a set of fragments. Thirdly, each fragment is assigned to an LS. While traveling, the MO updates

its position on the LSs based on the calculated fragmentation. If the predicted trajectory deviates from the real trajectory where the MO is currently traveling, a new fragmentation is initiated using a new set of LSs.

The process of *TFA* is shown in Algorithm 12. Since MOs usually travel on fastest paths to reach their destination, we predict the MO's trajectory T at time t_{start} as the fastest path from the current position π_{start} to the known destination π_{end}. Then, we split up T into set $F = \{f_1, \ldots, f_n\}$ of n fragments, one for each LS, by using function

$$fragment(T, n) = \{f_1, \ldots, f_n\},$$

which is further introduced below. For each fragment $f_j \in F$, we calculate a new random pseudonym of the MO and select an LS to store f_j. To calculate the pseudonyms, also other approaches can be applied. For example, Rass et al. [RFSK08] derive pseudonyms for different trajectories and position fixes to anonymize floating car data and to hide the identity of a driver. Furthermore, Jorns et al. [JQJ07] provide different pseudonyms using transaction pseudonyms to preserve user privacy in LBAs. The LS storing the positions of fragment f_j is denoted as LS_j. We use the notation $\pi_i \in f_j$ to denote that position π_i is part of f_j. As formalized in our problem statement, the goal of function *fragment*(T, n) is to minimize the maximum revealed trace length $\Phi(f_j)$ for each fragment. To solve the presented optimization problem, we use a dynamic programming approach presented below.

After calculating fragmentation F, the MO updates its position $\pi_i \in f_j$ to LS_j while traveling on f_j. As soon as a new sensed position is part of f_{j+1}, the MO changes the LS from LS_j to LS_{j+1}. Furthermore, the used pseudonym is changed from id_j to id_{j+1}. In case the MO leaves the predicted trajectory T at position $\pi_k \notin T$, a new calculation of *TFA* is initiated for the new initial position $\pi_{start} = \pi_k$ and the destination π_{end} using a new set of LSs.

The trajectory reconstruction allows different clients to access the MO's real trajectory T by querying the LSs using the provided pseudonyms of the MO.

Algorithm 12 TFA: Process overview

Function: $TFA(n, \pi_{end})$

1: $\pi_{start} \leftarrow$ sensed position ▷ Initial positioning
2: $T \leftarrow FP(\pi_{start}, \pi_{end})$ ▷ Calculate fastest path
3: $F[1, \ldots, n] \leftarrow fragment(T, n)$ ▷ Fragmentation
4: $ID[1, \ldots, n] \leftarrow getIDs(F)$ ▷ Calculate pseudonyms for fragments
5: $\pi_i \leftarrow \pi_{start}$
6: **while** $\pi_i \in T$ **do**
7: $f_j \leftarrow getFragment(\pi_i)$ ▷ Map position to fragment
8: $LS_j \leftarrow getLS(f_j)$ ▷ Map fragment to LS
9: $id_j \leftarrow getID(f_j)$ ▷ Get calculated pseudonym
10: $update(\pi_i, LS_j, id_j)$ ▷ Update π_i to LS_j using pseudonym id_j
11: $\pi_i \leftarrow$ sensed position ▷ Positioning
12: **end while**

That is, LSs grant clients access to the stored position information of the assigned trajectory fragments based on the MO-defined access rights. For example, clients can access the complete movement trajectory T or only certain parts of T while the MO travels on the motorway or within a certain geographical area.

Trajectory Fragmentation

Next, we present function $fragment(T, n)$ calculating set $F = \{f_1, \ldots, f_n\}$ of fragments in Algorithm 13. As introduced, each fragment $f_j \in F$ defines to which LS_j position $\pi_i \in f_j$ should be sent. The part of trajectory T belonging to fragment f_j is the part of T between two *split positions*. For example, fragment f_j in Figure 4.3 is defined by the split positions of junction v_7 and junction v_9. We split up the MO's predicted trajectory T at nodes representing junctions instead of splitting up T within an edge or at an intermediate node. This approach has the advantage that all positions belonging to the same edge e_i are assigned to the same fragment which is stored by only one LS. Furthermore, we assume for our description that trajectory T starts and ends at a node representing a junction.

Algorithm 13 TFA: fragmentation

Function: $fragment(T, n)$

1: $G_T \leftarrow getGraph(T)$ ▷ Calculate fragmentation graph G_T
2: $M \leftarrow getAdjacencyMatrix(G_T)$ ▷ Calculate adjacency matrix M
3: $L \leftarrow n$ ▷ Set number of LSs
4: $M^L \leftarrow maxMatrixMult(M, L)$
5: $\Phi_{max} \leftarrow M^L[0, m-1]$ ▷ Store maximum value of Φ
6: $G_T^{max} \leftarrow removeEdges(G_T, \Phi_{max})$ ▷ Delete edges in G_T
7: $M_{max} \leftarrow getAdjacencyMatrix(G_T^{max})$ ▷ Calculate adjacency matrix M_{max}
8: $S_T \leftarrow getRandomPath(M_{max}, n, v_0, v_{m-1})$ ▷ Calc. random path of length n
9: $F[1, \ldots, n] \leftarrow getFragments(S_T)$ ▷ Calculate fragments for split nodes
10: **return** $F[1, \ldots, n]$

The problem is now how to find the split positions for trajectory T such that the corresponding set of fragments F is optimal considering the maximum revealed trace length $\Phi(f_j)$. To solve the proposed optimization problem, we first calculate the *fragmentation graph* $G_T = (V_T, E_T)$. The set of nodes $V_T = \{v_i \in T\}$ consists of all possible split positions of T, i.e., the set of nodes representing a junction on T. The set of edges E_T is generated by calculating for each node $v_i \in V_T$ an edge to each node $v_j \in V_T$ if the MO will visit the junction of v_i before visiting the junction of v_j. The generated edge from v_i to v_j is denoted as $e_{ij} \in E_T$. The weight of e_{ij} is $\Phi(f(v_i, v_j))$ representing the revealed trace length of the fragment that is defined by the split position v_i and v_j. Then, we calculate the adjacency matrix M of G_T, which is of size $m \times m$ with $m = |V_T|$. Each value $M[i; j] = \Phi(f(v_i, v_j))$ defines the weight of edge e_{ij}. Since we aim for an optimal fragmentation using n LSs, we have to find a path in G_T consisting of n edges from the start node $v_0 \in V_T$ of T to the destination node $v_{m-1} \in V_T$ of T minimizing the maximum edge weight. The L-th power of the adjacency matrix M is denoted as M^L and calculated using matrix multiplication. For each possible node $v_k \in V_T$, we calculate the maximum value of $M^{L-1}[i; k]$ and $M^1[k; j]$. Then, we select the minimum value from all possible nodes v_k

and store the determined maximum value in $M^L[i;j]$. Thus, $M^L[i;j]$ is the minimized maximum revealed trace length of a single fragment on the path of length L that leads from v_i to v_j. The maximum value for a path of length n from node v_0 to v_{m-1} for $L = n$ is the value of $M^L[0, m-1]$. This value is then stored in Φ_{max} and used to remove all edges in G_T with a higher value than Φ_{max}. The corresponding adjacency matrix after deleting these edges is M_{max}, which represents the former fragmentation graph G_T containing only edges with a weight up to Φ_{max}. Based on M_{max}, a random path of length n from v_0 to v_{m-1} is calculated. The selected nodes are stored in the set S_T of split nodes. The nodes of the calculated path represent the split positions for an optimal fragmentation. Finally, we determine set F of fragments using the calculated split positions of set S_T.

Minimizing the Number of Required LSs

After solving the problem of how to find an optimal fragmentation for trajectory T, we consider now the problem of minimizing the number n of required LSs to achieve a MO-defined maximum revealed trace length Φ_{max}^{MO}. To solve this problem, we adapt Algorithm 13 as follows (cf. Algorithm 14). Instead of using a fixed value of $L = n$ to calculate Φ_{max}, we stepwise increment L. As soon as $M^L[0, m-1] \leq \Phi_{max}^{MO}$, the minimum path length L and thus the minimum number of required LSs is found that can provide a maximum value of Φ_{max}^{MO}. After setting Φ_{max} to Φ_{max}^{MO}, we calculate a random path of length L based on M_{max} and calculate the corresponding fragmentation. If L reaches a value above L_{max}, which represents the maximum number of available LSs, no solution could be found for trajectory T and Φ_{max}^{MO}. Then, the MO has either to adjust Φ_{max}^{MO} or to use Algorithm 13 to find an optimal fragmentation using $n = L_{max}$ LSs.

4.1.4 Privacy Analysis

In this section, we evaluate the provided privacy of our trajectory fragmentation approach. First, we introduce our attacker model followed by our privacy metric.

Algorithm 14 TFA: fragmentation minimizing the number of required LSs

Function: $fragmentMinLS(T, \Phi_{max}^{MO})$

1: $G_T \leftarrow getGraph(T)$ ▷ Calculate fragmentation graph G_T
2: $M \leftarrow getAdjacencyMatrix(G_T)$ ▷ Calculate adjacency matrix M
3: $L \leftarrow 1$
4: **while** $(M^L[0, m-1] > \Phi_{max}^{MO})$ and $(L \leq L_{max})$ **do** ▷ Incremental search
5: $M_L \leftarrow maxMatrixMult(M, L)$
6: L++ ▷ Stepwise increment L
7: **end while**
8: $\Phi_{max} \leftarrow \Phi_{max}^{MO}$
9: $G_T^{max} \leftarrow removeEdges(G_T, \Phi_{max})$ ▷ Delete edges in G_T
10: $M_{max} \leftarrow getAdjacencyMatrix(G_T^{max})$ ▷ Calculate adjacency matrix M_{max}
11: $S_T \leftarrow getRandomPath(M_{max}, L, v_0, v_{m-1})$ ▷ Calc. random path of length L
12: $F[1, \ldots, L] \leftarrow getFragments(S_T)$ ▷ Calculate fragments for split nodes
13: **return** $F[1, \ldots, L]$

Attacker Model

Nowadays, map knowledge is widely available, for instance, provided by the OpenStreetMap project [Ope14]. Therefore, we assume that an attacker has map knowledge and knows the used fragmentation algorithm. In case an attacker compromises a client, all positions of the MO's movement trajectory provided to the client are revealed. To limit the revealed information of a client, the MO can individually specify which part of the trajectory should be accessible for each client by defining access rights on the LSs. Because the access control mechanisms do not prevent that the stored information of an LS is revealed to an attacker compromising an LS, we further consider in our evaluation that an attacker compromises a single or multiple LSs. If an attacker compromises an LS, all positions assigned to the stored fragment of the MO are revealed.

The goal of an attacker is to derive as much information as possible from his known positions. Therefore, we consider attackers using movement prediction methods to predict and reconstruct the MO's movement trajectory. More

precisely, we use the first order Markov model presented by Krumm in [Kru08] to simulate attackers using different turn probability estimations. First, we consider attacker A^{NC} that cannot correlate fragments that were provided to different LSs using different pseudonyms. Secondly, we consider attacker A^{AC} that can correlate adjacent fragments based on their spatio-temporal properties. That is, attacker A^{AC} analyses the positions of two fragments and merges both fragments if the analyzed positions belong to two adjacent edges on the road network graph.

In addition to the ability of correlating fragments, we distinguish two attackers which are of different strength based on their known information. The first attacker A_{MAP} uses map knowledge to determine the MO's trajectory from his known positions based on the introduced map matching and maximum movement boundary attack. To this end, attacker A_{MAP} estimates at each junction a uniform probability distribution where the MO probably came from or where the MO is probably going to. For instance, if three alternatives exist at a junction where the MO can continue traveling, attacker A_{MAP} assigns each edge the probability of 33.33%. Then, attacker A_{MAP} predicts and reconstructs the MO's trajectory by selecting step by step adjoining edges to his known fragments based on the calculated probability distribution. The second attacker A_{SMI} uses statistical movement information gained from a road network traffic analysis from trajectories of other MOs. The goal of attacker A_{SMI} is to improve his prediction by using more accurate turn probabilities. Considering the attackers of our attacker classification presented in Section 2.6.2, attacker A_{SMI} improves his attack based on additional personal context knowledge of different MOs.

Privacy Metric

To measure the provided privacy of fragmentation F, we analyze the maximum revealed trace length Φ^{A}_{max} an attacker A compromising different LSs can derive. For an attacker who is able to correlate adjacent fragments even if different pseudonyms are used, the maximum revealed trace length Φ^{A}_{max} is in the worst case equal to the length of the complete movement trajectory of the MO. There-

fore, we analyze also the probability that the maximum revealed trace length Φ^A_{max} is above a threshold length θ. To this end, we calculate for a given length θ the cumulated probability $Pr(\theta < \Phi^A_{max})$ as follows:

Let F_C be the set of all possible combinations of compromised fragments for fragmentation F. Furthermore, let $F^\theta_C = \{c_i \in F_C | \theta < \Phi^A_{max}\}$ be the set of all possible combinations $c_i \in F_C$ fulfilling that the maximum revealed trace length for attacker A is above length θ. The probability that attacker A compromises exactly the k fragments of $c_i \in F^\theta_C$ is

$$\alpha(c_i) = p^k * (1-p)^{n-k}, \tag{4.1}$$

where $p \in (0, 1]$ is the probability that attacker A compromises a single LS. Then, $Pr(\theta < \Phi^A_{max})$ is calculated as

$$Pr(\theta < \Phi^A_{max}) = \sum_{c_i \in F^\theta_C} \alpha(c_i). \tag{4.2}$$

Privacy Evaluation

We analyze the success of different attackers in predicting and reconstructing the MO's movement trajectory based on the compromised fragments by using movement prediction methods. In our evaluation, we use the real world dataset provided by Piorkowski et al. [PSDG09], which consists of the traces of about 500 taxi cabs collected over 30 days in 2008 in San Francisco, USA. The position of each taxi cab is measured by its GPS receiver and updated within an update time of less than 10 s on average [PSDG09]. The used map information is derived from the OpenStreetMap project [Ope14]. The turn probabilities of attacker A_{MAP} consider a uniform distribution for all possible paths at each junction. For attacker A_{SMI}, we performed a road network traffic analysis where we analyzed the movement behavior of all taxis of [PSDG09] for a complete day (2008/06/01) and derived for each junction the corresponding turn probability distribution. We selected for our evaluation from the next day (2008/06/02) a

short trajectory of 4.52 km length within the city (denoted as *city*) and a long trajectory of 17.04 km length mainly using highways (denoted as *highway*). We assume a probability of $p = 10\%$ that a single LS is compromised and calculate the probability that multiple LSs are compromised according to Equation 4.1. Next, we evaluate attacker A^{NC} and continue afterwards with attacker A^{AC}.

Attacker $\mathbf{A^{NC}}$: Since attacker A^{NC} does not correlate fragments, the case that A^{NC} compromises multiple LSs is identical to the case that A^{NC} compromises a single LS. To show the success of protecting movement trajectories against attacker A^{NC} when using fragmentation, we measure the maximum revealed trace length Φ^{A}_{max} that is revealed to attacker A^{NC}_{MAP} and A^{NC}_{SMI} for the introduced trajectories. The results are shown in Figure 4.5 for the city trajectory and in Figure 4.6 for the highway trajectory. For the city trajectory, Φ^{A}_{max} of A^{NC}_{MAP} decreases to 523 m (11.56%) when using up to 15 LSs. By considering statistical movement information in addition to map knowledge, Φ^{A}_{max} of A^{NC}_{SMI} decreases to 1157 m (25.58%). For the highway trajectory, Φ^{A}_{max} decreases to 5591 m (32.80%) for attacker A^{NC}_{MAP} respectively 9199 m (53.97%) for attacker A^{NC}_{SMI}. As we can see, each trajectory has a limiting value of Φ^{A}_{max} such that even if the number n of used LSs is increased, the maximum revealed trace length known to the attacker does not decrease further. This is based on the fact that the maximum revealed trace length Φ^{A}_{max} cannot decrease below the maximum revealed trace length of a single road segment on the considered trajectory.

By comparing the relative values of Φ^{A}_{max} from the highway trajectory with the city trajectory, A^{NC}_{SMI} receives a higher value of Φ^{A}_{max} for the highway trajectory. This is based on the fact that the prediction of A^{NC}_{SMI} works well on the highway due to a high statistical probability that the MO stays on the highway for longer times. For the city trajectory, many junctions exist with approximately equal turn probabilities for alternative roads such that the attacker cannot precisely predict the MO's movement.

Next, we analyze the minimum number of LSs that is required to reveal

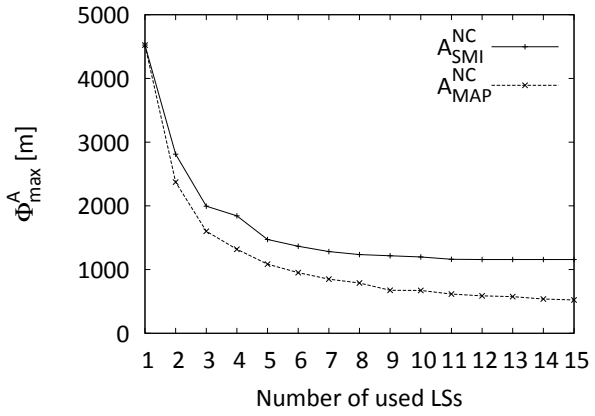


Figure 4.5: City trajectory evaluation

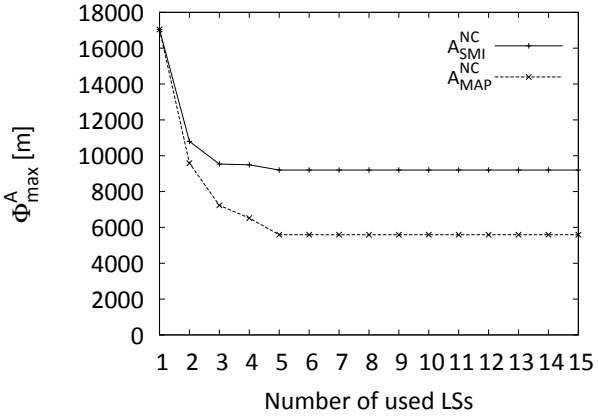


Figure 4.6: Highway trajectory evaluation

at most a trace length of Φ^{MO}_{max} to a single LS. Figure 4.7 and Figure 4.8 show which values of Φ^{MO}_{max} can be provided using the map based and the statistical movement based fragmentation for the city and the highway trajectory. By decreasing Φ^{MO}_{max} from the maximum length of the considered trajectory, the minimum number n of required LSs to provide Φ^{MO}_{max} increases. Again, each trajectory has a limiting minimum value of Φ^{MO}_{max} such that smaller values of Φ^{MO}_{max} cannot be provided even when increasing n as presented before.

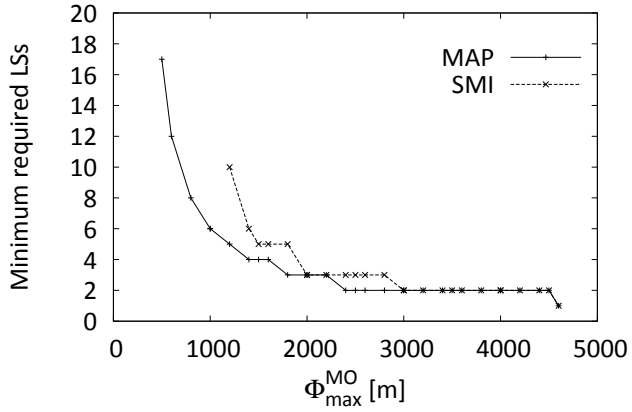


Figure 4.7: Minimum number n of required LSs for the city trajectory

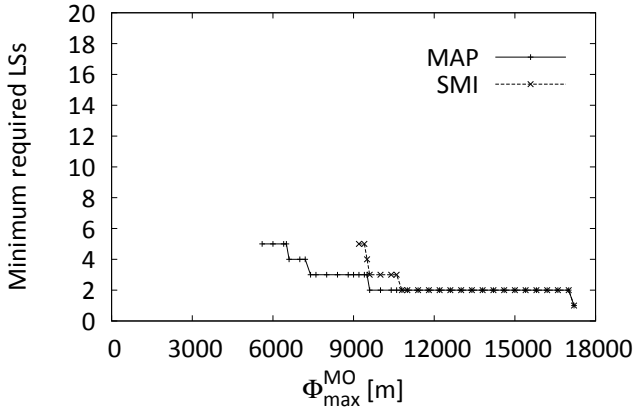


Figure 4.8: Minimum number n of required LSs for the highway trajectory

To show that attacker A^{NC} cannot derive a higher revealed trace length than specified by the MO in Φ^{MO}_{max}, we analyze for the city trajectory the cumulated probability distribution $Pr(\theta < \Phi^{A}_{max})$ using the fragmentation generated by optimizing n. We select a value of $\Phi^{MO}_{max} = 1.5\,\text{km}$, which leads to $n = 4$ LSs for the map based fragmentation and $n = 5$ for the statistical movement based fragmentation. As shown in Figure 4.9, the probability that attacker A^{NC}_{MAP} and attacker A^{NC}_{SMI} can derive a maximum revealed trace length above Φ^{MO}_{max} drops to zero such that the privacy requirement

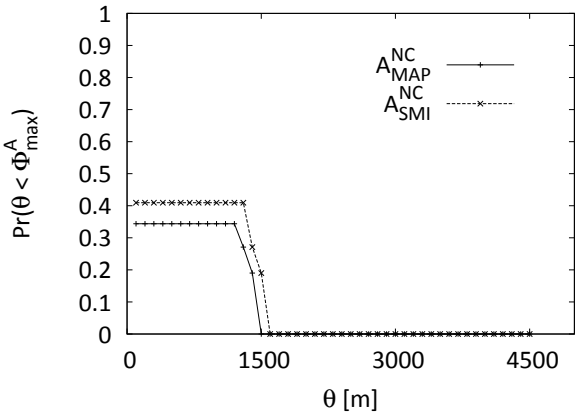


Figure 4.9: Cumulated probability distribution of A^{NC} for the city trajectory

is fulfilled and *TFA* prevents that an attacker can derive a higher revealed trace length than Φ_{max}^{MO}. Altogether, we can state that *TFA* effectively prevents that attacker A^{NC} can trace the MO over longer distances.

Attacker $\mathbf{A^{AC}}$: The powerful attacker A^{AC} can correlate adjacent fragments based on the spatio-temporal properties of the compromised positions. If attacker A^{AC} can compromise all used LSs, the complete movement trajectory of the MO is revealed. This results in a maximum revealed trace length of Φ_{max}^{A} equal to the length of the trajectory. To better understand this kind of strong attacker, we show the success of attacker A^{AC} to trace the movement of the MO over a certain length θ by analyzing the cumulated probability that attacker A^{AC} receives a maximum revealed trace length above θ by compromising a certain set of LSs. Figure 4.10 shows for the city trajectory the cumulated probability that attacker A_{MAP}^{AC} can derive a maximum value Φ_{max}^{A} above θ for different values of n used LSs. As we can see, increasing the number of generated fragments increases the probability that a small part of the trajectory is revealed, whereas the probability that longer parts of the trajectory are revealed decreases. For instance, attacker A_{MAP}^{AC} can trace the MO for a value of $\theta = 1\,\text{km}$, which represents 22.10% of the trajectory, only with a probability of 2.61% when

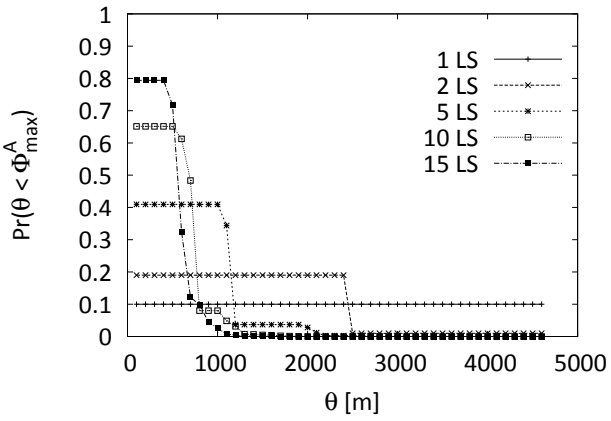


Figure 4.10: Cumulated probability distribution of A^{AC}_{MAP} for the city trajectory

using 15 LSs. Therefore, we can state that our trajectory fragmentation algorithms can prevent attacker A^{AC} from tracing the MO over longer distances with a high probability.

4.1.5 Evaluation

In this section, we present our runtime performance evaluation for our trajectory fragmentation algorithm. We evaluate the performance of *TFA* by measuring its runtime on a mobile device (HTC Desire HD, 1 GHz Qualcomm Snapdragon S2, 768 MB RAM). We measure the required time to calculate the corresponding fragmentation for the city and the highway trajectory using n LSs. The city trajectory is of 4.52 km length and passes 50 junctions. The highway trajectory is of 17.04 km length and passes 57 junctions. As shown in Figure 4.11, the calculation time of *TFA* stays below four seconds for the city trajectory and below five seconds for the highway trajectory even for a larger number of calculated fragments. The calculation time for the highway trajectory is higher than for the city trajectory because of the higher number of passed junctions. The runtime of *TFA* optimizing the number of required LSs for a value of $\Phi^{MO}_{max} = 1.5\,\mathrm{km}$ for the city trajectory stays below 1.0 s while a value of $\Phi^{MO}_{max} = 6.0\,\mathrm{km}$ for the highway trajectory leads to a calculation time below 1.6 s. Recall that the fragmentation

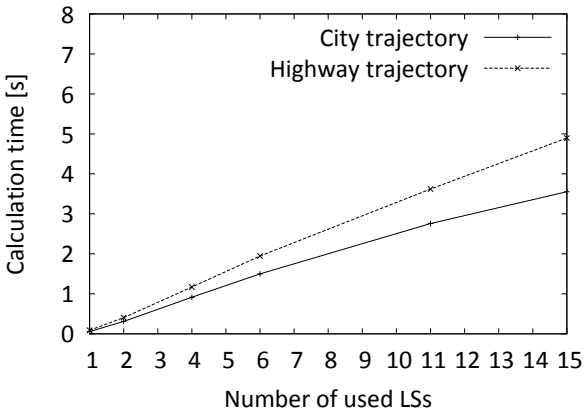


Figure 4.11: Runtime performance evaluation

is only calculated initially at the start of the MO's movement or if the MO leaves the predicted trajectory. While traveling, *TFA* only performs a simple lookup of the current position against the corresponding fragment before sending the position to the LS. Therefore, we can state that *TFA* supports real-time position updates and that the fragmentation time is reasonable.

4.1.6 Related Work

Trajectory k-anonymity [NASG09] protects the identity of a mobile object by making the mobile object indistinguishable from $k-1$ other objects for the complete trajectory. *Mix zones* [BS03] protect the identity of a mobile object by changing pseudonyms within an area where no mobile object provides position information to a client. *Trajectory obfuscation* [CM07] uses a group-based approach to collect positions of multiple objects to calculate obfuscated areas for the complete trajectory. However, all these approaches require a trusted third party, while we do not rely on a trusted third party.

Dummy approaches [YPL07, SGI09] provide false trajectories to the LS in addition to the real trajectory of the mobile object such that the LS cannot distinguish the received trajectories. However, since identifying dummy trajectories is possible as shown in [PS11], the provided privacy can be decreased.

The approaches presented by Ardagna et al. [XZZ+13] and Xue et al. [ALS12] protect the start and the destination of movement trajectories without assuming a trusted third party. To this end, Ardagna et al. [XZZ+13] rely on a so-called end-point generation method calculating position information of a trajectory that have to be deleted such that an attacker cannot derive the trajectory's destination with a probability above a certain threshold value. Instead of suppressing location information, Xue et al. [ALS12] protect the destination of a trajectory by publishing a cover story for the trajectory. Both approaches rely on a first order Markov model to predict possible trajectory destinations. With our approach, we focus on protecting the complete movement trajectory rather than only its start or destination. Finally, Yigitoglu et al. [YDAS12] protect sensitive positions in trajectories instead of protecting the complete trajectory.

4.2 Speed Protection

After showing in the previous section how movement trajectories can be protected, we focus in this section on the protection of private information that can be derived from movement traces, in more detail, the protection of *speed information*. Although the user is typically aware of the fact that LBAs, as the ones presented in Section 2.1.2, collect movement trajectories, he is seldom aware that this information can be used to derive further information beyond geographical positions. In particular, movement trajectories consisting of positions and timestamps can be used to calculate the speed of the user. Although this speed information is mandatory for many applications like real-time traffic monitoring—e.g., to detect traffic jams—, the user might involuntarily reveal information about his behavior that he is not willing to share, in particular, information about when and where he might have driven too fast.

At this point, we have to make clear that in the case of violations of speed limits, the protection of information is ambivalent. On the one hand, the protection of private information—in particular, location information—is a commonly accepted goal. On the other hand, it should be clear to everybody that speed

limits are there to protect people, and therefore, monitoring speed information is an important measure for law enforcement. Our decision to design mechanisms to protect speed information is based on the commonly accepted principle that everybody should be able to control which information about him is revealed to someone else. In other words: If the user is aware that the collected information might be used for detecting speeding violations—for instance, as part of a pay-as-you-drive insurance with special rates for safety-conscious drivers, or using a tachograph for trucks—, no protection mechanisms are necessary. On the contrary, the insurance or police might want to ensure that the driver does not manipulate the speed information using tamper-proof devices (which is a different research topic on its own right). However, if the driver does not explicitly agree on accurately monitoring his speed, our mechanisms will make sure that no information can be recorded that might later be used against him.

This is a very important prerequisite for ensuring the acceptance of LBAs based on movement trajectories and time information. Clearly, although everyone would assume to obey speed limits in general, the possibility to detect violations will deter users from participating in such applications like automatic traffic jam detection. Even if the recorded information cannot be used by the police due to legal restrictions, private companies like car insurances might use information found on the Internet (e.g., in OpenStreetMap GPS traces) to screen their customers and adjust rates. This poses a serious psychological barrier in providing unfiltered trajectory information to LBAs.

Various cases from the past have shown that speed information is indeed used without knowledge of the users. For instance, in 2001, a car rental company in the United States fined customers for speeding violations using GPS-equipped cars [Pre01]. One customer was billed $150 for each of his alleged speeding violations where the trace showed a speed faster than 79 miles per hour. For tracking, the company installed GPS-devices in their cars. Nowadays, sensing and tracking technology of mobile devices and car navigation systems can be used to track users. For instance, new navigation systems provide real-time speed and location data to servers calculating real-time traffic conditions, which

is clearly a service that many drivers find useful and would actively support. Or, consider the Waze application introduced in Section 2.1.2, which analyzes the speed information of users based on the positioning sensors provided in current smartphones to calculate information about the current traffic situation.

However, in 2010, a company providing such real-time traffic services sold their collected GPS records to the Dutch police, which used the data to target speed traps where they could catch most drivers [Wil11]. Immediately, the company stressed that they only stored anonymous data such that individual speeders could not be identified by the police. However, as shown in [HGXA06], user identification from anonymized trajectories is possible if, for instance, an anonymized trajectory starts in front of an individual home. Therefore, identifying individual speeders would be possible.

As we can see from these examples, publishing user trajectories without protecting the speed information can have severe monetary and legal effects on the user if speeding violations can be revealed, as well as for the acceptance of LBAs based on trajectory information. To remove these concerns and, therefore, to support the acceptance of LBAs, we present speed protection algorithms protecting the speed information of a user trajectory in real-time by adjusting the reported trajectory to the allowed speed limit such that users do not have to fear any negative impact due to speeding violations. To protect trajectories from indicating speedings, we either adjust temporal information by delaying position updates or adjust spatial information of positions. In our evaluation, we analyze real world traces and point out that protecting speed information of movement trajectories is necessary. Furthermore, we show that the accuracy decrease introduced by our speed protection algorithms only affects few position updates such that the speed protected trajectories are of high quality.

Next, we present our extended system model and introduce our problem statement. Then, we show the details of our speed protection algorithms. Afterwards, we present our privacy analysis including our proof of correctness for the algorithms. Then, we show evaluation results and related work. Finally, we conclude our approach protecting the speed information of movement trajectories.

4.2.1 Extended System Model

Our extended system model is based on the basic system model presented in Section 2.2 and shown in Figure 2.1a using a single location server. However, our speed protection algorithm can also be applied to the other presented models such as the model using multiple LSs storing speed protected position information. The mobile object provides its current position π to an LS storing and managing trajectory information of several MOs. To this end, the MO sends its position information to the LS by using function $update(\pi, t)$ consisting of its current position π and the temporal information defining the corresponding time t. The position updates are triggered based on the MO-selected location update protocol. The LS provides different clients access to the stored trajectory information and controls the access of different clients by using an access control mechanism as presented by Bonatti and Samarati [BS02]. Thus, only clients with the corresponding access rights can access the stored information of the MO's movement trajectory.

It must be guaranteed by existing trusted mobile computing approaches such as [GCJW10] that no other component than our speed protection component can directly access the MO's position π locally on the mobile device. Otherwise, a local client that could access π by directly querying the positioning system could maliciously reveal a speeding violation of the MO. In the following, we focus our description on remote clients querying the remote LS and mention local clients only if they behave differently.

The MO's position π is defined by its longitude and latitude value. Since vehicles typically move on streets, we map positions to a graph representing the road network. As introduced for our trajectory fragmentation algorithm in Section 4.1.1, we model the road network by a weighted graph $G = (V, E)$. Each node $v_i \in V$ represents either a junction or defines an intermediate node that is used to model the shape of a road. Each edge $e_i \in E$ represents a road segment between two nodes and has an assigned maximum allowed speed limit. To map "raw" GPS positions to the underlying road network, map matching algorithms

can be used. In the following, we do not consider the map matching any more, but assume that π is located on a road segment of the graph.

Trajectory T is defined as sequence $T = \{(\pi_{start}, t_{start}), \ldots, (\pi_{end}, t_{end})\}$ of different positions π_i, where the MO was located at time t_i. The MO's trajectory T can be split up into a set of segments, where each segment $\overline{\pi_i, \pi_{i+1}}$ defines the path the MO traveled from position π_i at time t_i to position π_{i+1} at time t_{i+1}. We assume that MOs travel on fastest paths, since usually a MO intends to reach its destination as quick as possible. Thus, $\overline{\pi_i, \pi_{i+1}}$ is the fastest path between π_i and π_{i+1} on the road network. The distance between π_i and π_{i+1} is

$$distance(\overline{\pi_i, \pi_{i+1}}) = length(FP(\pi_i, \pi_{i+1})) \tag{4.3}$$

defining the length of the fastest path (FP) from π_i to π_{i+1}.

The time the MO traveled from π_i to π_{i+1} on segment $\overline{\pi_i, \pi_{i+1}}$ is defined as

$$time(\overline{\pi_i, \pi_{i+1}}) = t_{i+1} - t_i. \tag{4.4}$$

4.2.2 Problem Statement

We tackle the problem that real-time position updates of a MO's movement trajectory T reveal speeding violations of the MO. To solve this problem, we use our speed protection algorithms transforming trajectory T into the speed protected trajectory $\widehat{T}$ (introduced below) guaranteeing that nobody can determine any speeding violation by analyzing $\widehat{T}$.

For trajectory $T = \{(\pi_{start}, t_{start}), \ldots, (\pi_{end}, t_{end})\}$, a speeding violation is detected if for at least one segment $\overline{\pi_i, \pi_{i+1}} \in T$ the MO traveled from π_i to π_{i+1} within shorter time than it takes the MO when driving at the maximum allowed speed of the segment:

$$time(\overline{\pi_i, \pi_{i+1}}) < timeMaxSpeed(\overline{\pi_i, \pi_{i+1}}). \tag{4.5}$$

This concept is also known as "section control", where the average speed over

a certain known distance is calculated. Here, $timeMaxSpeed(\overline{\pi_i, \pi_{i+1}})$ defines the time it takes the MO driving from π_i to π_{i+1} at the maximum allowed speed. In case the maximum allowed speed changes within segment $\overline{\pi_i, \pi_{i+1}}$, $timeMaxSpeed(\overline{\pi_i, \pi_{i+1}})$ is calculated by splitting up segment $\overline{\pi_i, \pi_{i+1}}$ into separate parts of different speed limits. Then, the required times to travel the distances of the different parts at the maximum allowed speed limits are calculated and summed up defining $timeMaxSpeed(\overline{\pi_i, \pi_{i+1}})$.

An attacker is interested in finding segment $\overline{\pi_i, \pi_{i+1}} \in T$ indicating that the MO was speeding:

$$\exists\ \overline{\pi_i, \pi_{i+1}} \in T : time(\overline{\pi_i, \pi_{i+1}}) < timeMaxSpeed(\overline{\pi_i, \pi_{i+1}}). \tag{4.6}$$

As stated previously, $\overline{\pi_i, \pi_{i+1}}$ defines the fastest path from π_i to π_{i+1}. If a speeding violation is detected for $\overline{\pi_i, \pi_{i+1}}$, then the MO is also speeding for all other slower paths from π_i to π_{i+1}. Thus, we call this a *definitely speeding* semantic, where the MO has no plausible way of denying a speeding violation.

The goal of our speed protection algorithms is to transform trajectory T into a speed protected trajectory $\widehat{T}$ such that for every segment $\overline{\pi_i, \pi_{i+1}}$ at least one possible path exists where the MO could have traveled from π_i to π_{i+1} without speeding. Then, the definitely speeding condition is not fulfilled for $\widehat{T}$:

$$\forall\ \overline{\pi_i, \pi_{i+1}} \in \widehat{T} : time(\overline{\pi_i, \pi_{i+1}}) \geq timeMaxSpeed(\overline{\pi_i, \pi_{i+1}}). \tag{4.7}$$

Obviously, transforming the original trajectory T into the speed protected trajectory $\widehat{T}$ introduces artificial inaccuracies. Therefore, another goal is to alter trajectory T as less as possible to reduce the introduced inaccuracies. Later, we will show that the speed protection algorithms either lead to spatial or temporal inaccuracies for positions of $\widehat{T}$. Therefore, the spatial or temporal difference should be as small as possible.

By guaranteeing that the definitely speeding condition is not fulfilled for the published trajectory $\widehat{T}$, the MO can plausibly deny a speeding violation since the MO could have traveled on each segment of trajectory $\widehat{T}$ without driving faster

than the allowed speed. In combination with the "in dubio pro reo" principle, this fulfills our goal to protect the speed of a MO's movement trajectory $\widehat{T}$ because no evidences of a speeding violation exist as long as the MO could have traveled along $\widehat{T}$ on the fastest path without any speeding violation.

4.2.3 Speed Protection Algorithms

In this section, we present our speed protection algorithms (*SPA* for short), which guarantee that the updated movement trajectory $\widehat{T}$ of the MO does not contain any speeding violation.

Overview

The general idea of *SPA* is to slow down the speed of trajectory $\widehat{T}$ to the maximum speed the MO is allowed to drive on each road segment. Since location update protocols are usually either time-based or distance-based, we use the methods of *position adjustment* (PA) and *temporal delay* (TD) to support both kinds of protocols. Later, we show that our position adjustment method can also be applied to dead reckoning-based update protocols, which can reduce the number of position updates.

We use position adjustment in *SPA-PA* to support time-based position update protocols, which periodically update π_{i+1} after a predefined update time period *TP*. Our position adjustment shifts position π_{i+1} to position $\widehat{\pi}_{i+1}$ if a speeding violation happened between the last updated position $\widehat{\pi}_i$ and the currently sensed position π_{i+1} (cf. Figure 4.12a). Here, position adjustment protects the speed of the MO by decreasing the spatial accuracy of π_{i+1} such that the speed protected position $\widehat{\pi}_{i+1}$ is updated instead of π_{i+1} at time t_{i+1}. As soon as the speed of the MO is below the speed limit, the spatial accuracy is increased until position $\widehat{\pi}_{i+1}$ is equal to π_{i+1} and the accurate position of the MO can be updated again without revealing a speeding violation. In case the MO does not drive faster than the speed limit, *SPA-PA* updates the movement trajectory of the MO without modification. Thus, clients can provide their service based on the

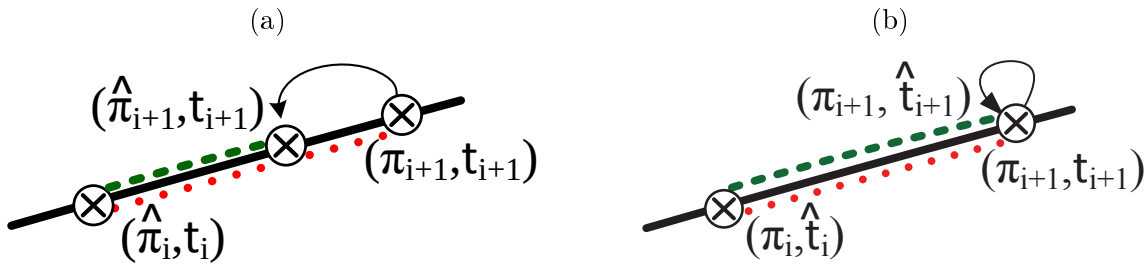


Figure 4.12: (a) Position adjustment and (b) temporal delay examples. Dotted segments indicate a speeding violation while the dashed ones do not

MO's movement trajectory without decreasing quality for non-speeding MOs. Later, we analyze the introduced inaccuracies of the speed protected trajectory depending on the MO's driving behavior.

We use temporal delays in *SPA-TD* to support distance-based position update protocols taking the traveled distance of the MO into account. With distance-based protocols a new position π_{i+1} is updated whenever the Euclidean distance to the last reported position π_i reaches a given threshold distance D. If a speeding violation occurred between the last updated position π_i and the new sensed position π_{i+1}, the update of position π_{i+1} is delayed until time $\widehat{t}_{i+1}$ such that no speeding violation can be recognized between π_i updated at time $\widehat{t}_i$ and π_{i+1} updated at time $\widehat{t}_{i+1}$ (cf. Figure 4.12b). Our temporal delay keeps the spatial information of the position accurate and decreases the temporal accuracy of the update. As soon as the speed of the MO is reduced and the MO drives slower than the allowed speed limit, the temporal accuracy is increased until the timestamps of the sensed and updated positions are identical and no temporal delay is needed any more. If no speeding violation of the MO occurred, *SPA-TD* updates the MO's movement trajectory without modification.

Figure 4.13 shows an overview of the complete process of our approach. The MO senses its position π_{i+1} at time t_{i+1} using the positioning sensor. For time-based updates, *SPA-PA* uses the position adjustment method and provides position $\widehat{\pi}_{i+1}$ at time t_{i+1} to the update algorithm. For distance-based updates, *SPA-TD* uses the temporal delay method to provide position π_{i+1} at the delayed

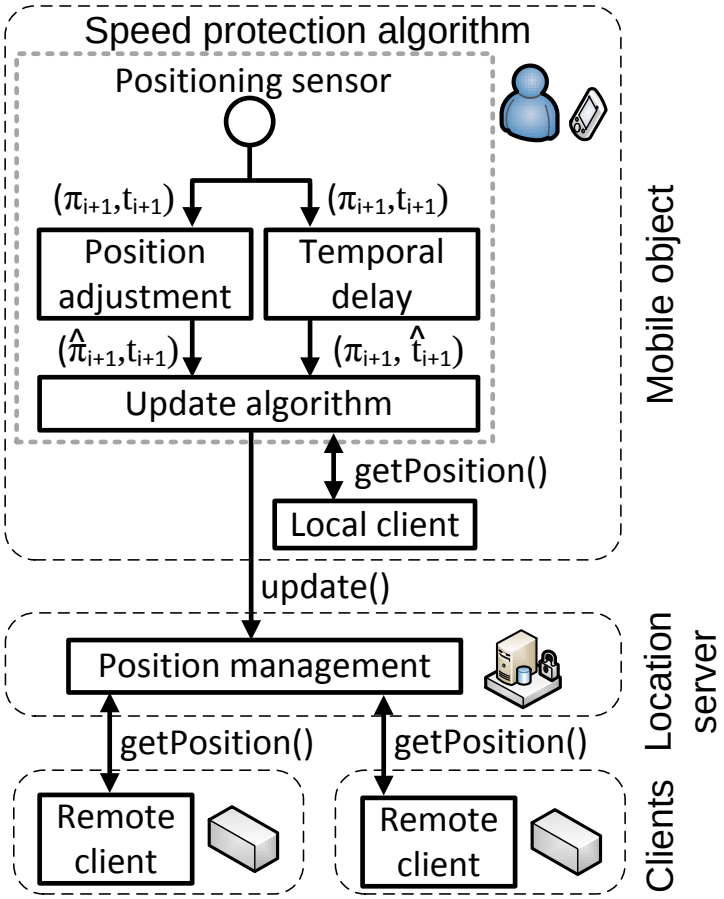


Figure 4.13: Speed protection process overview

time $\widehat{t}_{i+1}$ to the update algorithm. Then, function $update(\widehat{\pi}_{i+1}, t_{i+1})$ respectively $update(\pi_{i+1}, \widehat{t}_{i+1})$ is used by the update algorithm to provide the new position information of the MO to the LS. Finally, clients can access this information by using function $getPosition()$. Below, we present the two protection methods in more detail.

SPA with Position Adjustment

The detailed speed protection algorithm *SPA-PA* for time-based updates is shown in Algorithm 15. First, the initial position π_{start} is sensed at time t_{start}

Algorithm 15 SPA with position adjustment

Function: $SPA-PA()$

1: $\pi_{start}, t_{start} \leftarrow$ sensed position ▷ Initial positioning
2: $T \leftarrow \pi_{start}, t_{start}$ ▷ Sensed trajectory
3: $\widehat{\pi}_{start} \leftarrow \pi_{start}$ ▷ Initial position
4: $update(\widehat{\pi}_{start}, t_{start})$ ▷ Initial position update
5: **while** report movement **do**
6: $\pi_{i+1}, t_{i+1} \leftarrow$ sensed position ▷ Triggered time-based
7: $T \leftarrow T \cup \pi_{i+1}, t_{i+1}$
8: $\widehat{\pi}_i, t_i \leftarrow$ last updated position
9: **if** $time(\overline{\widehat{\pi}_i, \pi_{i+1}}) < timeMaxSpeed(\overline{\widehat{\pi}_i, \pi_{i+1}})$ **then**
10: $\delta t \leftarrow time(\overline{\widehat{\pi}_i, \pi_{i+1}})$ ▷ Speeding detected
11: $\widehat{\pi}_{i+1} \leftarrow getReachablePosition(\widehat{\pi}_i, \delta t, T)$
12: **else**
13: $\widehat{\pi}_{i+1} \leftarrow \pi_{i+1}$ ▷ No speeding occurred
14: **end if**
15: $update(\widehat{\pi}_{i+1}, t_{i+1})$ ▷ Update $\widehat{\pi}_{i+1}$ at time t_{i+1}
16: **end while**

and updated on the LS as position $\widehat{\pi}_{start}$ by using function $update(\widehat{\pi}_{start}, t_{start})$ at time t_{start}. Afterwards, a new position π_{i+1} is sensed at time $t_{i+1} = t_i + TP$ based on the MO-selected update time period TP. Next, *SPA-PA* uses the last updated position $\widehat{\pi}_i$ to evaluate $time(\overline{\widehat{\pi}_i, \pi_{i+1}}) < timeMaxSpeed(\overline{\widehat{\pi}_i, \pi_{i+1}})$. If the MO reached π_{i+1} from $\widehat{\pi}_i$ within shorter time than it takes a MO driving at the maximum allowed speed, then updating π_{i+1} would reveal a speeding violation. In this case position π_{i+1} has to be adjusted before it can be published. To this end, *SPA-PA* uses function $getReachablePosition(\widehat{\pi}_i, \delta t, T)$ to calculate position $\widehat{\pi}_{i+1}$ as the position that can be reached from the last updated position $\widehat{\pi}_i$ within time $\delta t = time(\overline{\widehat{\pi}_i, \pi_{i+1}})$ when driving at the maximum allowed speed on segment $\overline{\widehat{\pi}_i, \pi_{i+1}}$. In case no speeding violation is detected, position π_{i+1} is used as position $\widehat{\pi}_{i+1}$. Finally, the calculated position $\widehat{\pi}_{i+1}$ is updated using function $update(\widehat{\pi}_{i+1}, t_{i+1})$.

An example of *SPA-PA* for time-based updates is presented in Figure 4.14a. For simplicity, we use a fixed value of $maxSpeed(e) = 100\,\mathrm{km/h}$ for each edge

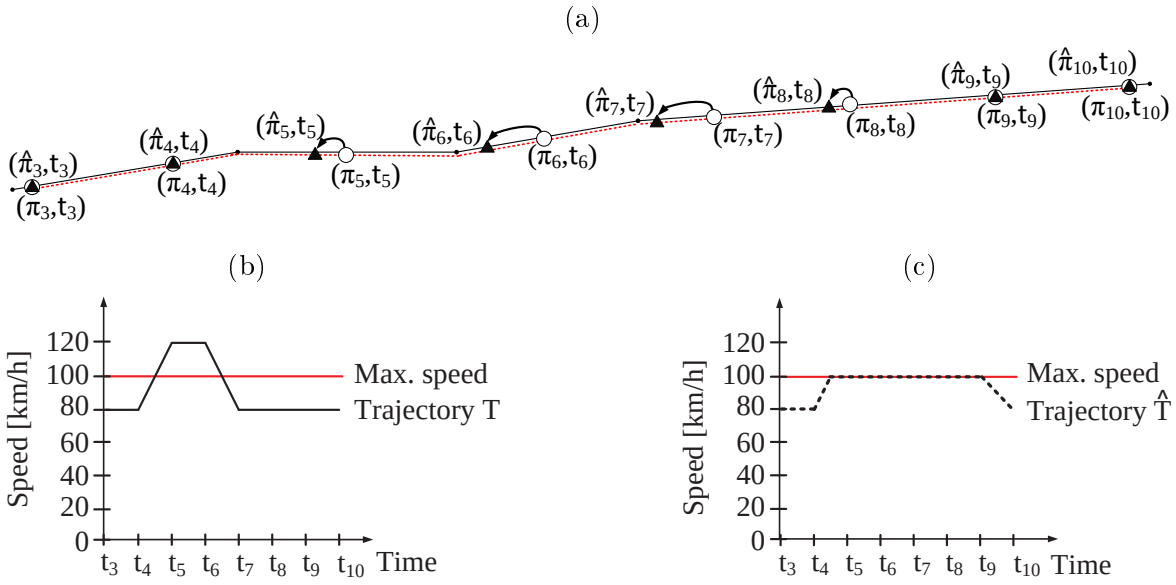


Figure 4.14: (a) *SPA-PA* example using time-based updates, (b) the time-speed diagram of trajectory T, and (c) the protected trajectory $\widehat{T}$

e of a road segment and assume that a new position is sensed after a time period of $TP = 5\,\text{s}$. Without *SPA-PA*, trajectory T shown in Figure 4.14a with the corresponding time-speed diagram shown in Figure 4.14b would be published. The MO's accurate positions of trajectory T are depicted as circles ($\circ$) in Figure 4.14a. The MO drives with a speed of 80 km/h at time t_3. At time t_4, the MO accelerates up to a speed of 120 km/h at time t_5 and travels at this speed until time t_6. Then, the MO slows down to 80 km/h at time t_7 that is kept until time t_{10}. By using *SPA-PA*, the maximum speed between two updated positions in $\widehat{T}$ is limited to the maximum allowed speed of 100 km/h. The generated updates are shown in Figure 4.14a, where the protected positions of trajectory $\widehat{T}$ are depicted as triangles ($\blacktriangle$). The corresponding time-speed diagram of the speed protected trajectory $\widehat{T}$ is shown in Figure 4.14c. While actually a speeding violation of the MO occurred, the reported speed is limited to the maximum allowed speed. The time after the MO's speeding violation is used to reduce the spatial difference between the updated and the actual position of the MO. Thus, the introduced spatial inaccuracy is reduced.

We assume for *SPA-PA* and *SPA-TD* that the MO updates its protected movement trajectory to the LS until the destination of the protected trajectory $\widehat{T}$ is equal to the destination of the MO's trajectory T. Otherwise, for instance, if the MO would turn off its mobile device before, trajectory $\widehat{T}$ could end in case of a speeding violation before reaching the actual destination of the MO.

SPA with Temporal Delay

The speed protection algorithm *SPA-TD* for distance-based updates is shown in Algorithm 16. The positioning sensor provides a new position to *SPA-TD* as soon as the Euclidean distance between the last sensed position π_i and the new sensed position π_{i+1} reaches the MO-defined threshold distance D. The initial position is updated as described for *SPA-PA*. For the following position updates, it is evaluated whether position π_{i+1} sensed at time t_{i+1} can be reached in time $time(\overline{\pi_i, \pi_{i+1}}) = t_{i+1} - \widehat{t}_i$ from the last updated position π_i updated at time $\widehat{t}_i$. A speeding violation is detected if $time(\overline{\pi_i, \pi_{i+1}}) < timeMaxSpeed(\overline{\pi_i, \pi_{i+1}})$. Then, the update of position π_{i+1} must be delayed until π_{i+1} can be reached from π_i without exceeding the speed limit. To this end, *SPA-TD* calculates time $\widehat{t}_{i+1}$ at which the MO can reach π_{i+1} from π_i without speeding violation. *SPA-TD* assumes a MO driving at the maximum allowed speed from π_i to π_{i+1} and uses the minimum required time $timeMaxSpeed(\overline{\pi_i, \pi_{i+1}})$ to calculate $\widehat{t}_{i+1}$ as $\widehat{t}_{i+1} = \widehat{t}_i + timeMaxSpeed(\overline{\pi_i, \pi_{i+1}})$. The update of position π_{i+1} is then delayed until time $\widehat{t}_{i+1}$. Otherwise, if π_{i+1} can be reached from π_i without a speeding violation, time t_{i+1} is used to define $\widehat{t}_{i+1} = t_{i+1}$. Finally, position π_{i+1} is updated at time $\widehat{t}_{i+1}$ using function $update(\pi_{i+1}, \widehat{t}_{i+1})$. After a speeding violation occurred and the MO is again driving at a speed below the speed limit, the temporal accuracy is increased until no delay is needed any more.

An example for *SPA-TD* is shown in Figure 4.15a. For this example, we assume the same movement of the MO and the same speed limitations as presented in our previous example for *SPA-PA*. Compared to the former example, the sensing of a new position is triggered based on the MO-defined threshold distance D that is set to $D = 100\,m$. The sensed positions of trajectory T and

Algorithm 16 SPA with temporal delay

Function: $SPA-TD()$

1: $\pi_{start}, t_{start} \leftarrow$ sensed position ▷ Initial positioning
2: $\widehat{t}_{start} \leftarrow t_{start}$ ▷ Initial update time
3: $update(\pi_{start}, \widehat{t}_{start})$ ▷ Initial position update
4: **while** report movement **do**
5: $\pi_{i+1}, t_{i+1} \leftarrow$ sensed position ▷ Triggered distance-based
6: $\pi_i, \widehat{t}_i \leftarrow$ last updated position
7: **if** $time(\overline{\pi_i, \pi_{i+1}}) < timeMaxSpeed(\overline{\pi_i, \pi_{i+1}})$ **then**
8: $\widehat{t}_{i+1} \leftarrow \widehat{t}_i + timeMaxSpeed(\overline{\pi_i, \pi_{i+1}})$ ▷ Speeding detected
9: **else**
10: $\widehat{t}_{i+1} \leftarrow t_{i+1}$ ▷ No speeding occurred
11: **end if**
12: $update(\pi_{i+1}, \widehat{t}_{i+1})$ ▷ Update π_{i+1} at time $\widehat{t}_{i+1}$
13: **end while**

the corresponding position updates of trajectory $\widehat{T}$ are shown in Figure 4.15a. The accurate positions are again depicted as circles (○), while the protected positions are depicted as triangles (▲). As shown, the spatial position information of each sensed position is kept accurate, while the point in time used to update the position is adjusted to protect the speed information of the MO. The corresponding distance-speed diagrams of trajectory T and the protected trajectory $\widehat{T}$ are shown in Figure 4.15b and Figure 4.15c.

4.2.4 Privacy Analysis

In this section, we prove that the MO's movement trajectory $\widehat{T}$ generated by our speed protection algorithms does not indicate any speeding violation to an attacker. First, we consider *SPA-PA* and proceed afterwards with *SPA-TD*.

Proof of Correctness for *SPA-PA*

We prove the correctness of *SPA-PA* by contradiction. Assume that there exists a segment in the published trajectory $\widehat{T}$ of the MO indicating a speeding violation. Furthermore, assume that trajectory $\widehat{T} = \{(\widehat{\pi}_{start}, t_{start}), \ldots, (\widehat{\pi}_{end}, t_{end})\}$

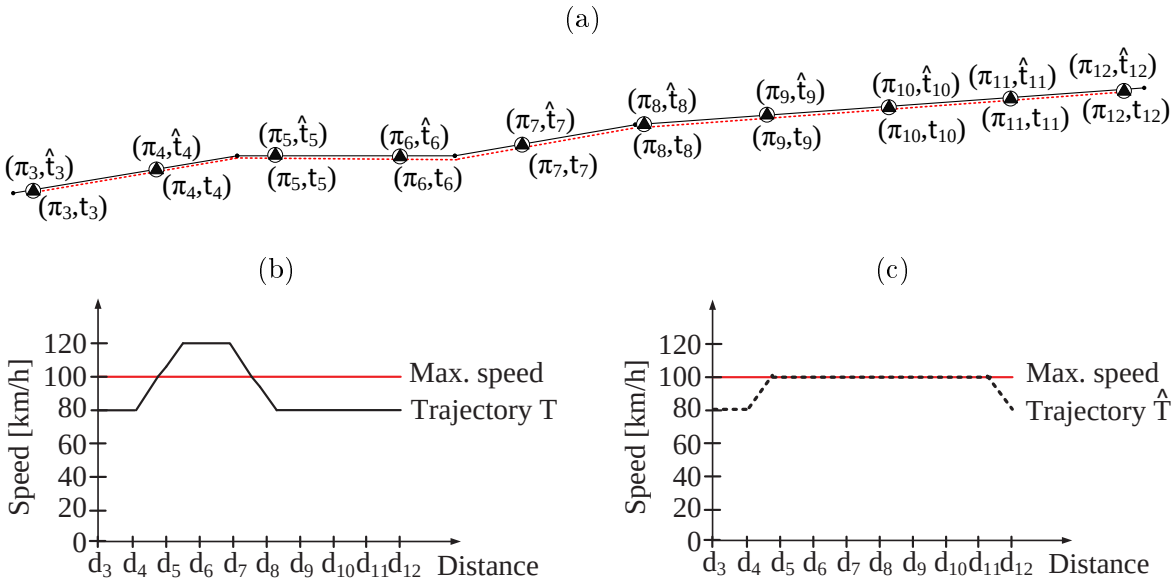


Figure 4.15: (a) *SPA-TD* example using distance-based updates, (b) distance-speed diagram for trajectory T, and (c) the protected trajectory $\widehat{T}$

generated by *SPA-PA* consists of at least two position updates, which is the minimum number of position updates required to derive speed information. Following our assumption, at least one segment $\overline{\widehat{\pi}_i, \widehat{\pi}_{i+1}} \in \widehat{T}$ has to exist indicating a speeding violation of the MO. Without loss of generality, let position $\widehat{\pi}_{i+1}$ updated at time t_{i+1} be the first update indicating a speeding violation. As we can see in Algorithm 15, $\widehat{\pi}_{i+1}$ can only be updated using function $update(\widehat{\pi}_{i+1}, t_{i+1})$ (cf. line 15). Moreover, our assumption requires that the MO drove faster than the allowed speed limit on segment $\overline{\widehat{\pi}_i, \widehat{\pi}_{i+1}}$. However, because *SPA-PA* detects the speeding violation for position π_{i+1} at time t_{i+1} (cf. line 9), the updated position $\widehat{\pi}_{i+1}$ is calculated as the position that can be reached from the last updated position $\widehat{\pi}_i$ without speeding violation (cf. line 11). Therefore, the updated position $\widehat{\pi}_{i+1}$ cannot indicate a speeding violation on segment $\overline{\widehat{\pi}_i, \widehat{\pi}_{i+1}}$. This contradicts our assumption that segment $\overline{\widehat{\pi}_i, \widehat{\pi}_{i+1}} \in \widehat{T}$ indicates a speeding violation. Thus, it is guaranteed that trajectory $\widehat{T}$ does not contain any speeding violation.

Proof of Correctness for *SPA-TD*

For *SPA-TD*, we can show in a similar way to *SPA-PA* that no segment $\overline{\pi_i, \pi_{i+1}} \in \widehat{T}$ can exist indicating a speeding violation of the MO. Assume trajectory $\widehat{T}$ provided to the LS consists again of at least two position updates. Moreover, assume that position π_{i+1} updated at time $\widehat{t}_{i+1}$ is the first position update indicating a speeding violation. The only function updating the MO's position π_{i+1} in Algorithm 16 is function $update(\pi_{i+1}, \widehat{t}_{i+1})$ (cf. line 12). However, a speeding violation is detected by *SPA-TD* (cf. line 7) and the update of position π_{i+1} is delayed until time $\widehat{t}_{i+1}$ (cf. line 8). Therefore, the update of position π_{i+1} does not provide any information that the MO drove faster than the allowed speed limit. This contradicts our assumption that segment $\overline{\pi_i, \pi_{i+1}} \in \widehat{T}$ indicates a speeding violation of the MO, and we can state that trajectory $\widehat{T}$ generated by *SPA-TD* protects the speed information of the MO.

4.2.5 Evaluation

In this section, we present a real world trace evaluation analyzing the speed information of taxi cabs in the San Francisco Bay Area. Moreover, we evaluate the runtime performance of our speed protection algorithms using a prototype implementation on a mobile device.

Analysis of Real World Traces for Speeding Violations

We select from the mobility traces provided by Piorkowski et al. [PSDG09] the time period of one day (2008/06/01) and analyze the behavior of the taxi cabs for this day. To determine the speed limit and the length of each road segment, we use the map information of OpenStreetMap [Ope14] providing road network data, speed limits, and further information.

First, we analyze the driving behavior and the occurred speeding violations that can be derived from the movement trajectories. To this end, we distinguish for each position update whether it indicates a speeding violation or not by analyzing the travel time and distance between succeeding updates as formalized

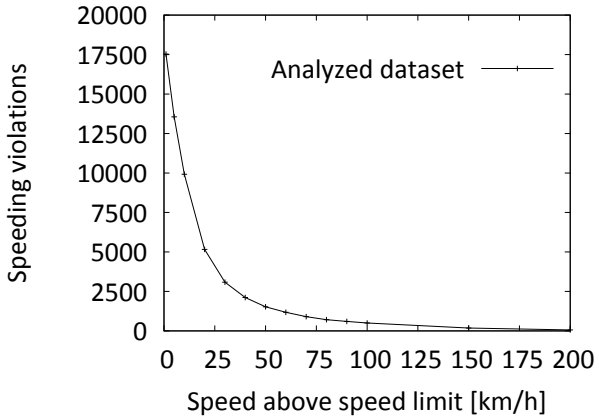


Figure 4.16: Speeding violation analysis

in Equation 4.5. Overall, we analyzed 365 348 position updates of 484 different taxi cabs. 347 818 of the updates (95.20%) conform to the local speed limits, while 17 530 position updates (4.80%) indicated a speeding violation.

We further analyzed these speeding violations and calculated the speed

$$\Delta v_{speeding} = v - v_{max} \tag{4.8}$$

as the difference between the MO's speed v and the allowed speed limit v_{max}. The results are shown in Figure 4.16, where we plotted the number of detected speeding violations over $\Delta v_{speeding}$. The majority (87,93%) of the speeding violations occurred for $\Delta v_{speeding} \leq 40\,\text{km/h}$. On average, $\Delta v_{speeding}$ is 20.47 km/h for all detected speeding violations. From these results we can see that protecting the speed information of movement trajectories is a relevant problem.

Analysis of Introduced Inaccuracies

Next, we formalize the spatial and temporal inaccuracy introduced by *SPA-PA* and *SPA-TD*. The protection of the speed information leads to an artificial inaccuracy in case the MO drives faster than the allowed speed limit. We define $\Delta v^{max}_{speeding}$ as the maximum speed difference between the MO's speed v and the

allowed speed limit v_{max}. Therefore, $\Delta v^{max}_{speeding}$ depends on the individual driving behavior of the MO. Furthermore, we define position $\pi_j \in T$ measured at time t_j as the last measured position in trajectory T where the MO drove equal to or below the allowed speed limit. Then, we can calculate for each position $\widehat{\pi}_k \in \widehat{T}$ updated at time $t_k \geq t_j$ the maximum spatial inaccuracy introduced by *SPA-PA* as

$$\Delta d_{max}(\pi_j, \widehat{\pi}_k) = time(\overline{\pi_j, \widehat{\pi}_k}) \cdot \Delta v^{max}_{speeding}. \tag{4.9}$$

For *SPA-TD*, the maximum introduced temporal inaccuracy for each position $\pi_k \in \widehat{T}$ and the last measured non-speeding position π_j is calculated as

$$\Delta t_{max}(\pi_j, \pi_k) = \frac{distance(\overline{\pi_j, \pi_k}) \cdot \Delta v^{max}_{speeding}}{(v_{max})^2 + (v_{max} \cdot \Delta v^{max}_{speeding})}. \tag{4.10}$$

As we can see, the maximum spatial and temporal deviation introduced by our speed protection algorithms depends on the MO's driving behavior defining $\Delta v^{max}_{speeding}$ and the duration of the speeding violation of the MO respectively the traveled distance between position π_j and position π_k.

To get an insight into real world user driving behavior, we analyze the spatial and temporal inaccuracy introduced by *SPA-PA* and *SPA-TD* for the presented real world dataset. To this end, we analyze the spatial inaccuracy introduced by *SPA-PA* by measuring the Euclidean distance between position $\widehat{\pi}_{i+1}$ calculated by *SPA-PA* and the original position π_{i+1}, which would be updated without speed protection. Formally, the spatial inaccuracy is calculated for time t_{i+1} as

$$\Delta d(t_{i+1}) = dist(\widehat{\pi}_{i+1}, \pi_{i+1}). \tag{4.11}$$

The temporal inaccuracy introduced by *SPA-TD* is the time between t_{i+1} when position π_{i+1} is updated without *SPA-TD* and time $\widehat{t}_{i+1}$ *SPA-TD* updates π_{i+1}. The temporal inaccuracy is calculated for position π_{i+1} as

$$\Delta t(\pi_{i+1}) = \widehat{t}_{i+1} - t_{i+1}. \tag{4.12}$$

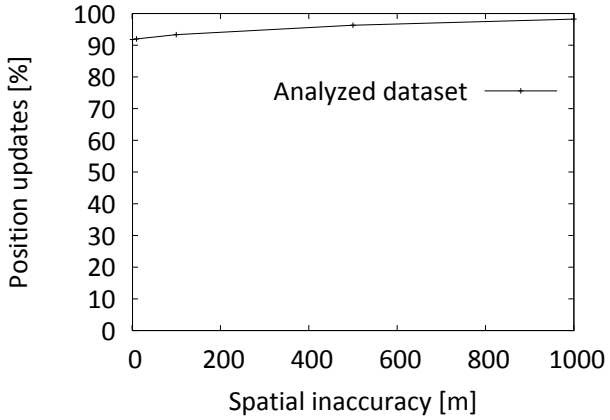


Figure 4.17: Cumulative distribution of spatial inaccuracy

Because the taxi cabs of the analyzed dataset triggered their position updates in an irregular manner, the used update strategy is neither strictly distance-based nor time-based. However, since our algorithms are also applicable for irregular update intervals and distances, we used the provided positions from the original dataset with the original timestamps.

We analyze *SPA-PA* by calculating the spatial inaccuracy $\Delta d(t_{i+1})$ for each position update π_{i+1} triggered at time t_{i+1}. As shown in Figure 4.17, 93.3% of the position updates have an inaccuracy below 100 m. Therefore, we can state that the speed protected movement trajectories are of high quality and only few updates have a low spatial accuracy.

For our analysis of *SPA-TD* we calculate the temporal inaccuracy $\Delta t(\pi_{i+1})$ for each position update triggered by position π_{i+1}. The evaluation results are shown in Figure 4.18. As we can see, 94.56% of the position updates have a temporal inaccuracy below ten seconds. An inaccuracy of 60 seconds covers 99.14% of all updates. Therefore, we can state that the temporal inaccuracy introduced by *SPA-TD* is low.

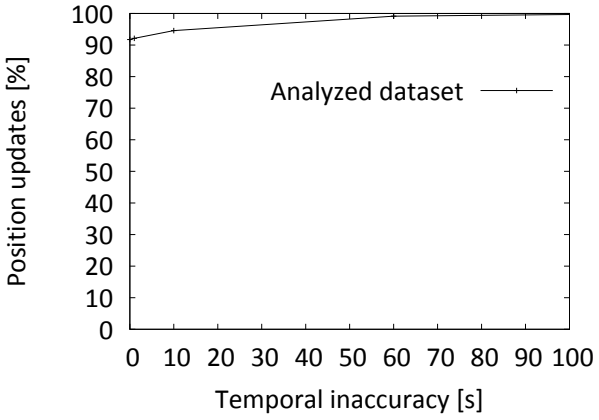


Figure 4.18: Cumulative distribution of temporal inaccuracy

Performance Evaluation

Next, we analyze the efficiency of *SPA-PA* and *SPA-TD*. Since the protection algorithms have to be executed online on the mobile device, which typically has relatively low computational power, efficiency is a critical factor for the algorithms. Moreover, a small computational overhead also reduces the energy consumption, which is desirable for battery-operated mobile devices.

We evaluate *SPA-PA* and *SPA-TD* using a prototype implemented on a mobile device (HTC Desire HD, 1 GHz Qualcomm Snapdragon S2, 768 MB RAM). The used map is the road network of Stuttgart, Germany. The time required to calculate a new speed protected position depends on the selected update protocol and its update parameter.

For *SPA-PA*, we analyze randomly selected positions on the map with different update time periods TP ranging from one second to 60 seconds. Figure 4.19 shows the evaluation results for a movement trajectory over several runs for each measurement. As we can see, the required time to determine a new speed protected position increases by increasing the update time period TP. This is based on the fact that the speed protection algorithm has to calculate the fastest path on the road network from the last updated position to the currently evaluated position. By increasing TP, the traveled distance also increases. The measured

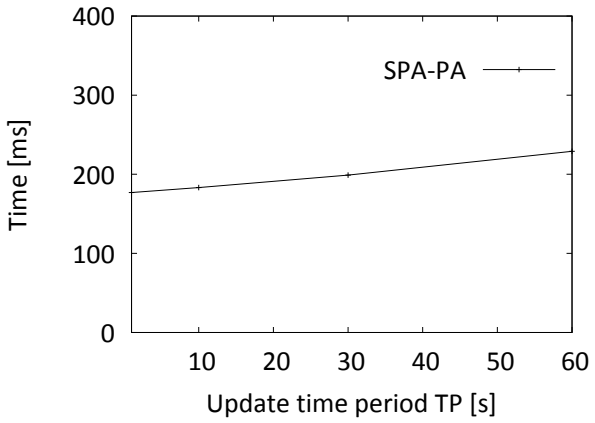


Figure 4.19: Performance evaluation using time-based updates

time includes the time required to calculate the map matching on the road network for the sensed position of the MO. The map matching is required because position π provided by the positioning system has to be mapped to the road network before it can be used by our speed protection algorithms. On average, the map matching takes 150 milliseconds. As the introduced calculation of *SPA-PA* for time-based position updates is well below 250 milliseconds even for larger update time periods, we can state that protecting the speed information of a MO is possible without affecting the real-time property of position updates.

For *SPA-TD*, we vary the threshold distance D from ten meters to 1 000 meters. As shown in Figure 4.20, the processing time of *SPA-TD* stays well below 250 milliseconds. The time for the map matching part of *SPA-TD* is identical to the evaluation of *SPA-PA*. Generally, calculating the temporal delay of a position update in *SPA-TD* takes nearly the same time than calculating the next reachable position not indicating a speeding violation in *SPA-PA*. This is based on the fact that the functions used in the introduced algorithms mainly differ only in line 11 in Algorithm 15 and line 8 in Algorithm 16, where we calculate the next reachable position and the next reachable point in time that can be used without indicating a speeding violation for the next update. Therefore, *SPA-TD* can also be calculated very fast and support real-time position updates.

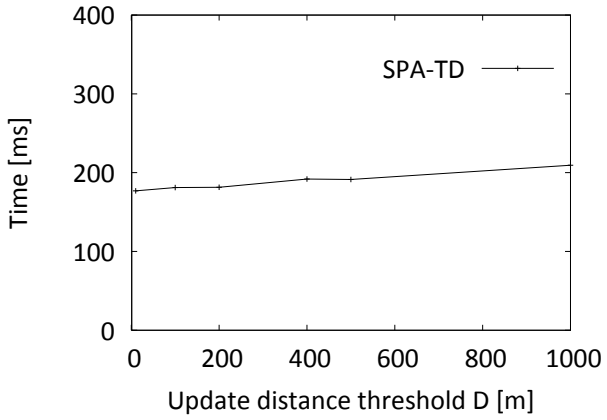


Figure 4.20: Performance evaluation using distance-based updates

Analysis of Update Costs

As we can see from Algorithm 15 and Algorithm 16, each sensed position of the MO leads to exactly one (protected) position update. Thus, the update costs of *SPA-PA* and *SPA-TD* are identical to the costs without speed protection.

To reduce the number of required position updates sent to the LS, advanced update protocols based on dead reckoning can be used. In these protocols, the LS estimates the position of the MO based on its last known position, its movement speed, and its movement direction. The MO calculates the same estimation and updates its position, its movement speed and its movement direction as soon as its actual position differs from the estimated position by more than a certain distance threshold DT. To this end, the MO evaluates for its current position π_i and its estimated position π_{est} at time t_i the update criteria

$$dist(\pi_i, \pi_{est}) > DT. \tag{4.13}$$

If the update criteria is fulfilled, the MO updates its movement information to the LS. Otherwise, no update is required. Typically, linear prediction mechanisms are used to estimate the position of the MO. By using prediction functions of higher order or map-based functions, the number of required updates can fur-

ther be reduced. For an overview of different variants of dead reckoning-based update protocols, we refer to the work of Leonhardi et al. [LNR02].

To allow for speed protected position updates based on dead reckoning, the time-based location update protocol *SPA-PA* provides the MO's speed protected position $\widehat{\pi}_i$ at time t_i to the update algorithm implementing dead reckoning. Then, the MO evaluates based on the speed protected position $\widehat{\pi}_i$ instead of position π_i the update criteria

$$dist(\widehat{\pi}_i, \pi_{est}) > DT. \tag{4.14}$$

This approach guarantees that the dead reckoning algorithm does not reveal any speeding violation when updating the MO's movement information to the LS. Since dead reckoning-based update protocols explicitly update the speed of the MO, which is calculated based on the speed protected trajectory $\widehat{T}$, this speed is equal to or below the maximum allowed speed limit. The correctness of this approach relies on the correctness of *SPA-PA* presented in Section 4.2.4. That is, the MO's speed protected position is provided by *SPA-PA* to the algorithm implementing dead reckoning instead of providing the position directly to the LS. Thus, the correctness arguments presented for *SPA-PA* also apply to speed protected position updates based on dead reckoning.

4.2.6 Related Work

Existing *k-anonymity approaches* [KGMP07] try to protect the user identity by preventing an attacker from linking a trajectory—which is not modified and, therefore, contains all speeding violations—to an individual user. However, as shown in [ZB11, MHVB13], spatial and temporal information can be used to identify individual users and, therefore, to identify individual speeders.

Spatial obfuscation approaches [ACG09, DSB11] protecting user position information can reveal speeding violations if multiple obfuscated positions are provided to the LS. Thus, these approaches do not protect the speed information of the user. This applies also to the *position sharing approaches* [DSR11, SDR12]

and our position sharing approaches presented in Chapter 3.

Our *trajectory fragmentation approach* presented in Section 4.1 may reveal violations of given speed limits to the LS storing a trajectory fragment which includes a speeding violation of the user. Therefore, trajectory fragmentation does not protect the speed information of movement traces. However, our trajectory fragmentation approach could be extended by our speed protection algorithms to protect the updated positions of the shared fragments.

Dummy approaches [KYS05, SGI09] do not consider the speed information of dummies. Thus, if all dummies and the real user trajectory contain speeding violations, an attacker knows that the user is definitely speeding. As the dummy approach is orthogonal to our approach, our speed protection algorithms could be integrated to protect dummies from revealing speeding violations.

4.3 Conclusion

In this chapter, we presented novel approaches to share movement trajectories in a privacy-aware manner in system environments of location servers and clients without assuming a trusted third party. To this end, we focused on the protection of movement trajectories and speed information that can be derived from trajectories. First, we introduced our trajectory fragmentation algorithm splitting up a trajectory into different trajectory fragments that are distributed among LSs of different providers to limit the maximum revealed trace length of an LS. In case an LS gets compromised, only the positions of the stored fragment are revealed instead of the complete movement trajectory.

Second, we presented our speed protection algorithms protecting mobile objects from revealing violations of given speed limits when sharing movement trajectories. To this end, the maximum speed of a shared trajectory is limited to the speed limit of the corresponding road segment.

5 Conclusion and Outlook

In this chapter, we summarize this thesis and give an outlook onto possible future research directions.

5.1 Conclusion

Driven by the widespread availability of powerful mobile devices with integrated positioning sensors, location-based applications are attracting millions of users today. For example, users share position check-ins to restaurants or their movement trajectory with friends. When sharing location information, users provide precise information to a location service managing location information of multiple users in a scalable and efficient manner. Location services provide clients, for instance different location-based application, access to their stored location information. However, sharing location information raises severe privacy concerns, especially if service providers cannot be assumed to be fully trustworthy. These concerns are intensified by the increasing number of reported privacy breaches where service providers did not succeed in protecting private user information adequately. Therefore, approaches protecting user location privacy are required, and user location information should not be exposed carelessly. Otherwise, an attacker could derive information the user is not willing to share, like his precise position, his home location, or where he is currently traveling.

In this thesis, we presented a classification of existing location privacy approaches and location privacy attacks. We analyzed which privacy approaches can be applied to achieve a certain protection goal and which attacks a privacy approach resists. To allow for privacy-aware sharing of location information, we

presented different approaches taking into account that location service providers are non-trusted.

First, we focused on the protection of user position information and introduced a novel position sharing concept. Position sharing splits up precise position information into position shares of limited precision that are distributed to multiple location services of different providers. Therefore, location services store only position information of decreased precision instead of precise position information. Nevertheless, clients can combine multiple shares to increase their precision to different well-defined precision levels up to the precise position information. Thus, position sharing improves existing obfuscation-based privacy approaches by not limiting the maximum precision that can be provided to clients by the trustworthiness of the location service. Furthermore, position sharing provides a graceful degradation of privacy in case of compromised location servers. Our first position sharing approach is based on the concept of multi-secret sharing. It supports geometric and symbolic location models and resists advanced attackers using map knowledge. Our second position sharing approach is based on the concept of binary space partitioning. This position sharing approach allows to increase the efficiency of position sharing by reducing the number of required share updates for multiple position updates significantly.

Second, we focused on the protection of movement trajectories and presented a trajectory sharing concept to allow for storing movement trajectories on non-trusted servers. Instead of providing the complete movement trajectory of a user to a single location service, our trajectory fragmentation approach splits up the user's movement trace into a set of shorter trajectory fragments that are shared among multiple location servers. The trajectory fragmentation approach minimizes the maximum revealed trace length an attacker compromising a location server can reveal. As shown in our evaluation, protecting movement trajectories is possible even against advanced attackers using map knowledge and movement prediction mechanism.

Finally, we presented algorithms protecting the speed information of movement trajectories. Movement trajectories can reveal violations of given speed

limits, which can lead to monetary or legal effects for the user. Therefore, the speed information of the user's movement trajectory must be protected when sharing movement traces. To this end, the speed protection algorithms slow down the user's speed information that can be derived from a shared movement trajectory to the maximum allowed speed limit. Thus, it is guaranteed that no speeding violation can be revealed from the user's movement trace unless the user explicitly agrees on sharing this information.

5.2 Outlook

Driven by the tremendous number of users sharing location information and the increasing number of reported privacy breaches, protecting user privacy will remain a big research topic in future. In the following, we discuss two possible research directions:

Protection of semantic locations: Shared location information typically have a certain semantic. For instance, the semantic location information of a user position can be that the user is currently located at the university or in a restaurant. However, semantic information can also be related to critical locations, such as churches, hospitals, or buildings with a political association. If users do not want to reveal that they visit such locations, privacy approaches protecting semantic location information are required. This must be possible even if location services are non-trusted and if attackers use advanced attacks. To this end, our position sharing approach taking road network and building data into account could be extended by assigning each location in a first step its semantic, and by calculating afterwards the number of effectively covered locations with a certain semantic instead of calculating the effective area size of an obfuscated position.

Preventing user profiling: Beyond revealing semantic location information, shared location information can be used by an attacker to create user profiles. For instance, a user profile can reveal how frequently a user visits a

bar or a shopping mall. Compared to the protection of semantic locations, preventing user profiling has to consider all shared information of the user instead of only certain locations. As an example, consider that visiting a bar once may represent a typical user behavior, which is not critical to be revealed. However, if the user visits the bar frequently, this could reveal a characteristical behavior of the user, his habits, or personal interests allowing to derive further information of the user. If the user does not want to reveal such information, mechanisms preventing user profiling based on revealed location information are required. This is important especially for geosocial networks, where users frequently share location information. Furthermore, it would be interesting how mechanisms providing different privacy levels of user profiles ranging from a stranger to an individual user could be implemented.

Bibliography

[ABN08] Osman Abul, Francesco Bonchi, and Mirco Nanni. Never walk alone: Uncertainty for anonymity in moving objects databases. In *Proceedings of the 24th International Conference on Data Engineering (ICDE '08)*, pages 376–385, Cancun, Mexico, April 2008.

[ACD+06] Claudio A. Ardagna, Marco Cremonini, Ernesto Damiani, Sabrina De Capitani di Vimercati, and Pierangela Samarati. Supporting location-based conditions in access control policies. In *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security (ASIACCS '06)*, pages 212–222, Taipei, Taiwan, March 2006.

[ACDS11] Claudio A. Ardagna, Marco Cremonini, Sabrina De Capitani di Vimercati, and Pierangela Samarati. An obfuscation-based approach for protecting location privacy. *IEEE Transactions on Dependable and Secure Computing*, 8(1):13–27, January 2011.

[ACG09] Claudio A. Ardagna, Marco Cremonini, and Gabriele Gianini. Landscape-aware location-privacy protection in location-based services. *Journal of Systems Architecture*, 55(4):243–254, April 2009.

[Aco09] Byron Acohido (USA Today). Hackers breach heartland payment credit card system. `www.usatoday.com/money/perfi/credit/2009-01-20-heartland-credit-card-security-breach_N.htm`, January 2009.

[ALS12] Claudio A. Ardagna, Giovanni Livraga, and Pierangela Samarati. Protecting privacy of user information in continuous location-based services. In *IEEE 15th International Conference on Computational Science and Engineering (CSE '12)*, pages 162–169, Paphos, Cyprus, December 2012.

[App14a] Apple. Find my friends. `www.apple.com/apps/find-my-friends/`, accessed January 2014.

[App14b] Apple. Iphone 5s technical specification. `www.apple.com/iphone-5s/specs/`, accessed June 2014.

[BAB+09] Ken Barker, Mina Askari, Mishtu Banerjee, Kambiz Ghazinour, Brenan Mackas, Maryam Majedi, Sampson Pun, and Adepele Williams. A data privacy taxonomy. In *Dataspace: The Final Frontier*, volume 5588 of *Lecture Notes in Computer Science*, pages 42–54. Springer Berlin Heidelberg, 2009.

[BD05] Christian Becker and Frank Dürr. On location models for ubiquitous computing. *Personal Ubiquitous Computing*, 9(1):20–31, January 2005.

[BE09] Andrew J. Blumberg and Peter Eckersly. On locational privacy, and how to avoid losing it forever. `www.eff.org/wp/locational-privacy`, April 2009.

[BKSS90] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The r*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data (SIGMOD '90)*, pages 322–331, Atlantic City, New Jersey, USA, June 1990.

[BLPW08] Bhuvan Bamba, Ling Liu, Peter Pesti, and Ting Wang. Supporting anonymous location queries in mobile environments with privacy-

grid. In *Proceeding of the 17th International Conference on World Wide Web (WWW '08)*, pages 237–246, Beijing, China, April 2008.

[BMW+09] Claudio Bettini, Sergio Mascetti, Xiaoyang Sean Wang, Dario Freni, and Sushil Jajodia. Anonymity and historical-anonymity in location-based services. In *Privacy in Location-Based Applications*, volume 5599 of *Lecture Notes in Computer Science*, pages 1–30. Springer Berlin Heidelberg, 2009.

[BS02] Piero A. Bonatti and Pierangela Samarati. A uniform framework for regulating service access and information release on the web. *Journal of Computer Security*, 10(3):241–271, September 2002.

[BS03] Alastair R. Beresford and Frank Stajano. Location privacy in pervasive computing. *IEEE Pervasive Computing*, 2(1):46–55, January 2003.

[BWJ05] Claudio Bettini, X. Sean Wang, and Sushil Jajodia. Protecting privacy against location-based personal identification. In *Secure Data Management*, volume 3674 of *Lecture Notes in Computer Science*, pages 185–199. Springer Berlin Heidelberg, 2005.

[CC05] Chao-Wen Chan and Chin-Chen Chang. A scheme for threshold multi-secret sharing. *Applied Mathematics and Computation*, 166(1):1–14, July 2005.

[CCLS11] Zhiyuan Cheng, James Caverlee, Kyumin Lee, and Daniel Z. Sui. Exploring millions of footprints in location sharing services. In *Proceedings of the 5th International AAAI Conference on Weblogs and Social Media (ICWSM 2011)*, Barcelona, Spain, July 2011.

[CCW+12] G.B. Colombo, M.J. Chorley, M.J. Williams, S.M. Allen, and R.M. Whitaker. You are where you eat: Foursquare checkins as indicators of human mobility and behaviour. In *Proceedings of the 10th*

IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom '12 Workshops), pages 217–222, Lugano, Switzerland, March 2012.

[Cle14] Privacy Rights Clearinghouse. Chronology of data breaches. `www.privacyrights.org/data-breach`, accessed June 2014.

[CM07] Chi-Yin Chow and Mohamed F. Mokbel. Enabling private continuous queries for revealed user locations. In *Proceedings of the 10th International Conference on Advances in Spatial and Temporal Databases (SSTD '07)*, pages 258–273, Boston, MA, USA, July 2007.

[CM11] Chi-Yin Chow and Mohamed F. Mokbel. Trajectory privacy in location-based services and data publication. *SIGKDD Explorations*, 13(1):19–29, August 2011.

[CMA09] Chi-Yin Chow, Mohamed F. Mokbel, and Walid G. Aref. Casper*: Query processing for location services without compromising privacy. *ACM Transactions on Database Systems*, 34(4):1–48, December 2009.

[CZBP06] Reynold Cheng, Yu Zhang, Elisa Bertino, and Sunil Prabhakar. Preserving user location privacy in mobile data management infrastructures. In *Proceedings of the 6th International Conference on Privacy Enhancing Technologies (PET '06)*, pages 393–412, Cambridge, UK, June 2006.

[DBS10] Maria Luisa Damiani, Elisa Bertino, and Claudio Silvestri. The probe framework for the personalized cloaking of private locations. *Transactions on Data Privacy*, 3(2):123–148, August 2010.

[Dea14] Deals Nearby You. Local free coupons on your mobile phone! `www.dealsnearbyyou.com`, accessed June 2014.

[DK05] Matt Duckham and Lars Kulik. A formal model of obfuscation and negotiation for location privacy. In *Proceedings of the Third International Conference on Pervasive Computing (Pervasive '05)*, pages 152–170, Munich, Germany, May 2005.

[DSB11] Maria Luisa Damiani, Claudio Silvestri, and Elisa Bertino. Fine-grained cloaking of sensitive positions in location-sharing applications. *Pervasive Computing*, 10(4):64 –72, April 2011.

[DSR11] Frank Dürr, Pavel Skvortsov, and Kurt Rothermel. Position sharing for location privacy in non-trusted systems. In *Proceedings of the 9th IEEE International Conference on Pervasive Computing and Communications (PerCom '11)*, pages 189 –196, Seattle, USA, March 2011.

[Efr13] Amir Efrati (The Wall Street Journal). Google confirms antitrust review of waze deal. `http://blogs.wsj.com/digits/2013/06/22/google-confirms-antitrust-review-of-waze-deal/`, June 2013.

[Fac14] Facebook. `www.facebook.com`, accessed June 2014.

[FB74] Raphael A. Finkel and Jon Louis Bentley. Quad trees a data structure for retrieval on composite keys. *Acta Informatica*, 4(1):1–9, April 1974.

[Fou14a] Open Security Foundation. DataLossDB. `www.datalossdb.org`, accessed June 2014.

[Fou14b] Foursquare. `www.foursquare.com`, accessed June 2014.

[Fou14c] Foursquare. About foursquare. `www.foursquare.com/about`, accessed June 2014.

[FR13] Josef Federmann and Max J. Rosenthal (Yahoo News). Waze sale signals new growth for israeli high tech. `http:`

//news.yahoo.com/waze-sale-signals-growth-israeli-high-tech-174533585.html, June 2013.

[Fre03] Elias Frentzos. Indexing objects moving on fixed networks. In *Advances in Spatial and Temporal Databases*, volume 2750 of *Lecture Notes in Computer Science*, pages 289–305. Springer Berlin Heidelberg, 2003.

[FRVM+10] Dario Freni, Carmen Ruiz Vicente, Sergio Mascetti, Claudio Bettini, and Christian S. Jensen. Preserving location and absence privacy in geo-social networks. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM '10)*, pages 309–318, Toronto, Canada, October 2010.

[GCJW10] Peter Gilbert, Landon P. Cox, Jaeyeon Jung, and David Wetherall. Toward trustworthy mobile sensing. In *Proceedings of the 11th Workshop on Mobile Computing Systems & Applications (HotMobile '10)*, pages 31–36, Annapolis, Maryland, February 2010.

[GdAD06] Hartmut Güting, Teixeira de Almeida, and Zhiming Ding. Modeling and querying moving objects in networks. *The VLDB Journal*, 15(2):165–190, June 2006.

[GDSB09] Gabriel Ghinita, Maria Luisa Damiani, Claudio Silvestri, and Elisa Bertino. Preventing velocity-based linkage attacks in location-aware applications. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS '09)*, pages 246–255, Seattle, USA, November 2009.

[GDVM09] Aris Gkoulalas-Divanis, Vassilios S. Verykios, and Mohamed F. Mokbel. Identifying unsafe routes for network-based trajectory privacy. In *Proceedings of the 9th SIAM International Conference on Data Mining (SDM '09)*, pages 942–953, Sparks, Nevada, USA, April 2009.

[GG03] Marco Gruteser and Dirk Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services (MobiSys '03)*, pages 31–42, San Francisco, California, USA, May 2003.

[GHP07] Gyözö Gidofalvi, Xuegang Huang, and Torben Bach Pedersen. Privacy-preserving data mining on moving object trajectories. In *Proceedings of the 2007 International Conference on Mobile Data Management (MDM '07)*, pages 60–68, Mannheim, Germany, May 2007.

[GKdPC10] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. Show me how you move and i will tell you who you are. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS (SPRINGL '10)*, pages 34–41, San Jose, California, USA, November 2010.

[GKK+08] Gabriel Ghinita, Panos Kalnis, Ali Khoshgozaran, Cyrus Shahabi, and Kian-Lee Tan. Private queries in location based services: anonymizers are not necessary. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD '08)*, pages 121–132, Vancouver, Canada, June 2008.

[GL04] Marco Gruteser and Xuan Liu. Protecting privacy in continuous location-tracking applications. *IEEE Security Privacy*, 2(2):28–34, March 2004.

[GL08] Buğra Gedik and Ling Liu. Protecting location privacy with personalized k-anonymity: Architecture and algorithms. *IEEE Transactions on Mobile Computing*, 7(1):1–18, January 2008.

[Gly13] Glympse Inc. Localization support for global audience; company

surpasses 10 million users milestone. `www.glympse.com/news/press/15`, August 2013.

[Gly14] Glympse Inc. `www.glympse.com`, accessed June 2014.

[Goo14a] Google. Introducing the all-new 5" phone from google. `www.google.com/nexus/`, accessed June 2014.

[Goo14b] Google Maps. `www.maps.google.com`, accessed June 2014.

[Goo14c] Google Plus. `www.plus.google.com`, accessed June 2014.

[GP09] Philippe Golle and Kurt Partridge. On the anonymity of home / work location pairs. In *Proceedings of the 7th International Conference on Pervasive Computing (Pervasive '09)*, pages 390–397, Nara, Japan, May 2009.

[Gut84] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data (SIGMOD '84)*, pages 47–57, Boston, Massachusetts, USA, June 1984.

[GZPK10] Gabriel Ghinita, Keliang Zhao, Dimitris Papadias, and Panos Kalnis. A reciprocal framework for spatial k-anonymity. *Information Systems*, 35(3):299–314, May 2010.

[HG05] Baik Hoh and Marco Gruteser. Protecting location privacy through path confusion. In *First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SecureComm '05)*, pages 194–205, Athens, Greece, September 2005.

[HGH+08] Baik Hoh, Marco Gruteser, Ryan Herring, Jeff Ban, Daniel Work, Juan-Carlos Herrera, Alexandre M. Bayen, Murali Annavaram, and Quinn Jacobson. Virtual trip lines for distributed privacy-preserving traffic monitoring. In *Proceeding of the 6th Interna-*

tional Conference on Mobile Systems, Applications, and Services (MobiSys '08), pages 15–28, Breckenridge, CO, USA, June 2008.

[HGXA06] Baik Hoh, Marco Gruteser, Hui Xiong, and Ansaf Alrabady. Enhancing security and privacy in traffic-monitoring systems. *IEEE Pervasive Computing*, 5(4):38–46, October 2006.

[HGXA10] Baik Hoh, Marco Gruteser, Hui Xiong, and Ansaf Alrabady. Achieving guaranteed anonymity in gps traces via uncertainty-aware path cloaking. *IEEE Transactions on Mobile Computing*, 9(8):1089–1107, August 2010.

[HL06] Haibo Hu and Dik Lun Lee. Range nearest-neighbor query. *IEEE Transactions on Knowledge and Data Engineering*, 18(1):78–91, January 2006.

[Hol14] Martin Holland (Heise Verlag). Shazam räumt weitergabe von nutzerdaten ein. `http://heise.de/-2125289`, February 2014.

[HOO+14] Peter Händel, Jens Ohlsson, Martin Ohlsson, Isaac Skog, and Elin Nygren. Smartphone based measurement systems for road vehicle traffic monitoring and usage-based insurance. *IEEE Systems Journal*, 8(4):1238–1248, December 2014.

[HS05] Urs Hengartner and Peter Steenkiste. Access control to people location information. *ACM Transactions on Information and System Security*, 8(4):424–456, November 2005.

[IMI10] Sergio Ilarri, Eduardo Mena, and Arantza Illarramendi. Location-dependent query processing: Where we are and where we are heading. *ACM Computing Surveys*, 42(3):1–73, March 2010.

[Ins14a] Instagram. `www.instagram.com`, accessed June 2014.

[Ins14b] Environmental Systems Research Institute. Geoloqi. `www.geoloqi.com`, accessed June 2014.

[Int14] Intelligent Apps GmbH. Mytaxi. `www.mytaxi.com`, accessed June 2014.

[JKPT03] Christian S. Jensen, Jan Kolářvr, Torben Bach Pedersen, and Igor Timko. Nearest neighbor queries in road networks. In *Proceedings of the 11th ACM International Symposium on Advances in Geographic Information Systems (GIS '03)*, pages 1–8, New Orleans, Louisiana, USA, November 2003.

[JQJ07] Oliver Jorns, Gerald Quirchmayr, and Oliver Jung. A privacy enhancing mechanism based on pseudonyms for identity protection in location-based services. In *Proceedings of the Fifth Australasian Symposium on ACSW Frontiers (ACSW '07)*, volume 68, pages 133–142, Ballarat, Victoria, Australia, January 2007.

[KGMP07] Panos Kalnis, Gabriel Ghinita, Kyriakos Mouratidis, and Dimitris Papadias. Preventing location-based identity inference in anonymous spatial queries. *IEEE Transactions on Knowledge and Data Engineering*, 19(12):1719–1733, December 2007.

[Kru07] John Krumm. Inference attacks on location tracks. In *Proceedings of the 5th International Conference on Pervasive Computing (Pervasive '07)*, pages 127–143, Toronto, Canada, May 2007.

[Kru08] John Krumm. A markov model for driver turn prediction. In *Society of Automotive Engineers (SAE) World Congress*, Detroit, Michigan, USA, April 2008.

[Kru09] John Krumm. A survey of computational location privacy. *Personal and Ubiquitous Computing*, 13(6):391–399, August 2009.

[KS10] Ali Khoshgozaran and Cyrus Shahabi. A taxonomy of approaches to preserve location privacy in location-based services. *International Journal of Computational Science and Engineering*, 5(2):86–96, November 2010.

[KV04] Jonathan Krim and David A. Vise (Washington Post). AOL employee charged in theft of screen names. `www.washingtonpost.com/wp-dyn/articles/A860-2004Jun23.html`, June 2004.

[KYS05] Hidetoshi Kido, Yutaka Yanagisawa, and Tetsuji Satoh. An anonymous communication technique using dummies for location-based services. In *Proceedings of the International Conference on Pervasive Services (ICPS '05)*, pages 88–97, Santorini, Greece, July 2005.

[LLV07] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *Proceedings of the 23rd IEEE International Conference on Data Engineering (ICDE '07)*, pages 106–115, Istanbul, Turkey, April 2007.

[LM98] Ulf Leonhardt and Jeff Magee. Security considerations for a distributed location service. *Journal of Network and Systems Management*, 6(1):51–70, March 1998.

[LNR02] Alexander Leonhardi, Christian Nicu, and Kurt Rothermel. A map-based dead-reckoning protocol for updating location information. In *Proceedings of the 16th International Parallel and Distributed Processing Symposium (IPDPS '02)*, Ft. Lauderdale, Florida, USA, April 2002.

[LR01] Alexander Leonhardi and Kurt Rothermel. Architecture of a large-scale location service. Technical Report 2001/01, Faculty of Computer Science, University of Stuttgart, Germany, January 2001.

[LZP+12] Xinxin Liu, Han Zhao, Miao Pan, Hao Yue, Xiaolin Li, and Yuguang Fang. Traffic-aware multiple mix zone placement for protecting location privacy. In *Proceedings of the 31st Annual IEEE International Conference on Computer Communications (INFOCOM '12)*, pages 972–980, Orlando, Florida, USA, March 2012.

[MAA+10] Anna Monreale, Gennady Andrienko, Natalia Andrienko, Fosca Giannotti, Dino Pedreschi, Salvatore Rinzivillo, and Stefan Wrobel. Movement data anonymity through generalization. *Transactions on Data Privacy*, 3(2):91–121, August 2010.

[Map14] MapMyFitness Inc. Mapmyfitness. `www.mapmyfitness.com`, accessed June 2014.

[MBW+09] Sergio Mascetti, Claudio Bettini, X. Sean Wang, Dario Freni, and Sushil Jajodia. ProvidentHider: An algorithm to preserve historical k-anonymity in LBS. In *Proceedings of the 10th IEEE International Conference on Mobile Data Management (MDM '09)*, pages 172–181, Taipei, Taiwan, May 2009.

[MDKG05] Giannis F. Marias, Constantinos Delakouridis, Leonidas Kazatzopoulos, and Panagiotis Georgiadis. Location privacy through secret sharing techniques. In *Proceedings of the 1st International IEEE WoWMoM Workshop on Trust, Security and Privacy for Ubiquitous Computing (WOWMOM '05)*, volume 3, pages 614–620, Taormina, Sicily, Italy, June 2005.

[Met14] MetroMile Inc. Metromile. `www.metromile.com`, accessed June 2014.

[MFB+11] Sergio Mascetti, Dario Freni, Claudio Bettini, X. Sean Wang, and Sushil Jajodia. Privacy in geo-social networks: proximity notification with untrusted service providers and curious buddies. *The VLDB Journal*, 20(4):541–566, August 2011.

[MGKV06] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE '06)*, pages 24–35, Atlanta, Georgia, USA, April 2006.

[MHVB13] Yves-Alexandre de Montjoye, César A. Hidalgo, Michel Verleysen, and Vincent D. Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific Reports*, 3(1376):1–5, March 2013.

[MRC09] Joseph Meyerowitz and Romit Roy Choudhury. Hiding stars with fireworks: Location privacy through camouflage. In *Proceedings of the 15th Annual International Conference on Mobile Computing and Networking (MobiCom '09)*, pages 345–356, Beijing, China, September 2009.

[MYYR10] Chris Y.T. Ma, David K.Y. Yau, Nung Kwan Yip, and Nageswara S.V. Rao. Privacy vulnerability of published anonymous mobility traces. In *Proceedings of the 16th Annual International Conference on Mobile Computing and Networking (MobiCom '10)*, pages 185–196, Chicago, Illinois, USA, September 2010.

[NASG09] Mehmet Ercan Nergiz, Maurizio Atzori, Yücel Saygin, and Baris Güç. Towards trajectory anonymization: a generalization-based approach. *Transactions on Data Privacy*, 2(1):47–75, April 2009.

[Nat14] National General Insurance. Smartdiscounts for smart drivers. `www.nationalgeneral.com/auto-insurance/smart-discounts.asp`, accessed June 2014.

[Ope14] OpenStreetMap. `www.openstreetmap.org`, accessed June 2014.

[PBB09] Raluca Ada Popa, Hari Balakrishnan, and Andrew J. Blumberg. Vpriv: Protecting privacy in location-based vehicular services. In *Proceedings of the 18th Conference on USENIX Security Symposium (SSYM '09)*, pages 335–350, Montreal, Canada, August 2009.

[Pil09] Ed Pilkington (The Guardian). Us hacker charged with stealing 130m credit card ids. `www.theguardian.com/world/2009/aug/18/american-credit-card-hacker`, August 2009.

[PJT00] Dieter Pfoser, Christian S. Jensen, and Yannis Theodoridis. Novel approaches in query processing for moving object trajectories. In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB '00)*, pages 395–406, Cairo, Egypt, 2000.

[PL11] Balaji Palanisamy and Ling Liu. Mobimix: Protecting location privacy with mix-zones over road networks. In *Proceedings of the 27th IEEE International Conference on Data Engineering (ICDE '11)*, pages 494–505, Hannover, Germany, April 2011.

[Pre01] The Associated Press (USA Today). Gps system used to fine driver for speeding. `www.usatoday.com/tech/news/2001-07-03-car-tracking.htm`, March 2001.

[Pre14] International Data Corporation (IDC) Press. Worldwide smartphone shipments top one billion units for the first time, according to idc. `www.idc.com/getdoc.jsp?containerId=prUS24645514`, January 2014.

[Pro14] PriLoc Project. `www.PriLoc.de`, accessed June 2014.

[PS11] Sai Teja Peddinti and Nitesh Saxena. On the limitations of query obfuscation techniques for location privacy. In *Proceedings of the 13th International Conference on Ubiquitous Computing (UbiComp '11)*, pages 187–196, Beijing, China, September 2011.

[PSDG09] Michal Piorkowski, Natasa Sarafijanovoc-Djukic, and Matthias Grossglauser. A parsimonious model of mobile partitioned networks with clustering. In *Proceedings of the 1st International Conference on COMmunication Systems and NETworkS (COMSNETS '09)*, pages 1–10, Bangalore, India, January 2009.

[PXM12] Xiao Pan, Jianliang Xu, and Xiaofeng Meng. Protecting location privacy against location-dependent attacks in mobile services. *IEEE*

Transactions on Knowledge and Data Engineering, 24(8):1506–1519, August 2012.

[PZMT03] Dimitris Papadias, Jun Zhang, Nikos Mamoulis, and Yufei Tao. Query processing in spatial network databases. In *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB '03)*, volume 29, pages 802–813, Berlin, Germany, September 2003.

[QON07] Mohammed A. Quddus, Washington Y. Ochieng, and Robert B. Noland. Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation Research Part C: Emerging Technologies*, 15(5):312 – 328, October 2007.

[RFSK08] Stefan Rass, Simone Fuchs, Martin Schaffer, and Kyandoghere Kyamakya. How to protect privacy in floating car data systems. In *Proceedings of the Fifth ACM International Workshop on VehiculAr Inter-NETworking (VANET '08)*, pages 17–22, San Francisco, California, USA, September 2008.

[Say08] Peter Sayer (New York Times). T-mobile lost disk containing data on 17 million customers. `www.nytimes.com/idg/IDG_852573C400693880002574DA0034AE43.html`, October 2008.

[SDFMB08] Agusti Solanas, Josep Domingo-Ferrer, and Antoni Martínez-Ballesté. Location privacy in location-based services: Beyond ttp-based schemes. In *International Workshop on Privacy in Location-Based Applications (PILBA '08)*, pages 12–23, Malaga, Spain, October 2008.

[SDR12] Pavel Skvortsov, Frank Dürr, and Kurt Rothermel. Map-aware position sharing for location privacy in non-trusted systems. In *Pervasive Computing*, volume 7319 of *Lecture Notes in Computer Science*, pages 388–405. Springer Berlin Heidelberg, 2012.

[Seg11] Laurie Segall (CNN Money). Facebook buys gowalla. `http://money.cnn.com/2011/12/02/technology/gowalla_facebook/`, December 2011.

[SGI09] Pravin Shankar, Vinod Ganapathy, and Liviu Iftode. Privately querying location-based services with sybilquery. In *Proceedings of the 11th International Conference on Ubiquitous Computing (Ubicomp '09)*, pages 31–40, Orlando, Florida, USA, 2009.

[Sha49] Claude E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, October 1949.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, November 1979.

[STLBH11] Reza Shokri, George Theodorakopoulos, Jean-Yves Le Boudec, and Jean-Pierre Hubaux. Quantifying location privacy. In *Proceedings of the 31st IEEE Symposium on Security and Privacy (SP '11)*, pages 247 –262, Berkeley, California, USA, May 2011.

[TA10] Nilothpal Talukder and Sheikh Iqbal Ahamed. Preventing multi-query attack in location-based services. In *Proceedings of the third ACM Conference on Wireless Network Security (WiSec '10)*, pages 25–36, Hoboken, New Jersey, USA, March 2010.

[TM08] Manolis Terrovitis and Nikos Mamoulis. Privacy preservation in the publication of trajectories. In *Proceedings of the 9th International Conference on Mobile Data Management (MDM '08)*, pages 65–72, Beijing, China, April 2008.

[Tom14] TomTom International. `www.tomtom.com/en_gb/licensing/products/traffic/real-time-traffic/`, accessed June 2014.

[Twi14] Twitter Inc. `www.twitter.com`, accessed June 2014.

[Van13] Kyle Vanhemert (Wired.com). Amazing maps of 3 billion tweets reveal iphone vs. android neighborhoods. `www.wired.com/design/2013/07/design_06272013_tweetmaps/`, July 2013.

[Waz14] Waze Ltd. `www.waze.com`, accessed June 2014.

[WDR12] Marius Wernke, Frank Dürr, and Kurt Rothermel. PShare: position sharing for location privacy based on multi-secret sharing. In *Proceedings of the 10th IEEE International Conference on Pervasive Computing and Communications (PerCom '12)*, pages 153 – 161, Lugano, Switzerland, March 2012.

[WDR13a] Marius Wernke, Frank Dürr, and Kurt Rothermel. Efficient position sharing for location privacy using binary space partitioning. In *Mobile and Ubiquitous Systems: Computing, Networking, and Services*, volume 120 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 263–275. Springer Berlin Heidelberg, 2013.

[WDR13b] Marius Wernke, Frank Dürr, and Kurt Rothermel. PShare: ensuring location privacy in non-trusted systems through multi-secret sharing. *Pervasive and Mobile Computing*, 9(3):339 – 352, June 2013.

[WDR13c] Marius Wernke, Frank Dürr, and Kurt Rothermel. Speed protection algorithms for privacy-aware location management. In *Proceedings of the IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob '13)*, pages 355–362, Lyon, France, October 2013.

[WDR14] Marius Wernke, Frank Dürr, and Kurt Rothermel. Protecting movement trajectories through fragmentation. In *Mobile and Ubiquitous Systems: Computing, Networking, and Services*, volume 131 of *Lecture Notes of the Institute for Computer Sciences, Social*

Informatics and Telecommunications Engineering, pages 303–315. Springer International Publishing, 2014.

[Web10] Webroot Inc. Webroot survey finds geolocation apps prevalent amongst mobile device users, but 55% concerned about loss of privacy. `www.webroot.com/us/en/company/press-room/releases/social-networks-mobile-security`, July 2010.

[Wil11] Christopher Williams (The Telegraph). Police use tomtom data to target speed traps. `www.telegraph.co.uk/technology/news/8480195/Police-use-TomTom-data-to-target-speed-traps.html`, April 2011.

[WL09] Ting Wang and Ling Liu. Privacy-aware mobile services over road networks. *Proceedings of the VLDB Endowment*, 2(1):1042–1053, August 2009.

[WSDR14] Marius Wernke, Pavel Skvortsov, Frank Dürr, and Kurt Rothermel. A classification of location privacy attacks and approaches. *Personal and Ubiquitous Computing*, 18(1):163–175, January 2014.

[WZ08] Haojun Wang and Roger Zimmermann. Snapshot location-based query processing on moving objects in road networks. In *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS '08)*, number 50, pages 1–4, Irvine, California, November 2008.

[WZ11] Haojun Wang and Roger Zimmermann. Processing of continuous location-based range queries on moving objects in road networks. *IEEE Transactions on Knowledge and Data Engineering*, 23(7):1065–1078, July 2011.

[XXM08] Zhen Xiao, Jianliang Xu, and Xiaofeng Meng. p-sensitivity: A semantic privacy-protection model for location-based services.

In *Proceedings of the Ninth International Conference on Mobile Data Management Workshops (MDMW '08)*, pages 47–54, Beijing, China, April 2008.

[XZZ+13] Andy Yuan Xue, Rui Zhang, Yu Zheng, Xing Xie, Jin Huang, and Zhenghua Xu. Destination prediction by sub-trajectory synthesis and privacy protection against such prediction. In *Proceedings of the 2013 IEEE International Conference on Data Engineering (ICDE '13)*, pages 254–265, Brisbane, Australia, April 2013.

[YDAS12] Emre Yigitoglu, Maria Luisa Damiani, Osman Abul, and Claudio Silvestri. Privacy-preserving sharing of sensitive semantic locations under road-network constraints. In *Proceedings of the IEEE 13th International Conference on Mobile Data Management (MDM '12)*, pages 186–195, Bengaluru, India, July 2012.

[Yel14] Yelp Inc. `www.yelp.com`, accessed June 2014.

[YJML11] Man Lung Yiu, Christian S. Jensen, Jesper Møller, and Hua Lu. Design and analysis of a ranking approach to private location-based services. *ACM Transactions on Database Systems*, 36(2):1–42, May 2011.

[YPL07] Tun-Hao You, Wen-Chih Peng, and Wang-Chien Lee. Protecting moving trajectories with dummies. In *Proceedings of the 8th International Conference on Mobile Data Management (MDM '07)*, pages 278–282, Mannheim, Germany, May 2007.

[ZB11] Hui Zang and Jean Bolot. Anonymization of location data does not work: A large-scale measurement study. In *Proceedings of the 17th Annual International Conference on Mobile Computing and Networking (MobiCom '11)*, pages 145–156, Las Vegas, Nevada, USA, September 2011.