Universität Stuttgart

Position Sharing for Location Privacy in Non-trusted Systems

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

Vorgelegt von

Pavel Skvortsov

aus Donezk

Hauptberichter:	Prof. Dr. rer. nat. Dr. h. c. Kurt Rothermel
Mitberichter:	Prof. Dr.-Ing. Dr. h. c. Peter Göhner
Tag der mündlichen Prüfung:	30.04.2015

Institut für Parallele und Verteilte Systeme (IPVS)
der Universität Stuttgart
2015

Contents

ACKNOWLEDGEMENTS

I would like to gratefully thank Prof. Kurt Rothermel from the University of Stuttgart and Prof. Vladimir Svjatnyj from the Donetsk National Technical University for granting me the opportunity to conduct this research. Prof. Kurt Rothermel carefully supervised my work, especially in the early stages, which allowed this work to become of great scientific significance. Special thanks go to my supervisor Frank Dürr for his advice on improving the content and structure of this dissertation, as well as for his collaboration on scientific publications in the process.

I would also like to thank the students who were engaged in this work at various stages: Björn Schembera, Andreas Paul, Daniel del Hoyo and Simon Hänle.

I have enjoyed working with my colleagues from the Distributed Systems department including Ralph Lange, Marius Wernke, Stamatia Rizou, Stefan Föll, Lars Geiger, Harald Weinschrott and many others.

In terms of financial support, I am thankful for the scholarship from the land of Baden-Württemberg, and the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG), whose project Nexus gave rise to the topic of this work.

For their continued emotional and all-round support, I would like to thank my family – especially during my long years of study in school and at the Donetsk National Technical University.

Finally, I would like to especially thank Ben Carabelli and Ralph Lange who read my work prior to submission, and therefore gave many helpful suggestions to improve it.

Kurzfassung (German)

Ortsbezogene Dienste (engl. *location-based services*) dienen dazu, die aktuelle geographische Position des Nutzers zu bestimmen und im Rahmen einer Anwendung zu nutzen. Heutzutage sind viele ortsbezogene Anwendungen für Nutzer mobiler Endgeräte verfügbar und weit verbreitet, wie z.B. Google Now, Trace4You oder FourSquare. Diese Anwendungen sind auch in verschiedenen Umgebungen eingesetzt, in denen Positionsprivatheit ein kritisches Thema für Benutzer ist. Eine allgemeine Lösung für die Sicherung der Positionsprivatheit eines Benutzers ist, seine Positionsinformationen von geringerer Genauigkeit zu veröffentlichen. In dieser Arbeit schlagen wir einen Ansatz vor, der räumliche Verschleierung benutzt, um Positionsprivatheit mobiler Nutzer zu sichern.

Nach der Offenlegung der Position des Benutzers mit einem bestimmten Verschleierungsgrad ist der Kompromiss zwischen Datenschutz und Genauigkeit sehr wichtig, da das Vertrauen in die Dienstanbieter begrenzt ist. Eine höhere Verschleierung erhöht Positionsprivatheit, führt aber zu geringerer Qualität der Dienstleistung. Wir bieten das "Position Sharing"-Verfahren an, um dieses Problem zu lösen. Die Grundidee ist, dass Positionsinformationen zwischen mehreren Dienstanbietern in Form von separaten Datenstücken (in dieser Arbeit als Positionsshares bezeichnet) verteilt werden. Unser Ansatz ermöglicht die Nutzung von nichtvertrauenswürdigen Dienstanbietern und verwaltet flexibel mehrere Stufen des Datenschutzes für Benutzerpositionen, die auf probabilistischen Privatheitsmetriken basieren. In dieser Arbeit präsentieren wir den "Position Sharing" Ansatz für mehrere Dienstanbieter der ortsbezogenen Dienste, der die Algorithmen zur Erzeugung und Kombination von Positionsshares beinhaltet.

Eine wichtige Herausforderung im Rahmen des Ansatzes ist, dass der Umgebungskontext des Benutzers deutlich das Niveau der Verschleierung verringern kann. D.h., ein Flugzeug, ein Boot oder ein Auto stellt unterschiedliche Anforderungen an die zu verschleiernden Gebiete

dar. Deswegen ist es nötig, die Karteninformationen bei der Auswahl der verschleierten Gebiete zu berücksichtigen. Wir gehen davon aus, dass eine statische Karte einem Angreifer bekannt ist, die die echte Benutzerposition enthüllen kann. Wir analysieren, wie sich die Karteninformation auf die Erzeugung und Fusion der Positionsshares auswirkt. Wir zeigen auch den Unterschied zwischen dem kartebewussten “Position Sharing” Ansatz und seiner Version für unstrukturierte Gebiete. Unsere Sicherheitsanalyse zeigt, dass der vorgeschlagene “Position Sharing”-Ansatz gute Sicherheitsgarantien für unstrukturierte sowie strukturierte Raummodelle anbietet.

Die nächste Herausforderung ist, dass mehrere Positionsserver bzw. Serveranbieter unterschiedliche Vertrauenswürdigkeitswerte aus der Sicht des Benutzers haben können. In diesem Fall möchte der Benutzer unterschiedliche Genauigkeitsgrade der Positionsinformationen an jedem einzelnen Positionsserver offenlegen. Wir schlagen einen Ansatz für Platzierungsoptimierung vor, der sicherstellt, dass das Risiko der Positionsserver nach den individuellen Vertrauensniveaus ausgeglichen wird. Unsere Evaluierung zeigt eine signifikante Verbesserung der Positionsprivatheit nach der Anwendung der optimierten Shareverteilung, im Vergleich zu der Gleichverteilung der Shares.

Das letzte betrachtete Problem ist das Lokationsupdateverfahren. Laut unserem Basisansatz kann dies zu einem erheblichen Kommunikationsaufwand führen, wenn die Anzahl der unterschiedlichen Positionsserver (und der entsprechenden Privatheitsniveaus) n hoch ist: Jedes Update würde n Nachrichten von jedem mobilen Nutzer an die Positionsserver erfordern, vor allem im Fall einer hohen Aktualisierungsrate. Wir bieten daher ein optimiertes Lokationsupdateverfahren an, um die Anzahl der Nachrichten so zu verringern, dass die Positionsprivatheit der mobilen Nutzer unverändert bleibt.

Abstract

Currently, many location-aware applications are available for mobile users of location-based services. Applications such as Google Now, Trace4You or FourSquare are being widely used in various environments where privacy is a critical issue for users. A general solution for preserving location privacy for a user is to degrade the quality of his or her position information. In this work, we propose an approach that uses spatial obfuscation to secure the users' position information. By revealing the user's position with a certain degree of obfuscation, the *first* crucial issue is the tradeoff between privacy and precision. This tradeoff problem is caused by limited trust in the location service providers: higher obfuscation increases privacy but leads to lower quality of service. We overcome this problem by introducing the position sharing approach. Our main idea is that position information is distributed amongst multiple providers in the form of separate data pieces called position shares. Our approach allows for the usage of non-trusted providers and flexibly manages the user's location privacy level based on probabilistic privacy metrics. In this work, we present the multi-provider based position sharing approach, which includes algorithms for the generation of position shares and share fusion algorithms.

The *second* challenge that must be addressed is that the user's environmental context can significantly decrease the level of obfuscation. For example, a plane, a boat and a car create different requirements for the obfuscated region. Therefore, it is very important to consider map-awareness in selecting the obfuscated areas. We assume that a static map is known to an adversary, which may help in deriving the user's true position. We analyze both how map-awareness affects the generation and fusion of position shares and the difference between the map-aware position sharing approach and its open space based version. Our security analysis shows that the proposed position sharing approach provides good security guarantees for both open space and constrained space based models.

The *third* challenge is that multiple location servers and/or their providers may have different trustworthiness from the user's point of view. In this case, the user would prefer not to reveal an equal level (precision) of position information to every server. We propose a placement optimization approach that ensures that risk is balanced among the location servers according to their individual trust levels. Our evaluation shows significant improvement of privacy guarantees after applying the optimized share distribution, in comparison with the equal share distribution.

The *fourth* related problem is the location update algorithm. A high number of different location servers n (corresponding to n privacy levels) may lead to significant communication overhead. Each update would require n messages from the mobile user to the location servers, especially in cases of high update rate. Therefore, we propose an optimized location update algorithm to decrease the number of messages sent without reducing the number of privacy levels and the user's privacy.

Chapter 1 Introduction

In this chapter, we begin by presenting basic information about location-based services using private user information. Then, we explain the need for preserving the user's location privacy while using the location-based services. Finally, we outline the goals and main features of our approach.

1.1. Background: Location-Based Services

During the last few decades, we have experienced an unprecedented increase in access to digital geographical information. The major contributing factors include the advance of cloud services to store personal data remotely, the increased availability of fast internet connection and the widespread usage of mobile devices, in addition to the common usage of geographic positioning. The combined utilization of these modern technologies has allowed to create a large variety of applications for users. Among them, location-based applications (LBAs) are very popular nowadays. The availability of the user's position makes services possible which users could only dream of before: automated navigation with advanced geographical maps, geo-social networking, search for places and locations by the given criteria, etc.

LBAs are supported by location services, which store the positions of mobile objects (MOs) at location servers (LSs). At the same time, powerful mobile devices such as mobile phones, smartphones, PDAs and tablet computers are becoming increasingly widespread. Such mobile devices offer high processing power, large memory capacity and an integrated positioning system—for example, the satellite-based Global Positioning System (GPS), which was deployed between 1989 and 1994. Similarly to the Internet, GPS was initially developed for military use and then later made available for civilian and commercial use. The widespread flat-rate tariffs for network communication make the use of location-based applications affordable for many users.

A location-based service (LBS) is a service for the users of mobile devices, which allows for exchanging and processing the users' location data through the mobile network. For example, web services such as Google Now [Goo15], InstaMapper [Ins14], Trace4You [Tra14], Facebook Places [Fac15] and FourSquare [Fou14] support queries for obtaining a user's nearest neighbors, local points of interests or a set of friends in a given area, etc. The convergence of technologies which enables the functioning of an LBS is shown in Figure 1.1 [Mok07]. We can see that the use of advanced mobile devices with Internet access enables the mobile Internet. If spatial databases become available over the Internet, they are known as Web-GIS (Geographic Information System). Overall, the cooperation of all these technologies forms what is called the LBS. The basic system architecture of an LBS (Figure 1.2) consisting of mobile device, positioning system, location server and communication network [VMG$^+$01, RM03, SNE06]. Below, we describe each major component in more detail.

Mobile device. It is an MO carried by mobile user, which can be used to request various services and send them the required information. Today, the most widespread example of such a device is a smartphone, i.e., a mobile phone with advanced functionality including a GPS receiver.

Positioning system. This system allows the mobile device to automatically determine its position locally. The methods for determining the position may vary: for example, the

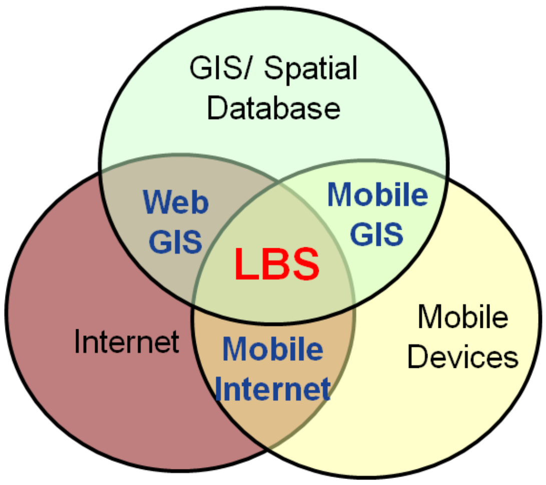


Figure 1.1.: Convergence of technologies for creating an LBS [Mok07]

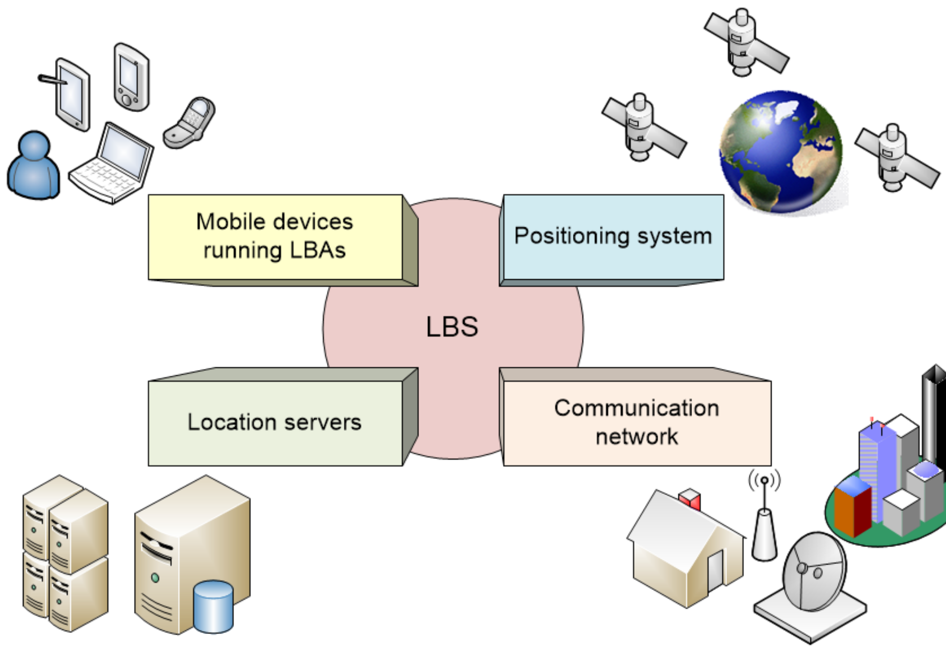


Figure 1.2.: System architecture of an LBS

localization can be done through GPS, or through a mobile radio system, which provides the ID of the current mobile radio cell (the cell ID).

Location server (LS). An LS is responsible for managing the position information of mobile devices and provides this information to the LBAs. Thus, LBAs make use of the location-based information stored in a spatial database at the LSs. The LS stores at least the current (i.e., the last known) position of each tracked mobile object; however, it can also store their movement history. Furthermore, LSs can store and manage the positions of various static objects, for instance, by providing detailed map knowledge from the spatial database.

Finally, a **communication network** is needed between the system components in order to enable the exchange of information between them.

It should be noted that a simple LBS could also operate without having an LS. In such case, the positions of mobile devices are sent directly to the LBS, and the mobile devices receive the requested information related to their current positions. However, as soon as position information is shared by multiple LBAs, LSs are necessary to store this information. There are two major advantages if LSs are available in the system and they store spatial database of multiple user positions: first, the availability of LSs provides *scalability* and *efficiency*, since multiple LBAs can share information about the MO's position updates; second, it allows for spatial queries over *multiple moving objects* to be implemented and processed.

The LBS user allows LBS to obtain position information of his or her mobile device by communicating through the mobile network. In return, mobile users get access to services provided by the LBS [VMG+01, ACD+07]. In other words, LBS offers a variety of services for mobile users, by making use of their position information. A typical service is when a user requests location information about an interesting object called a Point of Interest (POI), for example, searching for nearby hotels or getting information about the current traffic situation in the locality. Such services are pro-active (query-based). Other services are reactive (event-based), i.e., they run in the background and react to events such as a shopping center visit, which could trigger a location-based advertisement. In both cases, location-based services analyze the user's environmental context primarily depending on his or her position.

1.2. Motivation

As we have shown, to answer location-based queries of a mobile user, LBAs require the mobile user to reveal his or her position. The position information sent by a user to the LBS should be precise enough to provide an acceptable quality of service for the variety of location-based applications. The user's location privacy is a critical issue, since the user's position must be provided to a third party—LSs and/or service providers—which are usually considered non-trusted from the user's point of view. An LS can be compromised by an attacker, or its provider can be malicious and misuse the user's private information. Therefore, it is important to have alternative solutions in order to provide better security.

According to the well-known definition of privacy for information handling by Alan Westin [Wes67], privacy is the user's ability to determine independently how to deal with his or her own information. In other words, the user must be able to protect the information from unauthorized access and undesired processing. Simply put, the user wants to maintain control over his or her personal information, including position information.

The need for location privacy was studied by Brush et al. [BKS10]. The majority of respondents in that study said that they were willing to disclose their location information if it was in the public (i.e., not personal) interest such as for managing traffic jams or for planning future bus routes. However, in those cases, when the processed information was very personal, the respondents were concerned about their privacy (e.g., while using services such as a recommended place advisor service or daily route tracking).

A large number of examples of leaked information exist; there are web-resources that monitor such cases on a daily basis [Pri14]. Moreover, the recent WikiLeaks and PRISM scandals [Wik14, BBC14, The14] show not only that privately shared data can be maliciously tracked and later disclosed but also that even well-protected centralized data storage facilities of high-level governmental institutions cannot be trusted, and therefore they cannot guarantee 100% data security and resilience against attacks.

Further critical problems exist in addition to problems of unauthorized or undesired disclosure of information. If other persons can access the user's position at any time, this lack of security may have consequences for his or her personal well-being [DK06]. Thus, the user's position is always connected with privacy and security. Moreover, by using the information obtained about visited locations and POIs, an adversary can create a user profile, which will include the social behavior, health status, and personal interests of the targeted user [DK06, DF03]. For example, if it is known that a politically-sensitive event was held at the same place and time where a person was located, information about his or her political views can be derived. As a result, people can be persecuted by third parties based on the obtained position information and location-based context information. Dobson and Fisher describe "geoslavery" as a potential undermining of privacy through the use of LBS, and warn of the possibility of dire consequences if the people's location privacy continues to be undermined [DF03]. In particular, the authors describe "stalking" as a situation in which a person is pursued or harassed to be then forced to perform actions against his or her will.

The protection of privacy in LBS should be provided at different levels [Eyo08]:

First, personal privacy protection must be highlighted to members of society, so that people are aware of and can assess the potential dangers and possible consequences related to the privacy-critical services available and offered to them.

Second, a legal framework to regulate the protection exists. For example, § 98 of the Germany's Telecommunications Act [Bun14] states that the use of location information is prohibited if the target person has not agreed to such usage. Anyone who acts contrary to this principle can be prosecuted.

However, these rules and policies are not sufficient for privacy protection, since the legal framework may lag behind the technological developments over time. Moreover, the legal framework can be consciously or unconsciously betrayed and undermined. There are many recent incidents which illustrate such behavior; for example, the case of Apple smartphones, which stored all the user's position data without authorization and notification. Moreover, since legal prosecution generally involves long delays after the attack occurred and is therefore not a sufficient countermeasure, because the dire consequences have already taken place with regard to the victim. Thus, the problem of LBS users' privacy must be ensured through technical measures.

In this work, we propose an approach that allows an LBA to define the level of position precision required, while at the same time it allows the user to preserve his or her location privacy according to individual preferences. This approach includes flexible management of the user's position precision and the corresponding location privacy levels. The flexible management is achieved by utilizing multiple LSs of different providers. Emerging technology trends such as federated systems [CB04] and "mashups" (web application hybrids) [ZP07] show that multiple providers can cooperate. The increasing availability of large distributed (including cloud-based) infrastructures at a reasonable price provides scalable and efficient management of large amounts of location data. Suitable infrastructures are already offered by major operators, for example, Amazon [Ama14], Google [Goo14], Microsoft, IBM and some smaller companies like ElasticHosts [Ela14], Rackspace [Rac15] and XCalibre Communications [XCa14].

1.3. Focus and Contributions

In this work, we propose an approach for preserving location privacy of an LBS user in order to address the location privacy challenges described above.

Our *primary contribution* is a novel concept for the management of position information, which preserves the user's location privacy by using multiple non-trusted service providers and utilizing the spatial obfuscation technique (originally published in [DSR11]). The objective is to solve the problem of providers' trustworthiness and user's vulnerability through the utilization of multiple location servers of different providers, instead of using servers of a trusted third party. Each location server of the user-selected set stores un-encrypted position information, which we call a position share. The main idea of our approach is to split the user's precise position information into position shares containing position information of limited precision, the number of which can be defined by the user. The user distributes the generated position shares among n multiple LSs of different providers. The precision level available for different LBAs can be flexibly managed by defining the number of shares accessible for each of them: By obtaining k out of n shares ($0 < k \leq n$), LBAs can get position information with a certain obfuscation degree.

Our position sharing approach decreases the vulnerability of user's position information against possible attacks and provides the principle of graceful degradation of privacy. This means that a compromised LS reveals position information only of a limited precision, thus overcoming the problem of a single trusted third entity. Moreover, our approach provides a gradual increase of position precision after obtaining every new position share. Users can allow different location-aware applications to access an individual number of shares k, which corresponds to their trust in these applications, thus they can flexibly manage multiple levels of location privacy.

As our primary contribution, we present algorithms for share generation and fusion suited for open space (without considering map knowledge).

The *second contribution* is the adaptation of the basic position sharing approach to the map-aware scenario (originally published in [SDR12]). We present an extension of our approach to make it map-aware during the obfuscation process, i.e., to take into account space constraints like topography, land surface, roads, buildings, etc. The obfuscation shape in our basic position sharing approach is generated independently from the map as a simple circle. This could lead to privacy problems. If an obfuscation shape covers an area where the mobile user cannot possibly be located, this decreases the user's privacy level dramatically. For instance, a car's level of obfuscation is far lower than desired if it is located within an obfuscation area such as a circle, 90% of which are agricultural fields and only 10% are roads. Therefore, we extend the basic position sharing approach to the *map-aware* obfuscation in order to resist the privacy attacks caused by the constrained space environment. Namely, we perform the adjustment of obfuscation shape's size, depending on the user type and the corresponding representation of privacy sensitivity of various map regions. As a result, the share generation algorithm creates obfuscation shapes by taking into account map-based knowledge.

The *third contribution* is that having considered individual trustworthiness levels for LSs we improve privacy by optimizing share placement onto LSs based on their trustworthiness. In the basic approach, we assume that each LS has the same probability to be compromised. Therefore, we place the same degree of position information onto each LS. However, if many of the selected LSs have low trustworthiness (i.e., they can be hacked easily, or if they are malicious themselves), this lack of security could lead to the user's position being almost exactly revealed. Thus, if the information about trustworthiness of each LS or its provider is available to the mobile user, we can adapt the share placement in such a way that less trusted LSs get only the position information of lower precision, while more trusted LSs are allowed to store position information with higher precision.

The *fourth contribution* is a location update approach which reduces communication overhead caused by *multiple consecutive updates* without decreasing the user's privacy. If the number of different LSs n is large, and/or the position update rate is high, this may lead to significant communication overhead, since at every update we have to update all of n shares in a naïve approach. We propose an optimized location update algorithm to reduce the number of messages to be sent without impacting the user's privacy.

The *fifth contribution* is a novel classification of major techniques for protecting location privacy based on which protection goals they fulfill and which attacks they can resist (originally published in [WSDR14]). This contribution includes an analysis of possible attacker knowledge and types of mobile user's information that must be protected. In the proposed classification, we analyze which combinations of attacks are currently not considered in the literature, and we show how our position sharing approach relates to other approaches.

The contributions of this thesis have also been presented in several publications [DWSR10, DSR11, SDR12, WSDR14]. In [DWSR10], the author contributed to the basic principle of position sharing together with Frank Dürr. In [DSR11], the author introduced the algorithms of secure position sharing based on geometric transformations. Also, the author implemented the approach, while the measurements were performed together with Frank Dürr. In [SDR12], the author developed the concept of map-aware secure position sharing based on geometric transformations, implemented the approach and conducted the evaluation. In [WSDR14], the author provided the basic principle of classification of location privacy approaches. The classification of location privacy goals and attacks was provided by Marius Wernke. The above mentioned contributions were refined in collaboration with Frank Dürr.

There were several student theses supervised by the author [Sch11, Hae12, Pau11, Hoy12], which have also contributed to this work: Björn Schembera developed the basics of share placement optimization [Sch11]; Simon Hänle contributed to the concept and evaluation of the location update approach [Hae12]; Andreas Paul [Pau11] and Daniel del Hoyo [Hoy12] improved the mechanisms of reading and representing map information.

This work is structured as follows. In the second chapter, we present our basic position sharing approach and its extended map-aware version. This includes an analysis of the privacy guarantees provided by our approach as well as measurements of its runtime performance. In the third chapter, we describe an algorithm for share placement optimization. In the fourth chapter, we propose a location update algorithm which reduces communication overhead without affecting privacy levels. Finally, we summarize the results of this work and outline possible future research directions.

Chapter 2

Position Sharing Approach

In this chapter, we present our basic position sharing approach. It is based on work that has been previously published [DSR11, SDR12, WSDR14]. First, we describe our system model and privacy metrics, and define the problem. The approach includes four versions of share generation and share fusion algorithms, that depend on the methods of share generation, randomness of share fusion (fixed vs. free order of fusing shares) and availability of map knowledge (open space model vs. constrained space model). Next, we analyze the location privacy guarantees by evaluating the privacy metrics defined in the problem statement, and we evaluate the runtime performance of the approach. After that, we analyze and classify the related work in the field of location privacy in location-based services. We conclude the chapter with a summary.

2.1. System Model

The components of our LBS system are shown in Figure 2.1. They include a *mobile object (MO)*, i.e., a user with a mobile device, *location servers (LSs)* of multiple service providers, and *location-based applications (LBAs)*, which provide location-based services.

Mobile objects (MOs) are the objects whose positions are managed on LSs and used by LBAs. MOs correspond to users carrying a mobile device such as a smartphone with a positioning system such as GPS. By using this positioning system, the MO can determine its current position, which is denoted as π and represented by two-dimensional coordinates. For the sake of simplicity, we assume the position reported by the positioning system to be perfectly precise and accurate. In a real system, the detected position may already be imprecise and inaccurate to a certain degree (depending on the positioning system). However, we assume that this sensing error is much smaller than the artificial imprecision introduced by position obfuscation. A position of certain *precision* is defined by a circular area which we call *obfuscation area*, where radius r of this circular area defines *precision* $\text{prec}(\pi) = \phi = r$ of position π. A smaller radius corresponds to a higher precision, i.e., *precision level*: if $r_1 = \text{prec}(\pi_1), r_2 = \text{prec}(\pi_2)$ and $r_1 < r_2$, the precision of π_1 is higher than the precision of π_2.

The MOs issue location-based queries (through interfaces provided by the LBAs), for which they have to send their position information to LSs. However, we assume that the mobile user does not want his or her precise position to be revealed to a third party. For that reason, a local component installed on the MO runs a *share generation* algorithm. We assume that this component can be implemented in a trustworthy way, for example, by using TCP (Trusted Computing Platform) [DND07]. Given a precise position π, a number of n shares, and a lowest precision ϕ_{min}, the share generation algorithm generates *position shares* denoted as the *master share* s_0 and the set S of n *refinement shares* $S = \{s_1, s_2, \ldots, s_n\}$:

$$\text{generate}(\pi, n, \phi_{min}) = s_0, S \tag{2.1}$$

The master share s_0 is generated so that the position p_0 given through s_0 has the minimal

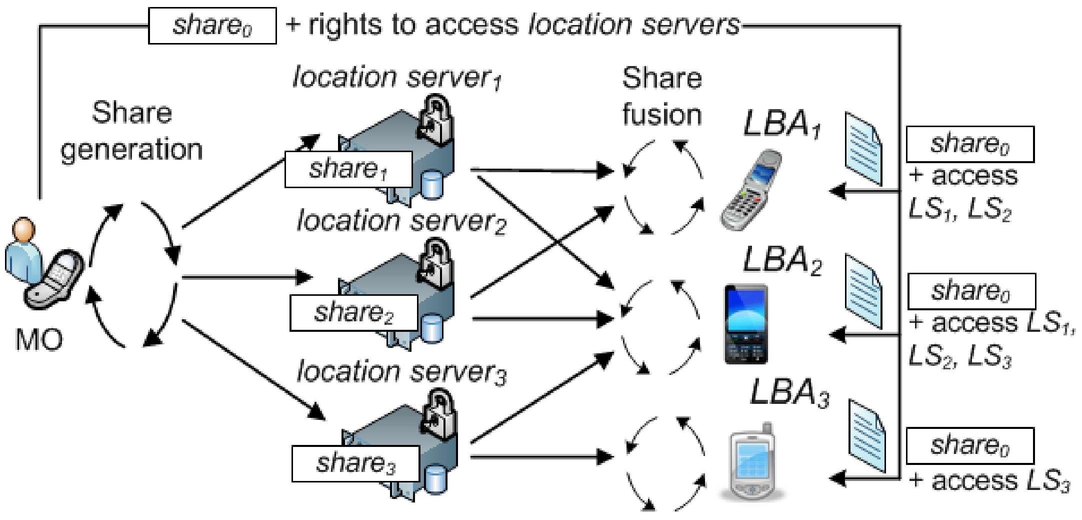


Figure 2.1.: System model: mobile object (MO) sends information to location servers (LSs), which provide this information to location-based applications (LBAs)

precision $\text{prec}(p_0) = \phi_0 = \phi_{min}$, which satisfies the maximal privacy demands of the user, i.e., it has the highest obfuscation level. Given s_0 and a subset $S_k \subseteq S$ of k refinement shares ($k \leq n$), a refined position p_k can be calculated using a *share fusion* algorithm:

$$\text{fuse}(s_0, S_k) = p_k \tag{2.2}$$

Each further refinement share s_{k+1} provides a more precise position p_{k+1}:

$$\text{fuse}(s_0, S_{k+1}) = p_{k+1}, \tag{2.3}$$

$$\text{where} \ \ \text{prec}(p_{k+1}) \leq \text{prec}(p_k), \ \ \text{i.e.,} \ \ \phi_{k+1} \leq \phi_k$$

Only after obtaining all the position shares, the last share s_n reveals the exact MO's position $p_n = \pi$ of highest precision $\phi_n = \phi_{max}$ without obfuscation:

$$\text{fuse}(s_0, S) = \pi, \tag{2.4}$$

$$\text{with} \ \ \text{prec}(\pi) = \phi_{max}$$

The obfuscated positions $p_0, p_1, \ldots, p_{n-1}$ correspond to precision levels $\phi_0, \phi_1, \ldots, \phi_{n-1}$. Thus, n different location privacy levels are provided.

We say that shares are *heterogeneous* if for a subset of refinement shares $S_k \subseteq S$ with size $|S_k| = k$ ($|S| = n$, $k \leq n$), after fusing k shares into p_k, we obtain the required precision level ϕ_k only after fusing these shares in a certain fixed order. In cases when shares are heterogeneous, each refinement share s_k increases the position precision by an individual pre-defined value Δ_k^{ϕ}.

If a share generation algorithm produces *homogeneous* shares, only the number k of obtained shares defines the resulting precision level ϕ_k, and the fusion order of the refinement shares can be arbitrary. In this case, the precision increase is equal for each share: $\Delta_1^{\phi} = \Delta_2^{\phi} = \ldots = \Delta_n^{\phi} = \phi_{max}/n$.

According to our position sharing approach, each share decreases the size of the obfuscation area, i.e., it increases the precision level of the given MO's position. An example of precision increase through share fusion for circular obfuscation areas is shown in Figure 2.2. Later in this chapter, we will introduce share generation algorithms with different assumptions regarding the shares' precisions and fusion order.

We assume that multiple *LSs* from different independent service providers are available, and each LS corresponds to a separate provider. We consider a provider's LS as a single entity, but this LS can be implemented by a number of physical severs on a lower level, for example, in a data center. Since no single LS is trusted completely by the MO, no LS is allowed to store the MO's precise position.

After share generation, the master share is known to everybody—in particular, every LBA—for instance, through full replication at every LS and unrestricted access by LBAs. Hence, every LBA can track MOs with (at least) a precision of ϕ_{min}. Therefore, ϕ_{min} is usually chosen large, i.e., as a large radius corresponding to a low precision.

The open (i.e., non-secret and available for each system actor) information also includes the number of LSs and the algorithms of share generation and share fusion. The only secret information is the set of refinement shares $s_1, s_2, \ldots, s_{n-1}$. The MO distributes the refinement shares among n selected LSs:

$$\text{place}(\{s_1, s_2, \ldots, s_n\}, L) : S \longrightarrow L \tag{2.5}$$

LSs store the shares of position information sent to it by MOs and deliver this information to authorized LBAs. Each LS has a common access control mechanism, which allows for the

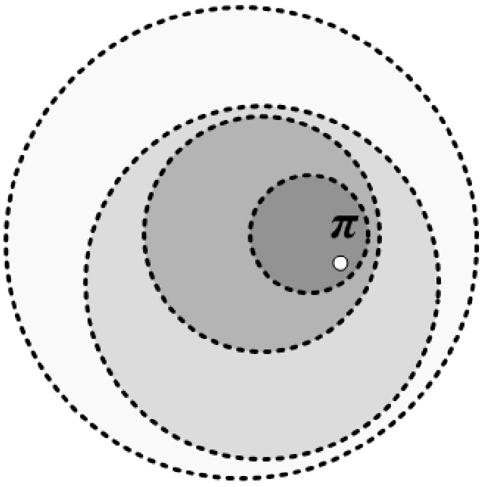


Figure 2.2.: Basic idea of position sharing approach: after getting each new share, the precision is increased until we get the exact MO's position π

specification of access rights (given by a user) for the LBAs' access to shares stored at this LS.

The mapping of shares to LSs or the precision increase Δ^{ϕ} could be adjusted to the individual trustworthiness of the LS, giving more trusted LSs better or more shares. In this chapter, we assume the exact levels of MO's trust to LSs to be unknown; therefore, the trustworthiness of every LS provider is assumed to be equal. Later in Chapter 3, we extend our system model by considering various trustworthiness levels of LSs stored in a trust database.

Refinement shares are only known to authorized *LBAs*. The MO specifies which precision each LBA should get (see Figure 2.1). Usually, this decision defines a trade-off between the quality of service an LBA can provide with a certain precision of information and the privacy requirements of the MO. The trusted share generation component running on the MO's device Next, the MO assigns access rights to a number of refinement shares, which provides this precision. Shares and the respective access rights are sent together to the LSs. The LSs use common access control mechanisms to deliver refinement shares only to authorized LBAs. LBAs receive the necessary access rights (credentials) together with the relevant LS addresses from the MO. Then LBAs fuse the obtained shares in order to get the MO's position within the defined level of precision.

Since the MO's position information is distributed among LSs, a compromised LS reveals only a position of strictly limited precision. This ensures an important property of our position sharing approach: graceful degradation of privacy (increase of precision) with the number of compromised LSs.

LBAs can subscribe to receive continuous position updates from the target MOs; or they can issue and process location-based queries to get the MOs' positions. However, in this chapter we assume that share generation is only triggered sporadically rather than with every update of the positioning system. Typically, this is the case when using a "check-in" usage pattern, where the user manually publishes his or her position sporadically at certain locations. Although the presented algorithms could also work with continuous positions updates, subsequent (close) positions might reveal additional information to an attacker. Such problems arising from continuous updates are addressed in Chapter 4. However, at this point we assume that a minimum position update interval is ensured and thus the succeeding obfuscation shapes of precision ϕ_k do not intersect.

2.2. Privacy Metrics

The user's privacy levels are primarily defined by *precision* levels ϕ_k, which are pre-defined by the user for each $0 \leq k \leq n$ as radii r_k of a *circular obfuscation areas*. Higher precision ϕ_k corresponds to a smaller obfuscation area, and vice versa. The problem is as follows: an attacker can derive a precision $\phi_{\text{k,attack}}$ higher than ϕ_k if the attacker knows the share generation algorithm and the k shares. As we will show later, analysis of k obtained shares in addition to the knowledge of share generation algorithm can provide stochastic knowledge about the true user position π. This makes the precision levels probabilistic. Thus, we need a probability distribution that ensures that an attacker is not able to predict the MO's position π for a given precision ϕ_k with sufficiently high probability $P_{\text{k,attack}}$. The following distribution $P_{\text{k,attack}}(\phi_{\text{k,attack}})$ defines the *probability* of an attacker obtaining a position $\pi_{\text{k,attack}}$ of a certain precision $\phi_{\text{k,attack}} = \text{prec}(\pi_{\text{k,attack}})$ depending on the number k of compromised LSs:

$$P_{\text{k,attack}}(\phi_{\text{k,attack}}) = Pr[\phi_{\text{k,attack}} \leq \phi_k] \tag{2.6}$$

This metric can be used by the MO to define the acceptable *probabilistic guarantees* represented as a set of probability thresholds $P_k(\phi_k)$ corresponding to various precision levels ϕ_k. For example, an MO can specify that an attacker must not be able to obtain a position of precision $\phi_1 \leq 1$ km with probability $P_{\text{1,attack}} > 0.2$, and precision $\phi_2 \leq 2$ km with probability $P_{\text{2,attack}} > 0.1$.

By using this metric, we can define the security of the given share generation algorithm. Namely, we determine the probability levels corresponding to the precision levels guaranteed for each number of known shares k and the given master share. These levels allow the user to decide whether his or her privacy levels are acceptable, after the user has selected ϕ_{min}, n and the generation share algorithm. If the privacy guarantees do not provide the required level of security, the user can improve them by adjusting the user-defined parameters P_k, ϕ_{min} and n, or by using a different share generation algorithm.

2.3. Problem Statement

The problem is to find a secure approach for the generation and fusion of shares, such that the following property is fulfilled for the generated shares: Given the master share s_0 and a set S_k of refinement shares, it must not be possible to derive a position with higher precision than the intended precision ϕ_k with probability $P_{\text{k,attack}}$ higher than a user-defined probability P_k (cf. Equation 2.6). The user can analyze the values of $P_{\text{k,attack}}$ provided by share generation algorithm and decide whether the given probabilistic guarantees of privacy levels are acceptable.

We define the following as given:

- n location servers,
- the MO's precise position π,
- the probability distribution $P_k(\phi_k)$, which specifies the required probabilistic guarantees for each precision level ϕ_k.

Problem: Find a share generation algorithm generate(...) (cf. Equation 2.1) which randomly generates set S containing a master share s_0 and n refinement shares $s_1 \ldots s_n$, and a share fusion algorithm fuse(...) (cf. Equations 2.2-2.4) which concatenates the shares $s_0 \ldots s_n$ such that the resulting point is π:

$$S = \{s_0 \ldots s_n\} \quad : \quad \sum_{k=0}^{n} s_k = \pi \tag{2.7}$$

such that the set of shares S satisfies the current user's privacy requirements, i.e., each further kth share must provide the pre-defined probabilistic guarantees of privacy levels $P_k(\phi_k)$:

$$\forall \ \phi_{\text{k,attack}} \quad : \quad P_k(\phi_k) > Pr[\phi_{\text{k,attack}} \leq \phi_{\text{k}}]; \tag{2.8}$$

finally, the precision ϕ_k of each imprecise position p_k^{i+1} derived by share fusion after obtaining the minimized set S has to be pre-defined:

$$\text{prec}(p_k) = \phi_k \tag{2.9}$$

2.4. Share Generation and Share Fusion Algorithms

In this section, four different algorithms for generating and fusing the position shares are presented (see Figure 2.3).

First, we present the basic position sharing approach, which does not consider space constraints and assumes any share order during share fusion; it has two versions: “a-posteriori” and “a-priori”. Then, we propose an approach with fixed share order during share fusion, which allows for the intersection of obfuscation circles and provides size adjustment of the obfuscation area resulting from the intersection. Finally, we present the map-aware position sharing approach, which adapts the area adjustment for the constrained space model. For each of these approaches, we present a share generation and share fusion algorithm.

2.4.1. Open Space – Any Share Order: “a-posteriori” Share Generation

Here we describe the position sharing approach for open space, later referred to as OSPS-ASO (Open Space Position Sharing with Any Share Order), first introduced in [DSR11]. Within the *open space* model, we assume that the prior probability for the mobile user to be located at each point in space is uniform. We present the main principles of share generation and share fusion, which are also the basis for the more advanced versions of our approach as well.

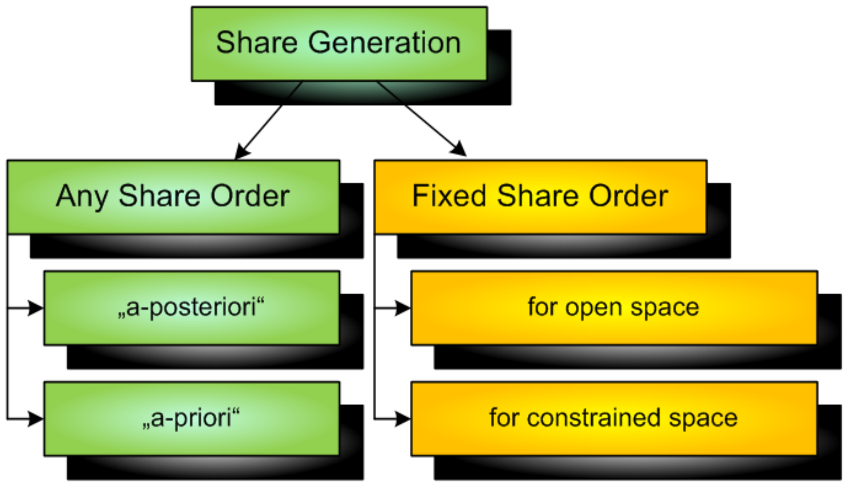


Figure 2.3.: Classification of position sharing algorithms

2.4.1.1. Share Fusion Algorithm

Algorithm 1 shows the share fusion algorithm of OSPS-ASO. As input parameters we have the number of LS providers (and correspondingly the total number of refinement shares) n, the obtained refinement shares $\vec{s_1} \dots \vec{s_k}$ ($k < n$), and the master share s_0.

The master share is the initial obfuscation circle c_0 with center p_0 (line 3) and radius r_0 (line 4). The refinement shares are shift vectors $S = \{\vec{s_1} \dots \vec{s_n}\}$. In the fusion algorithm, starting from the initial obfuscation circle c_0 (lines 3-4), step-by-step for k shares (line 5) each of the vectors $\vec{s_i}$ shifts the center p_i of the current obfuscation circle c_i (line 6) while reducing the radius r_i (line 7) of the current obfuscation circle by a pre-defined value $\Delta r = r_0/n = \Delta_\phi$ (line 2). The resulting obfuscation circle is c_k (line 8); an example for $n = 4; k = 3$ is shown in Figure 2.4.

Algorithm 1 OSPS-ASO: fusion of shares

```
1: function $fuse\_k\_shares\_OSPS\_ASO(n, s_0, \vec{s_1} \dots \vec{s_k})$
2: $\Delta r \leftarrow r_0/n$
3: $\vec{p} \leftarrow \vec{p_0}$
4: $r \leftarrow r_0$
5: for $i = 1$ to $k$ do
6:     $\vec{p} \leftarrow \vec{p} + \vec{s_i}$;
7:     $r \leftarrow r - \Delta r$
8: return $c_k = \{\vec{p}, r\}$
```

As shown in Figure 2.4, the order of obtainment of the refinement shares can be arbitrary, while the precision (namely, radius and area) of every obfuscation circle c_k is pre-defined. This is achieved by limiting the maximal length of shift vectors by $\Delta r = \Delta_\phi$.

For the same circle c_0, the maximal acceptable vector length decreases with the increase of n ($r_0 = 25$ km, $0 < k < 5$), as shown in Figure 2.5. As a result, each obfuscation circle c_k is inside the previous obfuscation circle c_{k-1} for any order of refinement shares obtainment.

Note that according to the algorithm presented, even if one tries to intersect the circles resulting from combinations of the same k vectors in a different order, this will not bring higher precision: the resulting obfuscation circle c_k will be the same for any share fusion order.

2.4.1.2. "A-posteriori" Share Generation Algorithm

The share generation algorithm that provides "a-posteriori" definition of s_0 related to π is presented in Algorithm 2. First, we determine the maximal shift length $\Delta r = \Delta_\phi = r_0/n$

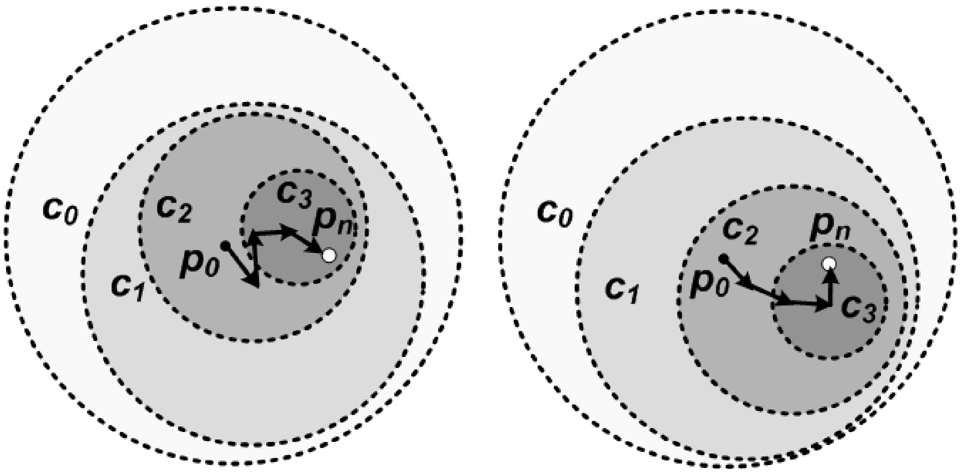


Figure 2.4.: OSPS-ASO: fusion of the same set of shares in an arbitrary order

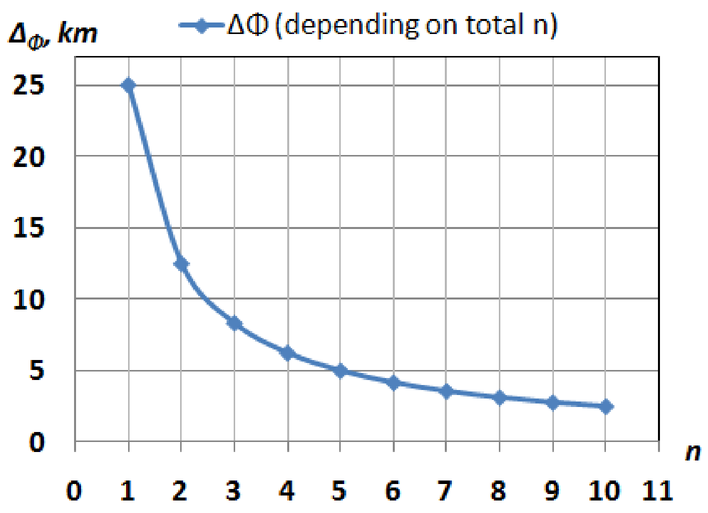


Figure 2.5.: OSPS-ASO: maximal vector length depending on n

(line 2). Then, we generate n shift vectors $s_1 \ldots s_n$ with randomly selected direction in $[0; 360°]$ and randomly selected length in the interval $[0; \Delta r]$ (line 4). After that, the position p_0 of the master share s_0 is calculated so that its concatenation with all of the refinement vectors results in the precise MO position $\pi = p_n$ (line 5).

We generate the set of shift vectors $s_1 \ldots s_n$ randomly (lines 2-3), having the master share's radius $r_0 = \phi_{min}$, the target number of shift vectors n, the exact MO position $\pi = p_n$ and the maximal shift length Δr. The direction of the generated vectors is chosen uniformly at random, while their lengths are chosen uniformly at random from the interval $[0; \Delta r]$.

Note that if the length of each vector is smaller than Δr and if c_0 contains π, we get a

Algorithm 2 OSPS-ASO: generation of shares "a-posteriori"

```
1: $\Delta r \leftarrow r_0/n$
2: function $gen\_n\_shares\_a\_posteriori(s_0, n, \pi)$
3: for $i = 1$ to $n$ do
4:     select randomly $\vec{s_i}$ with $|\vec{s_i}| \leq \Delta r$
5: $\vec{p_0} \leftarrow \pi - \sum_{i=1}^{n} \vec{s_i}$
6: return $\vec{s_0} \ldots \vec{s_n}$
```

symmetric permutation group property independent of the vectors' direction. That is,

$$\forall i \in [1; n] : |\vec{s_i}| \leq \Delta_\phi \text{ and } p_n \in c_0 \tag{2.10}$$

is a sufficient condition for creating a vector set, i.e., a vector set which allows for the fusion of shares in any order and provides the required precision levels ϕ_k. Thus, the center p_0 of the initial circle c_0 is defined a-posteriori by the concatenation of the random shift vectors from S to the true MO's position π (line 4).

Figure 2.6 shows the distribution of the precise MO position π inside c_0 for "a-posteriori" share generation Algorithm 2, having a total number of LSs $n = 5$ and 1000 runs of the Monte Carlo method. We can see that the end point of the concatenation of all vectors tends to be closer to the center of circle c_0. The resulting probability of finding the user's position π inside c_0 is high (as we will show in our evaluations later in more detail). In order to overcome this shortcoming, we propose an alternative "a-priori" share generation algorithm.

2.4.2. Open Space – Any Share Order: "a-priori" Share Generation

In this section we will present another approach to the open space scenario which we call "a-priori" share generation. The "a-posteriori" share generation algorithm is very simple and fast, but produces shares that make the exact MO's position π predictable, as it is illustrated in Figure 2.6. The "a-priori" algorithm aims to improve the stochastic properties of the share set by preventing such high-density regions within the obfuscation circles of probability distributions for π.

The share fusion algorithm for "a-posteriori" share generation is the same as for "a-posteriori" share generation (see Algorithm 1). Thus, the "a-priori" modification of our approach changes only the share generation algorithm.

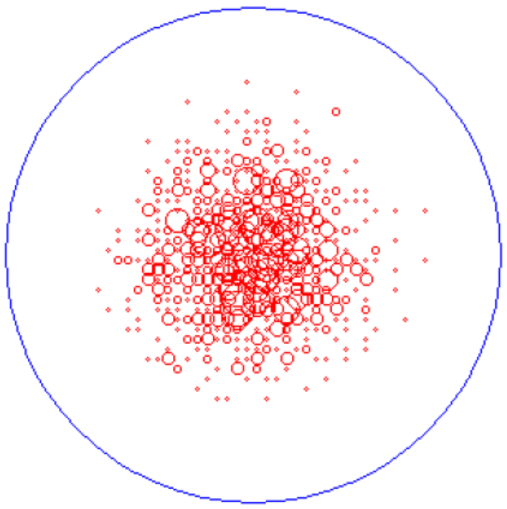


Figure 2.6.: Distribution of π inside c_0 for "a-posteriori" share generation Algorithm 2 ($n = 5$, Monte Carlo with 1000 runs) [DSR11]

2.4.2.1. "A-priori" Share Generation Algorithm

The generation of shares in OSPS-ASO with "a-priori" definition of the master share's center is presented in Algorithm 3 and works as follows: The input parameters are the MO-defined radius $r_0 = \phi_{min}$ of the initial obfuscation circle c_0, the total number of shares n and the precise user position $\pi = p_n$. First, we determine the maximal shift length $\Delta r = \Delta_\phi = r_0/n$ (line 2). Then, the position p_0 of the initial circle c_0 is selected randomly according to a uniform distribution, such that $\pi = p_n$ is inside c_0 (line 3). The set of the refinement shift vectors $S = \{\vec{s_1} \ldots \vec{s_{n-1}}\}$ is generated randomly (lines 4-6), such that starting from the center of c_0 the concatenation of all shift vectors of S gives the resulting point $\pi = p_n$ (with $r_n = 0$ correspondingly), which coincides with the user's position π within c_0 (line 8). The consistency of the operation (line 8) with the maximal vector length is guaranteed by the restriction of line 7.

Finally, the MO sends the position information to n LS, including the master share s_0, the size of the radius decrease after every shift Δr (in OSPS, Δr is constant for all shifts), and one share $\vec{s_i}$ for each LS.

The important condition of line 7 of Algorithm 3 defines whether the required share set has not been found yet (Figure 2.7a) or it has been found (Figure 2.7b). The problem is that before the required set is found, the share generation algorithm should traverse many randomly generated share sets $s_1 \ldots s_{n-1}$. The number of such sets is not pre-defined: for example, if π is located far away from the center of c_0, this process takes more time, since

Algorithm 3 OSPS-ASO: generation of shares "a-priori"

```
1: function $gen\_n\_shares\_OSPS\_ASO\_a\_priori(s_0, n, \pi)$
2: $\Delta r \leftarrow r_0/n$
3: select randomly $p_0$ such that $distance(\vec{p_0}, \pi) \leq r_0$
4: do
5:     for $i = 1$ to $n-1$ do
6:         select randomly $\vec{s_i}$ with $|\vec{s_i}| \leq \Delta r$ such that $\pi \in c_i$
7: while $distance(\vec{p_0} + \sum_{i=1}^{n-1} \vec{s_i}, \pi) > \Delta r$
8: $\vec{s_n} \leftarrow \pi - (\vec{p_0} + \sum_{i=1}^{n-1} \vec{s_i})$
9: return $\vec{s_0} \ldots \vec{s_n}$
```

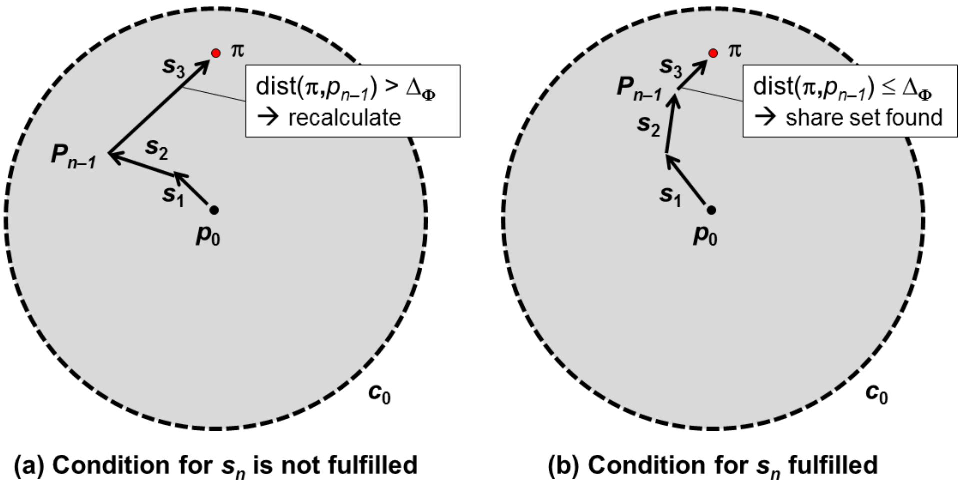


Figure 2.7.: "A-priori" share generation Algorithm 3, line 7: (a) condition is not fulfilled; (b) condition is fulfilled

the probability of generating a random share set which reaches π is lower in this case. Note that the main cycle (lines 4-7) is guaranteed to terminate in case of a sufficiently large vector sampling. The execution times of the "a-priori" share generation algorithm are evaluated later in this chapter.

An example pdf for the "a-priori" share generation Algorithm 3 is illustrated in Figure 2.8a, while Figure 2.8b shows an example of a correlated vector set where vectors are biased towards the North-East area. This bias means that shift vectors are correlated and therefore by knowing k of them ($0 < k < n$) an attacker can obtain a pdf of the resulting user position π. Such analysis and its effect on the probabilistic guarantees of privacy levels will be evaluated later in this chapter. However, note that this correlation of vectors does not affect the initial uniform probability distribution when the number of known shares $k = 0$ (cf. Figure 2.8a).

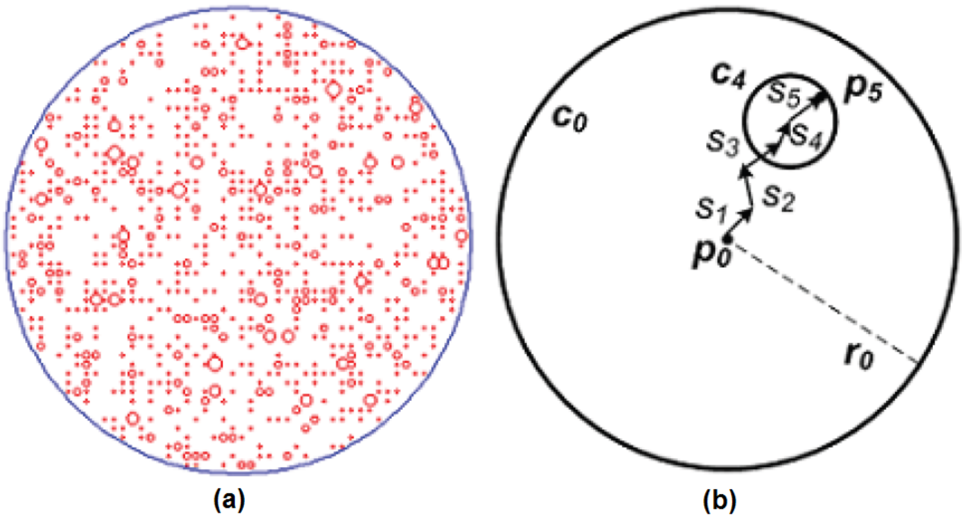


Figure 2.8.: (a) Example pdf for "a-priori" share generation Algorithm 3 ($n = 5$, Monte Carlo with 1000 runs); (b) example of a correlated vector set where vectors are biased towards the North-East area [DSR11]

2.4.3. Open Space – Fixed Share Order

Now, we relax the restrictions on the maximal vector length, which cause uneven probability distributions and biased vector sets of OSPS-ASO. This results in a non-arbitrary (as in the OSPS-ASO approach) but fixed (pre-defined) order of shares during their fusion. An example of a share fusion in the fixed order is shown in Figure 2.9. We define this approach as Open Space Position Sharing with Fixed Share Order (OSPS-FSO).

However, the allowed intersections (overlapped areas) of the obfuscation circles decrease the size of the actual obfuscation area for any $k > 0$. Thus, an area adjustment is needed for the intersected circles even without considering map-based knowledge. The goal of OSPS-FSO is to keep the obfuscation area above a certain threshold through the adjustment of the obfuscation circles' radius. Thus, OSPS-FSO eases the limitations on the maximal vector length by allowing for circle intersections, and has a flexible radius for each obfuscation circle.

In OSPS-FSO, the obfuscation area for k shares of precision ϕ_k is not equal to the pre-defined size of the circle c_k alone (as it is in OSPS-ASO), but is defined through the area A_k of intersection of k circles $c_1 \ldots c_k$ (see Figure 2.10a). Now, the goal of the radius increase is to adjust the intersection area A_k up to the size of area of c_k denoted as area(c_k), which it

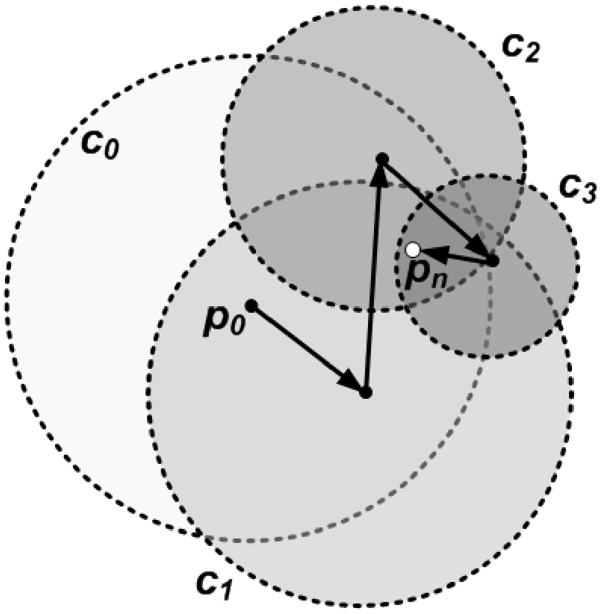


Figure 2.9.: Fusion of shares in a fixed order without area adjustment

would have without intersections:

$$A_k = \text{area}(c_0 \cap c_1 \cap \ldots \cap c_k) = \pi * r_k^2 \tag{2.11}$$

At the same time, the radius adjustment must be secure, i.e., an attacker should not be able to derive the original (non-adjusted) obfuscation area.

2.4.3.1. OSPS-FSO: Share Fusion Algorithm

The share fusion algorithm (Algorithm 4) for OSPS-FSO is illustrated in Figure 2.10a and includes the following steps. First, having the master share s_0, the obfuscation area A_k is defined by the initial obfuscation circle c_0 (line 2) and the center of the current obfuscation circle is set as p_0 (line 3). Then, the concatenation of k shift vectors $\vec{s_1} \ldots \vec{s_k}$ is performed iteratively for $i = 1 \ldots k$ (line 5), defining at each step the circle c_i with individual radius r_i (line 6). Each obfuscation circle intersects with the previously obtained obfuscation area, by which the current obfuscation area A_k for k shares is defined (line 7).

Note that vectors can be added only in a fixed order, otherwise the consistency of obfuscation areas (namely, the sizes of A_k) cannot be preserved: the obfuscation area A_k does not have a pre-defined shape such as a circle, but it has a pre-defined size. Therefore, we present

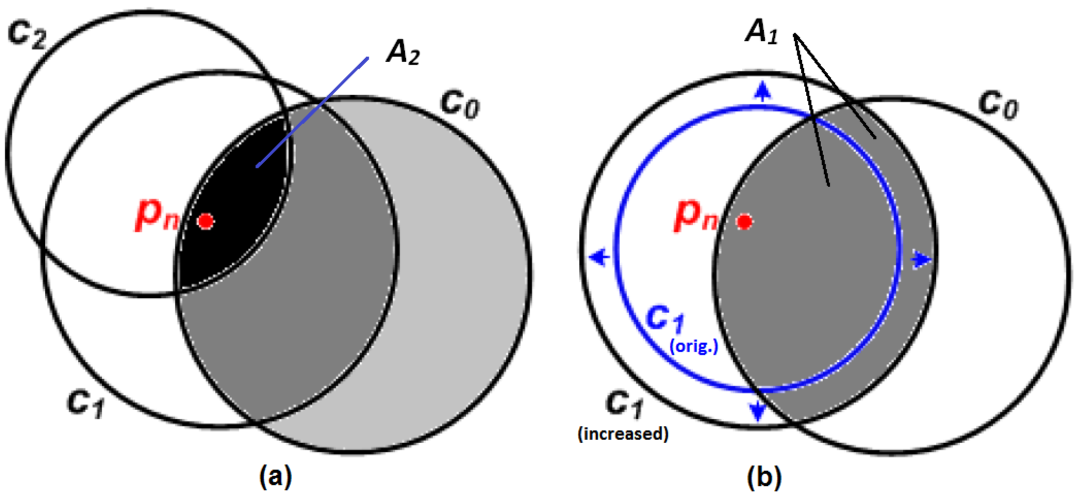


Figure 2.10.: OSPS-FSO: a) intersection A_2 of three circles c_0, c_1, c_2; b) adjustment of intersection area through radius increase for c_1: $A_1 = \text{area}(c_0 \cap c_1)$

Algorithm 4 OSPS-FSO: fusion of shares

```
1: function $fuse\_k\_shares\_OSPS\_FSO(n, c_0, \vec{s_1} \ldots \vec{s_k}, r_1 \ldots r_k)$
2: $A_k \leftarrow c_0$
3: $\vec{p} \leftarrow p_0$
4: for $i = 1$ to $k$ do
5:     $\vec{p} \leftarrow \vec{p} + \vec{s_i}$
6:     $c_i \leftarrow \{\vec{p}, r_i\}$
7:     $A_k \leftarrow A_k \cap c_i$
8: end for
9: return $A_k$
```

the OSPS-FSO share generation algorithm, which preserves the required position precision (and correspondingly the user's privacy guarantees) by adjusting the size of A_k.

2.4.3.2. OSPS-FSO: Share Generation Algorithm

Next, we present the share generation algorithm for the OSPS-FSO approach. The problem to solve here is the reduction of the obfuscation area A_k due to the intersections of c_k and the previous obfuscation circles $c_0 \ldots c_{k-1}$. In order to preserve the needed obfuscation level, we increase the radius r_k until the area of A_k achieves the value of the non-intersected area of c_k (see Equation 2.11), as we show in Figure 2.10b.

The first step of the share generation algorithm (Algorithm 5) is to randomly select the center p_0 of the initial obfuscation circle c_0 according to the uniform probability distribution (a similar principle to that in the "a-priori" version of the OSPS share generation algorithm)

Algorithm 5 OSPS-FSO: generation of shares

```
1: function $gen\_n\_shares\_OSPS\_FSO(n, r_0, \pi)$
2: select randomly $p_0$ with $distance(p_0, \pi) \leq r_0$
3: $A_0 \leftarrow$ area($c_0$)
4: for $i = 1$ to $n-1$ do
5:     $r_i \leftarrow r_0 * (n-i)/n$
6:     select randomly $\vec{s_i}$ with $|\vec{s_i}| \leq 2 * r_{i-1}$ and $\pi \in c_i$
7:     $A_i \leftarrow$ area($c_i$)
8:     while area($\cap_{j=1}^{i}(c_j)$) $< A_i$ do
9:         $r_i, p_i \leftarrow$ increase_and_adjust($r_i, p_i, \Delta r$)
10:    end while
11: end for
12: $\vec{s_n} \leftarrow \pi - (\vec{p_0} + \sum_{i=1}^{n-1} \vec{s_i})$
13: return $\vec{s_0} \dots \vec{s_n}$, $r_0 \dots r_n$
```

within radius r_0 around the given true user position $\pi = p_n$ (line 2). After that, the shift vectors $\vec{s_1} \dots \vec{s_{n-1}}$ are generated, which connect p_0 and p_{n-1} (lines 4-6). The corresponding radii are increased as well, taking into account not only the space constraints, but also the intersections with the previous obfuscation circles (lines 8-10). Finally, the last shift vector s_n is defined as the connection of the point p_{n-1} with the true MO's position π (line 12).

The result is that each share is represented by the shift vector $\vec{s_i}$, the individual radius r_i and the sequence number i itself. Thus, a fixed order of shares fusion is pre-defined in the OSPS-FSO approach; in contrast, the OSPS-ASO approach presented previously relies on an arbitrary sequence of share fusion.

2.4.3.3. Adjustment of p_i During the Radius Increase

There is an important aspect concerning the **increase_and_adjust($r_i, \dots$)** function for radius increase (Algorithm 5, line 9): if we use a deterministic algorithm for the area adjustment, an attacker can calculate the inverse function to decrease the size of the obfuscation area. Namely, if we increase radius r_i without changing the circle's position p_i, an attacker can reduce the obfuscation area A_i by simply decreasing the obtained radius r_i (see Figure 2.11a). This is possible if an attacker knows the share generation algorithm and therefore knows the initial (non-increased) value of the radius $r_{i(a)}$.

In order to avoid such a situation, the position of circle c_i must be adjusted so that the original position of $c_{i(a)}$ within c_i cannot be found, and therefore the possible location of the exact MO's position $p_n = \pi$ cannot be restricted within a smaller area. Figure 2.11b illustrates that after adjusting p_i, p_n can be located anywhere within c_i, and is not restricted

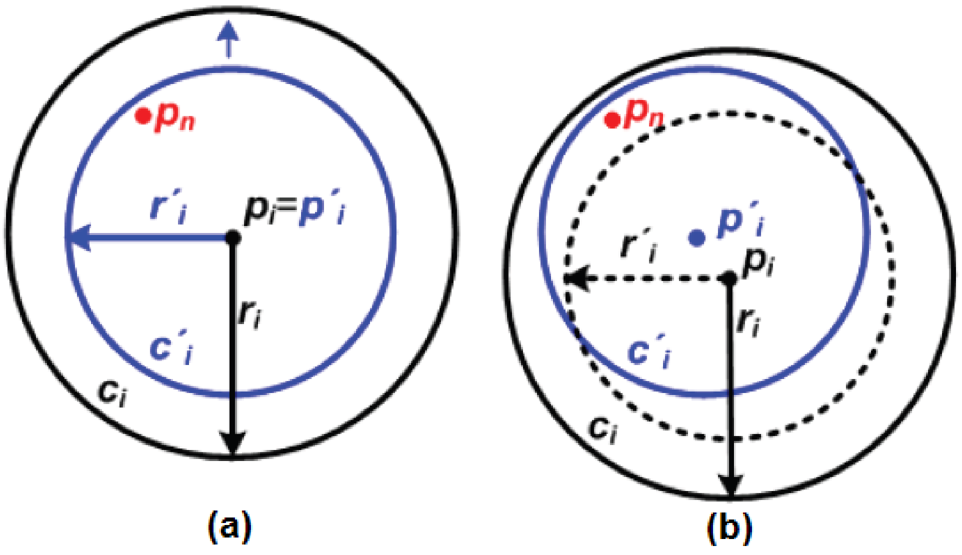


Figure 2.11.: Adjustment of p_i during radius increase: (a) no adjustment of p_i; (b) randomized adjustment of p_i

by the lesser radius $r_{i(a)}$. In other words, an attacker is not able to reduce the obfuscation area A_i just by knowing the share generation algorithm.

Algorithm 6 Radius increase with adjustment of p_i for OSPS-FSO

```
1:  function $increase\_and\_adjust(r_i, p_i, \Delta r)$
2:  $r_{i(a)} \leftarrow r_i$
3:  $A_i \leftarrow \text{area}(c_{i(a)})$
4:  while $\text{area}(\cap_{j=1}^{i}(c_j)) < A_i$ do
5:      $r_i \leftarrow r_i + \Delta r$
6:  end while
7:  $x_{shift} \leftarrow$ get_random_shift($p_i, r_{i(a)}, r_i$)
8:  $y_{shift} \leftarrow$ get_random_shift($p_i, r_{i(a)}, r_i$)
9:  $p_i \leftarrow$ shift($p_i, x_{shift}, y_{shift}$)
10: if $\text{area}(\cap_{j=1}^{i}(c_j)) < A_i$ then
11:     $r_i, p_i \leftarrow$ increase_and_adjust($r_i, p_i, \Delta r$) //recursive call
12: else
13:     while $\text{area}(\cap_{j=1}^{i}(c_j)) > A_i$ do
14:         $r_i \leftarrow r_i - \Delta r$
15:     end while
16:     $r_i \leftarrow r_i + \Delta r$
17: end if
18: return $p_i, r_i$
```

The function **increase_and_adjust(**$r_i, \ldots$**)** for radius increase combined with the adjustment of the obfuscation circle's position p_i is presented in Algorithm 6. First, for the current

p_i, we determine the radius r_i which makes the intersection area large enough (lines 2-6). Then we perform the random shift of p_i, not longer than $r_i - r_{i(a)}$ (lines 7-9). After that we check whether the current radius r_i satisfies the area condition (line 10). If the intersection area is still not large enough, we call the function **increase_and_adjust($r_i, \ldots$)** recursively (line 11). If the intersection area now exceeds the target value A_i, we simply decrease the current radius r_i until it achieves the required size (lines 12-16).

2.4.3.4. Computation of Arbitrary-shaped Area Size

In several lines of the above presented **increase_and_adjust($r_i, \ldots$)** algorithm, we need to calculate the intersection area of multiple circles area($\cap_{j=1}^{i}(c_j)$), i.e., an arbitrary-shaped area size. The size of such intersection area is calculated through space discretization with the discretization step, i.e., the distance between the lattice lines selected as $r_i/100$ (cf. Figure 2.12 with larger discretization step for illustrative purposes). After defining a virtual lattice, we count the number of the lattice crossing points $d_{k,intersect}$ covered by the intersection shape. Knowing the number of points $d_{k,original}$ located within the original (non-adjusted) circle $c_{i(a)}$ and the area of a non-intersected circle $c_{i(a)}$, we convert $d_{k,intersect}$ into the corresponding area value:

$$\text{area}(\cap_{j=1}^{i}(c_j)) = \frac{d_{k,intersect} \cdot \text{area}(c_{i(a)})}{d_{k,original}} \tag{2.12}$$

Note that later we will apply the same area computation principle when we have the map knowledge M_u as an additional intersection factor: $d_{k,intersect}$ is then the number of points located within area($M_u \cap_{j=1}^{i} (c_j)$).

In Figure 2.12, we show the space discretization principle for $k = 3$, with black points within the intersection shape of c_0, c_1 and c_2 and white points within the remaining area of c_2. Here, $d_{k,original}$ is the number of white and black points together, while $d_{k,intersect}$ is the number of white points.

2.4.4. System Model Extension: Map Knowledge

Now, the system model is extended by assuming the availability of *map knowledge*, which means that the MO has locally stored map information for the surrounding region. Moreover, the MO knows his or her own type of mobility and can distinguish between cars and pedes-

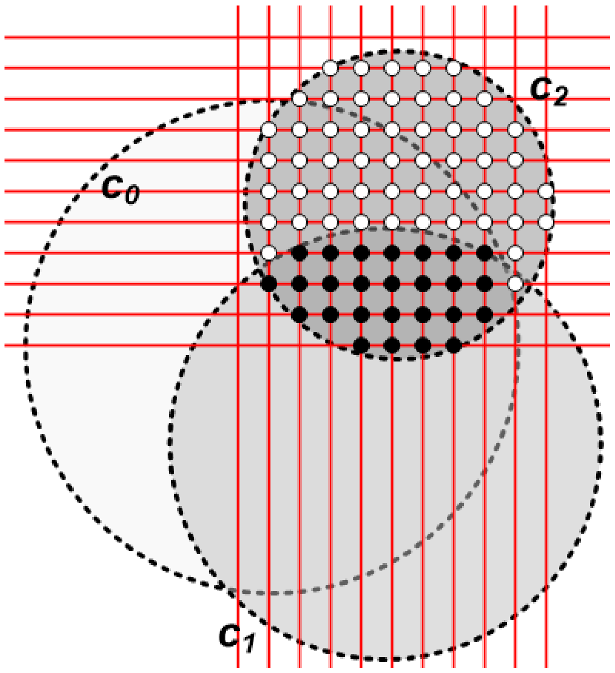


Figure 2.12.: Computation of an arbitrary-shaped area size for c_2 based on space discretization

trians, as well as boats, trains and planes. With this information, the MO is able to specify a map representation M_u that defines the map regions where he or she might be located, and use these regions during share generation.

The map-based knowledge allows each map to be considered as a binary *map representation* with a Boolean attribute assigned to different map regions. "True" means that a given MO can possibly be located there; "false" means that it is impossible that a given MO is located in this area. Thus, this Boolean attribute of a region can be different for each given MO and depends on the mode of MO's movement. For example, cars can only drive on paved roads, while pedestrians are not supposed to use highways; and neither cars nor pedestrians are expected to be located in hard-to-reach regions like mountains. The map representation is generated individually for each user type by analyzing the map-based knowledge. The Boolean attribute can be assigned for every single map feature (e.g., a building, a bridge, etc.), set of features, land surface of a certain type (e.g., a lake, an urban area, an agricultural field, etc.), or any combination of those.

We assume that each map region marked with "true" has equal probability of the MO to be located in any point of the region. Simple example of visual map representation for the given map of Figure 2.13a is presented in Figure 2.13b: grey areas show "true" regions M_{u1}

for a moving user u_1.

Similarly, Figure 2.14 shows two map representations: the left one shows where cars can move, while the right one shows where pedestrians can move. Here, green areas indicate "true", while dark-red and white areas indicate "false".

2.4.5. Constrained Space – Fixed Share Order (Map-aware Approach, CSPS)

Having developed OSPS-FSO with radius adjustment, this approach can be easily adapted to the constrained space model. In the map-aware position sharing approach for constrained space (CSPS), we define the obfuscation area A_k for k shares of precision ϕ_k through the intersection not only of k circles $c_1 \ldots c_k$, but also of the map representation M_u (see Figure 2.15a):

$$A_k = \text{area}(M_u \cap c_0 \cap c_1 \cap \ldots \cap c_k) = \pi * r_k^2 \tag{2.13}$$

CSPS includes a share fusion algorithm and a share generation algorithm, which allow us to overcome the disadvantages of OSPS described previously. CSPS is applicable for both open space and constrained space models; it assumes fixed share order.

Before the share generation, the user has to select the map representation M_u, which defines the map regions where he can possibly be located according to his movement mode. M_u is individual for each user, since different users can be possibly located in different map regions. Generally, the user u_1 corresponds to the region M_{u1}, u_2 corresponds to M_{u2}, and so on.

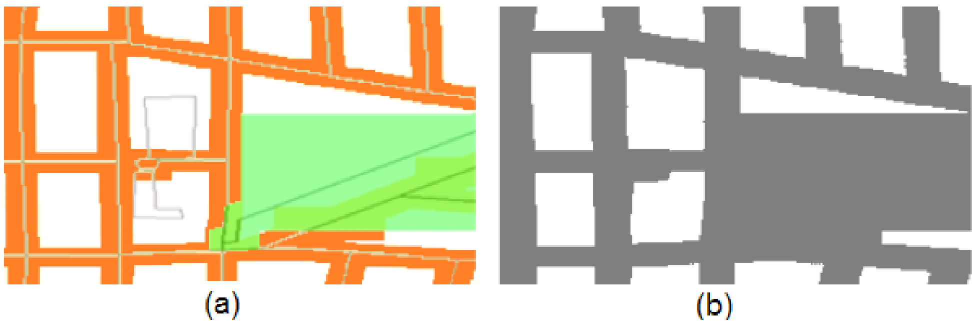


Figure 2.13.: (a) Basic map; (b) map representation M_{u1} for a moving user u_1

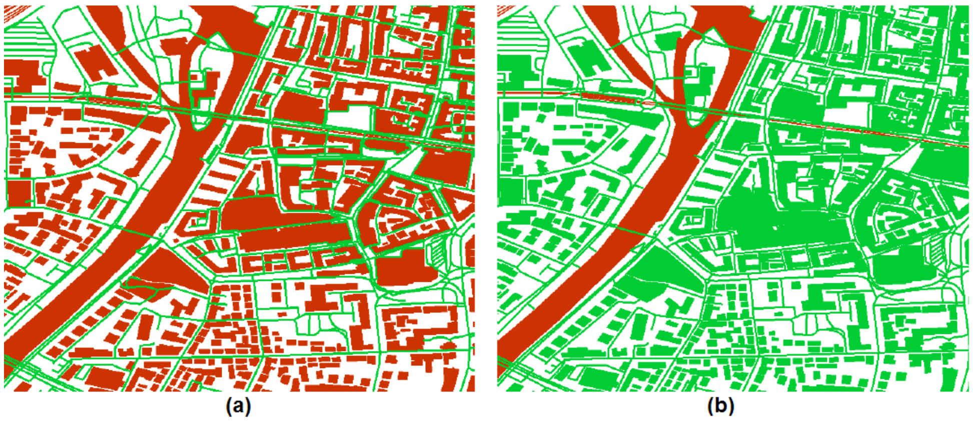


Figure 2.14.: (a) Map representation M_{u1} for an MO as a car u_1; (b) map representation M_{u2} for an MO as a pedestrian u_2; green areas indicate “true”, dark-red and white areas indicate “false” [Pau11]

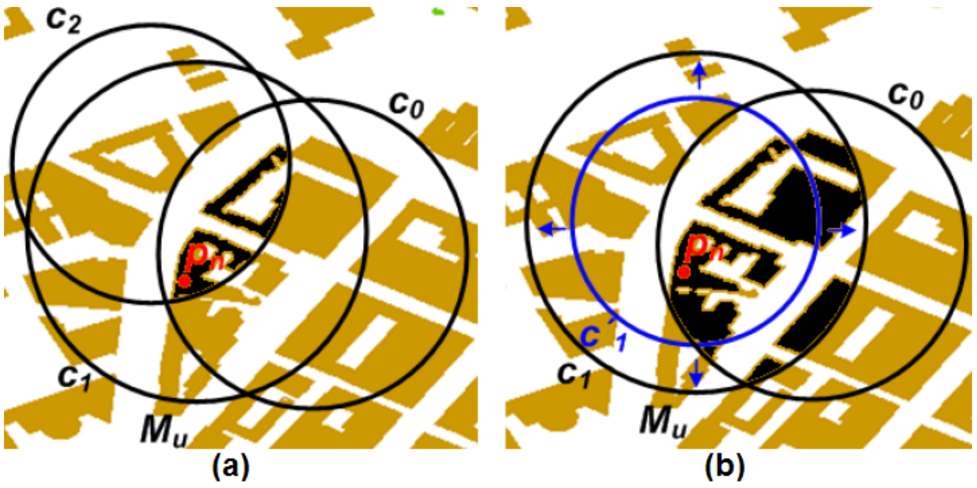


Figure 2.15.: CSPS: a) intersection of 3 circles c_0, c_1, c_2 and the map representation M_u; b) adjustment of intersection area through radius increase for c_1: $A_1 = \text{area}(M_u \cap c_0 \cap c_1) = \text{area}(c_{1a})$

For the inclusion of map representation M_u into the computation of A_k's area, the algorithms of share fusion and share generation require only small changes.

2.4.5.1. Share Fusion Algorithm

The share fusion algorithm for CSPS (Algorithm 7) requires a single modification: at first, the obfuscation area A_k is defined not only by the initial obfuscation circle c_0 but also by its intersection with the map representation M_u (line 2), as shown in Figure 2.15a. The next steps are the same as in Algorithm 4.

Algorithm 7 CSPS: fusion of shares

```
1: function $fuse\_k\_shares\_CSPS(M_u, n, c_0, \vec{s_1} \ldots \vec{s_k}, r_1 \ldots r_k)$
2: $A_k \leftarrow M_u \cap c_0$
3: $\vec{p} \leftarrow \vec{p_0}$
4: for $i = 1$ to $k$ do
5:     $\vec{p} \leftarrow \vec{p} + \vec{s_i}$
6:     $c_i \leftarrow \{\vec{p}, r_i\}$
7:     $A_k \leftarrow A_k \cap c_i$
8: end for
9: return $A_k$
```

2.4.5.2. Share Generation Algorithm

The share generation algorithm for CSPS also takes M_u into consideration: the radius r_k is increased until the area of $M_u \cap (c_0 \cap c_1 \cap \ldots \cap c_k)$ achieves the value of the non-intersected area of c_k (see Figure 2.15b).

Algorithm 8 has additional lines (4-6): the radius r_0 of the initial circle c_0 is increased taking into consideration the map representation M_u in order to adjust the size of $A_0 = M_u \cap c_0$. Then, in order to adjust the radii of shares $\vec{s_1} \ldots \vec{s_{n-1}}$, M_u is included in the condition of line 11. The rest of steps of the share generation algorithm for CSPS are the same as in Algorithm 5.

Algorithm 8 CSPS: generation of shares

```
1: function $gen\_n\_shares\_CSPS(n, M_u, r_0, \pi)$
2: select randomly $p_0$ with $distance(p_0, \pi) \leq r_0$
3: $A_0 \leftarrow \text{area}(c_0)$
4: while $\text{area}(M_u \cap c_0) < A_0$ do
5:     $r_0 \leftarrow$ increase_and_adjust_CSPS$(r_0, p_0, \Delta r)$
6: end while
7: for $i = 1$ to $n - 1$ do
8:     $r_i \leftarrow r_0 * (n - i)/n$
9:     select randomly $\vec{s_i}$ with $|\vec{s_i}| \leq 2 * r_{i-1}$ and $\pi \in c_i$
10:    $A_i \leftarrow \text{area}(c_i)$
11:    while $\text{area}(M_u \cap \cap_{j=1}^{i}(c_j)) < A_i$ do
12:        $r_i, p_i \leftarrow$ increase_and_adjust_CSPS$(r_i, p_i, \Delta r)$
13:    end while
14: end for
15: $\vec{s_n} \leftarrow \pi - (\vec{p_0} + \sum_{i=1}^{n-1} \vec{s_i})$
16: return $\vec{s_0} \ldots \vec{s_n}$, $r_0 \ldots r_n$
```

Furthermore, the function **increase_and_adjust_CSPS(...)** (line 12) is similar to

increase_and_adjust(...) (Algorithm 6) and presented in Algorithm 9. Note that the diffenrence is that M_u is now included in the conditions of lines 4, 10 and 13 of Algorithm 6. The computation of an arbitrary-shaped area's size area($M_u \cap \cap_{j=1}^{i}(c_j)$) is done according to the same principle based on space discretization as described in Section 2.4.3.4.

Algorithm 9 Radius increase with adjustment of p_i for CSPS

```
 1: function $increase\_and\_adjust\_CSPS(r_i, p_i, \Delta r)$
 2: $r_{i(a)} \leftarrow r_i$
 3: $A_i \leftarrow$ area$(c_{i(a)})$
 4: while area$(M_u \cap_{j=1}^{i} (c_j)) < A_i$ do
 5:     $r_i \leftarrow r_i + \Delta r$
 6: end while
 7: $x_{shift} \leftarrow$ get_random_shift$(p_i, r_{i(a)}, r_i)$
 8: $y_{shift} \leftarrow$ get_random_shift$(p_i, r_{i(a)}, r_i)$
 9: $p_i \leftarrow$ shift$(p_i, x_{shift}, y_{shift})$
10: if area$(M_u \cap_{j=1}^{i} (c_j)) < A_i$ then
11:     $r_i, p_i \leftarrow$ increase_and_adjust$(r_i, p_i, \Delta r)$ //recursive call
12: else
13:     while area$(M_u \cap_{j=1}^{i} (c_j)) > A_i$ do
14:         $r_i \leftarrow r_i - \Delta r$
15:     end while
16:     $r_i \leftarrow r_i + \Delta r$
17: end if
18: return $p_i, r_i$
```

2.4.6. Summary: Comparison of Algorithms

Table 2.1 compares the important properties of the position sharing approaches described in this chapter. The second column represents the properties of the OSPS-ASO "a-posteriori" and OSPS-ASO "a-priori" algorithms, while the third column contains the properties of OSPS-FSO and the map-aware CSPS approach. Only CSPS is map-aware, but note that OSPS-FSO shares most of the properties of CSPS with the exception of the actual map reading, which is done by CSPS in the same way as OSPS-FSO determines the arbitrary obfuscation shape A_k.

The obfuscation shape A_k can be arbitrary in OSPS-FSO and CSPS, since these two approaches support intersection of k obfuscation circles and therefore provide more flexibility within share generation, i.e., the generated shift vectors can be longer. However, the resulting disadvantage of such approach is that the generated shares provide individual radii reduction at each kth step of share fusion, and therefore the shares are not interchangeable by order as in both OSPS-ASO versions. For OSPS-FSO and CSPS, radius decrease is also individual for every fusion step k, while OSPS-ASO radius decrease is pre-defined and equal for each share.

Note that in spite of the described differences between the algorithms, the resulting precision in each algorithm is guaranteed to be preserved corresponding to the radius r_k. This principle requires more complex area adjustment for OSPS-FSO and CSPS, since they provide non-circular obfuscation shapes as opposed to OSPS-ASO.

Only OSPS-ASO "a-posteriori" is based on the "a-posteriori" selection of user's position π within the master share's obfuscation circle c_0. All the other approaches use "a-priori" selection, which provides better probabilistic guarantees of precision levels (as it will be shown in Section 2.5), but requires more runs during the share generation.

Only in OSPS-ASO "a-priori", the generated vector set is biased, i.e., shift vectors tend to form a correlated set. The reason for this bias is the "a-priori" c_0 selection, which is combined with limited shift vector lengths; as a result, the vectors are often stretched in order to connect the center of the initial obfuscation circle p_0 and the precise user's position pi. Other algorithms either have free p_0 selection (OSPS-ASO "a-posteriori") or free shift vector lengths, and therefore provide non-biased vector sets.

In general, share generation in OSPS-FSO and CSPS is more complex, since they often require to shift centers of obfuscation circles c_k during the area adjustment phase even after initial share generation. This process is required to make the arbitrary obfuscation shape A_k large enough to satisfy the basic precision requirements.

In the next section, we will present the evaluation of the presented algorithms in order to analyze the impact of the listed algorithm properties on their security and performance.

Comparison of position sharing approaches		
	OSPS-ASO	OSPS-FSO and CSPS
Map-awareness	No (disadvantage)	OSPS-FSO: no CSPS: yes (advantage)
Fusion: allow for intersection of obfuscation circles	No	Yes
Fusion: independent from shares order	Yes: arbitrary sequence of shares during fusion (advantage)	No: the sequence of shares is pre-defined (disadvantage)
Fusion: radius decrease	r_0/n; same for every k	Depends on obfuscation shape A_k; individual for every k
Fusion: precision (area) for k shares	Pre-defined: $\pi \cdot r_k^2$	Pre-defined: $\pi \cdot r_k^2$
Fusion: obfuscation shape	circle	Intersection of circles and map-based areas
Generation: basic method	Algorithm 2: “a-posteriori” Algorithm 3: “a-priori”	A-priori
Generation: biased vector set	OSPS-ASO “a-priori”: no (advantage); OSPS-ASO “a-posteriori”: yes (disadvantage)	No (advantage)
Generation: need to shift centers of c_k	No	Yes

Table 2.1.: Comparison of position sharing algorithms

2.5. Security Analysis

This section introduces the attacker model and analyzes privacy guarantees provided by our position sharing approaches. For evaluation, we use the privacy metrics already defined in Section 2.2.

2.5.1. Attacker Model

On the one hand, attackers can circumvent the access control mechanisms of LSs to get access to as many secret refinement shares as possible. On the other hand, attackers can be represented by malicious LSs or providers. In general, attackers have access to k out of n shares, e.g., a compromised LBA has access to k out of n shares for which it received access rights from the MO. As already described in Section 2.1, k defines a trade-off between the QoS that can be offered by the LBA due to the limited precision of position information, and the degree of lost privacy should the LBA misuse the position information. Therefore, in our approach, adjusting k is the basic means of controlling privacy risks. We should note that we do not explicitly consider the case of cooperating attackers, i.e., multiple malicious LS or LBA providers that exchange their shares to increase the number of (compromised) shares. To handle such a case, the MO needs to assess the risk that providers cooperate, which is a different problem of defining suitable trust relations and modeling relations between providers: for example, which LS are sharing the same server (cloud) infrastructure operated by the same third-party provider, or which providers have to reveal their data to the same legal entity because they fall under the same jurisdiction, etc.

As already mentioned, adjusting k is then an effective means to control privacy only if the precision of positions derived from these shares are well-defined. If the share generation algorithm is perfectly secure, an attacker with k compromised shares can calculate a position with at least the precision ϕ_k. However, due to a certain predictability of share generation, the attacker can even increase the precision beyond that value as already discussed in Section 2.4. Since we assume that the share generation algorithm is known to everybody, the attacker can use a *Monte Carlo Simulation* to simulate the process of share generation and predict further possible refinement shares from the known shares (as described in the next section).

To quantify the (undesired) effect of share prediction and the resulting effective security of shares, we use the probabilistic metrics $P_{\text{k,attack}}$ and $P_{k,10\%}$. As it was described earlier in Section 2.2, the privacy metric $P_{\text{k,attack}}$ defines the probability of an attacker refining the MO's position to an area with precision $\phi_{k,\text{attack}}$ where $\phi_{k,\text{attack}} \leq \phi_k$. This metric gives insight into the absolute precision that an attacker can acquire; for instance, an attacker can calculate a position of 500 m precision with 90% probability.

The second privacy metric $P_{k,10\%}$ is a special case of $P_{\mathrm{k,attack}}$. $P_{k,10\%}$ defines the probability of an attacker pinpointing the MO's position π to an area of 10% size of p_k covering the highest probability, i.e., a worst-case 10%-area. A perfectly secure set of shares leads to $P_{k,10\%} = 0.1$, meaning that the position of MO π is uniformly distributed within the obfuscation area. A non-uniform probability distribution of MO within c_k increases $P_{k,10\%}$ to values greater than 10%. This metric is based on a relative area size compared to p_k. It has to be noted that the choice of using 10% instead of another value is based on two reasons. First, the empiric observations have shown that the peaks of high density of non-uniform distributions are usually concentrated in the smaller parts of the obfuscation circle c_k. The selection of an area fraction which is much larger than 10%, e.g., 50%, would hide these higher peaks of probability concentrated in the smaller sub-areas within the selected larger area. For example, consider a pdf with the following parameters: $P_{k,10\%} = 0.9$ (90% of probability corresponds to 10% of precision), showing a very vulnerable π with disclosure of probability 10 times higher than in the ideal case, while $P_{k,50\%} = 0.95$, which is only ca. 2 times worse than the ideal value (which is $P_{k,50\%} = 0.5$). Second, the $P_{k,10\%}$ metric is intuitively understandable for a user: with the base of 10, it is easier to understand the practical meaning of the probabilistic values of $P_{k,10\%}$. For example, $P_{k,10\%} = 0.25$ means that the probability of disclosure is 2.5 times higher than in a perfectly secure (uniform) case. At the same time, for an area other than 10%, e.g., 15%, it would be more difficult to see the actual level of probabilistic guarantees by looking into the resulting value: $P_{k,15\%} = 0.375$ would be an equivalent of a probability which is 2.5 times higher than a uniform probability distribution.

2.5.2. Monte Carlo Simulation

The general idea of Monte Carlo simulation is to obtain a statistically significant number of output samples of a method (or a phenomenon) by executing the given method with randomized inputs [Eck87]. The Monte Carlo simulation is usually employed when analytical description of the simulated method is not available, as we will show later in Section 2.5.6.

In our case, in order to evaluate the probabilistic privacy guarantees provided by share generation algorithms of our position sharing approach, we simulate the attacker's action by repeatedly running a share generation algorithm and sampling the probability distribution of the MO's position π. Then, we analyze the resulting position distribution to determine the most likely area in which the MO is located. The randomized inputs are position shares and, with the exception of the OSPS-ASO "a-priori" algorithm, the center of the master share p_0.

In more detail, this method works as follows: Assuming that an attacker knows the share generation algorithm and k of n shares, we perform the share generation algorithm so many times that 100 full share sets with distance$(p_k^{known}, p_k^{MC}) \leq \epsilon$ are found. Here, p_k^{known} denotes the points resulting from the concatenation of k shares known to an attacker, p_k^{MC} denotes the points resulting from the concatenation of k shares generated by the Monte Carlo simulation, and ϵ is a maximal deviation of p_k^{MC} from p_k^{known}. The deviation ϵ is selected as $\Delta_\phi/10$ empirically, such that the computation does not take more than several seconds. As the result, we obtain a set of 100 target MO's positions $\pi = p_n$ providing a discrete probability distribution. Then, we calculate the probabilistic guarantees $P_{k,10\%}$ by counting the number of π samples inside the worst case 10% area of A_k, i.e., such 10% area of A_k where the maximal number of π points is located.

The results of the Monte Carlo method running 100 times were illustrated above in Section 2.4, with red dots depicting the samples of π: Figure 2.6 shows the resulting pdf for the OSPS-ASO "a-posteriori" share generation algorithm, Figure 2.8a shows the resulting pdf for the OSPS-ASO "a-priori" algorithm.

2.5.3. Open Space Evaluation

Next, we evaluate the security of the share generation algorithm. We assume that the attacker has compromised k out of n shares (or has access to k shares, if we consider a malicious LBA) and uses a Monte Carlo simulation to further increase the MO's precision beyond $\phi_{max} - k\Delta_\phi$. In this section, we begin with the assumption that MOs can move without restrictions in an open space. This evaluation shows the difference between our first approach, OSPS-ASO, presented in [DSR11] that fuses the refinement shares in an arbitrary order, and the fixed order fusion approach, OSPS-FSO.

2.5.3.1. Comparison of OSPS-ASO "a-posteriori" and OSPS-ASO "a-priori"

In Figure 2.16, we depict the dependency of the probability $P_{k,10\%}$ on different privacy (precision) levels k for share generation Algorithm 2 and various n values. The horizontal axis defines the radii sizes for various privacy levels k; the steps of precision decrease are smaller for higher n values, since $\Delta_i^\phi = \phi_{max}/n$, according to our assumptions in this chapter. The vertical axis defines $P_{k,10\%}$, i.e., the probability of π being located within 10% of the current obfuscation area (as explained before in Section 2.2 and Section 2.5.1).

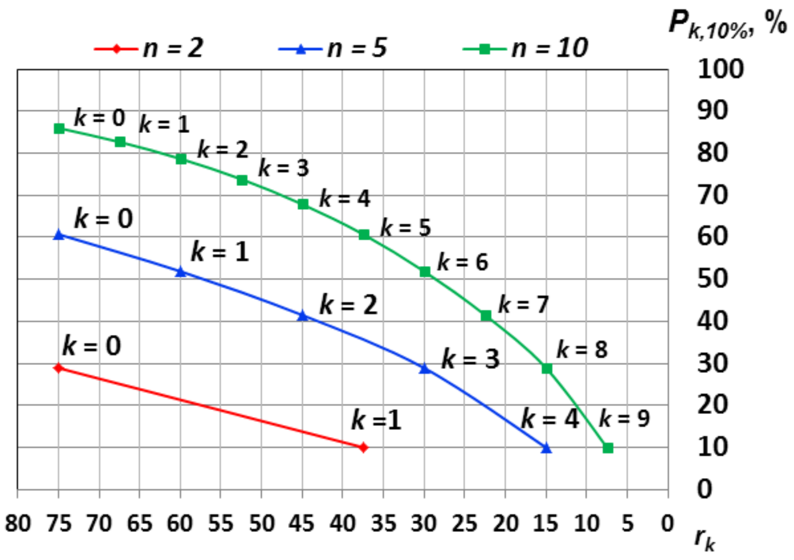


Figure 2.16.: Dependency of probabilistic guarantees $P_{k,10\%}$ on different precision levels represented through the corresponding radii r_k for various n values with $r_0 = 75$ km: "a-posteriori" share generation Algorithm 2; 100 runs of the Monte Carlo method

The probability values $P_{k,10\%}$ are lower for higher precision (i.e., for smaller radii r_k), since the convolution of a lesser number of remaining shift vectors results in a less biased probability distribution. However, note that the $P_{k,10\%}$ values are relative with regard to the current c_k: they show better predictability of π for smaller k within the given c_k, but by knowing fewer shares, an attacker knows a lower precision corresponding to a larger radius r_k. This property is based on fact that in the "a-posteriori" algorithm (as opposed to the "a-priori" algorithm), we do not reach a pre-defined point π but simply add generated random shares to each other. Thus, the more random variables were generated independently from each other, the more predictable is their sum, and the higher are the $P_{k,10\%}$ values for higher n. The basis for this property of the "a-posteriori" share generation will be also discussed in more detail later in Section 2.5.6.1.

We can see that although we have more privacy levels with $n = 10$, $P_{k,10\%}$ is much higher for larger n's if we employ the "a-posteriori" share generation Algorithm 2. Thus, excessively high n values are not preferred in terms of privacy. At the same time, the case of $n = 2$ provides the best probabilities $P_{k,10\%}$, yet with only two possible precision levels, which is not flexible.

We present the dependency of probabilistic guarantees $P_{k,10\%}$ on different precision levels for various n values for share generation Algorithm 3 ("a-priori") in Figure 2.17. The main

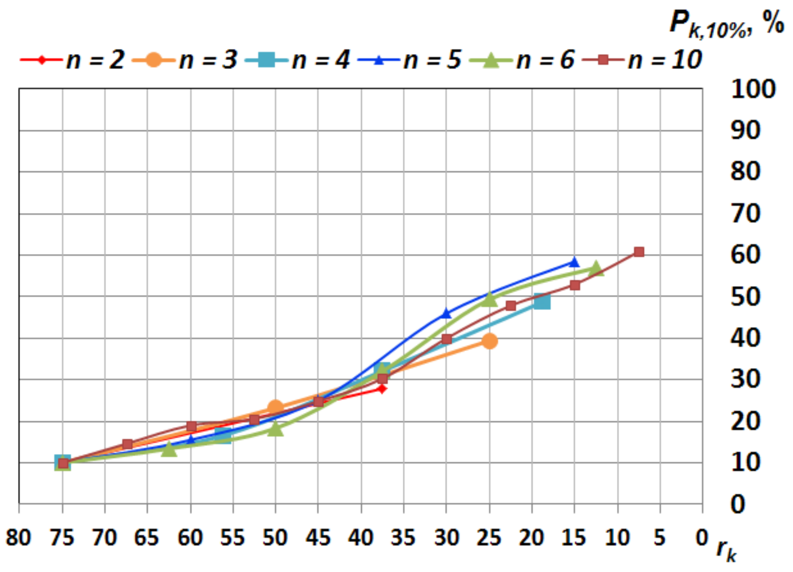


Figure 2.17.: Dependency of probabilistic guarantees $P_{k,10\%}$ on different precision levels represented through the corresponding radii r_k for various n values with $r_0 = 75$ km: "a-priori" share generation Algorithm 3; 100 runs of the Monte Carlo method

difference is that by employing Algorithm 3 we achieved higher $P_{k,10\%}$ values correspond to higher precision. This is due to the fact that with every next known kth share, an attacker derives (through the Monte Carlo simulation) stronger correlation between the known k shares that are directed toward π in more or less fuzzy way. Initially, in the circle c_0, π is distributed uniformly (cf. 10% corresponding to $r_0 = 75$ km in Figure 2.17), which means that π is most likely far away from the center of c_0. Having the maximal length of shift vectors limited by $\Delta_\phi = r_0/n$, the shift vectors are becoming more or less stretched (an example of such stretched vector set was shown in Figure 2.8 of Section 2.4.2.1).

Another important property of the "a-priori" share generation algorithm is that the $P_{k,10\%}$ values are located within a "tunnel", i.e., the correlation between vectors rather depends on the absolute values of precision (horizontal axis) than on k known shares, as it is in case of the "a-posteriori" share generation algorithm. Thus, the "a-priori" algorithm allows for the use of the largest possible value of n, without making the stochastic properties of the generated share sets worse, as happens when increasing n in terms of the "a-posteriori" share generation algorithm.

2.5.3.2. Comparison of OSPS-ASO "a-priori" and OSPS-FSO

First, we analyze how deeply the generated obfuscation circles are mutually intersected in the case of OSPS-FSO. In Figure 2.18, we show the kth circle fraction after k circles intersected, without performing area adjustment. This means that for smaller k values, the intersection cuts a large portion of the previous obfuscation area, but with the increase of k, the circles become much smaller and cannot reduce the obfuscation shape as much as before. The radii are decreasing linearly, while obfuscation areas decrease quadratically.

As a result, the absolute difference between the original radius and the adjusted radius decreases with each k. To evaluate this, we measured how the average radius sizes change after the area adjustment is performed in OSPS-FSO. In Figure 2.19, we compare the average radii of obfuscation circles for OSPS with area adjustment (OSPS-FSO) and OSPS-ASO "a-priori". We can see that the difference between the two curves increases for smaller k values. This is due to the fact that the target area size $A_k = \pi * r_k^2$ is much higher for such circles.

Next, we present the maximal possible shift vector lengths, which the OSPS-ASO and OSPS-FSO approaches produce. This parameter is important, since small vectors lead to uneven

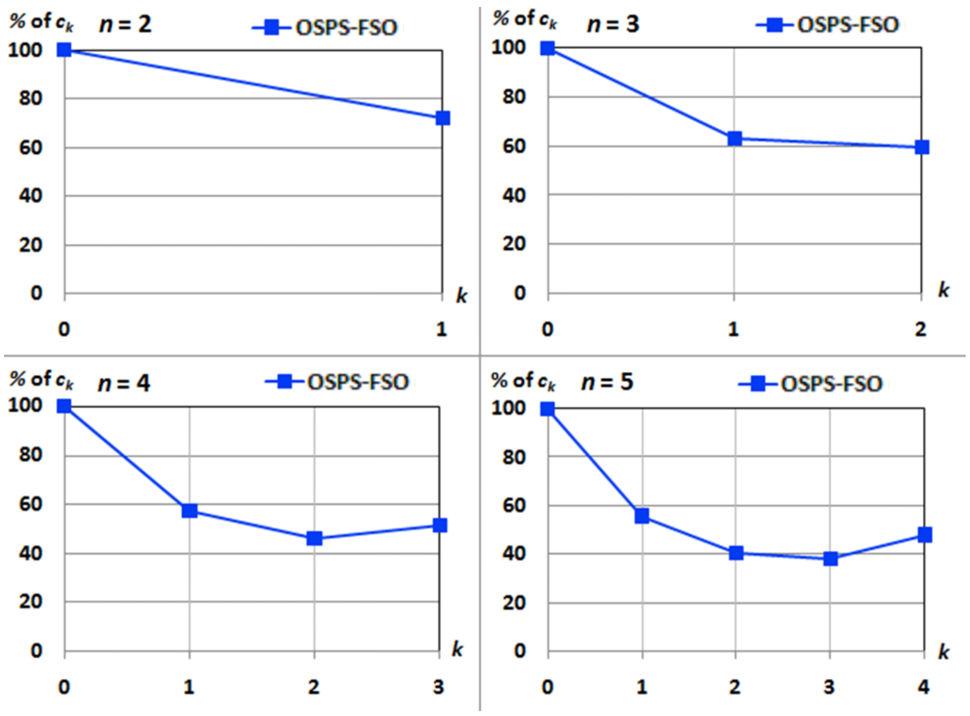


Figure 2.18.: OSPS-FSO: kth circle fraction after k circles are intersected, without area adjustment; 1000 runs of the Monte Carlo method

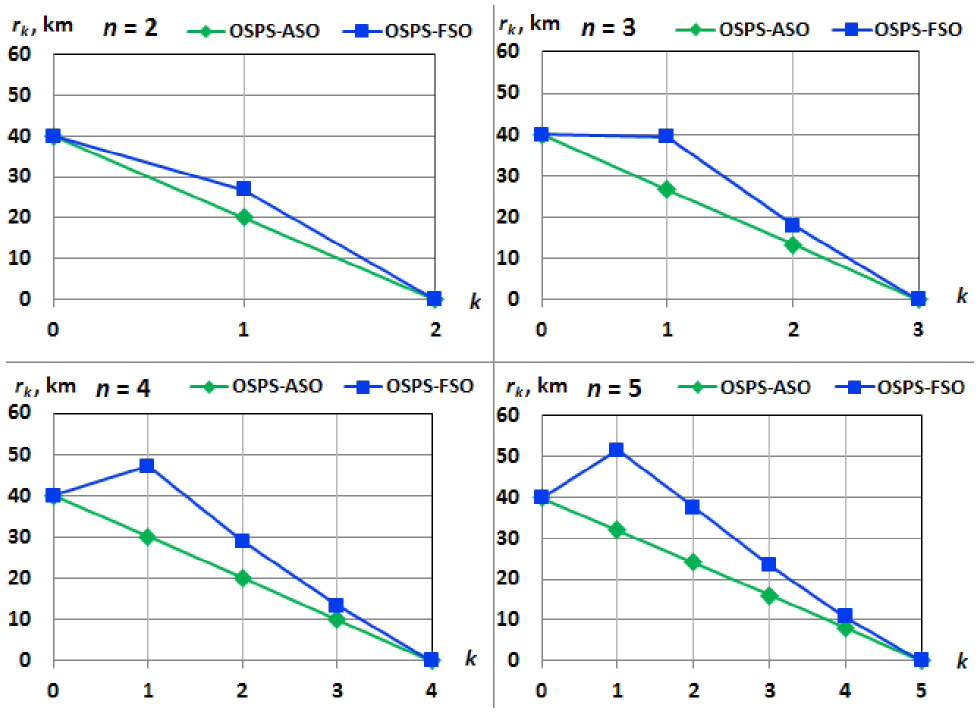


Figure 2.19.: Radii of obfuscation circles c_k depending on k of n shares for OSPS-ASO "a-priori" and OSPS-FSO; 1000 runs of the Monte Carlo method

distributions, while relaxed limitations on vector lengths help to achieve more uniform probability distribution over the obfuscation area. In Figure 2.20, we can see that even the maximal allowed shift value of OSPS-ASO (note that OSPS-ASO allows for any share order, cf. Table 2.1 in Section 2.4.6) is much lower than the average shift value of OSPS-FSO, which requires fixed share order. This is due to the following features of the algorithms: First, the limit of the shift length (r_0/n) always decreases with the increase of n for OSPS-ASO. Second, the average shift lengths of OSPS-FSO do not decrease with a larger total number of shares n. This is an important advantage of OSPS-FSO in comparison with OSPS-ASO, since it leads to a closer to uniform probability distribution.

Figure 2.21 presents different probabilities $P_{k,10\%}$ for different n's, such that we can analyze the effect of area adjustment on OSPS-FSO. The first curve depicted as "OSPS-FSO (no area adj.)" shows the $P_{k,10\%}$ values for the 10% area of the intersection area of k obfuscation circles (without area adjustment). The second curve depicted as "OSPS-FSO (no area adj.; corrected m.)" shows a corrected measurement of $P_{k,10\%}$, where the $P_{k,10\%}$ values are calculated for the 10% area of the the non-intersected area of circle c_k (also without area adjustment). The second curve is needed in order to have comparable $P_{k,10\%}$ values, since after the area

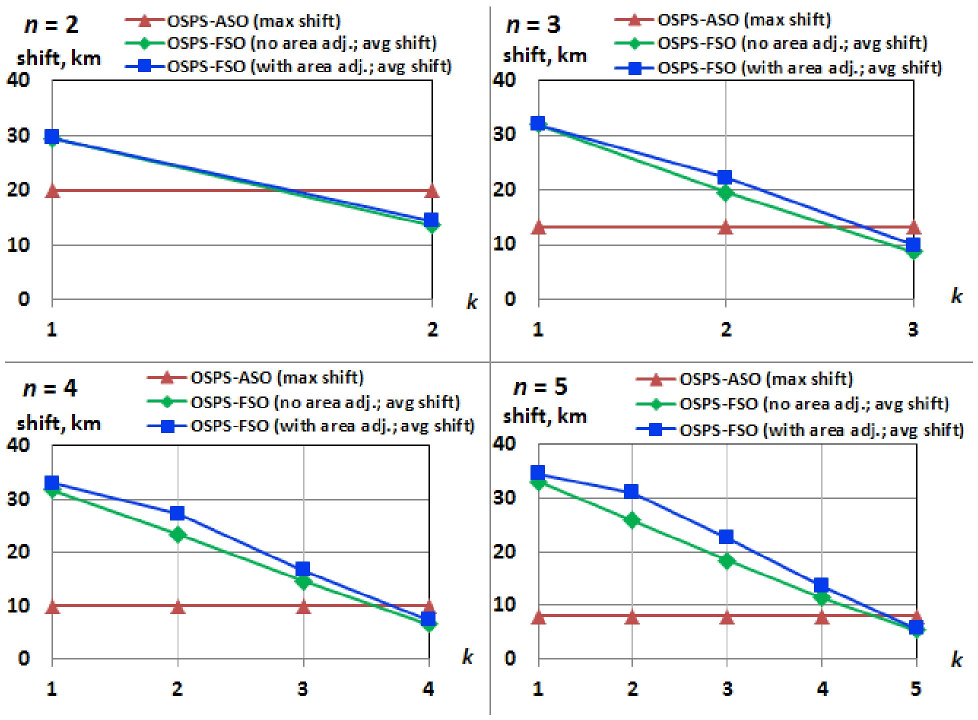


Figure 2.20.: Maximal possible shift of OSPS-ASO algorithms compared with the average shift of the fixed order based OSPS-FSO algorithm without area adjustment, and OSPS-FSO algorithm with area adjustment; $r_0 = 40$ km; 1000 runs of the Monte Carlo method

adjustment, the size of resulting obfuscation area A_k equals the size of c_k. With the third curve depicted as "OSPS-FSO (with area adj.)", we can see that after the area adjustment is done, the probabilities $P_{k,10\%}$ are lower for any $k > 0$.

Next, we show how the computed obfuscation area decreases in accordance with k known shares (Figure 2.22a). In the open space model, this curve is the same for OSPS-ASO and OSPS-FSO: OSPS-FSO does not have any intersections, as it adjusts the obfuscation areas after intersections up to their initial size.

The comparison of OSPS-FSO with the OSPS-ASO "a-priori" approach is shown in Figure 2.22b. By using the privacy metrics defined earlier, we measured the maximal probability of an attacker to derive that the target MO's position π is located within 10% of the current obfuscation circle c_k: $P_{k,10\%} = Pr[10\% * \text{area}(A_k) \leq \phi_{attack}]$. We can see that OSPS-FSO, similarly to OSPS-ASO "a-priori", has the initial uniform distribution of the $p_n = \pi$ over the initial obfuscation area having $k = 0$. For the larger k, the disclosure probability provided by OSPS-FSO is far lower than the one provided by OSPS-ASO "a-priori" and does not exceed

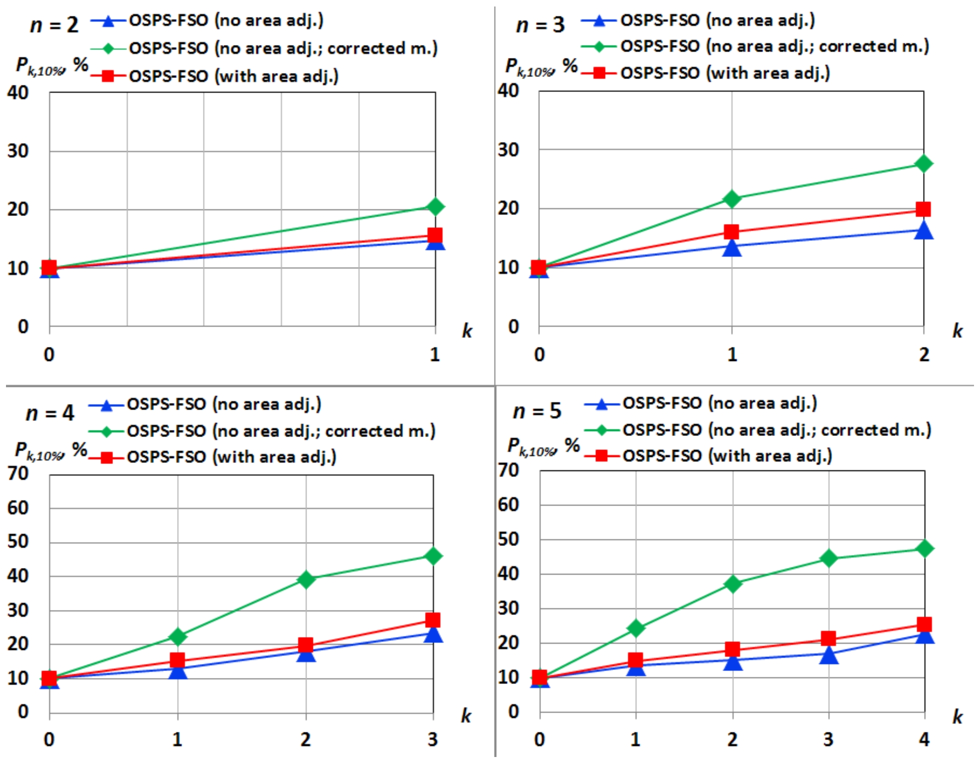


Figure 2.21.: Analysis of OSPS-FSO algorithm with and without area adjustment depending on k of n shares; 1000 runs of the Monte Carlo method

$\approx$ 25% of probability corresponding to 10% of area. Moreover, the increase of $P_{k,10\%}$ probabilities with k is significantly smaller for OSPS-FSO. Generally, OSPS-ASO "a-priori" makes π very predictable for the values of k that are closer to n. Thus, the OSPS-FSO provides lower probability values $P_{k,10\%}$ for the wider spectrum of k within the open space model.

Another view of the user's privacy is shown in Figure 2.23. Figure 2.23a illustrates which fraction of the obfuscation area covers the given probabilities $P_{\mathrm{k,attack}}$ for each number of known shares k out of $n = 5$. The straight diagonal lines represent the linear precision values $P_{\mathrm{k,attack}} = 100\%$, relevant when the share generation algorithm is not known to an attacker. The three other curves represent the precision values corresponding to k levels provided by the OSPS-ASO "a-priori" (Figure 2.23a) and OSPS-FSO (Figure 2.23b) share generation algorithms. Similarly as in Figure 2.22b, we can see in Figure 2.23 that the OSPS-FSO approach provides higher $P_{\mathrm{k,attack}}$ values corresponding to larger radii (i.e., lower precision) than the OSPS-ASO "a-priori" approach. Therefore, OSPS-FSO guarantees lower probability of the precision's disclosure to an attacker than OSPS-ASO "a-priori".

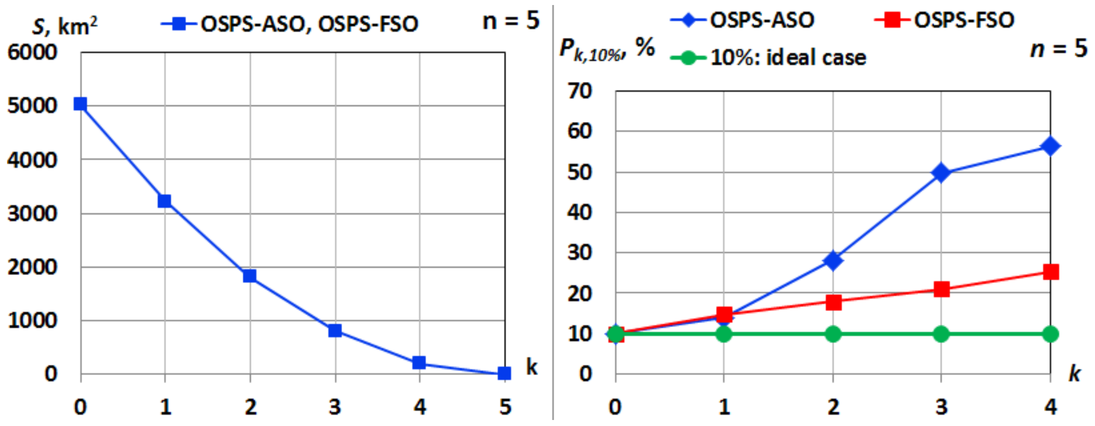


Figure 2.22.: (a) Obfuscation area computed for different k for both OSPS-ASO "a-priori" and OSPS-FSO; (b) comparison of share generation algorithms: probability of deriving that the target MO's position π is located within 10% of the current obfuscation circle c_k ($P_{k,10\%}$); $n = 5$; $r_0 = 25$ km; 100 runs of the Monte Carlo simulation

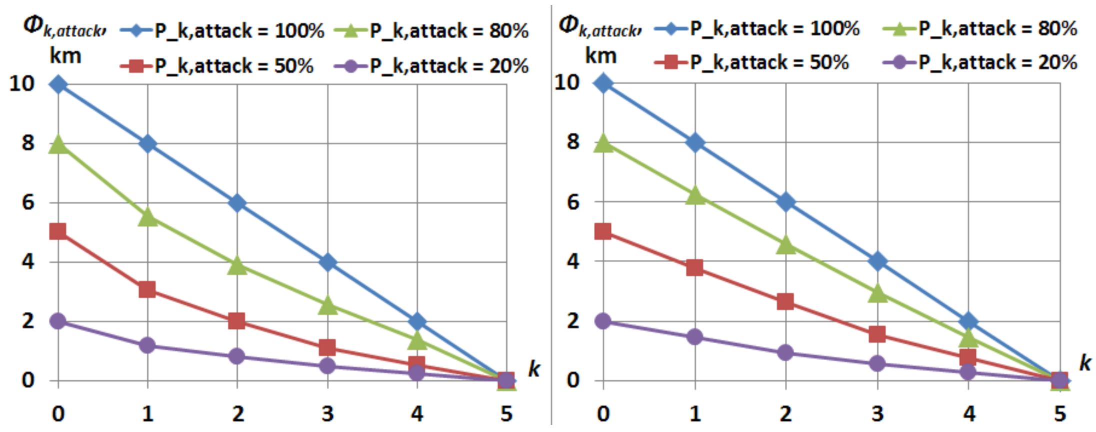


Figure 2.23.: Precision $\phi_{k,attack}$ corresponding to probability $P_{\text{k,attack}}(\phi_{k,attack})$ depending on k for (a) OSPS-ASO and (b) OSPS-FSO; $n = 5$, $r_0 = 10$ km; 100 runs of the Monte Carlo simulation

2.5.4. Constrained Space Evaluation

In this section we first describe the Shapefile format that we use for reading map information. Then we analyze the security of the map-aware share generation algorithm (CSPS).

2.5.4.1. Map Information Format

Shapefile [Esr98] is one of ESRI[1] file formats developed for representing spatial information. It became a de facto standard, as it is supported by many applications, and many open spatial databases are currently available in the Shapefile format. The information represented by the Shapefile format for the same map is stored in several files. They contain geometric objects (features) with corresponding attributes in the form of "key / value" pairs assigned to each map object. The files of the Shapefile format include:

- *.SHP – contains the geometrical data;
- *.DBF – defines the attributes of the geometrical data in form of the "key / value" pairs;
- *.SHX – contains the attributes from the *.DBF file linked to the geometrical data from the *.SHP file;
- .SHP.XML (optional) – contains metadata in XML format;
- .PRJ (optional) – specifies the coordinate system.

Shapefiles distinguish three basic geometrical types: *Point*, *PolyLine*, *Polygon*. By using additional properties, these three basic types can be extended into further types. One extension is based on the multi-set principle: for example, a *Multi-Point* object can contain several points, but it will be considered as a single object. Another advanced geometrical type is so called *Measured Shapetype*. It defines the M parameter assigned to each point in addition to the X and Y coordinates. The M parameter can be used for different purposes, e.g., to represent the height of buildings. Another Shapefile geometrical type is *Multipatch*, which forms a surface of multiple surface parts. In contrast to polygons, the surface parts can be described as "triangle fans" or "triangle strips". These are areas consisting of triangles that are formed by lying next to each other or in a circle.

In our evaluation, we check whether any geometrical types are present in the given location. The most frequently used types are *polygons* and *multipolygons*, since they are applicable to such common map objects as buildings and squares.

[1]ESRI stands for Environmental Systems Research Institute.

2.5.4.2. Analysis of CSPS

This subsection evaluates the privacy characteristics of the map-aware algorithm by considering two different kinds of movement constraints. The effect of map-based constraints for share generation is highly dependent on each map and the user-specific map representation M_u.

As we explained in Section 2.4.5, the only difference of CSPS compared to OSPS-FSO is that the arbitrary-shaped non-circular obfuscation area A_k now includes the intersection with the underlying map knowledge (see Section 2.4.3.4 for details of an arbitrary-shaped area size computation). Otherwise, CSPS is the same algorithm as OSPS-FSO with the same area adjustment procedure (cf. Algorithm 5, Algorithm 8). Therefore, in this evaluation we analyze the effect of the underlying map knowledge on the size of obfuscation area A_k.

We analyzed the obfuscation area reductions caused by two different map types. The first one is a more fine-granular (small-scale) map where M_u is assumed to consist of only the roads and squares of the City of Los Angeles (see Figure 2.24a). The second example shows a more coarse-granular (large-scale) map, where M_u includes all areas except forests in the German state of Baden-Württemberg, i.e., MOs can be located everywhere but in forests (see Figure 2.24b).

We analyze the major difference between the OSPS-FSO and CSPS approaches by calculating the obfuscation areas formed after the intersection of the generated obfuscation circles with the underlying maps. The CSPS generates the obfuscation circles in such a way that

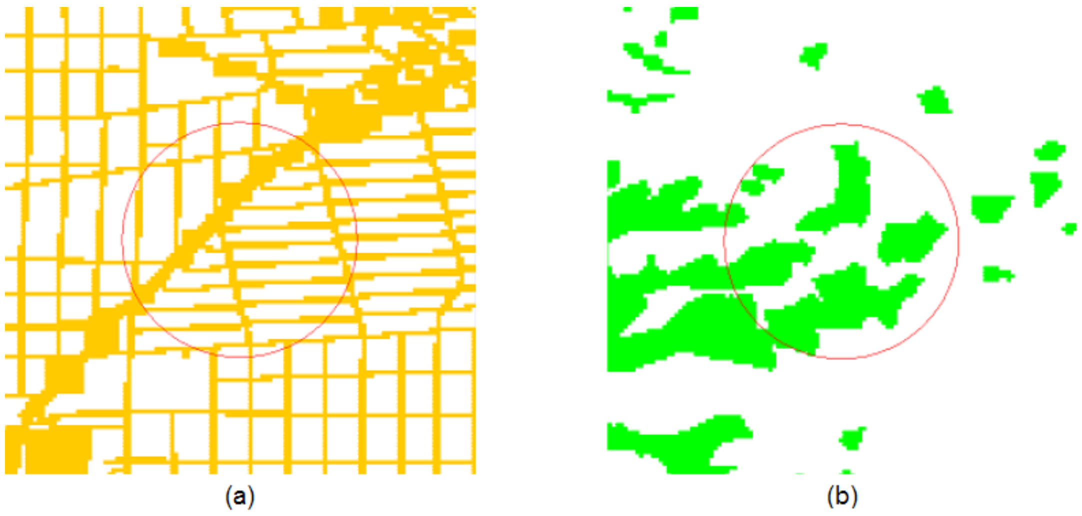


Figure 2.24.: (a) Roads and squares of the City of Los Angeles; (b) forests of Baden-Württemberg

the size of the obfuscation area is never below the desired threshold, as was described in Section 2.4.5. This adjustment is not performed by OSPS-FSO; therefore, the resulting size of the generated effective obfuscation area is smaller and the location privacy of the user is lower.

Our metric is the obfuscation area size which can be expressed as $A_k = M_u \cap \cap_{j=1}^{k}(c_j)$ (cf. Algorithm 8).

In Figure 2.25, we show the resulting sizes of obfuscation areas A_k remaining after intersections. We have generated 1000 sets of shares and calculated the average obfuscation circle fraction of c_k, which is left after intersecting with circles $c_0, \dots c_{k-1}$ for various share numbers n. This value for CSPS is at least 100% or larger due to space discretization and corresponding computational inaccuracy. This is the result of the applied adjustment, while in the case of OSPS-FSO, the value can vary a lot depending on each map.

The two lower curves represent the results of OSPS-FSO. The middle curve shows the result of intersections with a coarse-granular map representation M_1 of Figure 2.24b, while the lowest curve shows the result of intersections with a fine-granular map representation M_2 of Figure 2.24a. “SPS-FSO+M1” depicts the results for the City of Los Angeles map, while “SPS-FSO+M2” depicts the results for the forests of Baden-Württemberg map. We can see that these curves slowly increase from 40% to 60% with increasing number of circles intersected k. This increase is due to the linear-based radius decrease of each kth circle, which causes quadratic area decrease. Since the latter circles are much smaller, they tend to be fully covered by the previous circles despite their large number; moreover, they tend to be overlapped by even smaller map regions where the user can be located. Also, we can see that the intersection with the coarse-granular map representation M_2 decreases the area more than the overlapping with the fine-granular M_1.

The demonstrated results are very important in preserving user’s location privacy: the adjustment function of CSPS keeps the obfuscation area not smaller than 100% of the original size and therefore preserves the user’s privacy (precision) requirements (which depend on both k and the map knowledge). In contrast, if we apply OSPS-FSO, which does not take map knowledge into account within the same adjustment function, the obfuscation would be reduced by 40%–60% for the given maps.

2.5.5. Summary: Comparison of Algorithms

Table 2.2 summarizes the selected properties of probabilistic privacy guarantees provided by the approaches presented in this chapter. In columns two and three, we present the

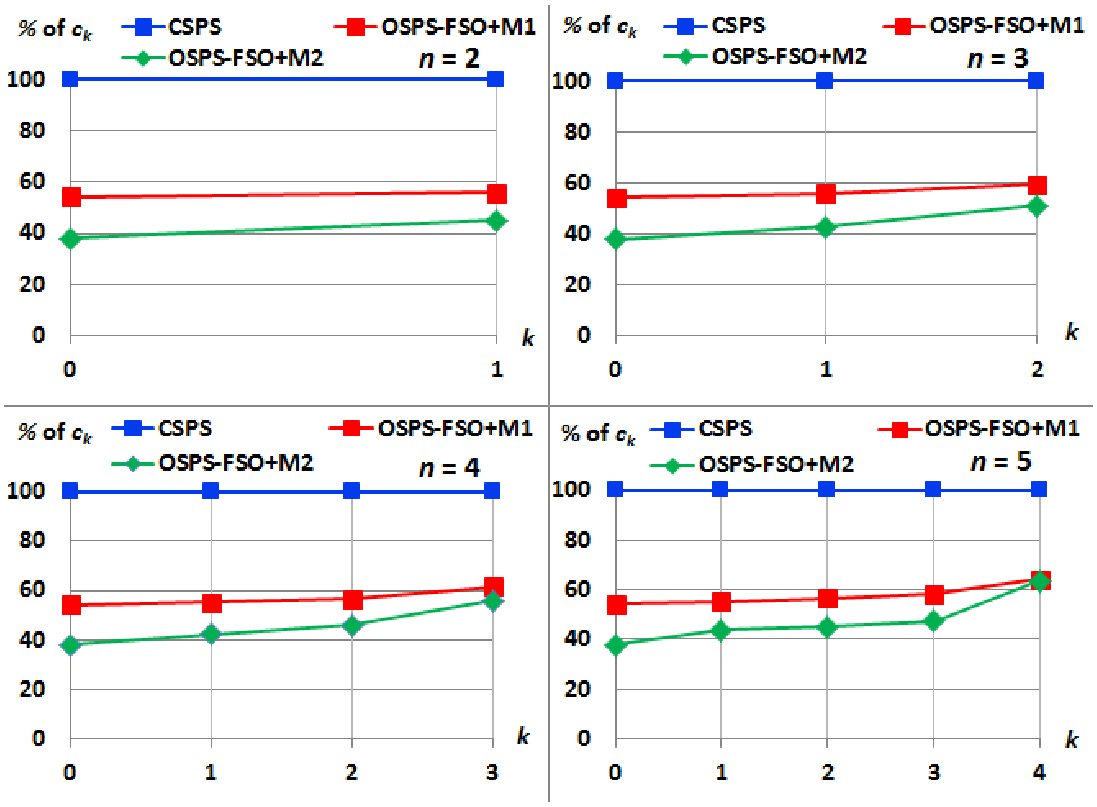


Figure 2.25.: The intersection area of circles $c_0 \cap c_1 \cap \ldots \cap c_k$ of CSPS compared to OSPS-FSO with no area adjustment based on map knowledge; 1000 runs of the Monte Carlo method

properties of OSPS-ASO “a-posteriori” and OSPS-ASO “a-priori” algorithms. In column four, we show the properties of OSPS-FSO and CSPS, since OSPS-FSO has the same properties as CSPS in this context as they represent the same share generation algorithm in its core. The only difference is that CSPS also includes the underlying map reading.

To estimate a pdf, we always require the Monte Carlo simulation, with the exception of the OSPS-ASO “a-posteriori” share generation algorithm, where pdf can be also determined analytically due to its simplicity. However, OSPS-ASO “a-posteriori” provides a pdf that is close to normal distribution, i.e., the precise user position π is easier to predict for an attacker. The pdf of OSPS-ASO “a-priori” is also non-uniform for $k > 0$, yet it is uniform for the important case of $k = 0$. The pdf provided by OSPS-FSO and CSPS are uniform-line, since these algorithms do not require strict limitation of shift vector lengths, which cause the non-uniform pdf character of both OSPS-ASO algorithms.

An important question is whether the increase of n (i.e., generating more shares and distributing them among a larger number of LSs) has a positive effect on security. Our evaluations have shown that the only share generation algorithm that makes π more predictable for higher n is OSPS-ASO “a-posteriori”. All the other algorithms allow for using the benefits of larger n (e.g., for having more precision levels) without making the probabilities $P_{k,10\%}$ worse, i.e., without having increased the probability peaks in their pdf’s. However, it is also interesting to note that the difference between the probabilities $P_{k,10\%} - P_{k-1,10\%}$ of two consecutive precision levels (defined as “probabilistic guarantees delta” in the table) decreases in the OSPS-ASO “a-posteriori” algorithm, thus making the probabilistic peak lower for each next k. In OSPS-ASO “a-priori”, this delta is decreasing, while in OSPS-FSO and CSPS it remains close to constant. Therefore, we can state that each algorithm has its stronger and weaker properties, while CSPS generally has the most positive properties and does not have major drawbacks in comparison with the other algorithms.

2.5.6. Alternative Estimations of Security

In this section, we analyze the security characteristics of the share generation algorithms presented earlier by analytical methods rather than simulation. We mainly target the OSPS-ASO “a-posteriori” and the OSPS-ASO “a-priori” share generation algorithms. At the same time, the conclusions made for OSPS-ASO “a-priori” are also valid for OSPS-FSO and CSPS,

Privacy guarantees of share generation algorithms			
	OSPS-ASO “a-posteriori”	OSPS-ASO “a-priori”	OSPS-FSO and CSPS
Method to estimate pdf	analytical and Monte Carlo	Monte Carlo for k $>$ 0 (for k $=$ 0 not needed)	Monte Carlo for k $>$ 0 (for k $=$ 0 not needed)
Resulting pdf	normal-like	biased (exception: $k = 0$)	Distribution is closer to uniform $\Rightarrow$ better privacy guarantees
Effect of n’s increase	positive (more precision levels); negative (increase of probability peak)	Positive only	Positive only
Probabilistic guarantees delta	Decreasing	Increasing	Constant

Table 2.2.: Comparison of various position sharing algorithms: privacy guarantees

since the latter two algorithms also rely on the “a-priori” selection of MO’s position π within the master share’s obfuscation circle c_0.

2.5.6.1. OSPS-ASO “a-posteriori” and Convolution of Shares

For a rough estimation of how the MO’s position π is distributed in the current obfuscation c_i, the Central Limit Theorem for random walk can be used in the “a-posteriori” algorithm version (Algorithm 2). Assume that the attacker wants to know the distribution of π inside of c_0, i.e., n refinement shares are unknown to the attacker.

According to the Central Limit Theorem, for a sum of independent and identically distributed random elements $S_n = X_1 + X_2 + \ldots + X_n, n \in N$, if each element X_i has finite values of expectation μ and dispersion σ^2, then:

$$\frac{S_n - \mu n}{\sigma \sqrt{n}} \to N(0,1), \tag{2.14}$$

where $N(0,1)$ denotes the normal density distribution. In our case:

$$\sigma \sqrt{n} = \frac{d_{max}^2}{12}; \mu = \frac{d_{max}}{2} \tag{2.15}$$

Here, we can do an a-priori pdf estimation: it is well-known that the convolution of $n \geq 6$ uniformly distributed random variables follows approximately a normal distribution [GS97]. The problem, however, is that for a small n (for $n < 6$) the pdf character is not so clearly defined.

Convolution of Probability Distributions. In general, convolution is a sum of variables or functions. In our case, we need to find how the target MO’s position π is concentrated inside the known obfuscation circle from the point of view of an attacker (in the “a-posteriori” algorithm version). Here, the functions to be convoluted are the unknown shares viewed as vector variables X_i uniformly distributed in the given interval.

Although for $2 < n < 6$ the pdf is still similar to normal distribution according to the Central Limit Theorem, we can use the formula for convolution of multiple probability distributions [KC01] in order to determine the pdf more precisely. The convolution of one-dimensional variables, each of which is uniformly distributed in the interval $[a;b]$, is determined by:

$$f^{(n)}(x) = \begin{cases} \frac{1}{(n-1)!(b-a)^n} \sum_{i=0}^{\tilde{n}(n,x)} (-1)^i \binom{n}{i} (x - na - i(b-a))^{n-1}, & \text{if } na \leq x \leq nb, \\ 0, & \text{otherwise,} \end{cases} \tag{2.16}$$

where $\tilde{n}(n,x) = \left\lceil \frac{x-na}{b-a} \right\rceil$ is the largest integer lesser than $\frac{x-na}{b-a}$.

In special cases when the vector variables X_i are uniformly distributed in the interval $[0,1]$, then $f_{S_n}(x)$ is given by the simplified formula [Usp37]:

$$f_{S_n}(x) = \begin{cases} \frac{1}{(n-1)!} \sum_{i=0}^{x} (-1)^i \binom{n}{i} (x-i)^{n-1}, & \text{if } 0 \leq x \leq n, \\ 0, & \text{otherwise} \end{cases} \tag{2.17}$$

As an example, Figure 2.26 shows the result of two uniformly distributed variables: the special case of convolution known as Triangular Distribution (or Simpson Distribution) [KVD04].

The illustration for $n = 2, 4, 6, 8$ and 10 ($a = 0; b = 1$ for each element) is shown in Figure 2.27. Applying this to our approach, it shows the obfuscation decrease from $n = 10$ down to 2 with two shares obtained at each step. Triangular Distribution is obtained at $n = 2$.

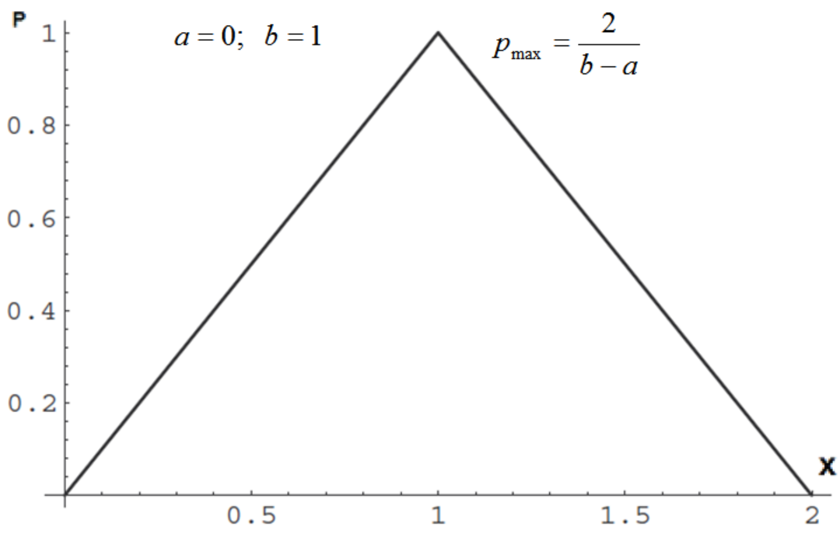


Figure 2.26.: Convolution of two uniform probabilities [GS97]

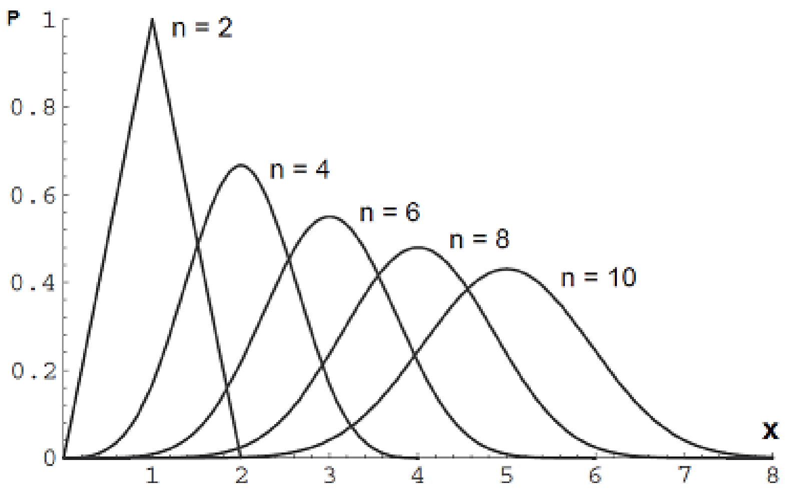


Figure 2.27.: Convolution of n uniform probabilities [GS97]

Note that in Figure 2.27 the coordinate values of the distributions are summed, while in our approach we have different numbers of providers n for the same radii of obfuscation circles. The resulting level of probability changes for different values of n, if the sum of the convoluted variables is fixed as required when generating n shares for the given radius r_0 of the master share (Figure 2.28). The stochastic properties of the share set become worse (i.e., the target MO's position becomes more predictable) with increasing of n in the case of the "a-posteriori" share generation algorithm. This occurs despite the fact that the larger number of shares allows for the distribution of position information among a larger amount of LSs and therefore reduces the risks of its disclosure. This property was analyzed in more detail in Section 2.5.3; also see Figure 2.16.

The previous examples deal with one-dimensional variables. In order to get a two-dimensional convolution of uniform probability distributions, we need to multiply two one-dimensional functions such as shown in Equation 2.16, one for X and one for Y [GS97, Mat00]. The resulting two-dimensional pdf is called circular bivariate distribution and retains the properties of the one-dimensional distributions, i.e., it is normal-like (see Figure 2.29).

2.5.6.2. OSPS-ASO "a-priori" and Impossibility of Deconvolution of Shares

Here, we explain why for OSPS-ASO "a-priori" (Algorithm 2) we cannot apply a convolution-based approach as for the "a-posteriori" algorithm version (Algorithm 3), which makes impossible the use of a mathematical formula to determine the probability distribution of π

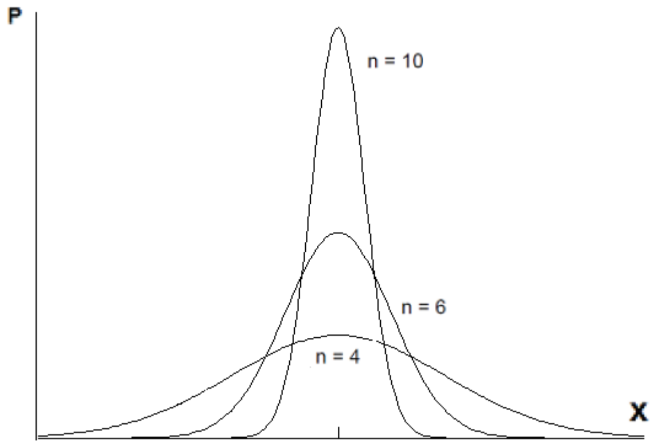


Figure 2.28.: One-dimensional pdf's for different n values, with the sum of the convoluted variables fixed: curves with higher peaks correspond to higher n

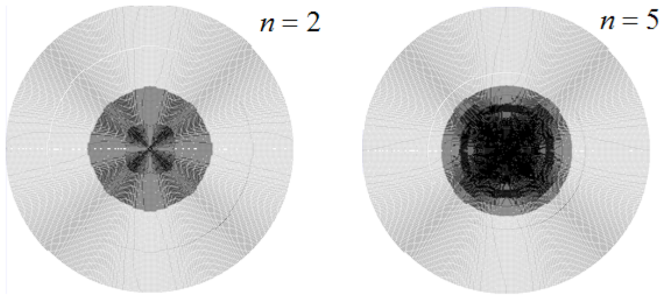


Figure 2.29.: Circular bivariate distributions for $n = 2$ and $n = 5$

for the "a-priori" algorithm version (Algorithm 2).

To find the precise probability distribution of π resulting from the convolution of $n - i$ unknown shift vectors by knowing i shift vectors ($i = 0 \ldots n - 1$), we would need to use the deconvolution procedure [PZ02]:

$$\begin{aligned} \text{convolution: } h(x) = (f * g)(x) = \int_{-\infty}^{\infty} f(t)g(x-t)dt \\ \text{deconvolution: } f(x) = F^{-1}[\hat{h}/\hat{g}], \end{aligned} \tag{2.18}$$

where F is the Fourier transform of function f.

Our goal is to find an intermediate distribution after convoluting $n - i$ shift vectors by knowing that the resulting probability distribution, i.e., that the convolution of n shift vectors results in a uniform distribution.

However, such a deconvolution is “ill-posed”, meaning that we only know the resulting distribution, while a probability distribution of any intermediate variable (i.e., shift vector) is unknown. Even if an attacker knew the first distribution (i.e., the distribution of already obtained i vectors’ concatenation points), the deconvolution result is represented by a multidimensional family of functions, and not by a single function [PZ02].

At the same time, statistical solutions of Circular Random Walk [Ste63] are not applicable for OSPS-ASO “a-priori” due the following reasons: First, there is an insufficient number of variables must be calculated (usually, the number of shares n is not expected to be much larger then 10). Second, although the length deviation is known, the angular deviation required for statistical analysis is unknown: at the first step ($i = 1$), no deviation is known; at steps $i > 1$ only an intermediate deviation is known, since p_n is not revealed until the last share.

2.5.6.3. Alternative Estimations of Security: Summary

An attacker can estimate a pdf for the OSPS-ASO “a-priori” algorithm by using the Central Limit Theorem of convolution of probabilistic variables. However, these estimations can be only precise in case of large n, while we expect that the number of shares n within the position sharing approach usually is not going to be larger than 10. If n is smaller or equals 10, as in our evaluations, the Monte Carlo simulation provides better, i.e., more precise pdf’s.

In case of the OSPS-ASO “a-priori” algorithm, we have shown that it is not possible to determine the probability distribution of π through a non-heuristic (analytical) solution such as deconvolution or Circular Random Walk, since we know only the resulting pdf and cannot determine stochastic properties of the intermediate variables, i.e., the probability distributions of the refinement shares generated by OSPS-ASO “a-priori”. Therefore, we have used the Monte Carlo simulation earlier in this chapter to evaluate the probabilistic guarantees of the precision levels provided by our share generation algorithms.

2.6. Performance Evaluation

Since the processing power and storage capacities of today’s server-side (e.g., cloud) infrastructures are much more advanced than the MO’s device characteristics, the client side becomes the bottle-neck of the system, whereas the server side is less important with regard to cost optimization. Therefore, not only the security guarantees but also the processing costs of algorithms are important, especially for small hand-held mobile devices.

First, we analyze the execution times of our share generation algorithms on a mobile device. Second, we make an estimation of attacker's computational overhead assuming more powerful hardware. Third, we consider the communication cost required to transmit the necessary shares from MOs to LSs. Since in this chapter we assume snapshot position updates, we estimate the size of the messages and not their number.

2.6.1. Evaluation Setup

According to our approach, the share fusion is done on the LBA side, while share generation must be done locally by the MO. Moreover, while share fusion algorithms always have linear complexity $\mathcal{O}(n)$ to sum n vectors, the share generation in all algorithms except the first ("a-posteriori" OSPS-ASO) version is non-deterministic, being usually more complex and requiring more time. We therefore focus on analyzing the computational overhead required by the share generation algorithms.

We implemented our share generation algorithms and ran them on the HTC Desire HD smartphone with Android OS (CPU: 1 GHz Qualcomm QSD8250 Snapdragon, memory: 576 MB RAM). We measured the average time it took for our share generation algorithms to generate the full new set of position shares for a single position update. For this evaluation, we used the Google Caliper micro-benchmarking framework, which executes each share generation algorithm multiple times and calculates the average time required.

It is reasonable to assume that the adversary has a more powerful device available to analyze the pdf of the MO's position. Thus, we simulated an attacker's Monte Carlo analysis for FSO and CSPS algorithms by using hardware with Intel Core i7 CPU (1.60 GHz) and 4GB RAM.

2.6.2. Processing Overhead

In Table 2.3, we can see the average execution time in milliseconds (ms) of the first two algorithms based on open space (OSPS-ASO "a-posteriori" and OSPS-ASO "a-priori") for the sets of shares of different sizes n generated for a single position update. As expected, the execution time of OSPS-ASO "a-posteriori" is always significantly smaller than for OSPS-ASO "a-priori", since OSPS-ASO "a-priori" has to traverse many more vector sets in order to find the one that connects the starting point p_0 with the end-point p_n.

n	2	4	8	16	32	64	128
OSPS-ASO “a-posteriori”	0.06	0.21	1.22	8.65	16.2	561	$\gg 1\,\mathrm{s}$
OSPS-ASO “a-priori”	3.4	32.6	164.27	2273	$\gg 1\,\mathrm{s}$	$\gg 1\,\mathrm{s}$	$\gg 1\,\mathrm{s}$
OSPS-FSO without area adj.	0.9	1.1	1.4	1.6	1.7	2.2	2.9
OSPS-FSO with area adj.	3.5	4.4	11.0	1 s	$\gg 1\,\mathrm{s}$	$\gg 1\,\mathrm{s}$	$\gg 1\,\mathrm{s}$
CSPS	17.1	104.6	1039	$\gg 1\,\mathrm{s}$	$\gg 1\,\mathrm{s}$	$\gg 1\,\mathrm{s}$	$\gg 1\,\mathrm{s}$

Table 2.3.: Processing overhead of OSPS-ASO algorithms, ms

The time needed to generate the set of shares for OSPS-FSO without area adjustment is always smaller than for both OSPS-ASO versions. The reason for this difference is that OSPS-ASO applies more constraints to the vectors, and therefore many more randomly generated sets are dismissed before the suitable set (which guarantees the arbitrary order of share fusion) is found.

OSPS-FSO with area adjustment is slower than OSPS-FSO without area adjustment but still faster than both OSPS-ASO versions, since intersection area computation is simple when only circles are considered. In turn, the processing times required for the map-aware CSPS approach are highly dependent on the map granularity and the efficiency of reading the map data. In Table 2.3, we provided the CSPS execution results for the City of Los Angeles map presented in the previous section. We can see that CSPS is the slowest share generation algorithm, since it requires to read the map data during the computation of the intersection area.

We can analyze absolute execution times by assuming a maximum position update rate of 1 Hz, which is the maximum rate of common GPS receivers. This assumption about update rate can be considered as the worst case in terms of communication overhead, representing frequent and continuous tracking scenario. Under this assumption, OSPS-ASO “a-posteriori” is able to generate more than 64 shares online in real time until one second elapses, whereas OSPS-ASO “a-priori”, OSPS-FSO (with area adjustment) and CSPS can generate at least 8 shares within one second, i.e., provide 8 different privacy levels. For most LBAs, 8 privacy levels should be sufficient; moreover, we expect that the majority of LBAs will have much smaller position update rates than 1 Hz. Therefore, we can conclude that real-time position obfuscation is feasible with our algorithms.

2.6.3. Attacker's Overhead

An important issue is how efficiently an attacker can obtain a pdf of the user's position by performing the Monte Carlo simulation for different values of k number of already known shares and different share generation algorithms. However, note that even if an attacker had have a very powerful hardware and could obtain the pdf's within a reasonable time, those pdf's would provide no worse probability distributions $P_{\text{k,attack}}(\phi_{\text{k,attack}})$ than the ones guaranteed by the corresponding share generation algorithms.

Here, we analyze the processing time required for an attacker if OSPS-ASO "a-posteriori" (Algorithm 2), OSPS-ASO "a-priori" (Algorithm 3), OSPS-FSO (Algorithm 5) and CSPS (Algorithm 8) are employed.

In our analysis, the attacker uses the Monte Carlo simulation to sample the pdf according to our attacker model (see Section 2.5.1) and the simulation methodology described in Section 2.5.2. The attacker executes the known share generation algorithm (in the way as a regular user would perform this for generating shares) enough times in order to obtain a set of 100 samples of the exact MO's position π over the obfuscation area. An attacker tries to utilize the k known shares to get the resulting pdf that represents the correlation between the known k shares and the resulting position π. For each further share of k known shares, the Monte Carlo simulation looks for a matching k of the newly generated random share sets, while the previously found set match for $k-1$ shares cannot be reused. If k generated shares match with k known shares, position π is obtained and saved as a sample point for building a pdf. As a result, the time required for the Monte Carlo tests grows exponentially with the increase in k, as our measurements show (cf. Table 2.4). The only exception is the OSPS-ASO "a-posteriori" algorithm, since in this algorithm each generated share is stochastically independent, and therefore $k-1$ shares can be reused.

We also can see that it is very costly for an attacker to find share sets that match the k known shares for the share generation algorithms OSPS-FSO and CSPS, since these algorithms require area size computations. Moreover, CSPS is dependent on the concrete map and its granularity during the computations. Thus, more time (more than one hour for $k > 2$ and $n > 4$) is required to simulate the share generation algorithm of CSPS than of OSPS-FSO due to the map analysis. In contrast, 100 runs of the Monte Carlo simulation for the both versions of OSPS-ASO ("a-posteriori" and "a-priori") is much faster than in case of CSPS and OSPS-FSO. This simulation is done within one minute for the given n and k combinations, and therefore it is easier for an attacker to derive the desired pdf for OSPS-ASO.

n	3		4			5		
k	1	2	1	2	3	1	2	> 2
OSPS-ASO “a-posteriori”	0.08	0.11	0.09	0.12	$\ll 1\,\mathrm{s}$	0.09	0.16	$\ll 1\,\mathrm{s}$
OSPS-ASO “a-priori”	1.69	30.6	2.07	38.28	$> 1\,\mathrm{m}$	6.34	63.75	$> 1\,\mathrm{m}$
OSPS-FSO	2.13	88.6	7.74	191	$> 1\,\mathrm{h}$	12.7	271	$> 1\,\mathrm{h}$
CSPS	419	$> 1\,\mathrm{h}$	814	$> 1\,\mathrm{h}$	$\gg 1\,\mathrm{h}$	973	$> 1\,\mathrm{h}$	$\gg 1\,\mathrm{h}$

Table 2.4.: Attacker’s processing overhead for Monte Carlo analysis resulting in 100 samples of position π, having k known shares out of n shares, s

2.6.4. Communication Overhead

Next, we estimate the size (payload) of the update messages and the corresponding communication overhead. Despite the fact that all of the n shares for each position update must be sent, each single share is quite small in size: user id (≈ 32 bytes) + 2 floating point numbers for the shift vector (8 bytes). If this piece of information is sent using the UDP protocol with a binary format, we can estimate that the transmission of, for example, 8 shares requires only about 550 bytes to send. Taking into account that the most modern LBSs use HTTP protocol with JSON or XML payloads, the size of a single position update might be around two times higher. The major part of the traffic amount generated by a location update is required for the establishment of secure sessions; for example, 1810 bytes would be used by SSL protocol [PRRJ06]. The given values demonstrate that in our approach it is not a problem (in terms of messages size) to send the required amount of position information after each update.

Another important issue regarding communication overhead is that our approach produces a large number of messages to be sent in cases where a naïve update protocol is applied. For instance, shares do not have to be updated as long as the user moves within the smallest obfuscation circle. If he or she moves further, the complete share set does not necessarily need to be updated at once. In many situations, it might be sufficient to update only a single or a few shares. Our optimized location update protocol will be presented in Chapter 4.

2.7. Related Work: Privacy in Location-based Services

This section gives an overview of major existing techniques for protecting location privacy: cryptography, position dummies, mix zones, k-anonymity, spatial obfuscation, coordinate transformation and secret sharing. Then, we classify them according to their privacy goals and the adversarial attacks they are able to resist. The major contributions of this section were originally published in [WSDR14].

2.7.1. Cryptography-based Approaches

A classic solution to ensure the confidentiality of the user's position is to use *cryptography*. However, by encrypting user positions stored on servers, server-side query processing of advanced queries like range queries over the encrypted data is usually impossible, or possible only at a very high cost [RPB08].

Another example of a cryptography-based approach for location privacy was proposed by Mascetti et al. [MFB$^+$11] in *proximity services* of geo-social networks. The authors assume that service providers are untrustworthy and consider the scenario where mobile users want to notify their friends called *buddies* of their proximity. The main idea is that the secret keys are shared with the selected buddies in a distributed fashion and remain unknown to the service providers. The authors use a precision metric which is defined through the union of multiple discrete space cells called *granules*. A drawback of this approach is that it requires a complex implementation of the encryption functionalities, and it is mainly suitable for the specific case of proximity calculation within geo-social networks.

2.7.2. Position Dummies

The goal of *position dummies* is to secure the user's true position by sending multiple false positions ("dummies") to the location servers together with the true position [KYS05]. The essential advantage of this approach is that the user himself/herself can generate dummies without the need for trusted third-party components that could introduce additional security problems. The user identity is not secured by such dummies, because new fake IDs are not generated together with the dummy positions.

Various examples of position data distribution by adding dummy positions with different ubiquity, congestion and uniformity parameters to the true user position are shown in Figure 2.30a-e. Obviously, it is challenging to create dummies that cannot be distinguished from the true user position, particularly, when an adversary has additional context information such as a roadmap and can track the user for longer times.

An advanced method to generate dummies is presented in the *SybilQuery* approach proposed by Shankar et al. [SGI09]. This approach assumes that the mobile user has a database with traffic history for the surrounding area, which allows him or her to create additional dummy positions along the past real trajectories of other MOs, so that these dummy positions cannot be distinguished from the real user positions. Figure 2.31 shows (a) a simple location query compared with (b) a location query using the SybilQuery approach. Note that the SybilQuery approach prevents the exclusion of fake dummies by employing the location tracking analysis, since all points lie along real movement paths. For this approach to be possible, a database with MOs' traffic history is required, which can be outdated or difficult to obtain.

Pareschi et al. presented another approach based on dummies, which resist *"shadow attacks"* [PRB08]. The shadow attack can be performed by an attacker that uses LBS and pretends to be another user, i.e., the attacker issues location-based queries being a "shadow" under a fake pseudonym of the target user. Then the attacker can get additional knowledge about the target user by analyzing the content of service responses. The proposed defense techniques include fake query generation and delaying user queries.

The general problem with the dummies approach is that dummy positions can be easily

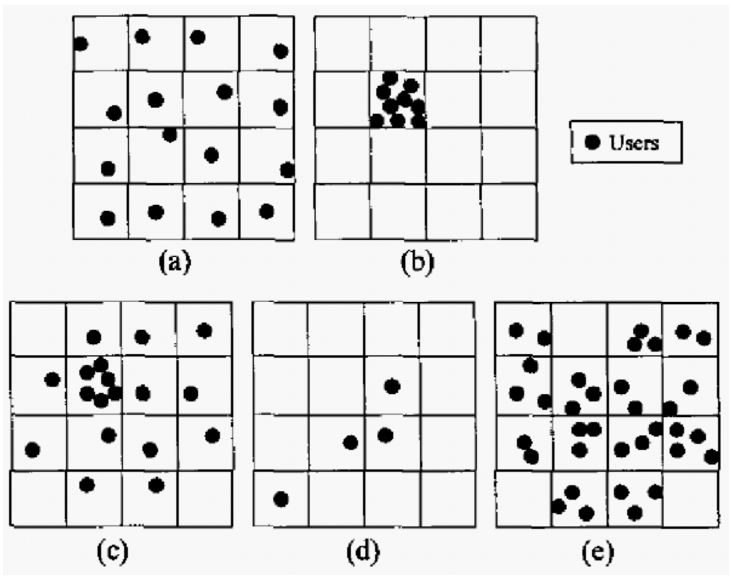
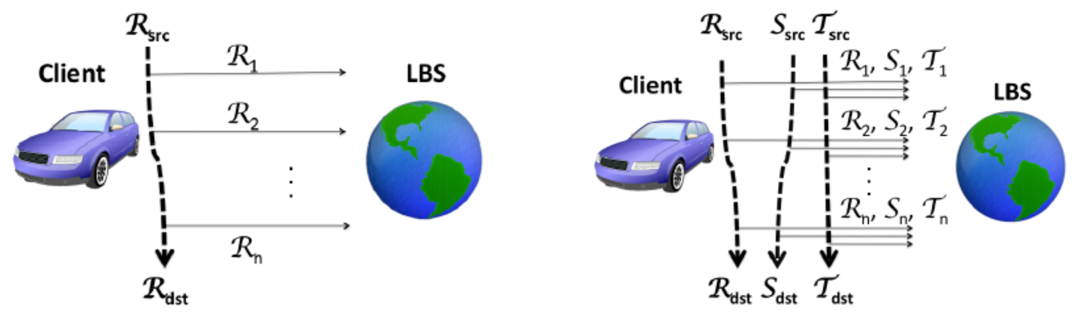


Figure 2.30.: Example of position data distribution with diverse ubiquity, congestion and uniformity parameters [KYS05]


$\mathcal{R}_{src}$
Client
$\mathcal{R}_1$
$\mathcal{R}_2$
$\mathcal{R}_n$
LBS
$\mathcal{R}_{dst}$
$\mathcal{R}_{src}$ $\mathcal{S}_{src}$ $\mathcal{T}_{src}$
Client
$\mathcal{R}_1, \mathcal{S}_1, \mathcal{T}_1$
$\mathcal{R}_2, \mathcal{S}_2, \mathcal{T}_2$
$\mathcal{R}_n, \mathcal{S}_n, \mathcal{T}_n$
LBS
$\mathcal{R}_{dst}$ $\mathcal{S}_{dst}$ $\mathcal{T}_{dst}$

(a) Without SybilQuery, the client sends its actual location to the LBS as it moves along path $\mathcal{R}$.

(b) With SybilQuery, the client sends both its actual location along path $\mathcal{R}$ and along Sybil paths $\mathcal{S}$ and $\mathcal{T}$.

Figure 2.31.: Querying an LBS: (a) without SybilQuery; (b) using SybilQuery with $k = 3$ [SGI09]

distinguished from the real positions if an attacker has some background information such as database of real user movements [SGI09]. Even if the real paths database is assumed to be available, the dummies approach does not provide robust privacy guarantees or multiple privacy levels. In addition, the transmissions of fake positions and operations on them cause excessive costs.

2.7.3. Mix Zones

The idea of the **mix zones** approach proposed by Beresford et al. [BS04] is to define privacy-sensitive areas called mix zones, where all users must be protected such that the user position is hidden within these zones. This is achieved by not sending any position updates within a zone. However, by observing the user positions right outside a mix zone and while entering and leaving zones, it is possible to infer their trajectory inside the zone and to link incoming and outgoing traces of a mix zone to obtain complete trajectories. Therefore, Beresford et al. introduced *dynamic pseudonyms* [BS04] as a part of the mix zone concept, which means that the mobile user changes his or her pseudonym upon entering and exiting the mix zone in order to protect his or her identity. As such, it becomes more difficult for an adversary to determine the user's trajectory within the mix zone by tracing the entry and exit points of a given user.

A mix zone with three mobile users moving through it is presented in Figure 2.32. The entry and exit points of the mix zone are reached by the mobile users under different pseudonyms, making it more difficult for an attacker to obtain their trajectories within the mix zone.

The *MobiMix* approach presented by Palanisamy and Liu [PL11] applies the mix zone

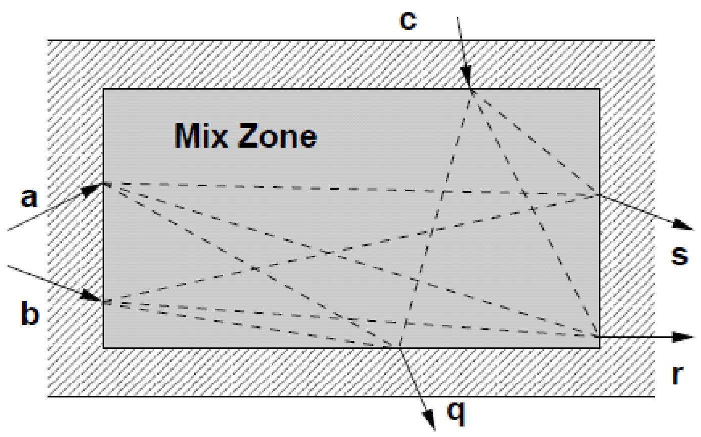


Figure 2.32.: Example of movement of 3 users through a simple mix zone [BS04]

concept to road networks. The authors extended the basic mix zones concept by taking into account diverse context information, which can be used by an attacker to derive detailed trajectories. The additional parameters include geometrical, temporal constraints and users' density. The MobiMix approach is illustrated in Figure 2.33, showing the mix zones concept applied over a road network, thus obfuscating direction of each car after entering the mix zone.

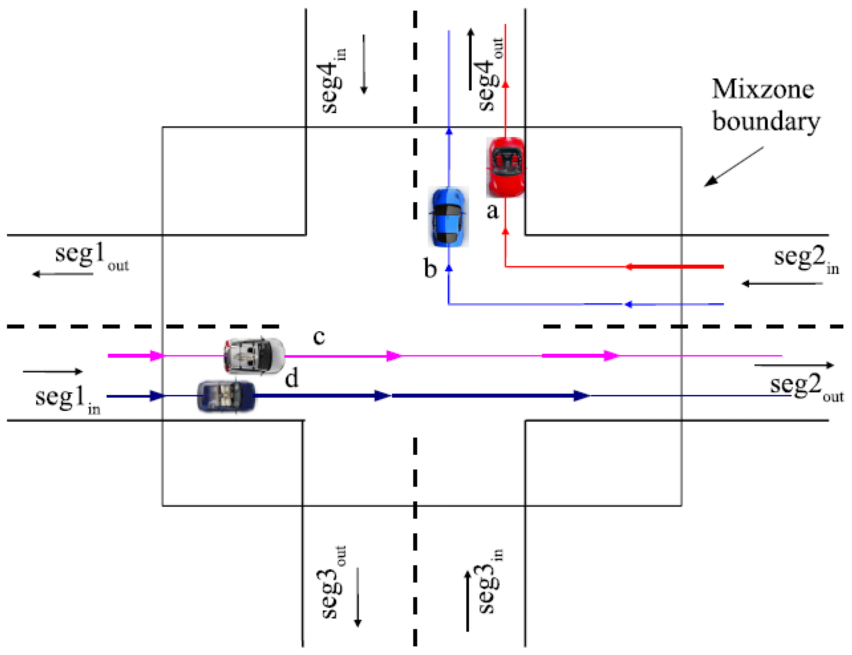


Figure 2.33.: Mix zone over a road network [PL11]

Another extension of the mix zones idea is the *dynamic mix zones* approach of Ouyang et al. [OXL+08]. It adapts the mix zones dynamically depending on the user movements instead of having mix zones with pre-defined static boundaries. Figure 2.34 shows the heuristic selection of a dynamic circular mix zone with its center in A, B and C for a single time point, such that it covers the maximum number of user positions.

The approaches based on mix zones lack flexibility, because they need a pre-defined location-based (or user-based, as in [OXL+08]) division of space into fixed zones, and they do not allow for different levels of privacy in different zones.

2.7.4. k-anonymity

k-anonymity is a widely accepted concept that guarantees that in a cluster (i.e., a set or tuple) of k objects (in our case, mobile users), only one of them is the target object and it is indistinguishable from the $k-1$ other objects of the cluster. Thus, the probability of determining the target user is $1/k$.

Many approaches exist that apply the general concept of k-anonymity to location privacy. The *adaptive cloaking* approach, as proposed by Mokbel et al. [MCA06], considers the minimal size of the obfuscation area including k users. It is based on hierarchical division of space into cells in order to cluster mobile objects in the form of a pyramid. Thus, by preserving the original precision and privacy, this approach reduces the cost of querying, i.e., the cost of search for the requested obfuscation area. The principle of adaptive cloaking is presented in Figure 2.35. Only the space cells adjacent (i.e., needed to meet privacy requirements) to the MO's movement are maintained, instead of maintaining the whole pyramid's index structure.

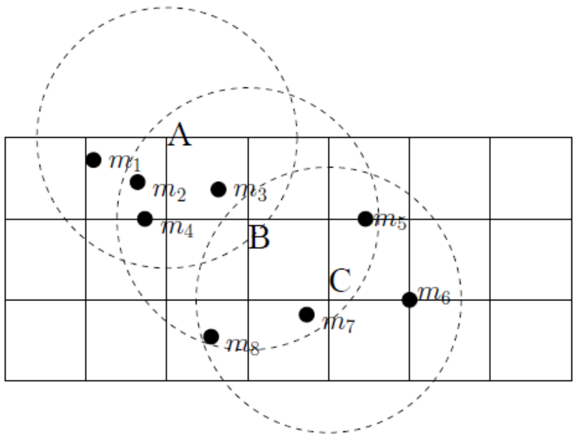


Figure 2.34.: Dynamic circular mix zones [OXL+08]

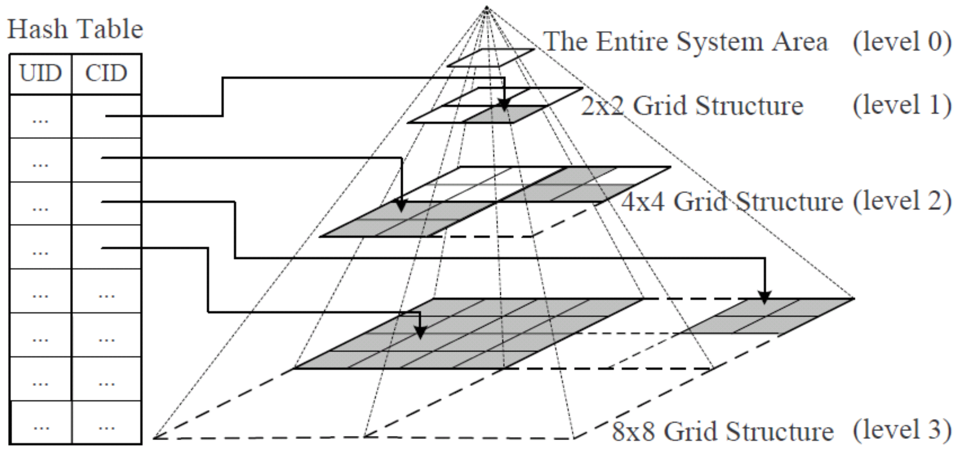

Hash Table
UID
CID
...
The Entire System Area (level 0)
2x2 Grid Structure (level 1)
4x4 Grid Structure (level 2)
8x8 Grid Structure (level 3)

Figure 2.35.: Adaptive location anonymizer [MCA06]

Gedik et al. proposed the *CliqueCloak* algorithm [GL05, GL08] based on k-anonymity, which also performs spatial and temporal cloaking. Similarly to the *adaptive cloaking* approach [MCA06], in order to preserve acceptable location privacy, a user can additionally define individual upper limits for both obfuscation area size and time periods associated with his or her positions. Figure 2.36 illustrates the Clique-Cloak algorithm with the resulting 3-dimensional spatio-temporal cloaking boxes (2.36e).

Usually, achieving k-anonymity requires a TTP which has a global view of the service users. An exception is the approach of Chow et al. [CML06] that avoids using a single trusted anonymizer by using P2P communication to find a spatial region so that it covers the needed number of other $k-1$ mobile users (i.e., a cluster). After the needed cluster is found, the user sends it to the client indirectly by using a randomly selected node to hide the identity of the query issuer. The system architecture for the P2P spatial cloaking is presented in Figure 2.37. It consists of LBS databases and mobile users who can communicate with each other and send queries to LBSs through the base station.

Zhang et al. [ZH09] introduced an approach to combine the use of a TTP and P2P principles. The proposed system can switch between these modes depending on privacy settings and other parameters, therefore balancing the work load and communication cost in a more efficient way. To improve the security of the P2P mode, the authors present the Random Range Shifting (RRS) algorithm (illustrated in Figure 2.38). To avoid centering the mobile object in the selected cluster of users, the space cluster is selected randomly while the algorithm is looking for k users. Thus, this version of k-anonymity guarantees that the calculated clusters of k users remain the same over several queries. This property of k-

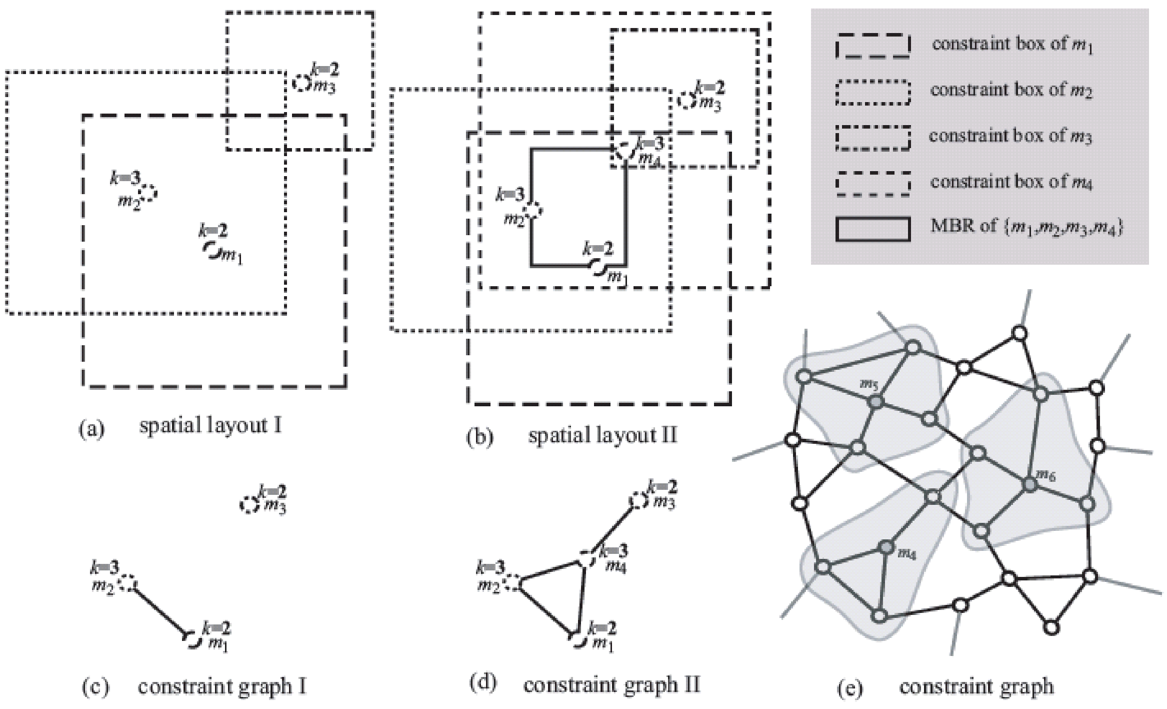
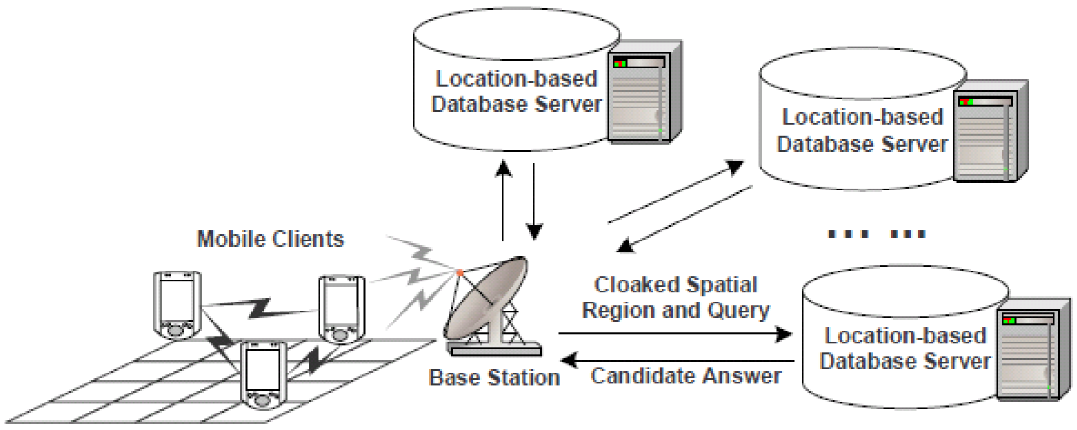


Figure 2.36.: The Clique-Cloak algorithm [GL05]



Location-based Database Server
Location-based Database Server
Location-based Database Server
Mobile Clients
Base Station
Cloaked Spatial Region and Query
Candidate Answer

Figure 2.37.: System architecture for P2P spatial cloaking [CML06]

clusters is called *reciprocity*, and it guarantees that attacks on several intersecting k-clusters of different queries cannot easily identify the target user.

Another approach to achieve reciprocity of k-clusters is presented by Ghinita et al. [GKS07]. The authors propose to use Hilbert space-filling curves for the indexing of service users and their locations. Figure 2.39 shows two examples of a Hilbert space-filling curves with different granularities of space distribution (4×4 cells and 8×8 cells). The advantage of the Hilbert space indexing is that physically close objects likely have numerically close indices. By utilizing such distance-aware spatial indexing, the anonymizer always selects the same fixed set of $k-1$ additional users for each query of a given user; therefore, it is impossible to decrease k by excluding some users after intersecting the clusters retrieved from multiple queries. Thus, so called **strong k-anonymity** is guaranteed by the reciprocity property.

Yet another approach to achieve reciprocity of k-clusters is proposed by Talukder and Ahamed [TA10]. The authors use *adaptive nearest neighborhood cloaking* to achieve this property. They describe how their approach resists two types of the multi-query attack: shrink region attack and region intersection attack. The aim of these attacks it to reduce the number k of cluster's users by issuing multiple queries with similar target location and then

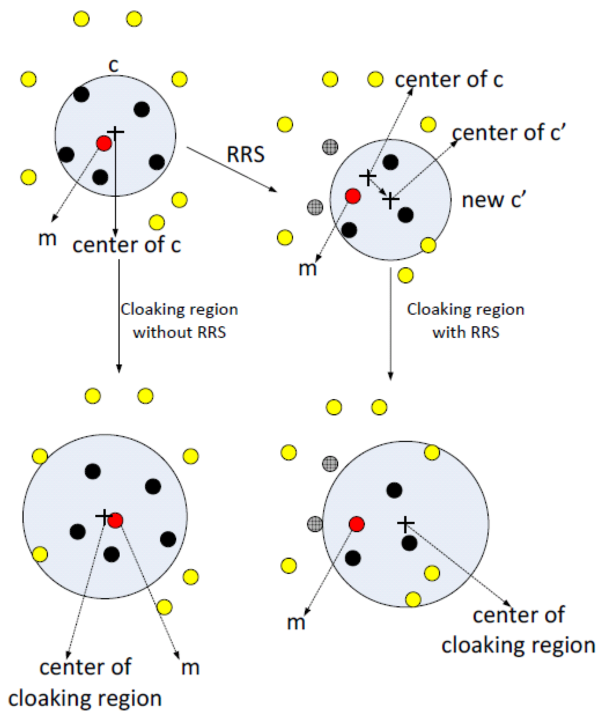

c
RRS
m
center of c
center of c
center of c'
new c'
m
Cloaking region without RRS
Cloaking region with RRS
center of cloaking region
m
m
center of cloaking region

Figure 2.38.: RRS algorithm for the number of clients in a mobile client's surrounding cell $K_m = 7$ [ZH09]

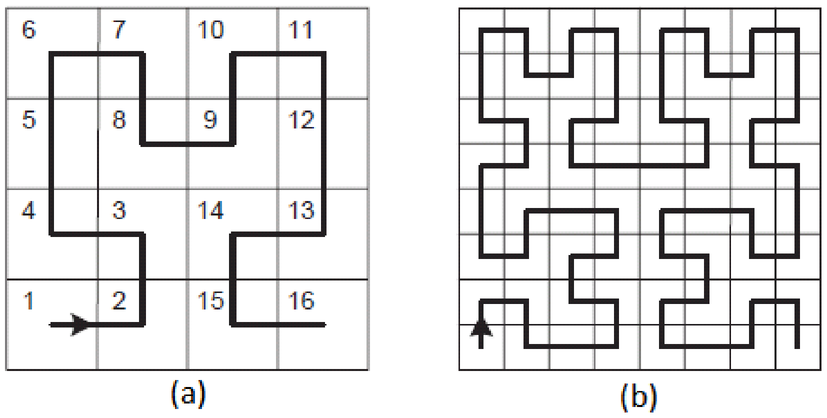


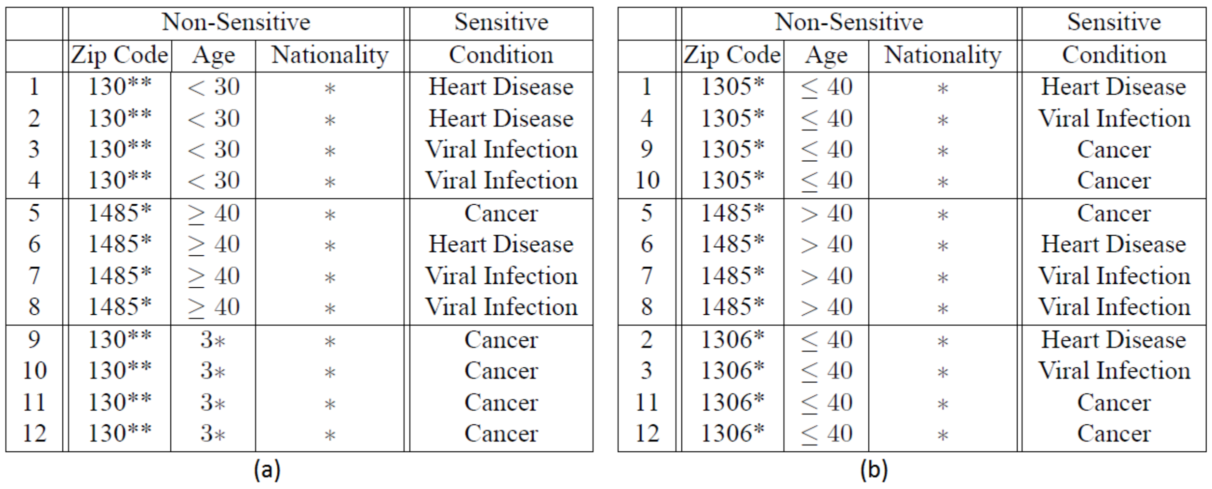
Figure 2.39.: Hilbert curve examples: (a) 4 × 4 cells; (b) 8× 8 cells [GKS07]

overlapping the received k-clusters.

There are also a number of approaches inspired by database privacy principles, which are also applicable in cases when an MO needs to send multiple attributes of itself to an LBS. Such approaches add **additional parameters** to the k-anonymity guarantee in order to improve it. Usually these ideas originate from the database privacy field, but they can also be applied in the field of location privacy if LBS users are represented by more attributes than just their identity and location.

For example, Machanavajjhala et al. introduced the l-*diversity* parameter to extend the k-anonymity guarantees [MKGV07]. Their approach preserves the diversity of the personal context for the given k-cluster. Thus, the target user position cannot be disclosed by analyzing non-spatial attributes of the MOs. To illustrate l-diversity, Figure 2.40 shows two tables containing 4-clusters. The information presented in the first table (Figure 2.40a) is less secure, since the user identity can be revealed by analyzing the sensitive attributes (here: condition of a patient) although the 4-anonymity of each cluster has been preserved. For example, in the case of the last 4-cluster, an attacker knows with 100% certainty that the target user has cancer. The second table (Figure 2.40b) is improved by providing the 3-diversity, so that the probability of a target user to have a certain disease is not higher than 33.3%. The additional security is achieved by grouping the entries in such a way that at least 3 different sensitive (condition) values are represented in each 4-cluster. This approach guarantees that 3 users within each cluster are indistinguishable, i.e., l-diverse (3-diverse).

Similarly, l-diverse k-anonymity was guaranteed by Bamba et al. [BLPW08] by using a sophisticated *dynamic cloaking* approach. The authors proposed various ways to find l-diverse k-clusters of users: top-down (starting with larger space cells and dividing them into smaller



(a)

	Non-Sensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	130**	< 30	$*$	Heart Disease
2	130**	< 30	$*$	Heart Disease
3	130**	< 30	$*$	Viral Infection
4	130**	< 30	$*$	Viral Infection
5	1485*	≥ 40	$*$	Cancer
6	1485*	≥ 40	$*$	Heart Disease
7	1485*	≥ 40	$*$	Viral Infection
8	1485*	≥ 40	$*$	Viral Infection
9	130**	$3*$	$*$	Cancer
10	130**	$3*$	$*$	Cancer
11	130**	$3*$	$*$	Cancer
12	130**	$3*$	$*$	Cancer

(b)

	Non-Sensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	1305*	≤ 40	$*$	Heart Disease
4	1305*	≤ 40	$*$	Viral Infection
9	1305*	≤ 40	$*$	Cancer
10	1305*	≤ 40	$*$	Cancer
5	1485*	> 40	$*$	Cancer
6	1485*	> 40	$*$	Heart Disease
7	1485*	> 40	$*$	Viral Infection
8	1485*	> 40	$*$	Viral Infection
2	1306*	≤ 40	$*$	Heart Disease
3	1306*	≤ 40	$*$	Viral Infection
11	1306*	≤ 40	$*$	Cancer
12	1306*	≤ 40	$*$	Cancer

Figure 2.40.: (a) 4-anonymous inpatient microdata; (b) 3-diverse inpatient microdata [MKGV07]

cells), bottom-up (starting with small space cells and merging them into larger cells) and hybrid spatial cloaking. The anonymization of spatio-temporal range queries is achieved by incorporating temporal cloaking.

Domingo-Ferrer et al. proposed a so-called *p-sensitivity* parameter to improve k-anonymity guarantees [SSDF08]. The idea is to prevent all mobile user profiles in a k-cluster from sharing a combination of confidential key attributes, which would otherwise disclose the confidential attributes of the user.

The next extension of k-anonymity and l-diversity parameters was proposed by Li et al. [LLV07]: t-*closeness* represents the distance between an attribute's distribution within the selected cluster of k users and the same attribute's distribution over the total set of users. This distance should not be smaller than a certain threshold.

Another database-driven approach was introduced by Wong et al. based on (α, k)-*anonymity* [WLFW06]. This approach is an extension of k-anonymity similar to l-diversity, where α denotes the maximum relative diversity of the given sensitive attribute in the set of tuples. Contrary to the number of multiple diverse attributes l in a given tuple (k-cluster), the α parameter is used to preserve the diversity of a given separate attribute.

Also, there are a number of approaches called *historical* k-anonymity, where the k-anonymity principle is applied to multiple position updates. Historical k-anonymity methods improve k-anonymity guarantees by taking into account the temporal component of the user's position information, i.e., to provide k-anonymity guarantees for moving objects [MBW+09]. Similarly as with achieving strong k-anonymity by clustering, the historical information of multiple users is divided into blocks with each block containing the positions of at least k users.

The problem of all location privacy approaches based on k-anonymity is that in order to select a k-set, a trusted anonymizer with a global view must be available in the system, i.e., an undesired TTP is required.

2.7.5. Spatial Obfuscation

Spatial obfuscation approaches preserve the user's location privacy by deliberately reducing the precision of position information sent by the user to an LBS. A classic *spatial obfuscation* approach is presented by Ardagna et al. [ACD+07], in which a mobile user sends circular areas instead of his or her exact positions to the location server. The obfuscation areas are generated in a secure way, which means that the target user position is distributed uniformly inside the obfuscation shape. Figure 2.41 illustrates different obfuscation techniques: (a) enlarging the radius, (b) shifting the center, and (c) reducing the radius. We apply these techniques in our position sharing approach, while providing additional properties such as graceful degradation and multiple privacy levels.

An important advantage of spatial obfuscation is that it requires no TTP, since the user himself or herself can calculate the obfuscation area (in contrast to k-anonymity and other techniques relying on a TTP). However, this advantage comes at a price, which means that the clients are not provided with the precise user position. This trade-off between privacy and precision was studied by Cheng et al. [CZBP06]. They introduced a probabilistic model of results of range queries, depending on the overlapping size of query area and the obfuscation shapes. Figure 2.42 illustrates the query score of ILRQ (Imprecise Location-based Range Query). Here, p denotes the probability that user S yields R_i as the query answer, while V is

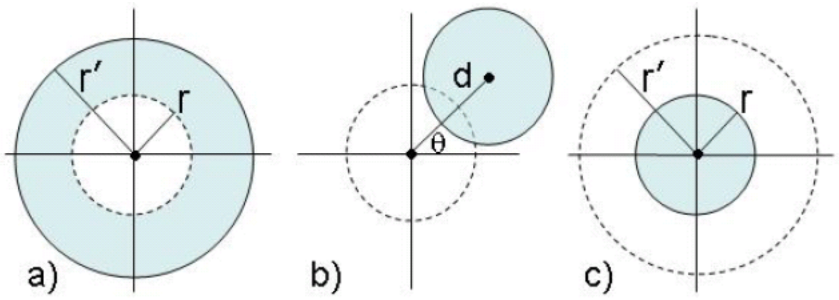


Figure 2.41.: Obfuscation by: (a) enlarging the radius; (b) shifting the center; (c) reducing the radius [ACD+07]

the precision of R_i with respect to the full set of answers R, where R gives the exact desired query answer.

Instead of geometric obfuscation shapes like circles, Duckham and Kulik have used *obfuscation graphs* to apply the concept of location obfuscation to road networks [DK05]. With the addition of fake vertexes, their obfuscation graphs can be applied if the road network is available. Figure 2.43 illustrates the addition of dummy vertex s to the obfuscated multi-source graph algorithm with a set of query locations $Q = q_1, \ldots, q_5$ (gray vertices) and a set of obfuscation locations $O = o_1, \ldots, o_6$ (black vertices).

An example of a location privacy approach based on geometric obfuscation is the *n-CD* approach of Li et al. [LSTL13]. According to this approach, a mobile user generates n so called concealed disks (CDs), such that their combination provides the "anonymity zone" around his or her position. The unpredictability of the resulting anonymity zone is preserved, since the overlapping and rotation of the CDs is done in a randomized fashion. The advantage of the *n-CD* approach is that, similarly to our position sharing approach, it does not require a trusted third party, which would store the precise location information of mobile users. The disadvantage, however, is that the *n-CD* approach does not provide multiple levels of privacy, i.e., precision levels depending on the user's trust in different location-aware applications. The addition of each further kth CD refines the obfuscation area, but there is no pre-defined value of this refinement, and, therefore, there are no guaranteed k precision levels. Another drawback of the *n-CD* approach is that the proposed privacy metric reflects only the area

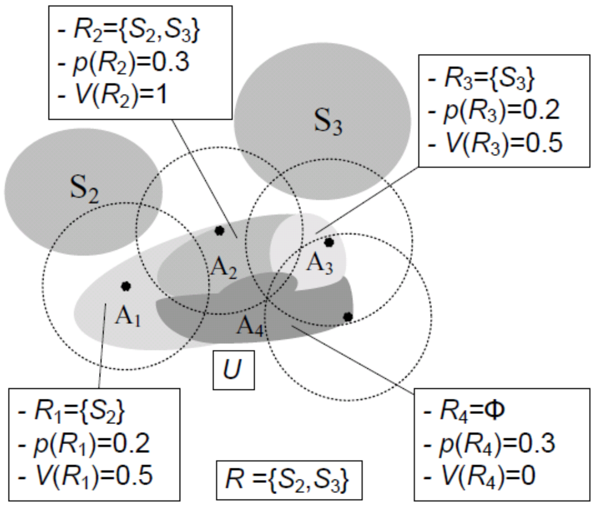


Figure 2.42.: Query score of ILRQ; p – probability that user S obtains R_i as the query answer; V – precision of R_i with respect to the full set of answers R [CZBP06]

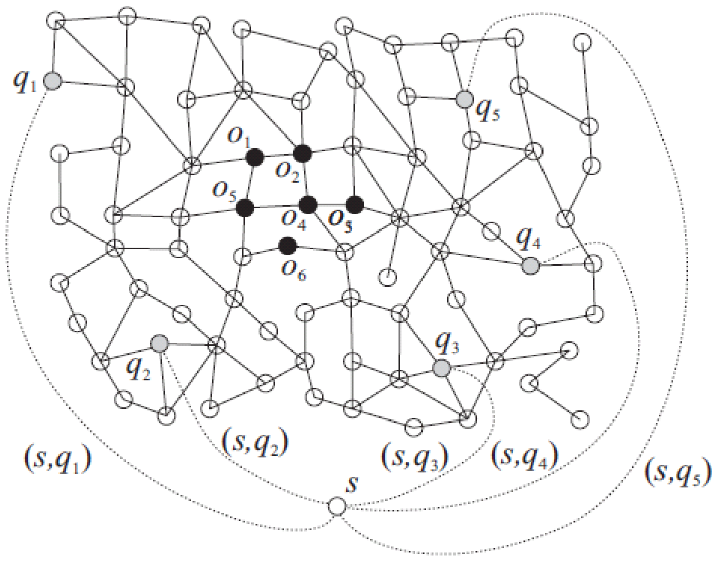


Figure 2.43.: Addition of a dummy vertex s to the obfuscated multisource graph, with a set of query locations $Q = q_1, \ldots, q_5$ (gray vertices) and a set of obfuscation locations $O = o_1, \ldots, o_6$ (black vertices) [DK05]

size of the anonymity zone, while the probability distribution of the precise user position over the anonymity zone is not considered [PSD15].

The problem of the obfuscation's accuracy was addressed by Perazzo et al. [DP12]. The authors proposed an approach based on spatial obfuscation, which guarantees uniform probability distribution of the true user's position over the obfuscated area while taking into consideration the inaccuracy parameters of the positioning system. Figure 2.44 shows examples of two different assumptions about the size of the measurement inaccuracy errors, which usually reduce the probability density near the borders of the obfuscation area.

Our position sharing approach is also based on spatial obfuscation, but additionally it provides graceful degradation of position precision depending on the number of missing position shares and therefore supports multiple obfuscation levels. Moreover, the mobile user is able to define the position precision levels that each location-based application is authorized to obtain, depending on its trustworthiness from the user's individual perspective.

2.7.6. Coordinate Transformation

A location privacy approach using *coordinate transformation* was proposed by Gutscher et al. [Gut06]. The mobile users perform some simple geometric operations (shifting, rotating) over their positions' coordinates before sending them to the location-based service. In order

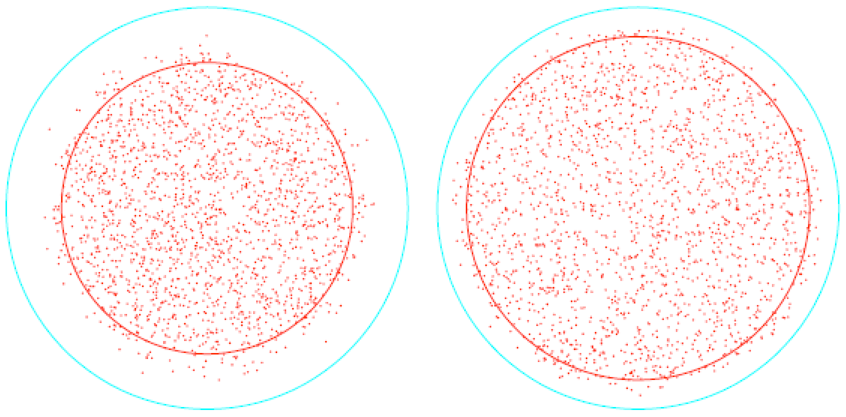


Figure 2.44.: Obfuscation circles with two different error measures [DP12]

to recover the original position, the transformation function needs to be distributed among the clients. Thus, it is not possible to compare the positions of different users obfuscated with different transformation functions, for instance, to perform range queries. Figure 2.45 shows how point $\vec{p}$ is represented in two different coordinate systems k_A and k_B, with corresponding coordinates $\vec{c_{p,A}}$ and $\vec{c_{p,B}}$, where $\vec{d_{B,A}}$ is the transformation vector.

Our position sharing approach is also based on coordinate transformations; however, to provide the LBS with different levels of granularity and thus enable multiple privacy levels, we combine coordinate transformation and spatial obfuscation.

Another method based on the concept of *spatial transformation* called SpaceTwist was presented by Yiu et al. [YJHL08]. The goal of this approach is to support private kNN-queries sent by the mobile user, which pretend that he or she has issued the query from a fake location. The user sends multiple queries to the server containing his or her fake location

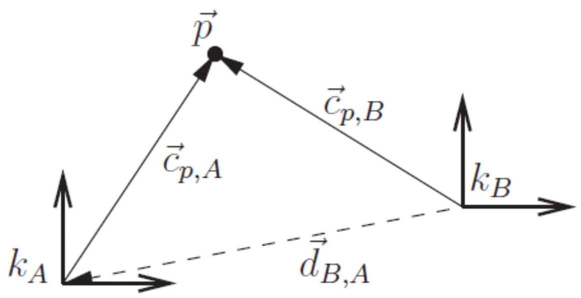


Figure 2.45.: Representation of point $\vec{p}$ in two coordinate systems k_A and k_B, with corresponding coordinates $\vec{c_{p,A}}$ and $\vec{c_{p,B}}$; $\vec{d_{B,A}}$ is the transformation vector [Gut06]

instead of the actual one, and then filters the received data in order to get the needed results for his or her true position. Thus, location privacy is achieved by degrading the query accuracy. Figure 2.46 shows the so called demand space and supply space. The supply space expands by iteratively querying and adding the nearest neighbors of the fake anchor location. The demand space is only known to the mobile user, and therefore he or she can perform the query without letting the LBS know his or her true location – until the number of obtained neighbors expands the supply space so much that it completely covers the demand space. Thus, in order to finally get the precise position, excessive querying is required, which leads to high communication cost.

2.7.7. Trajectory Privacy

Next, we will describe approaches that consider the temporal dimension in addition to the user location information and therefore aim to preserve trajectory privacy. One such approach based on k-anonymity was already presented above (*historical* k-anonymity [MBW+09]). As an alternative method, *spatio-temporal obfuscation* can be applied in order to protect movement trajectories of users [GG03]. In addition to decreasing the precision of positions, they also decrease the precision of the temporal information associated with positions until a specified k-anonymity criterion is achieved.

A similar idea was also presented by Ghinita et al. in their *spatio-temporal cloaking* approach [GDSB09]. To improve the security of spatial cloaking, the authors take into consideration attacks based on background map knowledge represented by a set of privacy-sensitive features as well as attacks based on the known maximum speed of objects (also called maximum movement boundary attack). As an example, Figure 2.47 shows the attack model

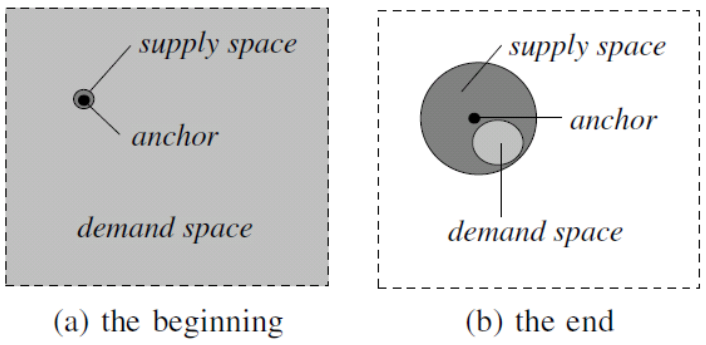


Figure 2.46.: Demand space and supply space before and after the refinement [YJHL08]

with background knowledge about the surrounding sensitive locations (e.g., hospital and night club), so that the area where the target user can be possibly located is reduced by intersecting the sensitive areas with the maximal user movement possible.

The principle of temporal cloaking by delaying the location update described by Ghinita et al. [GDSB09] is illustrated in Figure 2.48a; for comparison, the spatial cloaking principle in shown in Figure 2.48b. The delay for temporal cloaking is calculated as the distance between two locations A and B divided by a sufficiently slow MO's speed, so that the possible movement area is large enough to satisfy privacy requirements.

In addition to these approaches, a number of similar approaches developed to protect spatio-temporal location privacy exist, including *trajectory clustering* [LHW07], *trajectory transformation* [TM08], *uncertainty-aware path cloaking* [HGXA07], *virtual trip lines* [HGH+08], *fake paths* [LLLZ09], etc.

An approach that utilizes the concept of k-anonymity for securing a *complete* published user trajectory was presented by Abul et al. [ABN08]. The authors apply an enhancement of k-anonymity for spatial-temporal cloaking called (k, δ)-anonymity. The idea is that before publishing, the trajectories of at least k users are co-located into a "space tunnel" of radius $\delta/2$, which defines the desired uncertainty (privacy) level. Figure 2.49 shows a $(2, \delta)$-anonymity set formed by two co-localized trajectories, their respective uncertainty levels, and the central cylindrical volume of radius $\delta/2$, which contains both trajectories [ABN08].

Note that we do not consider securing the whole trajectory as one of our goals, but future work could extend our approach in this way.

2.7.8. Map-aware Approaches

A significant problem with many spatial obfuscation-based approaches is that the desired size of the obfuscation areas can be reduced if an adversary applies background knowledge,

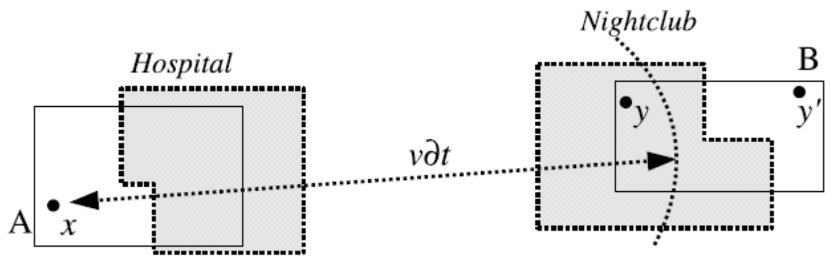


Figure 2.47.: Attack model with background map knowledge [GDSB09]

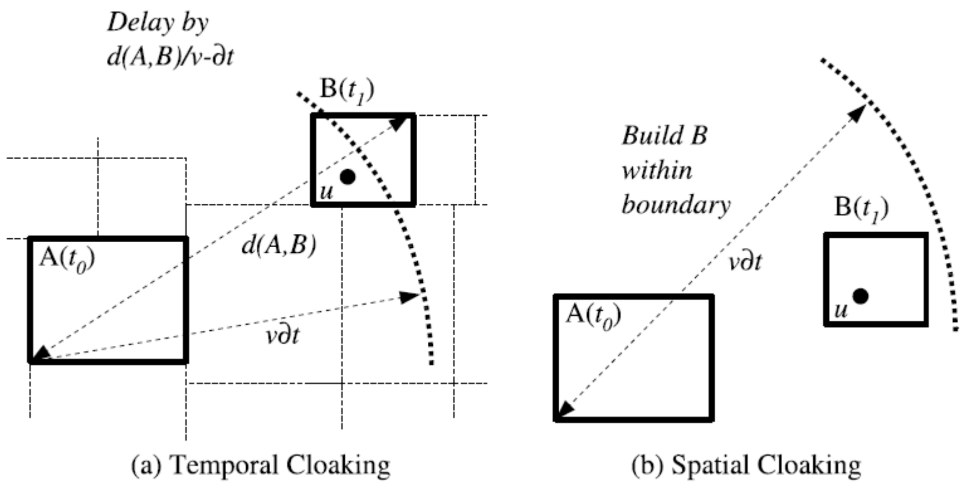


Figure 2.48.: Spatio-temporal cloaking: (a) temporal cloaking; (b) spatial cloaking [GDSB09]

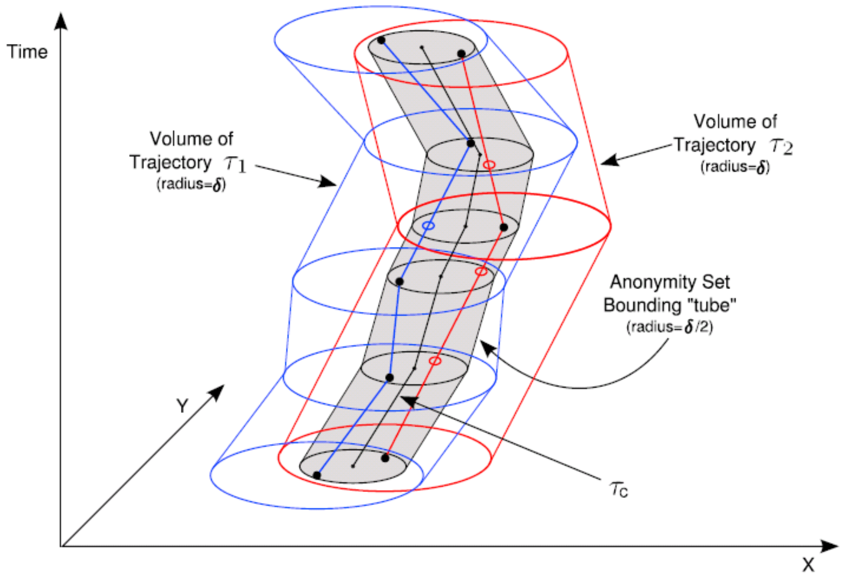


Figure 2.49.: A $(2, \delta)$-anonymity set formed by two co-localized trajectories, their respective uncertainty levels, and the central cylindrical volume of radius $\delta/2$, which contains both trajectories [ABN08]

in particular the map knowledge, in order to reduce the effective size of the obfuscation area.

To resist such map matching attacks, Ardagna et al. proposed a *landscape-aware* obfuscation approach [ACG09]. The work of Ardagna et al. provides theoretical background for map-awareness. Moreover, it presents the idea of adjusting the radius of the obfuscation disk in order to preserve the user's privacy from being affected by landscape knowledge.

This approach of Ardagna et al. is based on a probability distribution function that defines the probability of user being located in certain areas of a map. The obfuscation area is selected considering the probability of user being located in areas of the obfuscation shape. Figure 2.50 shows an example of a one-dimensional landscape prior to probability distribution $\lambda(t)$: the user is b times as likely to be localized between 0 and $2d$ than elsewhere (here, R denotes the obfuscation circle radius).

In our work, we adapt a similar principle to our position sharing approach with multiple non-trusted servers and provide flexible management of privacy levels.

Another advanced obfuscation approach by Damiani et al. [DBS10] called PROBE applies the map-awareness principle to protect semantic locations, i.e., to ensure that a user cannot be revealed by being located in certain sensitive locations. This obfuscation approach expands the obfuscation area adaptively, so that the probability of the user to be in a certain semantic location is below the given threshold. Figure 2.51 shows an obfuscated map generated by the location privacy algorithm for two hospitals taken as an example of critical semantic locations. The space is represented in a discrete fashion through polygonal regions, which are gradually added together in order to cover the required area. The resulting obfuscation region can have any shape, but the approach lacks flexibility due to the enforced cell-based

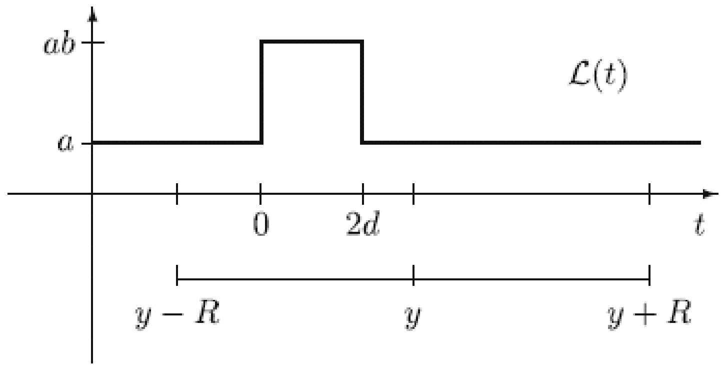


Figure 2.50.: Example of $1D$ landscape prior probability distribution $\lambda(t)$: the user is b times more likely to be localized between 0 and $2d$ than elsewhere (R – obfuscation circle radius) [ACG09]

space representation.

2.7.9. Secret Sharing and Position Sharing

In order to address the problem of non-trusted location server infrastructures, Marias et al. [MDKG05] proposed an approach for the distributed management of position information based on the concept of *secret sharing* [Sha79]. The basic idea of this approach is to divide position information into shares, which are then distributed onto a set of non-trusted location servers. In order to recover the precise positions, the LBS client (application) needs to retrieve the complete set of shares from multiple servers. The advantage of this approach is that a compromised server cannot reveal any position information since it does not have all the necessary shares. However, one important disadvantage of this approach is that location servers cannot perform any computations on the shares, for instance, in order to perform range queries. Thus, even if only a single location server is not available, the user's position is not accessible to location-based applications.

An approach based on secret sharing was proposed by Wernke et al. [WDR12] (*PShare*). Unlike our position sharing approach, which generates shares based on geometric transformations, the authors utilize the concept of *multi-secret sharing* for share generation [CC05]. PShare also supports symbolic location information in addition to geometric information. The idea of PShare is that the mobile user generates the shares based on the polynomial representation of his or her precise position π and the k positions of degrading precision, while π is the combination of k polynomials according to the multi-secret scheme [CC05] using the Chinese Remainder Theorem. Clients can partially reconstruct the allowed precision

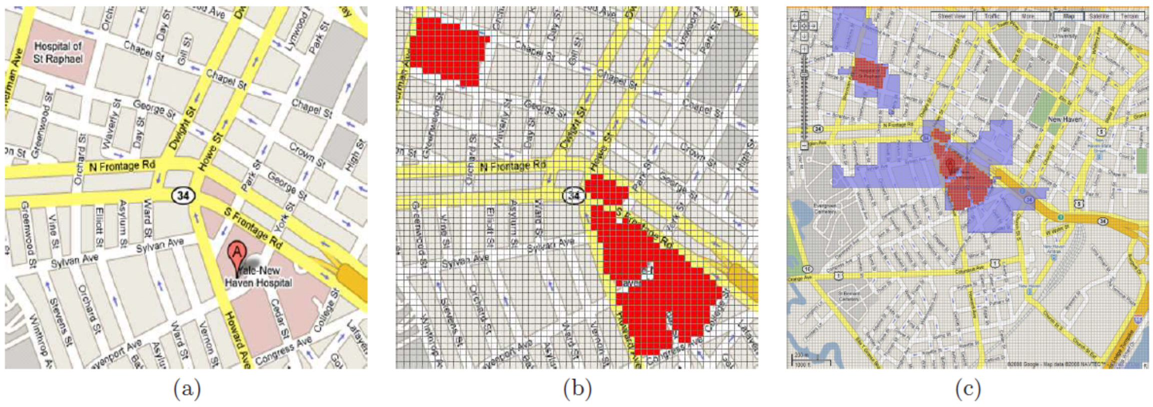


Figure 2.51.: Obfuscated map generated by the PROBE algorithm for two hospitals [DBS10]

by obtaining the k points and performing their Lagrange interpolation (see Figure 2.52). An important difference between PShare and our approach is that PShare uses discretized square-based space representation and such corresponding indexing that the precision change between the nearest privacy levels cannot be smaller that factor 4 (cf. Figure 2.52a). This significantly limits the flexibility of privacy levels management by the mobile user. Also, all clients are required to know the sophisticated cryptographic functions of PShare, which does not satisfy the requirements of interaction between large numbers of independent users within geosocial networks.

2.7.10. Classification of Location Privacy Approaches

There are a number of works that survey the state of the art of techniques of protecting location privacy and try to classify them [Kru09, SDFMB08, WL09, CM11]. For example, a survey of major existing location privacy approaches classified according to general techniques applied was presented in [Kru09], while Solanas et al. classified approaches based on their reliance on a trusted third party (TTP) [SDFMB08].

A basic classification according to schemes of communication applied between the mobile

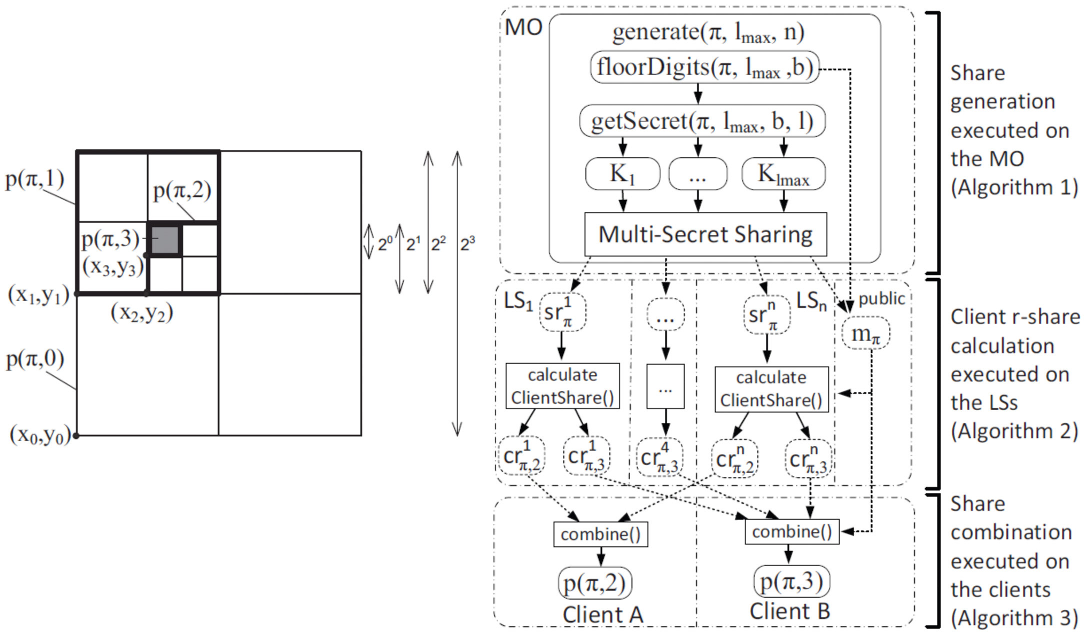


Figure 2.52.: (a) Geometric area of obfuscated MO position $p(\pi, l)$ for granularity of precision levels $b = 2$ and precision level $l_{max} = 3$; (b) PShare-GLM (geometric location model) process overview [WDR12]

user and the LBS was presented by Solanas et al. [SDFMB08]. The three major communication schemes are illustrated in Figure 2.53. According to the simplest scheme (Figure 2.53a), mobile user and LBS communicate directly with each other under the assumption that the LBS provider is fully trusted. In the second scheme (Figure 2.53b), the LBS provider is not trusted; therefore, there is another trusted entity (e.g., an anonymizer) called trusted third party (TTP) that operates between the LBS and the mobile user. TTP is responsible for making the users' locations secure. In the third scheme (Figure 2.53c), a set of mobile users collaborate to help each other to communicate with an untrusted LBS in a secure way.

Another classification of location privacy approaches proposed by Mokbel [Mok07] included an overview of location privacy challenges, techniques and attacks. Many possible attacks were considered, together with approaches for resisting such attacks. The author describes *location privacy concepts* such as location perturbation, spatial cloaking and k-anonymity. Additionally, he considers various *system architectures* used by location privacy approaches: (a) non-cooperative architectures (having independent system actors); (b) centralized architecture with a trusted third party; (c) peer-to-peer (P2P) architecture.

Location privacy approaches can be classified according to the possible attacks on the user's location privacy, which they can resist. Figure 2.54 shows the generalized representation of existing LBS privacy threats presented by Bettini et al. [BMW+09]. The authors proposed the following classification of privacy threats: (a) attacks exploiting quasi-identifiers in

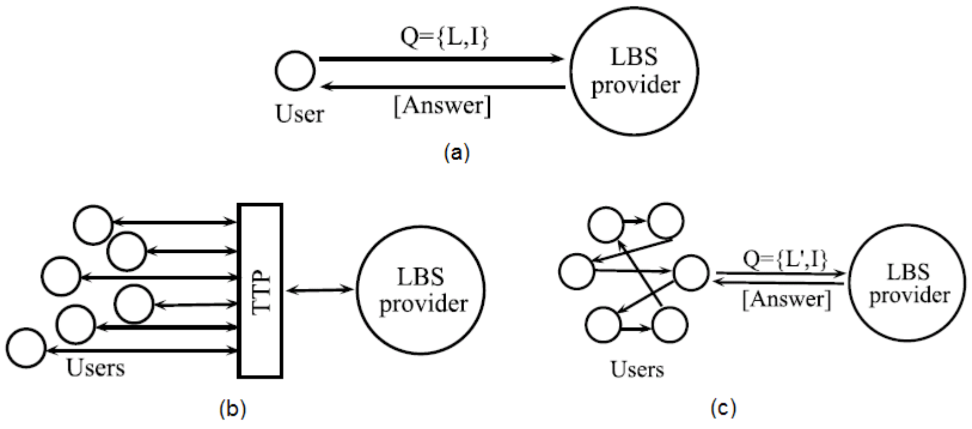


Figure 2.53.: (a) Scheme of direct communication between the mobile object and the LBS; (b) communication between the mobile object and the LBS through an intermediate trusted third party (TTP); (c) communication scheme between a set of collaborative users and an untrusted LBS. L is precise position; L' is obfuscated position; query is denoted as Q [SDFMB08]

requests; (b) snapshot vs. historical attacks; (c) single- vs. multiple-issuer attacks; (d) attacks exploiting knowledge of the defense. The authors also analyzed and categorized the existing approaches with respect to their ability to resist the attacks described.

Privacy threats can be also classified into the following categories [Kru09]: (a) analysis of movement patterns; (b) context inference; (c) simulated privacy attacks (when attacker sends queries by pretending to be the target user and derives location information from query answers).

By extending the concepts of the works presented above, we have proposed our own classification of existing location privacy attacks and techniques [WSDR14]. We grouped the location privacy techniques and approaches according to two major parameters: the *privacy goals*, i.e., the information secured by each approach, and the assumed *attacker knowledge*, i.e., methods or additional information, which an attacker can use in order to undermine the user's location privacy. As privacy goals, we considered securing user's identity (ID), position and timestamp associated with the position; these goals can be selected separately or in any combination.

To represent attacker knowledge, we classified it by distinguishing (a) whether single or multiple user positions are available, including various attacks based on multiple positions, and (b) whether the context (i.e., additional personal or environmental information) is known (see Figure 2.55). Thus, we considered the availability of additional knowledge to

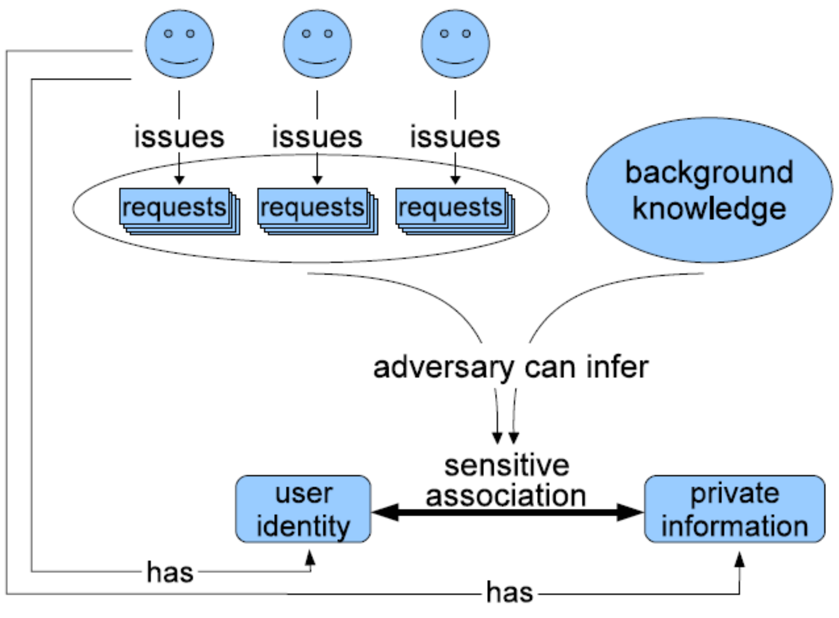


Figure 2.54.: General privacy threats in LBS [BMW+09]

an attacker such as any kind of user context and the user's movement history.

Our classification of attacks based on the given knowledge is shown in Figure 2.56. We distinguish between single position attacks, context linking attacks, multiple position attacks, attacks combining context linking and multiple position attacks, and attacks based on compromising a TTP component. For more details regarding the location privacy attacks, we refer to the original publication [WSDR14].

After analyzing the major location privacy techniques and approaches, we proposed their classification based on the analysis which protection goals they fulfill for different attacks (see Figure 2.57). Each protection goal is defined by whether the attribute identity, position and time should be protected (✓) or not (✗). The stated techniques provide the corresponding protection goal assuming a certain attacker knowledge. If the technique can resist an attacker with a certain attack, this is denoted by a ✓ in the main part of the table, whereas an empty cell denotes that the attack can be successful against the stated technique. The gray cells indicate possible future research directions not covered by the stated techniques. For each approach, we marked whether it needs a trusted third party (TTP) or not. We arranged the different approaches based on their primary protection goal. Approaches marked by "*" provide the protection goal as a sub-goal in addition to their primary protection goal.

We can see that most approaches protecting the user's identity against different attacks are based on k-anonymity. However, they usually require a TTP (i.e., an anonymizer). If the user wants to preserve location privacy without protecting his identity, the most popular technique to apply is spatial obfuscation. Its major drawback is that clients can only retrieve

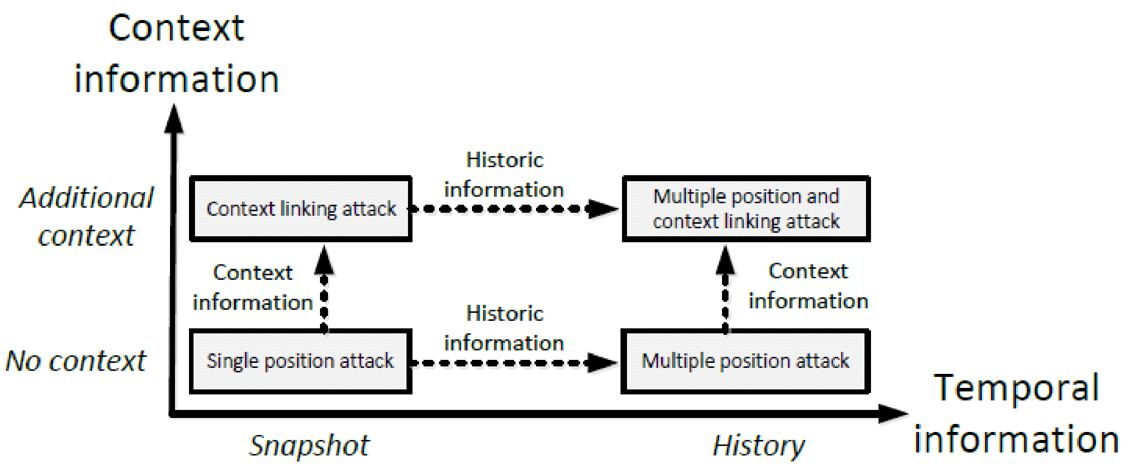


Figure 2.55.: Classification of attacker knowledge [WSDR14]

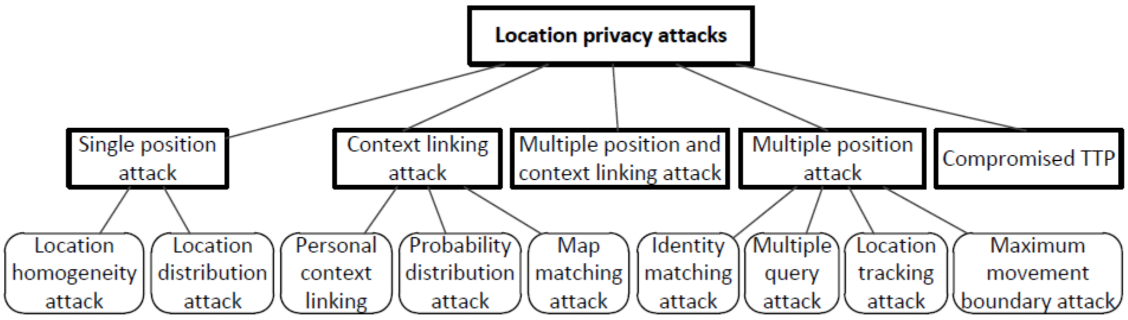


Figure 2.56.: Classification of location privacy attacks [WSDR14]

an obfuscation area instead of a precise user position. To overcome this problem, we proposed the position sharing approach, where the user can flexibly manage the precision provided to each application.

The most challenging goal is represented by the case when user trajectory and ID should be secured. We can also see that the most challenging attacks are those that link the user's location information with the user's personal context, and those that reduce privacy by matching the user's revealed location information with map knowledge.

In the given classification (Figure 2.57), we also show how our position sharing approach [DSR11, SDR12] relates to other approaches. The position sharing approach takes map knowledge into account in order to prevent de-obfuscation by map-based attacks; also, it resists "maximal movement boundary" attacks (as we will describe later in Chapter 4).

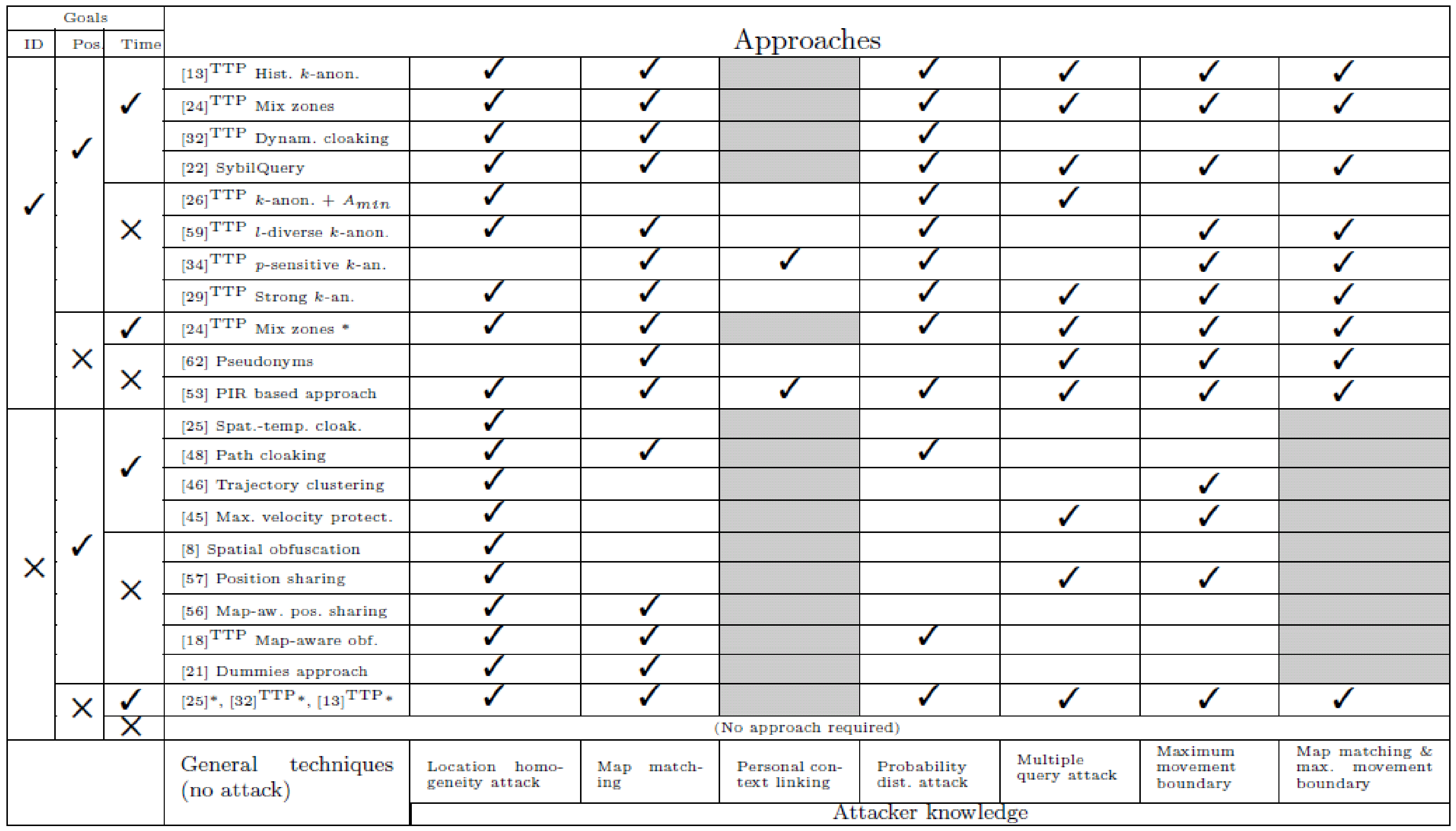
2.7.11. Related Work: Summary

As we have shown above, there is a great variety of methods for preserving user's location privacy, but usually they have significant limitations and drawbacks. For example, the *k-anonymity*-based location privacy approaches rely on a trusted anonymizer with a global view, i.e., an undesired TTP is required.

If *cryptography*-based approaches are applied to encrypt user positions stored on servers, the problem is that server-side query processing of advanced queries like range queries over the encrypted data is impossible, or require a very high cost.

The general problem with the *position dummies* approach is that dummy positions can be easily distinguished from the real positions; moreover, the transmissions of fake positions and operations on them cause excessive costs.

The approaches based on *mix zones* lack flexibility with regard to the user position's



Goals			Approaches							
ID	Pos	Time	General techniques (no attack)	Location homogeneity attack	Map matching	Personal context linking	Probability dist. attack	Multiple query attack	Maximum movement boundary	Map matching & max. movement boundary
				Attacker knowledge						
✓	✓	✓	[13]$^{\text{TTP}}$ Hist. k-anon.	✓	✓	▨	✓	✓	✓	✓
✓	✓	✓	[24]$^{\text{TTP}}$ Mix zones	✓	✓	▨	✓	✓	✓	✓
✓	✓	✓	[32]$^{\text{TTP}}$ Dynam. cloaking	✓	✓	▨	✓			
✓	✓	✓	[22] SybilQuery	✓	✓	▨	✓	✓	✓	✓
✓	✓	✗	[26]$^{\text{TTP}}$ k-anon. + A_{min}	✓			✓	✓		
✓	✓	✗	[59]$^{\text{TTP}}$ l-diverse k-anon.	✓	✓		✓		✓	✓
✓	✓	✗	[34]$^{\text{TTP}}$ p-sensitive k-an.		✓	✓	✓		✓	✓
✓	✓	✗	[29]$^{\text{TTP}}$ Strong k-an.	✓	✓		✓	✓	✓	✓
✓	✗	✓	[24]$^{\text{TTP}}$ Mix zones *	✓	✓	▨	✓	✓	✓	✓
✓	✗	✗	[62] Pseudonyms		✓			✓	✓	✓
✓	✗	✗	[53] PIR based approach	✓	✓	✓	✓	✓	✓	✓
✗	✓	✓	[25] Spat.-temp. cloak.	✓		▨				▨
✗	✓	✓	[48] Path cloaking	✓	✓	▨	✓			▨
✗	✓	✓	[46] Trajectory clustering	✓		▨			✓	▨
✗	✓	✓	[45] Max. velocity protect.	✓		▨		✓	✓	▨
✗	✓	✗	[8] Spatial obfuscation	✓		▨				▨
✗	✓	✗	[57] Position sharing	✓		▨		✓	✓	▨
✗	✓	✗	[56] Map-aw. pos. sharing	✓	✓	▨				▨
✗	✓	✗	[18]$^{\text{TTP}}$ Map-aware obf.	✓	✓	▨	✓			▨
✗	✓	✗	[21] Dummies approach	✓	✓	▨				▨
✗	✗	✓	[25]*, [32]$^{\text{TTP}*}$, [13]$^{\text{TTP}*}$	✓	✓	▨	✓	✓	✓	✓
✗	✗	✗	(No approach required)							

Figure 2.57.: Classification of location privacy techniques according to location privacy goals and attacker knowledge [WSDR14]

precision, because they need a pre-defined location-based or user-based division of space into fixed zones, and they do not allow for different levels of privacy in different zones.

An important advantage of *spatial obfuscation* is that it requires no TTP, since the user himself or herself can calculate the obfuscation area (in contrast to k-anonymity and other techniques relying on a TTP). However, this advantage comes at a price, which means that the clients are not provided with the precise user position. Our position sharing approach is also based on spatial obfuscation, but additionally it supports multiple obfuscation levels.

If the approach of *coordinate transformation* is applied to secure the user's position, the limitation is that it is not possible to compare the positions of different users obfuscated with different transformation functions, for instance, to perform range queries. Our position sharing approach is also based on coordinate transformations; however, to provide the LBS with different levels of granularity and thus enable multiple privacy levels, we combine coordinate transformation and spatial obfuscation.

The drawback of the original *secret sharing* approach is that even if only a single part of the secret is not available, the user's position is not accessible to location-based applications. The PShare approach was proposed to extend the idea of secret sharing and provide the graceful degradation property, i.e., the precision of positions revealed to a potential attacker can be incomplete, yet it will increase with the number of secret shares [WDR12]. The main difference to our approach is that PShare uses discretized square-based space representation and such corresponding indexing that the precision change between the nearest privacy levels cannot be smaller that factor 4 (cf. Figure 2.52a). This significantly limits the flexibility of privacy levels management by the mobile user. Also, all clients are required to know the sophisticated cryptographic functions of PShare, which does not satisfy the requirements of interaction between large numbers of independent users within geosocial networks.

In order to overcome the limitations described above, we proposed the concept of *position sharing* [DSR11, SDR12] for secure management of private position information stored on non-trusted location servers. Our method is that the mobile user splits the position information into so called position shares, where each share defines his or her position with strictly limited precision. These shares are distributed onto a set of non-trusted location servers, so that each server has only a position of limited precision, but which, at the same time, can be used to perform calculations. Through share fusion algorithms using simple *geometric operations*, multiple shares can be combined into positions of higher precision. Location-based applications can be provided with positions of different precision levels depending on the number of shares distributed to them. Within our approach, we also take into consideration the problems of map-awareness, share placement and position updates.

2.8. Conclusion

In this chapter, we have presented our basic position sharing approach and our classification of the existing location privacy approaches based on our previous publications [DWSR10, DSR11, SDR12, WSDR14]. The key idea of our position sharing approach is that the mobile user's position information is distributed among multiple location servers of different service providers in the form of separate data pieces, which we call position shares. The main advantages of this approach are that a trusted third party is avoided and that we are able to flexibly manage the revealed position's precision levels by defining the number of shares allowed to be obtained by each application.

We introduced four modifications to our approach. They include: OSPS-ASO with "a-posteriori" share generation, OSPS-ASO with the "a-priori" version of share generation, OSPS-FSO with fixed share order and the map-aware algorithm version called CSPS.

The difference between "a-priori" and "a-posteriori" algorithms lies in the predictability of the share set. They provide different dependencies of probabilistic guarantees corresponding on precision levels of the MO's position for each number of known shares k. OSPS-FSO lowers the peaks of probability distribution of both "a-priori" and "a-posteriori" versions of the OSPS-ASO. However, OSPS-FSO does so by introducing an additional requirement of fixing the order in which shares can be obtained and fused.

CSPS is basically an extension of OSPS-FSO, which provides map-awareness. This is done through adjustment of the obfuscation area, which is found through the intersection of the obfuscation circles and the map regions where the user can be located. Both OSPS-FSO and CSPS provide close-to-uniform distribution of probability of the MO's position inside the current obfuscation area, while CSPS provides this type of distribution if the map knowledge in form of a binary map representation is available. They both require a pre-defined order for obtaining and fusing the position shares.

All four of the presented approaches have the following limitations, which will be solved in the next chapters:

- In order to take into account different trustworthiness levels of the providers, it is necessary to optimize the share placement depending on the corresponding parameters of the providers.

- Since the current approaches consider only snapshot user positions, a mobility-aware share generation is needed, which includes efficient updating of shares, i.e., updating of the smallest required number of shares.

Chapter 3

Optimization of Share Placement

Up until this point, our basic position sharing approach (see Chapter 2) assumes that all LSs are equally trusted. However, this is not always a valid assumption. The probability of malicious behavior by LSs (i.e., their trustworthiness) can vary and can be estimated by using, for example, a feedback system providing the diverse trustworthiness levels of different LSs. In this chapter, we will extend our system model and present a share placement approach that takes LS trustworthiness into account in order to improve the user's location privacy. The basics of this approach were originally published in the diploma thesis of Björn Schembera [Sch11]. The author developed the main principles of share placement optimization, refined the optimization goals and algorithms, and supervised the diploma thesis as a whole.

We define the trustworthiness of LSs by using a probabilistic trust model. The trust value represents the probability that an LS behaves correctly, i.e., does not misuse the user's private position information, collaborate with other LSs, or become compromised by an external attacker.

The main goal is to optimize the placement of shares to LSs so as to decrease the risk of disclosure of large amounts of private location information. An LS with a higher trust level (i.e., a lower risk level) can obtain more position information (i.e., more shares, or "larger" shares) than a less trustworthy LS. This improves the user's location privacy measured according to our privacy metrics as the probability that a certain precision will be disclosed. Our privacy metrics guarantee that the end risk is balanced among LSs and that the probability of each precision level's disclosure does not exceed the pre-defined threshold.

Note that during optimization of share placement, we do not need to consider the actual content of a position share. In other words, our placement approach is suitable for both shares which contain geometrical information (such as in our main position sharing approach described in Chapter 2) and shares which contain numerical information (such as in the approach of Wernke et al. [WDR12]).

We distinguish a number of scenarios regarding the heterogeneity of shares and providers' trustworthiness. We analyze the privacy guarantees and computational complexity of the generalized placement algorithm, as well as special cases of share placement.

3.1. Problem Statement

In this section, we introduce our system model that was extended so as to address the trustworthiness of LSs. Then we describe our privacy metrics for optimizing the placement of position shares and formulate the problem statement.

3.1.1. Extended System Model

In order to extend our system to make it able to adapt according to the LSs trustworthiness, we need to extend our system model as originally presented in Figure 2.1 (Section 2.1). The new version of the system model is shown in Figure 3.1. In addition to the three main components (*mobile object (MO)*, *location server (LS)*, *location-based applications (LBA)*), it also contains a *trust database*.

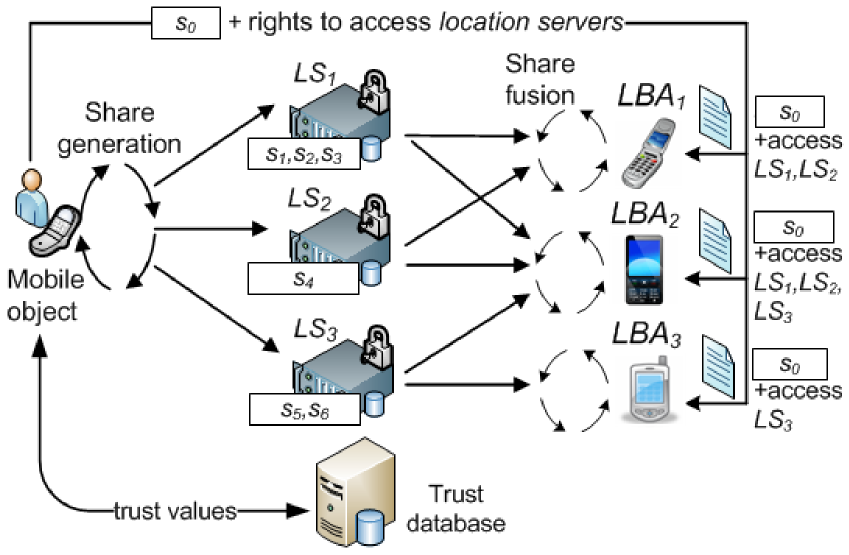


Figure 3.1.: Extended system model: now including a trust database

We assume that each LS is in principle non-trusted, and that it can be compromised with a known probability p_i. Therefore, we introduce into our system model the *trust database*, which maps LS_i to a risk value $p_i \in [0;1]$, providing the probabilities p_i that LS_i can be compromised. The risk value p_i represents the probability that LS_i will behave maliciously, i.e., misuse the user's private position information, or be compromised by an external attacker. Different LSs might have different risks depending, for instance, on the reputation of their provider. Moreover, different users might have personal trust in the same LS (and/or its provider). Thus, there is a personally or commonly determined trust value assigned to each available LS.

Note that in Figure 3.1 we depict the trust database as a single centralized entity. However, our assumptions and our placement approach are also valid in case if each MO has his or her own trust database locally.

The trust database answers the user's queries with the risk values p_i for the given LS. These values can be obtained, for example, by analyzing the feedback of other users through a reputation system [GHS08, Gut09]. The design of the trust database and the concepts for calculating p_i are beyond the scope of this work. Here, we rely on the *generic probabilistic trust model* and trust management concepts developed in [KBR05]. This model is generic in the sense that allows mapping of various representations of trust values to the probabilistic interval $[0;1]$. Based on the obtained risk values, the user can determine the number and set of LSs needed to satisfy his or her security requirements, as we will explain in the next sections.

Since each LS has an individual trust value, the user's position's privacy highly depends on the number of selected LSs and the placement of shares to different LSs. Previously, we assumed equal share placement, i.e., each LS stored shares of the same precision increase Δ_i^{ϕ}. Now, we want to make sure that an LS with a higher trust level can store more precise position information than an LS which has a higher risk of being compromised. Thus, each refinement share s_i increases the position precision by an individual pre-defined value Δ_i^{ϕ}.

Regarding the optimization of share placement, the exact content of a position share is not important: it can contain any cryptographic, geometric, or numerical information that increases the precision of the user's position. In the basic position sharing approach presented in Chapter 2, we introduced position sharing concept based on geometric transformations. At the same time, an alternative position sharing concept based on numerical operations on the user's position information was described by Wernke et al. [WDR12]. Our share placement optimization, proposed later in this chapter, is applicable to both the open space position sharing approaches and the map-aware approaches described in the previous chapter, as well as to the approach of Wernke et al. [WDR12].

3.1.2. Privacy Metric

To evaluate the user's privacy, we must determine which position precision of user position can an attacker derive from shares stored by k compromised LSs, and what is the probability of compromising these k LSs by the attacker. Assuming k compromised LSs, the shares must be placed such that an attacker – e.g., a malicious location service provider or LBA – cannot derive information of higher precision than required by user, with higher probability than required by user, in order not to violate the user's privacy requirements.

The formal definition of the probabilistic privacy metric defined in the previous chapter needs to be modified. Previously, the probability was dependent on the share generation algorithm's pdf over the obfuscation area corresponding to the current precision level k obtained by LBAs in an ordinary fashion. Now, the precision of a position ϕ_k defined as radius r_k of a circular obfuscation area corresponds to a level k that is obtained by an attacker and depends on the probabilities p_i of the correspondingly compromised LS_i.

Thus, the following distribution defines the probability $P_{\text{k,attack}}$ that an attacker can obtain a position $\pi_{\text{k,attack}}$ of a certain precision $\phi_{\text{k,attack}} = \text{prec}(\pi_{\text{k,attack}})$ depending on the number k of compromised LSs (consequently, depending also on the number and content of shares placed onto these k LSs):

$$P_{\text{k,attack}}(\phi_{\text{k,attack}}) = Pr[\phi_{\text{k,attack}} \leq \phi_k] \tag{3.1}$$

As before, this metric defines the acceptable probabilistic guarantees of privacy levels represented as a set of probability thresholds $P_k(\phi_k)$ corresponding to various precision levels ϕ_k. For example, an MO user can specify that an attacker should not be able to (compromise LSs and) obtain a position of precision $\phi_1 \leq 1$ km with probability $P_{1,\text{attack}} > 20\%$, and $\phi_2 \leq 2$ km with $P_{2,\text{attack}} > 10\%$, etc.

3.1.3. Problem Statement

The problem of share placement among the available LSs can be defined as a constrained optimization problem. The *constraint* is that an attacker cannot derive a position $\pi_{\text{k,attack}} =$

fuse(s_0, S_k) of precision prec($\pi_{\text{k,attack}}$) $> \phi_k$ with a probability $P_{\text{k,attack}}(\phi_{\text{k,attack}})$ higher than $P_k(\phi_k)$, where S_k denotes the set of compromised refinement shares. That is, the user defines probabilistic guarantees P_k for different precision levels.

The *optimization* goal is to provide the specified privacy levels and their probabilistic guarantees by (a) utilizing a minimal number of LSs which (b) store shares in an optimal way. By minimizing the number of required LSs, we limit the overhead required for updating (communicating) and storing shares at multiple servers.

We define the following as given:

- a master share s_0,
- a set S of n refinement shares $\{s_1, \ldots, s_n\}$ to provide the precision (privacy) levels ϕ_k for the LBAs,
- a set L of m_0 available LS, which can store shares, $L = \{\text{LS}_1, \ldots, \text{LS}_{m0}\}$,
- a set of risk values $\{p_1, \ldots, p_{m0}\}$ providing the probabilities for each LS$_i$ of L that it can be compromised ($p_i \in [0; 1]$),
- the probability values $P_k(\phi_k)$, which specify the required probabilistic guarantees for each precision level ϕ_k ($k = 0 \ldots n$).

Problem: Find the *minimal number m* of LSs in the range $2 \le m \le m_0$ and a *share placement* place(...) of n shares to a set of m LSs denoted as L':

$$\text{place}(\{s_1, \ldots, s_n\}, L) : S \rightarrow L' \subseteq L, \tag{3.2}$$

such that $m = |L'|$ is minimal and it satisfies the user's security requirements:

$$\forall\ \phi_{\text{k,attack}} : P_k(\phi_k) > Pr[\phi_{\text{k,attack}} \le \phi_{\text{k}}] \tag{3.3}$$

3.2. Background and Related Work

Next, we describe the existing trust models and some techniques of performing optimal placement, including exact and heuristic methods.

3.2.1. Trust Models

Trust is a vague term which cannot be precisely and universally defined [Gam88]. Among the existing trust models described in the current research [NWvL07], many are applicable for our system, since they define trust values[1] as probabilities of a certain trustee's behavior. For instance, Maurer [Mau96] formally described how a user can derive probabilistic trustworthiness values based on a set of experienced events. An evidential model where probabilistic trust values are generated by a distributed reputation system was presented by Yu and Singh [YS02]. Jøsang and Ismail [JI02] define a probabilistic trust value as the expectation of the beta probability function, which allows to determine the posterior probability of binary events based on the collected feedback.

Another usage of probabilistic values was presented by Singhal and Ou [SO09]. The authors presented a concept of how an attacker can combine a system's vulnerabilities to stage an attack such as a data breach, and a model of dependencies among vulnerabilities by using probabilistic graphs based on various input data.

A trust model of the feedback-based reputation system proposed by Gutscher allows trust values to be computed based on trust relations [Gut07, GHS08]. The authors distinguish between first-order probabilistic trust calculus, where trust is represented through real numbers in the interval [0;1], and second-order calculus, where trust is a discrete probability distribution of probabilities corresponding to the trust values (also in [0;1]).

A *generic trust model* was proposed by Kinateder et al. [KBR05], which utilizes different trust models in order to generalize them. The authors present a formal basis for the transformation of trust values of different trust models into a comparable probabilistic trust metric with values in [0;1].

3.2.2. Placement and Allocation Optimization Techniques

Our problem is to optimize the placement of position shares on non-trusted servers in order to balance security risks. This problem can be categorized as a problem of combinatorial optimization. A generalized combinatorial optimization problem is to find an optimal solution from a variety of possible solutions (i.e. *combinations*), while satisfying optimality criteria [KV06, Hu82]. Some well-known combinatorial optimization problems include Travelling Salesman Problem (TSP) [KV06], Bin Packing [KV06, MT90] and Knapsack Problems [MT90]. In the case of TSP, for example, the goal is to find the shortest route that visits all the nodes

[1]In this work, we consider only static values given at one point of time, without being changed over time.

of a given graph exactly once and then finishes at the starting point.

Formally, a combinatorial optimization problem is defined as follows [KV06]: Let L be the set of all possible solutions; then, the cost function f which assigns cost to each possible solution is:

$$f : L \rightarrow \mathbb{R} \tag{3.4}$$

To solve a combinatorial optimization problem, we need to find an optimal solution l_{opt} out of the set of all possible solutions L, which is the best in terms of the cost function and satisfies all the given requirements, formally:

$$(\exists \ l_{opt} \in L) \ \forall \ l \in L : f(l_{opt}) \leq f(l) \tag{3.5}$$

The objective of the combinatorial optimization is to find such an l_{opt}. Applied to our problem, this corresponds to traversing the set L of all possible placements of shares to LSs. The cost function f corresponds to the resulting probabilistic guarantees of privacy levels provided by the given share placement. We need to select the optimal placement, i.e., placement that satisfies the required probabilistic guarantees of privacy levels for the minimal possible n. The full problem description of share placement will be presented in Section 3.1.

Example: Knapsack Problem

The Knapsack Problem [KV06] is an example of a typical combinatorial optimization problem, which is similar to our share placement problem. The Knapsack Problem's goal is to distribute (allocate) *objects* among *knapsacks* in such a way that no single *knapsack* is overloaded, and at the same time the number of *knapsacks* used is minimized.

Every optimization problem includes an important decision: can the problem for the given requirements be solved or not. In the case of the Knapsack Problem, the question is whether n *objects* can be distributed among k *knapsacks* without the condition of Equation 3.5 being violated. In our case, we assume that a solution is possible. If not, the requirements of acceptable probabilistic guarantees of privacy levels of Equation 3.5 must be relaxed.

The Problem's Complexity

Combinatorial problems are often *NP-hard* [HU90]. The *NP-hardness* of a problem means that the given problem is at least as hard to solve as any other problem in the complexity class *NP*. If a problem is NP-hard, there exist no (known) deterministic algorithm which can solve it in polynomial time. Currently, for the class of NP-hard problems, no faster solutions than deterministic algorithms with exponential complexity are known.

One method to prove that problem A is NP-hard, is to select a problem B such that B is already known to be NP-hard [Sto01, MT90]. If B can be reduced in polynomial time to A, then A is NP-hard. The reduction algorithm should transform the initial problem in such a way that the same solutions are generated both before and after the reduction for the same inputs.

Some basic and widely known techniques of combinatorial optimization are described below. As stated previously, these problems are often NP-hard, and therefore the trade-off between the solution's accuracy and its performance is very important.

Complete Enumeration Method

The complete enumeration method enumerates and evaluates *all* the possible solutions (combinations), and then selects the best one. The weak side of this approach is the large number of possible solutions; this number increases exponentially with the size of the problem. Although a calculation may take an extremely long time for large combinatorial problems, taking into account the computing power of modern CPUs, the method of complete enumeration is acceptable for small combinatorial problems.

Branch-and-Bound Approach

Branch-and-Bound is an approach based on backtracking algorithms for optimization problems [LW66, Hu82]. Backtracking algorithms process a tree of solutions and try to construct the successive candidate solutions into an overall solution. Branch-and-Bound applies backtracking's structured search in a tree of solutions to optimization problems, where optimality is considered as a decision factor, along with the validity of the solution candidate. Prior to entering a tree's branch, the algorithm checks whether the following results in this direction will be non-optimal with regard to a given threshold. Only if the obtained value is below the threshold, will the search be continued in this direction. If there are no current branches that satisfy this condition, then there will be a step back in the tree of solutions. If at some decision step it becomes clear that no valid overall solution can be constructed, the complete remaining branch of candidates is discarded. If the search process reaches a dead end, the algorithm goes back one step to the next node, in order to process the unvisited nodes by following the same principle.

The Branch-and-Bound method is accurate, meaning that it always finds a solution and goes through a structured list of all candidate solutions, from which the invalid or non-optimal solutions are excluded as early as possible. Although this reduces complexity, the processing time of the tree of solutions is still $\mathcal{O}(c^n)$: in the worst case, all nodes must be visited. In practice, the performance of Branch-and-Bound is highly dependent on each specific problem. The definition of the problem also includes how to set the threshold and how many branches can be excluded at an early stage.

Heuristics

The problem-specific *heuristic* methods are those approaches that determine an approximate solution of NP-hard problems in an efficient amount of time. Some examples of the known heuristics for NP-hard problems can be found in the literature [KV06].

Metaheuristic describes a method for approximate determination of the solution for an optimization problem. This method is not problem-specific but rather generally applicable [Wei02]. Next, we make an overview of some metaheuristic methods, including more detailed descriptions of evolutionary algorithms, since later in this chapter we present a share placement approach based on this concept.

Metaheuristics (1): Local Search

According to the *local search* approach [Egl90], also known as the Hill Climbing algorithm, the first step is to determine an initial solution, which can be done at random or based on a heuristic. After that, starting from this initial solution, the neighboring solutions that are numerically close to it are processed. If there is a better solution in the neighborhood, then the algorithm proceeds further. The problem is that the process can get stuck in a local optimum, since it accepts the first local maximum as a solution.

The method of *Simulated Annealing* [Egl90] is a modification of the local search approach. This concept is based on the simulation of the annealing process in physics. This process follows the law that a system's transition from state Z to another state depends on $e^{energy(Z)/temp}$, where $energy(Z)$ is the energy of the system in that state, and $temp$ is the current temperature. If we consider temperature as the probability of acceptance of non-optimal neighboring state Z', then applied to optimization problems, this means that the lower is the temperature during the probabilistic search, the less likely is the selection of a non-optimal state.

Although the described methods are fast, they are not always sufficient. The pure local search can get stuck in a local optimum, whereas Simulated Annealing is very much dependent on each concrete optimization problem [Wei02].

Metaheuristics (2): Evolutionary Algorithms

Evolutionary algorithms simulate optimization problems as evolutionary processes [Wei02]. They can be defined as an extension of the local search approach; the difference is that now additional solution candidates are generated and combined with each other. In addition, mutations are performed to ensure that the entire possible range of values is covered. The following types of evolutionary algorithms are distinguished: evolution strategies, evolutionary programming and genetic algorithms.

For an evolutionary algorithm to be applied to an optimization problem, the problem must be encoded as a *genome*. The candidate solutions are usually represented as bit vectors (alternatively, as integer vectors). Potential solution candidates are called *individuals*. First, the initial solution candidates are determined either randomly or based on a heuristic. The set of all individuals at a given time is referred to as a *population* or *generation*.

Individuals of the initial population are crossed with each other, and different crossover operations are applied. Depending on the crossover operator, different parts of the genome are combined. Finally, a random exchange of some individuals is performed, which is called mutation. This operation is important to cover all potentially possible values in the range. The individuals generated from a population will then be evaluated by a *fitness* function. The *fitness* function rates the quality of an individual with respect to the optimization goal. Then, the top rated individuals are selected, and the cycle begins again. The whole cycle is shown in Figure 3.2; it ends either when a pre-determined number of generations is reached or when the target criterion is achieved.

Evolutionary algorithms provide a good approximation for combinatorial optimization problems, since they traverse a lot of combinations by using the mutation operator, and therefore, can reliably find the global optimum. Furthermore, the algorithm's parameters are easy to customize, e.g., through better selection of the initial population we can improve

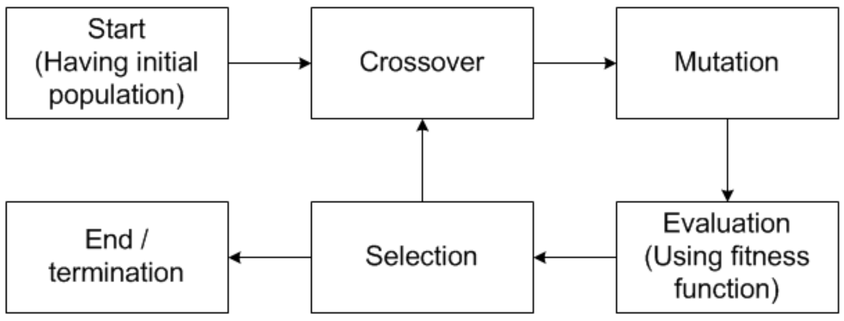


Figure 3.2.: Cycle of an evolutionary algorithm [Wei02]

the algorithm's runtime and accuracy.

In principle, evolutionary algorithms approach the optimal solution quickly, but then require more time as they continue searching for the exact globally optimal value. The disadvantage is the increased processing time compared to the local search approach, since the performed operations are more complex. The processing time of evolutionary algorithms depends on the termination condition: if the number of iterations of generations is fixed, the required time is rather constant and is determined primarily by the cost of the combination operators.

3.3. Analysis of Share Placement's Influence on Privacy

Before presenting our solution to the share placement problem, we make two basic assumptions. First, increasing the number m of LSs leads to higher security with regard to probabilistic guarantees of precision levels. At the same time, a large m is not desired, since it would increase the storage resources utilization and communication cost. Therefore, it is beneficial to incrementally increase m only until the security requirements are fulfilled. Second, we can increase security by optimizing the distribution of shares for a given m. In this section, we analyze the validity of both assumptions.

3.3.1. Influence of LS Risks on Probabilistic Privacy Guarantees

In this section, we illustrate the influence of LS risks on the user's probabilistic guarantees of precision levels. To differentiate it from other parameters, we assume that all LSs have equal risk p and that shares are uniformly distributed among the LSs, i.e., all Δ_i^{ϕ} are equal. p denotes the probability that an LS is compromised (equal for each LS) and therefore an attacker can access its refinement shares.

There are $\binom{m}{k_{attack}}$ combinations of LSs to compromise exactly k_{attack} LSs. Each combination has the probability $p^{k_{attack}}(1-p)^{m-k_{attack}}$. Therefore, $P_{\text{k,attack}}(\phi_{\text{k,attack}}) = Pr[\phi_{\text{k,attack}} \leq \phi_{\text{k}}]$ can be calculated as follows:

$$P_{\text{k,attack}}(\phi_{\text{k,attack}}) = \sum_{k=k_{attack}}^{m} \binom{m}{k} p^k (1-p)^{m-k} \tag{3.6}$$

Based on this formula, we can show the changes of different LS probabilistic guarantees of precision levels by varying p.

In Figure 3.3, we show the user's probabilistic guarantees of precision level (i.e., the probability that at least k LSs will be compromised) for $m = 10$ and different risk values p. As we can see, for $p = 0.1$ the probability of at least three LSs being compromised is less than 10%; for $p = 0.5$ with probability 50% at least five LSs will be compromised; and $p = 0.9$ provides almost no security at all. In general, user security has a very strong (reverse exponential) dependency on p.

3.3.2. Influence of Number of LSs on Probabilistic Privacy Guarantees

Next, we analyze the influence of the number m of LSs on the security, assuming that LSs have equal risk values p. The assumption of equal risks is similar as in Chapter 2 [DSR11, SDR12]. Obviously, in this case, each increase of the number of equally trusted LSs (which store equal shares) also improves security, since any potential attack on an LS can compromise a smaller piece of position information. To demonstrate this, we consider a scenario with $p = 0.2$ for each LS and two numbers of LSs, namely $m = 3$ and $m = 10$. In this scenario, $k = 100$ shares were distributed uniformly among the LSs.

Figure 3.4 depicts the resulting probabilistic guarantees of precision levels $P_{\text{k,attack}}$ of

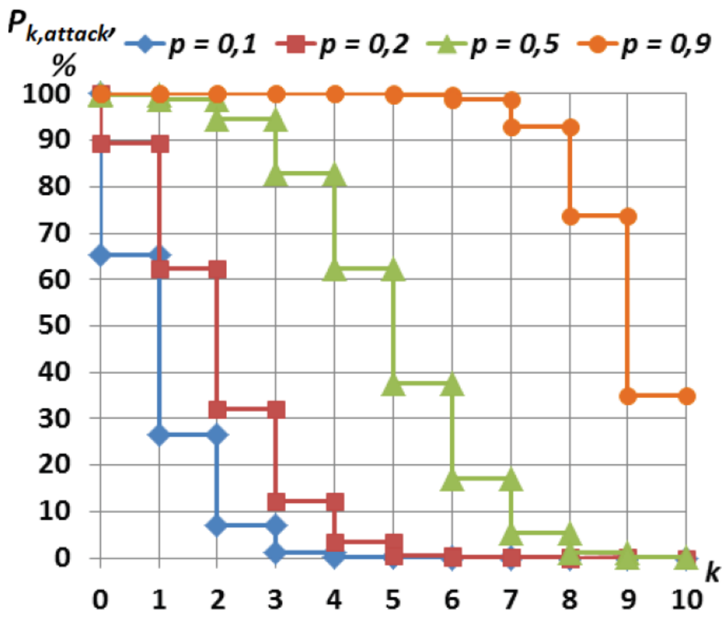


Figure 3.3.: Probabilistic guarantees of precision levels $P_{\text{k,attack}}$ depending on various LS risk values p

obtaining a certain increase of position precision ϕ for $\phi_{min} = 100$ km. Note that the depicted curves have different numbers of steps, since the precision revealed to an attacker depends on the number of compromised LSs $(0, \ldots, 3$ for $m = 3$; $0, \ldots, 10$ for $m = 10)$. We can see that a larger number of LSs increases the security, i.e., it provides lower probability of obtaining a position of certain precision. For instance, to get an increase in precision of $\phi = 30$ km, an attacker needs to compromise only one LS for $m = 3$, but he or she has to compromise three LSs for $m = 10$ in order to get the same result. Therefore, in addition to increasing the flexibility of generating different precision levels, using a larger number of LSs for storing position shares also increases security.

3.3.3. Influence of Number of LSs on Probabilistic Privacy Guarantees

In this section, we show how changing the number and set of selected LSs affects the user's probabilistic guarantees of precision levels. We calculate the probabilistic guarantees $P_{\text{k,attack}}(\phi_{\text{k,attack}})$ in a similar way as in Equation 3.6. With different risks, the difference is that now, instead of multiplying each kth probability by the number of $\binom{m}{k}$ combinations, we

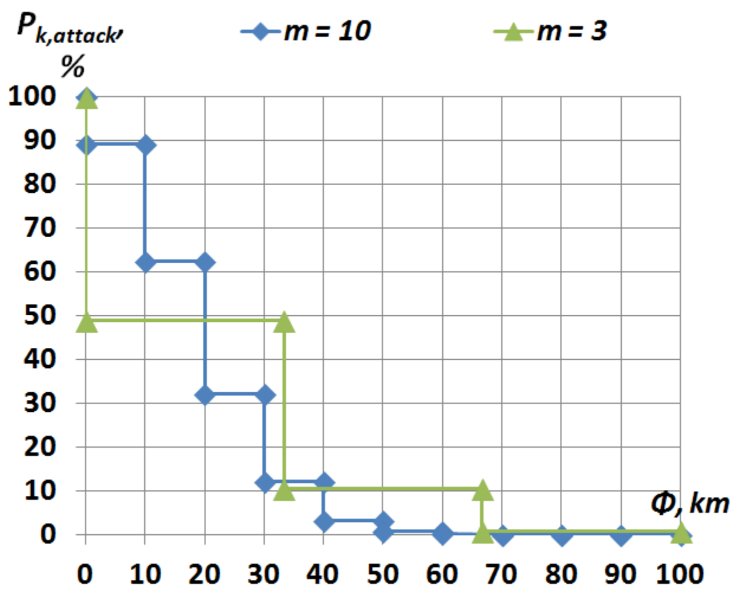


Figure 3.4.: Probabilistic guarantees of precision levels $P_{\text{k,attack}}$ depending on two different sets of LSs ($m = 3$ and $m = 10$) for $\phi_{min} = 100$ km; ϕ denotes the obtained increase of position's precision

summarize the probability of each k-combination separately:

$$P_{\text{k,attack}}(\phi_{\text{k,attack}}) = \sum_{k=k_{\text{attack}}}^{m} \sum_{i=0}^{\binom{m}{k}} p_{i,\text{incl}} \cdot p_{i,\text{excl}}, \tag{3.7}$$

$$p_{i,\text{incl}} = \prod_{j=0}^{m} p_j, \forall p_j \in P_{k,i} \tag{3.8}$$

$$p_{i,\text{excl}} = \prod_{j=0}^{m} (1 - p_j), \forall p_j \notin P_{k,i}, \tag{3.9}$$

where $P_{k,i}$ is the set of risks of the ith k-combination out of m LS risks. There are $\binom{m}{k}$ combinations of LSs to compromise exactly k LSs. Each combination has the probability defined by multiplying the risks p_j of (included) k LSs and the inverse risks $1 - p_j$ of the rest (excluded) $m - k$ LSs. To get a probability of *exactly* k compromised LSs, we have to summarize the probability of each k-combination. Finally, to get a probability of *at least* k compromised LSs, we summarize the probabilities corresponding to $\{k, k+1, \dots, m\}$ compromised LSs.

In the next sections, we will also consider placement of differently heterogeneous shares to LSs. In this case, Equation 3.7 must be extended to include the individual precision increase $\Delta\phi_j$ of share s_j as a multiplier in order to take into account the heterogeneity of shares (for homogeneous shares, $\Delta\phi_j$ is implicitly assumed to be 1 for each share s_j):

$$P_{\text{k,attack}}(\phi_{\text{k,attack}}) = \sum_{k=k_{\text{attack}}}^{m} \sum_{i=0}^{\binom{m}{k}} p^i_{\text{incl}} \cdot p^i_{\text{excl}} \cdot \Delta\phi_j \tag{3.10}$$

The examples of LS sets with less diverse and more diverse risks are shown in Figure 3.5 and Figure 3.6 for $m_0 = 5$. In the beginning, $m = 2$; then we incrementally add one LS with the lowest risk at a time, until $m = 5$. We can see that the positive effect of m's increase is neutralized by the growing risks of the newly included LS, especially in case of strongly different risks (Figure 3.6). Thus, there is usually no need to increase m further if the next

LSs to be selected are much less trusted than the ones previously selected.

As we can see, the equal share placement on LSs with different risks can increase the probability of the position's precision being revealed to an attacker, especially at lower precision levels. But an increase of the number of selected LS m alone may not solve this problem. In order to achieve the required probabilistic guarantees of precision levels for the given m, we optimize share placement to the LSs as we will show in the next sections.

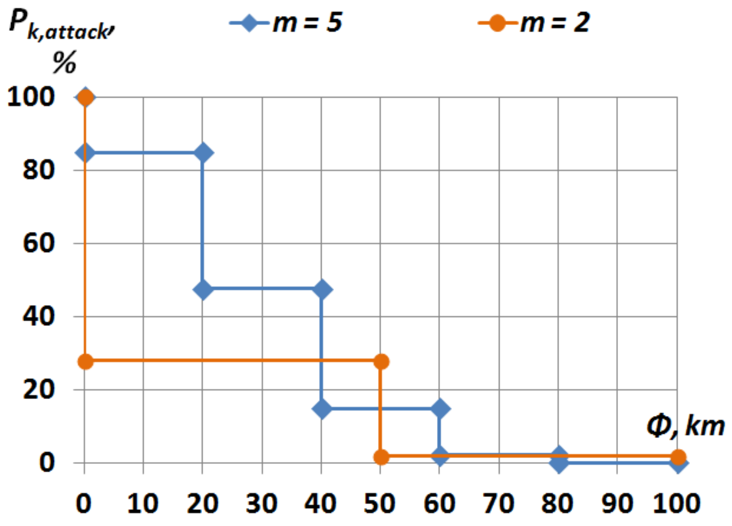


Figure 3.5.: Probabilistic guarantees of precision levels $P_{\text{k,attack}}$ depending on various sets of selected m LSs: the case of less diverse LS risks: $m_0 = 5; p_1 = 0.1; p_2 = 0.2; p_3 = 0.3; p_4 = 0.4; p_5 = 0.5$

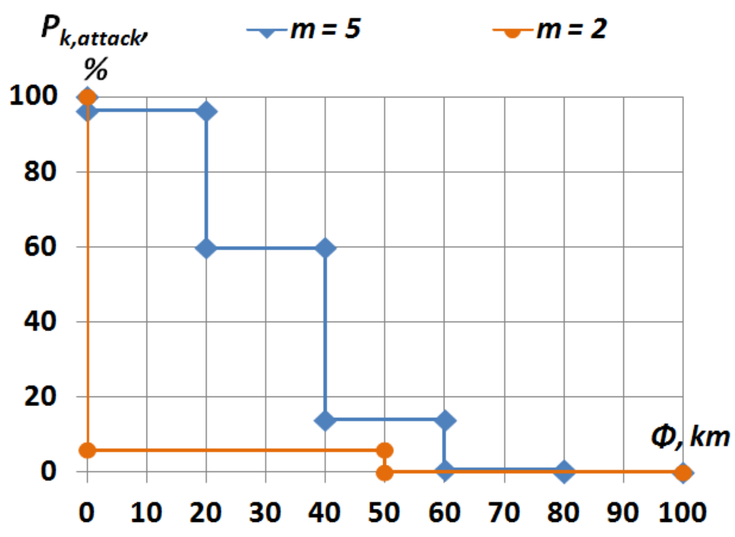


Figure 3.6.: Probabilistic guarantees of precision levels $P_{\text{k,attack}}$ depending on various sets of selected m LS: the case of more diverse LS risks $m_0 = 5; p_1 = 0.01; p_2 = 0.05; p_3 = 0.25; p_4 = 0.5; p_5 = 0.9$

3.4. General Selection & Placement Algorithm

In this section, we describe the general selection and placement algorithm (Algorithm 10), which presents the solution of two goals defined in our problem statement (Section 3.1.3) in two major steps without describing the second step in detail: (a) selection of m LSs $L' = \{\text{LS}_1, \text{LS}_2, \ldots, \text{LS}_m\}$; (b) optimization of placement of n shares among these LSs by mapping the shares to the selected LSs: $S \rightarrow L'$. The goal of optimization is to improve the provided probabilistic guarantees $P_{\text{k,attack}}$ by placing more position information on more trusted LSs.

The basic idea is to start with the smallest set of LSs and incrementally increase m until the security constraints (Equation 3.1) are fulfilled. As shown in Algorithm 10, first we calculate and check the probabilistic guarantees of precision levels for uniform share placement, where every LS manages exactly one share, independent of its individual risk value. This is the strategy used in our basic position sharing approach presented in Chapter 2 [DSR11, SDR12].

For each number of LSs, we first check whether a uniform placement (line 8) where each LS stores an equal number of shares (independent of its individual risk value) fulfills the user-defined probabilistic guarantees of privacy levels (lines 9-11). If a uniform placement fulfills these levels, we have already found a placement solution, as this placement fulfills the required probabilistic guarantees of precision levels and contains the minimum number of LSs. By optimizing the share placement (line 12), we can further improve security beyond the user-requested level (line 13) at the expense of using computational and energy resources for running the optimization algorithm.

Algorithm 10 General Selection & Placement Algorithm

```
1:  function $place(P_k(\phi_k), S, L, m_0, m_{min}, n)$
2:  $m \leftarrow m_{min} - 1$
3:  sort_by_ascending_p$_i(L)$
4:  $L' \leftarrow$ get_selected_set$(L, m)$
5:  $solution\_found \leftarrow false$
6:  repeat
7:      $m \leftarrow m + 1$
8:      distribute_equal$(P_k(\phi_k), S, L', m)$
9:      if $\forall\ \phi_k : P_k < P_{k,attack}(\phi_k)$ then
10:         $solution\_found \leftarrow true$
11:     else
12:         place_optimized$(S, L', m)$
13:         if $\forall\ \phi_k : P_k < P_{k,attack}(\phi_k)$ then
14:             $solution\_found \leftarrow true$
15:         end if
16:     end if
17: until $(m = m_0)||(solution\_found)$
18: return $S \rightarrow L'$
```

If the uniform (non-optimized) share placement on LSs already represents a solution that satisfies the problem statement's requirements, we skip the optimization algorithm to save the resources of the mobile device that executes this algorithm. If the uniform share placement does not satisfy the user's privacy requirements, we optimize the placement by relocating shares from less trusted to more trusted LSs, as will be presented later in detail.

If it is still impossible to satisfy the required probabilistic guarantees of precision levels for the current m, we increase the number of LSs. In each step, we add the next most trusted LS to set L', since the subset of the most trusted LSs provides the highest security. Therefore, the available LSs must be initially sorted by ascending risks p_i (line 3).

If m reaches the total number of available LSs m_0, while a solution has not been found, the current user's security requirements are too strict for the given constraints. Therefore, in order to perform an exhaustive search, the user should relax the constraints (i.e., probabilistic guarantees of privacy levels) given in Equation 3.1 (Section 3.1.2) step by step, and execute the algorithm again.

In lines 9 and 13 of Algorithm 10, we calculate the probabilistic guarantees of precision levels of a placement. In Section 3.3.3, we will show how we calculate the probabilistic guarantees of precision levels $P_{k,attack}$ for different numbers k_{attack} of compromised LSs for a given placement.

3.5. Optimizing Share Placement

In this section, we present an algorithm to solve the optimized share placement problem, which is executed within Algorithm 10 by calling function place_optimized(...). We consider the situation where a set L' of $m = |L'|$ LSs with lowest risks has been selected and is *fixed*. Thus, an optimized placement of n shares to these LSs in L' must be found, as the uniform placement strategy did not satisfy the user's security constraints (cf. problem statement in Section 3.1.3).

First, we show that this problem is NP-hard. Then, we propose a heuristic solution based on a genetic algorithm. After that, we show some simpler solutions for the special cases of share placement, which allow placement to be calculated in linear time.

3.5.1. Share Placement Problem and Its Complexity

At this point of the algorithm, we have a set of m LSs defined and fixed. Now, we aim to achieve a *balanced* placement of n shares among these LSs such that no single LS represents a higher security risk with regard to the stored precision than the other LSs, i.e., we guarantee that the probability of each precision level's disclosure does not exceed the pre-defined threshold (as defined in our Problem Statement in Section 3.1.3).

Our optimization of the share placement among LSs is based on the principle of allocation of capital (i.e., shares) between segments or business units (i.e., LSs). We take into account the important properties of our problem: (a) we first generate all the shares and distribute them all at the same time (ad not one by one), and (b) according to our system model, LSs are independent entities and there are no stochastic dependencies between them. If shares would be distributed in an additive manner, or there would be interdependencies between the LSs, *incremental* allocation, *marginal* allocation or *Myers-Read method* of allocation could be used [Alb03, VM03]. In our case, we require a non-incremental allocation, which is called absolute allocation.

There exist many methods of allocation, most of which however have different system model assumptions compared to our system model. For example, they assume more complex and differentiable definition of the risk measure (*Euler principle*), explicit collaboration between the LSs (*Game Theory*), or stochastic dependencies between the LSs and risks

(*Covariance principle*, *Conditional Expectation principle*, etc.) [Pav08, Alb03, VM03]. Thus, after selecting from methods of capital allocation, we employ *absolute proportional* allocation. The principle of proportional allocation is to equalize the expected information losses (also called Expected Monetary Values in the risk management theory [Pri97]) corresponding to different LSs and calculated as multiplication of the risk and the share value assigned to each LS. Proportional capital allocation guarantees that the allocated capital must not exceed the stand-alone risk-adjusted capital assigned to each LS, which fits our problem statement goal (Equation 3.3).

We call our placement problem the Balanced Risk Placement Problem (BRPP). Formally, a share placement $S \rightarrow L' \subseteq L$ has balanced risk if the proportion of position precisions $\phi_{\mathrm{i1,j}}$ and $\phi_{\mathrm{i2,j}}$ stored by $\mathrm{LS}_{\mathrm{i1}}$ and $\mathrm{LS}_{\mathrm{i2}}$ respectively ($j = 1 \dots n$) is inversely proportional to the corresponding risks p_{i1} and p_{i2} of $\mathrm{LS}_{\mathrm{i1}}$ and $\mathrm{LS}_{\mathrm{i2}}$:

$$S \rightarrow L' \subseteq L : \forall\ i1, i2 \in \{1, \dots, m\} : \frac{\sum_{j=1}^{n} \Delta\phi_{i2,j}}{\sum_{j=1}^{n} \Delta\phi_{i1,j}} = \frac{p_{i1}}{p_{i2}} \tag{3.11}$$

If the exact equality of proportions is not feasible due to the given risk values and other parameters, the goal is to find a share placement solution which is close to the best possible solution:

$$\text{minimize :}\quad max_{i=1}^{m} \sum_{j=1}^{n} p_i \Delta\phi_{i,j} - min_{i=1}^{m} \sum_{j=1}^{n} p_i \Delta\phi_{i,j}, \tag{3.12}$$

under the restrictions of probabilistic guarantees of precision levels given in Equation 3.3. This problem refines the placement goal (b) of our main problem statement (Section 3.1.3), i.e., defines how the function place_optimized(...) of Algorithm 10 must be realized.

In the general case, we must distribute n heterogeneous shares among m LSs with heterogeneous risks. The total number of possible combinations is m^n. The considered BRPP problem (Equation 3.12) is a NP-hard problem. This can be shown by reducing the Agent Bottleneck Generalized Assignment Problem (ABGAP), which is known to be NP-hard [MN88, AP98], to BRPP. The formal definition of ABGAP is:

$$\text{minimize}: \quad max_{i=1}^{m} \sum_{j=1}^{n} p_i \Delta\phi_{i,j} \tag{3.13}$$

$$\text{subject to}: \quad \sum_{i=1}^{m} \sum_{j=1}^{n} w_{i,j} \cdot x_{i,j} \leq w_i, \tag{3.14}$$

$$\text{where} \quad \sum_{j=1}^{n} x_{i,j} \leq 1; x_{i,j} \in \{0;1\}; i = 1, \ldots, m; j = 1, \ldots, n \tag{3.15}$$

ABGAP is equivalent to our placement problem (Equation 3.2), since one can be polynomially transformed into another: If we simplify our problem by adding an LS with zero risk, we can exclude the second term from Equation 3.12. This means that in order to solve our problem, we must also solve ABGAP. Thus, our problem is at least NP-hard. An exhaustive search to solve an NP-hard problem is infeasible in a reasonable amount of time.

The total number of possible placement combinations for distributing n shares among m LSs is $O(m^n)$. Since this number grows exponentially with the number of shares, an exhaustive search is very costly for larger m and n. Even relatively small numbers (e.g., $m = 5$ and $n = 15$) require analysis of more than $3 * 10^{10}$ combinations. At the same time, such an exhaustive search is not worth its computational costs, as a much faster linear-time heuristic can produce a solution that would not differ much in terms of probabilistic guarantees of privacy levels. Our goal is not to find the best placement among all possible combinations, but for a placement that is secure enough to satisfy the required probabilistic guarantees of privacy levels. Therefore, we need a strategy that guides our search for secure placement in a reasonable (linear) period of time.

3.5.2. Optimized Share Placement Algorithm

Since BRPP is NP-hard, we use a heuristic approach to solve it. In general, problem-specific heuristics or meta-heuristics can be used to find an approximation solution. We applied the meta-heuristic of *genetic algorithms* [Wei02]. Genetic algorithms belong to the class of evolutionary algorithms (see Section 3.2.2), as they reproduce the process of biological evolution. In general, they traverse multiple solution candidates by combining and mutating[1] them into new possible solutions. Each new solution (in our case, a share placement) is rated according to a fitness (objective) function defined by Equation 3.12. Then, the best placements in terms of the objective functions are selected, and the cycle can repeat until the goal is reached or the limit of cycles is achieved.

We implemented a genetic algorithm for share placement as shown in Algorithm 11. The input parameters are the probabilistic guarantees of precision levels $P_k(\phi_k)$, the set of LSs L' of size m, and the fixed set of shares S of size n. First, we define the initial population as 10 random placements (line 3). Then, we build a population of 40 new placements by recombining two placements with a uniform crossover (with a probability of 50%) (lines 5-11). Afterwards, the placement is mutated by changing one assignment randomly (line 12), ensuring that all theoretically possible placements can be created. The values of 10 initial placements and 40 recombined placements are selected in such a way that they provide a large number of combinations within each iteration.

Next, the 40 created placements are rated according to an objective function, and the 10 best placements are selected (lines 14-15). The 10 best placements are used as input for the next algorithm's iteration. Thus, the resulting placements are getting better after every iteration. This cycle is iterated 200 times or stopped if the conditions of Equation 3.12 are satisfied (lines 4-18). The value of 200 iterations is selected such that it ensures convergence. Our experiments have shown that we already achieve a near-optimal placement solution after ca. 20 iterations. If, after all cycles, the probabilistic guarantees of precision levels are still not acceptable (line 4), we say that the solution cannot be found for the given input parameters.

3.5.3. Placement Strategies for Special Cases

In this section, we analyze special placement cases in order to apply simpler solutions than the general one presented above under some given initial conditions.

The possible placement cases are summarized in Table 3.1. First, we distinguish whether the set of shares is pre-defined or if a user can freely generate a new share set that would better suit the needed placement. Second, we consider separately the homogeneous and heterogeneous trustworthiness (risk) levels of LSs. Finally, the generated shares themselves can provide different or equal increases of precision, i.e., be *homogeneous* or *heterogeneous*.

[1]The mutation operator ensures that we escape a local optimal solution and advance to new combinations.

Algorithm 11 Genetic Algorithm for Share Placement

```
1: function $place\_optimized(S, L', m)$
2: $t \leftarrow 0$
3: $Popul[1 \ldots 10] \leftarrow RandomPlacement(S, L', m)$
4: while $t < 200$ and $\forall\, P_k < P_{\text{k,attack}}$ do
5:     for $p = 1$ to $40$ do
6:         $i_1 \leftarrow RandomInteger(m)$
7:         $i_2 \leftarrow RandomInteger(m)$
8:         $u \leftarrow RandomBoolean()$
9:         if $u$ then
10:            $PopulTemp[p] \leftarrow$ Cross($PopulTemp, i_1, i_2$)
11:        end if
12:        $PopulTemp[p] \leftarrow$ Mutate($PopulTemp[p]$)
13:    end for
14:    Evaluate($PopulTemp$)
15:    $Popul \leftarrow$ Select10Best($PopulTemp$)
16:    $P_{attack}(\phi) \leftarrow$ BestLevels($Popul$)
17:    $t \leftarrow t + 1$
18: end while
```

We expect that the most relevant cases are **1d** and **2b**, but generally, all of them are realistic and define a placement problem that can be solved in different ways.

First, we consider special cases **1a-1d** under the assumption that the number of shares is not pre-defined, i.e., a user can generate the number of shares that allows for optimal placement (cf. Figure 3.7). Note that in each such case, the number of shares n_i to be stored at each LS_i can be proportionally increased without violating placement optimum criterion: $n_i \leftarrow c \cdot n_i$, where c is a multiplier constant for each i.

Set of shares is not pre-defined			Set of shares is pre-defined		
Case	LS risk	Share	Case	LS risk	Share weight
1a	homogeneous	homogeneous	2a	homogeneous	homogeneous
1b	heterogeneous	homogeneous	2b	heterogeneous	homogeneous
1c	homogeneous	heterogeneous	2c	homogeneous	heterogeneous
1d	heterogeneous	heterogeneous	2d	heterogeneous	heterogeneous
Required approach					
Number adjustment			Placement optimization		

Table 3.1.: Overview of share placement cases

Case 1a: LS risks are homogeneous and shares are homogeneous. Then we can simply generate the needed number of shares n equal to the number of LSs n and place each share to one LS.

$$n = m; \; n_i = \frac{n}{m} = 1 \tag{3.16}$$

Case 1b: LS risks are heterogeneous, while shares are homogeneous. The needed total number of shares n depends on the relation between LS risk values. First, we determine the maximal (worst case) sum: $n_{max} = \sum_{i=1}^{m} \frac{1}{p_i}$; next we divide the result through the greatest common divisor $d = gcd(\frac{1}{p_1} \; \dots \; \frac{1}{p_m})$: $n = \frac{n_{max}}{d}$. Thus, the numbers of generated shares and assigned shares to each LS are:

$$n = \sum_{i=1}^{m} \frac{d}{p_i}; \; n_i = \frac{d}{p_i} \tag{3.17}$$

Case 1c: LS risks are homogeneous, while shares are heterogeneous. This case is trivial and degrades into **1a**, since the shares can be generated homogeneous as risks here.

$$n = m; \; n_i = \frac{n}{m}, \text{ with homogeneous shares } s_i \tag{3.18}$$

Case 1d: LS risks are heterogeneous, and shares are heterogeneous. As in the case **1a**, $n = m$, and the share precision increases are determined as $\frac{1}{r_i}$, and then normalized.

$$n = m; \; n_i = \frac{n}{m}, \text{ with precision increases } \Delta\phi_i = \frac{1}{p_i} \tag{3.19}$$

Under the assumption of free share generation, the computational complexity of share placement for all cases (**1a**, **1b**, **1c** and **1d**) is linear: $\mathcal{O}(m)$. Thus, we do not need a real placement algorithm here, but an assignment of shares after generating suitable heterogeneous shares.

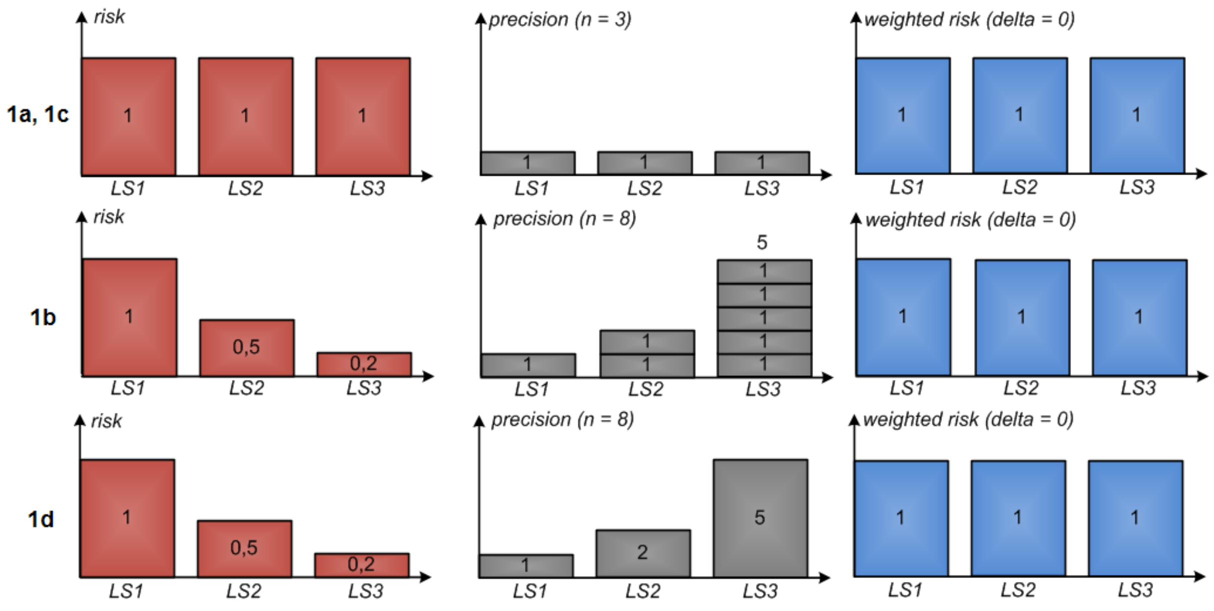


Figure 3.7.: Share placement cases 1a, 1c, 1b, 1d

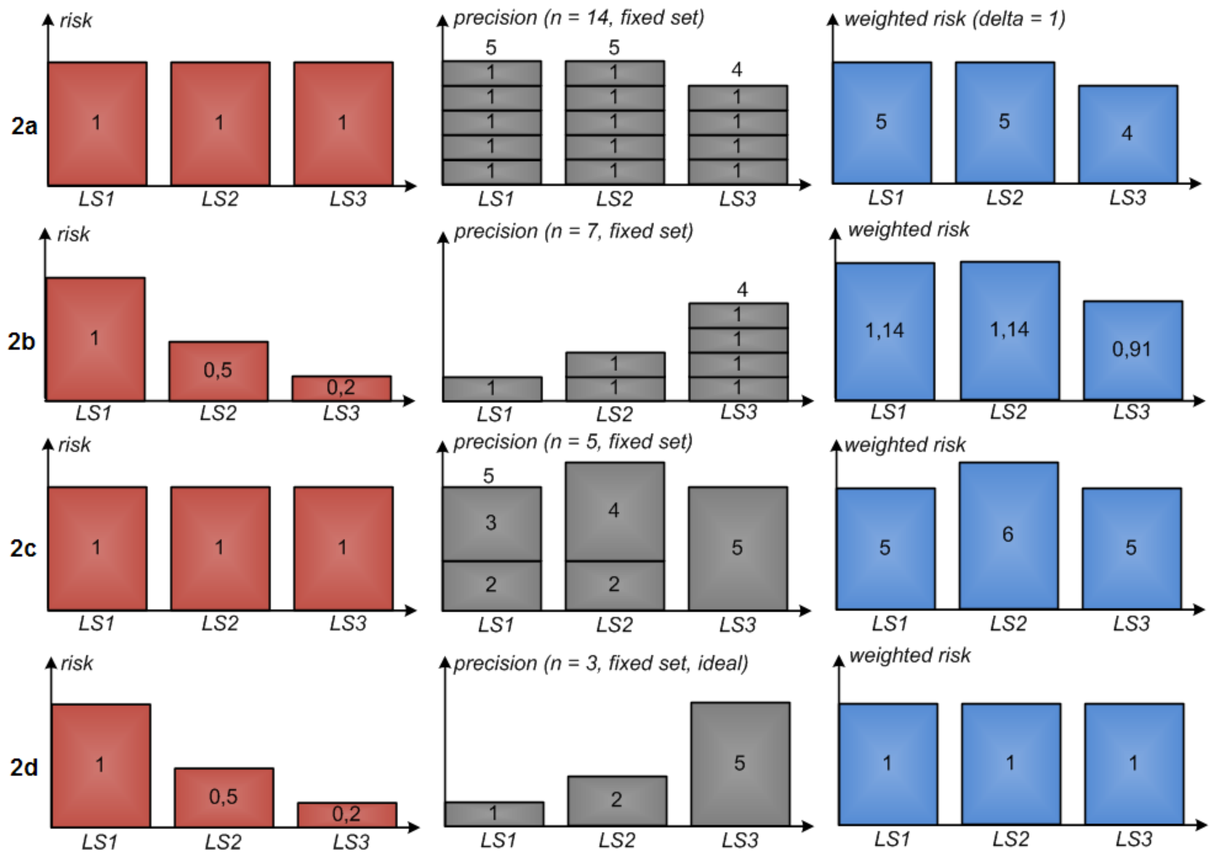


Figure 3.8.: Share placement cases 2a, 2b, 2c, 2d

Next, in cases **2a-2d** (cf. Figure 3.8), we assume that the number and composition of shares are pre-defined, i.e., we cannot generate another set of shares and should improve the placement of the already existing set. Note that in Figure 3.8 we use integer values of share precision increases for the sake of simplicity; they can be also defined by floating-point values.

Case 2a: LS risks are homogeneous, and shares are homogeneous. Having $n \geq m$, we assign shares, e.g., through the *round-robin* principle. The resulting balancing is not dependent on the order of share assignment. The placement imbalance can be relatively large for small m values and small $\frac{n}{m}$ relation, but it cannot be improved under the given assumptions.

$$n = \text{const}; n_i = n \bmod m + (1 - i \bmod m) \tag{3.20}$$

The computational complexity needed for share placement in case **2a** is $\mathcal{O}(m)$.

Case 2b: LS risks are heterogeneous, while shares are homogeneous. Such situations are likely to occur, but the placement problem in such conditions is not trivial and in worst case scenarios require an NP-hard solution. Therefore, we solve case **2b** in the same way as the most general case **2d** below.

Case 2c: LS risks are homogeneous, while shares are heterogeneous: in this case we have a “knapsack problem”. The computational complexity needed for an exact (ideal) share placement in this case is $\mathcal{O}(m^n)$, as in the most general case **2d** below.

Case 2d: LS risks are heterogeneous, and shares are heterogeneous. This situation, being the most general problem, also includes cases **2b** and **2c**. We described the general solution above in Section 3.5.2.

3.6. Evaluation

In this section, we analyze the performance and the improvement of the probabilistic guarantees of precision levels provided by the proposed placement algorithms.

3.6.1. Performance Evaluation

According to the principle of position sharing, share placement has to be calculated on the mobile device of the user, since it is the only trusted entity in our system model. Since mobile devices are typically restricted in terms of processing power and energy, the runtime of our share placement algorithm is crucial. Therefore, we measured the runtime of placing a set of shares on a state of the art mobile device. To evaluate the computational cost, we used a smartphone HTC Desire with Android OS (CPU: 1 GHz Qualcomm QSD8250 Snapdragon, memory: 576 MB RAM). We tested the full number of cycles of the genetic algorithm, without terminating the algorithm under the "solution found" condition (i.e., we have tested the worst case scenario, where the solution is not feasible for the given parameters). The number of LSs was given as $m = 5; 10; 20$, and the number of shares n is in the interval $[m; 50]$.

Figure 3.9 shows the average runtime for placing n shares on m LSs. As our evaluation shows, Algorithm 11 has linear complexity (cf. Figure 3.9) and is executed in less than one second even for larger input parameters ($m = 20$, $n = 50$). Therefore, we conclude that the algorithm is also suitable for resource-poor mobile devices.

3.6.2. Probabilistic Guarantees of Privacy Levels after Placement Optimization

Now, we consider the influence of share placement on probabilistic guarantees of privacy (precision) levels by analyzing several placement examples.

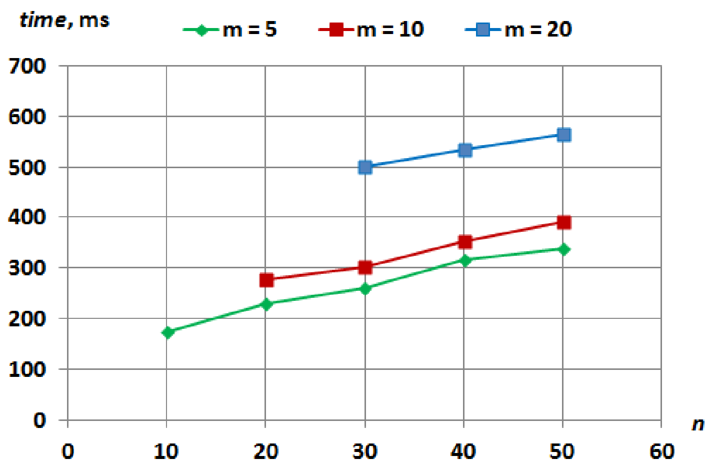


Figure 3.9.: Computational cost of genetic share placement algorithm (Algorithm 10)

As was shown previously in Figure 3.6 (Section 3.3.3), in the case of different LS risks, the probability of an LS compromise can be very high, especially for lower k (i.e., lower precision values). The reason is that the probability values of risky LS cause a substantial increase in some k-combinations' probability (Equation 3.7). We overcome this problem by optimizing share placement, so that LSs with higher security risks will receive less precise shares. The influence of various placements on user security is illustrated in Figure 3.10.

Naïve approach: Assuming that we have different risks $p_1 = 0.1; p_2 = 0.2; p_3 = 0.4$, equal share placement, $r_0 = 100$ km, and each LS has stores shares providing 33 km precision increase. Thus, the precision increase of 33 km (one third of r_0) can be achieved by compromising *at least* one LS with probability $P_{1,\text{attack}} = 56.8\%$ (cf. Figure 3.10).

Optimized approach: Having the same risks as above, we distribute 100 km of precision ($r_0 = 100$ km) proportionally to the risk values (57 km to LS$_1$, 29 km to LS$_2$ and 14 km to LS$_3$). Then the disclosed precision increase (which corresponds to exactly one compromised LS with probability $P_{1,\text{attack}} = 56.8\%$) is not 33 km but only 22 km. The security is improved compared to the naïve approach, since the same probability levels correspond to lower precision values (cf. Figure 3.10).

Next, we compare the resulting probabilistic guarantees of precision levels of optimized share placement compared to a basic (non-optimized) placement algorithm, having a more

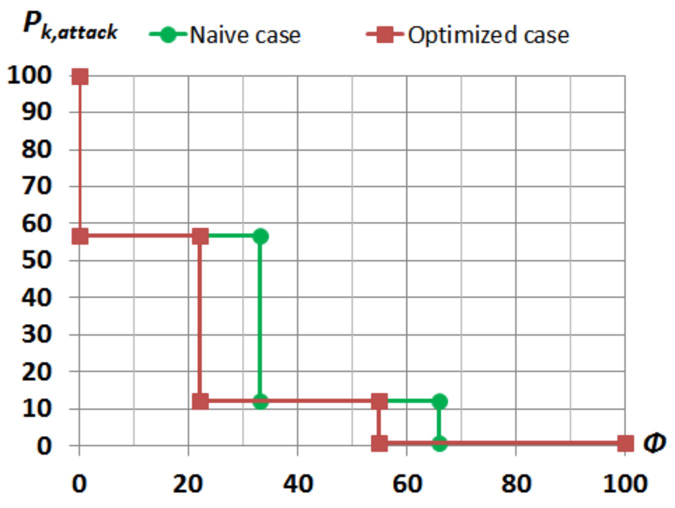


Figure 3.10.: Precision ϕ and probabilistic guarantees of precision levels $P_{\text{k,attack}}$ for different share placements

general scenario with randomly generated risk values. We place $n = 15$ shares on $m = 5$ LSs with heterogeneous risks; the risk values were chosen uniformly at random from the interval $[0; 0.5]$: $p_1 = 0.4932$; $p_2 = 0.3292$; $p_3 = 0.2344$; $p_4 = 0.1788$; $p_5 = 0.0925$. The basic algorithm distributes an equal number of shares (3) to each LS, while the optimized placement placed 1, 2, 2, 3 and 7 shares onto the given LSs.

Figure 3.11 depicts the probabilistic guarantees of precision levels $P_{\text{k,attack}}$ for the different precision levels ϕ. Note that the precision levels $\phi_{\text{k,attack}}$ which correspond to the probability levels $P_{\text{k,attack}}$ are calculated as the weighted average of position precisions of each possible k-combination. The figure shows that the optimized share placement algorithm leads to a significantly lower probability of compromising the respective shares for most precision levels.

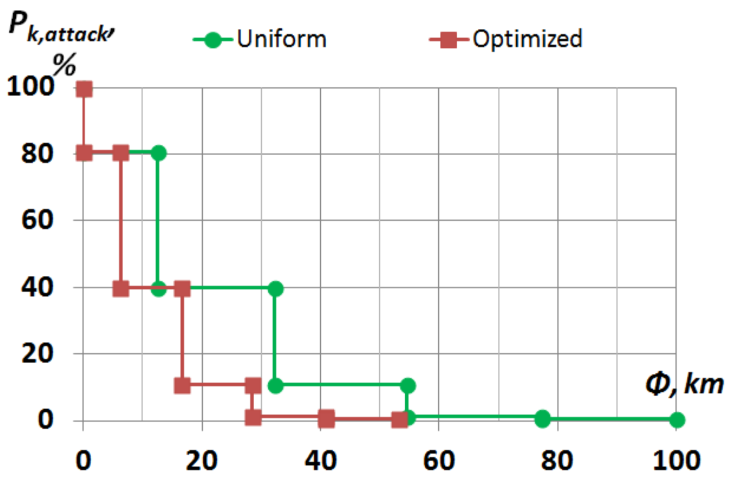


Figure 3.11.: Placement optimization: precision ϕ and probabilistic guarantees of precision levels $P_{\text{k,attack}}$

3.7. Conclusion

This chapter presented an extension for our basic position sharing approach, which improves the user's location privacy in the case when the available location servers are not equally trustworthy. We presented an algorithm to select the minimal required number of LSs and to optimize the distribution of position shares among them. The main result is that more position information is placed on the LSs that are more trusted. Thus, we avoid the situation where high precision can be disclosed with high probability after an attack on an LS. We have shown that our placement heuristic has linear runtime complexity, and therefore it can be executed on MOs with low processing power. We also considered special placement cases in order to apply simpler solutions than the general placement heuristic under suitable initial conditions.

Chapter 4

Location Update Algorithms for Position Sharing

The position sharing approach that was presented in the previous chapters is suited for snapshot (single) location updates, so that a complete set of shares has to be re-generated and sent to the corresponding set of LSs every time a position update event is triggered. This principle might produce a high communication overhead, e.g., if the update rate is high and the number of LSs is large. However, in many cases the re-generation and update of the whole share set causes redundancy (e.g., in cases where movements of the MO are insignificant). Hence, we aim to minimize the number of update messages in our system.

In this chapter, we present a location update approach, the basics of which were published in the diploma thesis of Simon Hänle [Hae12]. The author developed the main concept of the location update approach, contributed to the refinement of location update algorithms for different scenarios, and supervised the diploma thesis as a whole.

After giving an overview of related work, we define the problem of message reduction. Then, we describe two movement scenarios, and finally, we propose an optimized location update algorithm.

4.1. Background and Related Work

In this section, we first describe the location update protocols, focusing on dead-reckoning protocols. After that, we present some works, which aim to optimize message overhead.

4.1.1. Classification of Location Update Protocols

Next, we describe the different types of existing location update protocols. Figure 4.1 illustrates the different types of protocols, which, according to the survey of Leonhardi et al. [LR01], fall into the following three main categories:

- "*querying protocols*" – the server side initiates a location update of the target MO's position;
- "*reporting protocols*" – the decision for a location update is made from the client-side (by the MO);
- "*combined protocols*" – combination of the querying and reporting protocols principles.

Multiple classes of *querying* protocols are distinguished: *simple*, *caching* and *periodic protocols*.

According to *simple protocols*, the server always queries the location update from the MO if it needs the MO's current position, e.g., for a subscribed LBA. This ensures that the obtained MO's position is always accurate. However, this can lead to high numbers of update messages if the position is being queried very often.

Caching protocols represent an optimization of *simple protocols*: the server always stores the copy of the last transmitted position information. When the MO's position is queried by an LBA, the server performs an assessment of the position's accuracy. If the result is below a pre-determined limit, then the server forwards to the LBA the stored MO's previous position; otherwise, the server requests an update from the MO. Here, the estimation of accuracy

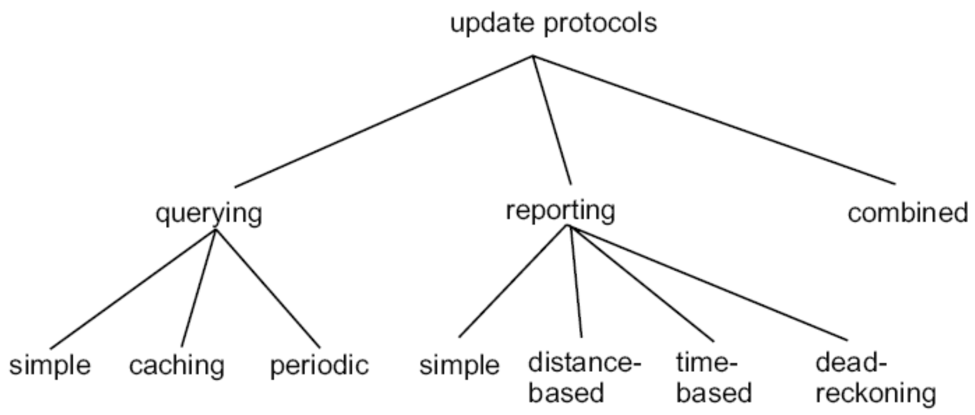


Figure 4.1.: Overview of different types of location update protocols [LR01]

can be either *pessimistic* or *optimistic*. The pessimistic variant is to calculate how far the MO could have moved since the last update based on its maximum speed (the maximum speed value must be defined in advance). In the optimistic variant, not maximal but average velocity of the MO is assumed. Thus, a situation might occur where the position deviation between the last saved MO position and the actual MO position exceeds the limit ϵ, e.g., if the MO has moved since the last update at a speed greater than the average speed. As a consequence, fewer update messages must be sent as compared to the pessimistic variant.

Periodic protocols are organized in such a way that the server requests a location update from the MO periodically, i.e., after a pre-defined time interval. This principle is very similar to the time-based update protocol presented later.

Reporting protocols can be classified as *simple*, *time-based*, and *distance-based* reporting protocols and *dead reckoning* protocols.

According to the principle of the *simple reporting protocol*, a position update is triggered every time a sensor system has detected a position change. In the case of an excessively accurate sensor system, this may lead to a very high number of update messages produced.

Time-based protocols: an update message is always sent to the server after a specified time interval Δt. The accuracy of location data in this case depends on the speed of the MO: if the MO is moving at high speed and the time interval Δt is large, then its actual position differs significantly from the last position stored on the server, i.e., the accuracy is decreased.

Distance-based protocols: a position update message is always sent if the geographical distance since the last update exceeds a specified threshold d. This protocol is well suited to MOs that are moving very slowly or almost not moving at all.

Dead-reckoning protocols represent an optimization of the distance-based protocols, where the server predicts the current position of the MO based on its last position, speed and movement direction. The MO also calculates this position and sends an update message when the geographical difference between the actual position and the calculated position is greater than a specified bound ϵ (Figure 4.2). This protocol can save a lot of messages, provided that the MO is moving at a constant speed in a predetermined direction, or if the MO's destination is known.

4.1.2. A Combined Location Update Protocol

Leonhardi and Rothermel [LR01] proposed a combined protocol that utilizes the ideas of both distance-based update protocols and querying protocols. The system works as follows: the basic approach is the same as for the distance-based protocols, but in the case of insufficient

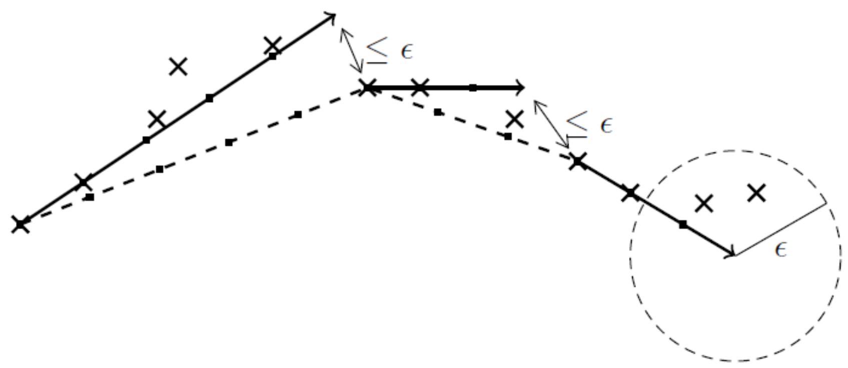


Figure 4.2.: Linear dead-reckoning principle [LDR08]

accuracy, the server may request an update message at any time with regard to the last saved MO's position. To minimize the number of update messages, the limit of distance d can be dynamically adjusted (in the distance-based protocol) in correspondence with the velocity of the MO.

Furthermore, the authors consider the protocol's behavior in the case of an absence of communication between the MO and the server and present the analytical comparison of various location update protocols in terms of location accuracy and the number of update messages produced.

The analytical results of this comparison can be summarized as follows: in general, the performance of the distance-based protocol is better than that of the time-based protocol, and the optimistic caching querying protocol is better than the pessimistic one. The number of update messages increases in the case of querying protocols with the number of requests; therefore the querying protocols are better than the reporting protocols in cases of low query rates. When the query rate is high, distance-based reporting protocols perform better than pessimistic caching querying protocols.

The analysis of the update protocols' accuracy says that pessimistic querying protocols can guarantee a fixed inaccuracy value ϵ, regardless of the MO's speed. The optimistic querying protocol usually produces more inaccuracy. The time-based reporting protocol has low average inaccuracy at low movement speeds, whereas distance-based reporting protocol has constant average inaccuracy for all speed levels. In general, reporting protocols do not guarantee an inaccuracy threshold ϵ for MOs.

The performance of the combined location update protocol proposed by Leonhardi and Rothermel [LR01] was analyzed based on real position data. Its efficiency depends on the query rate and the parameter that determines whether the last MO's position stored on the server should be updated. At high query rates, it is less efficient than the distance-based reporting protocol; however, at low rates it requires far fewer update messages. In contrast to the reporting protocols, it provides a pre-defined bound ϵ for the MO's location inaccuracy.

4.1.3. Dead Reckoning Protocols

Now, we will consider dead-reckoning protocols in more detail. In [LNR02], the authors present an overview of various existing dead-reckoning protocols and classify them as follows (Figure 4.3).

Linear prediction: The most simple dead-reckoning protocol, which assumes that the MO moves linearly (e.g., a car on a highway). This protocol is easy to implement and usually requires fewer update messages than naïve update protocols.

Prediction with higher-order function: This protocol allows for the prediction of non-linear movements of the MO, such as curves or splines. The speed of the MO can be also calculated based on the measured acceleration.

Map-based dead-reckoning: This case assumes that the MO usually moves along a road network (e.g., a car driving through town). The aim of the map-based dead-reckoning protocols is to compare the MO's position with the map of the surrounding environment. Such maps can be obtained, for example, from navigation systems. The protocol must decide the direction in which the MO is most likely to move at every road crossing.

Map-based dead-reckoning with probabilities: This improved map-based dead-reckoning protocol uses probability information at road crossings. This method can be user-independent,

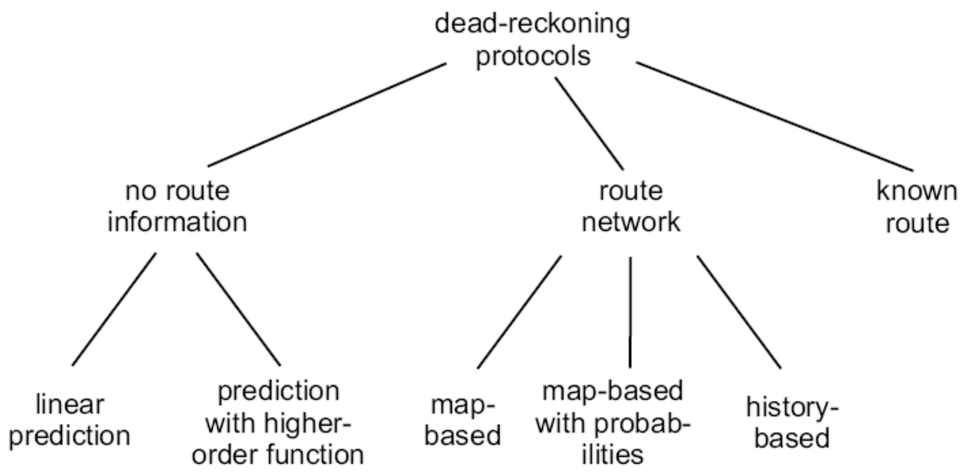


Figure 4.3.: Overview of different types of dead-reckoning protocols [LNR02]

if provided with information about the percentage of users who at crossing X select junction Y; or it can be user-specific, if provided with information about how often user U at crossing X selects junction Z. Of course, to be able to obtain such information we need to have a large spatial database included in the system model.

History-based dead-reckoning: This protocol is based on the assumption that the movement history of a given user (profile) is recorded over a long period of time. Since many users have a regular daily routine (e.g., driving to work every morning and home again in the evening), the performance of this protocol is similar to the map-based dead-reckoning with additional probability information. The resulting profile can either be user-specific or user-independent.

Dead-reckoning with known route: If the route of the given MO is pre-determined, then only its speed has to be considered, since it is already known which junction will be chosen by this MO at each crossing. In this case, the protocol operates just like the map-based dead-reckoning with ideal movement prediction, since the predicted MO's route matches its true route exactly.

4.1.4. Map-based Dead-Reckoning

In addition to an overview of the existing dead-reckoning approaches, a map-based dead reckoning algorithm was developed in [LNR02]. The proposed location update algorithm uses a map extracted from the automotive navigation system. The map will be interpreted as a graph, where each node represents a crossing, and the roads between the crossings are represented as graph edges. This graph-based representation helps to calculate the MO's position offline more precisely, with the most challenging part being in the right selection of edges after the crossings. The authors call the search for a right edge along the neighboring graph part forward-tracking and backward-tracking.

As the evaluation of [LNR02] shows, the proposed location update algorithm achieves a significant reduction in the number of update messages and thus reduces communication overhead between the MO and the server. However, it offers no way of obfuscating the exact location of the user, and also has the problem that the server that performs the calculations is a single TTP in the system, i.e., the proposed algorithm assumes that the server is always trustworthy. Since the server stores the exact position of the MO, the MO's position information can be revealed in the case of a compromised or malicious server.

4.1.5. Summary

The existing approaches in the field of location update algorithms cover many aspects of the optimization of LBS operation. In general, many of them achieve good results in optimizing the update messages. However, traditional approaches reducing single updates such as dead-reckoning are not applicable to our position sharing approach, since they are not suited to obfuscated positions. As illustrated in Figure 4.4, the calculation of the predicted position causes the resulting radius to double in size, since the MO can be originally located in any point of the previous obfuscation circle: obfuscated locations increase the prediction's deviation by $2 * r$. As a result, the position prediction becomes worthless and unhelpful. While the MO is still able to calculate his or her position, the server cannot do it.

Moreover, there are currently no approaches that would provide obfuscation-based location privacy, simultaneous usage of multiple location servers, and a reduction in the number of update messages all at the same time. In this work, we solve these problems in an integrated way that is specific to the basic position sharing approach introduced in Chapter 2.

In this chapter, we will describe how the required update message overhead can be significantly reduced: depending on the situation, up to 70-80% of communication overhead can be saved without violating the privacy guarantees provided or the precision levels of the MO's position.

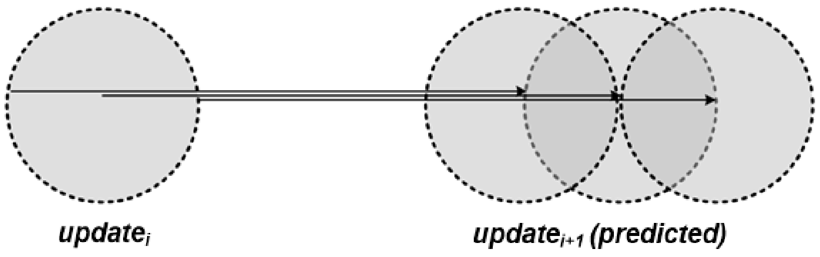


Figure 4.4.: Optimization of location updates: why dead reckoning is not applicable

4.2. Problem Statement

Next, we formulate the reduction of location updates as a constrained optimization problem.

The *optimization* goal is to reduce the total number of location update messages sent in the system. According to our system model (Figure 3.1), the communication overhead consists of messages being sent from MOs to LSs (denoted as the number of messages N^{MO-LS}) and from LSs to LBAs (N^{LS-LBA}). LBAs can subscribe to receive continuous MO's position updates from the LSs (with each update triggered by the LSs in this case); or they can pro-actively issue and process location-based queries to get the MOs' positions from the LSs. The *constraints* say that there should be no change of position precision ϕ_k as a result, as well as no reduction of the user's probabilistic guarantees $P_{\text{k,attack}}(\phi_{\text{k,attack}})$ of precision (privacy) levels.

We define the following as given:

- n location servers,
- the MO's previous consecutive precise position π_i, i.e., position defined before π_{i+1} (the algorithm is run on the MO side, which means that the MO's own precise positions are available),
- the MO's next consecutive precise position π_{i+1}, i.e., position defined after π_i,
- a master share s_0 generated for π_i,
- a set S^i of n refinement shares $s_1 \dots s_n$ generated for π_i,
- the probability distribution $P_k(\phi_k)$, which specifies the required probabilistic guarantees for each precision level ϕ_k.

Problem: Find the set of shares S^{i+1}_{opt}, where the concatenation of all shift vectors of S^{i+1}_{opt} must point to π_{i+1} according to the basic requirements of share generation (Algorithm 3, Section 2.4.2.1):

$$S^{i+1}_{opt} = \{s^{i+1}_0 \dots s^{i+1}_n\} \; : \; \sum_{k=0}^{n} s^{i+1}_k = \pi_{i+1}, \tag{4.1}$$

such that S^{i+1}_{opt} requires the minimal number of update messages, i.e., there is no other set of shares S^{i+1}_{other} that can produce a smaller number of update messages $N(S^{i+1}_{other})$ than

$N(S^{i+1}_{opt})$:

$$\forall S^{i+1}_{other},\ \ S^{i+1}_{other} \neq S^{i+1}_{opt} \ \ : \ \ N(S^{i+1}_{opt}) \leq N(S^{i+1}_{other}); \tag{4.2}$$

the set of shares S^{i+1}_{opt} must also satisfy the current user's privacy requirements, i.e., each further kth share must provide the pre-defined probabilistic guarantees of privacy levels $P_k(\phi_k)$:

$$\forall\ \phi_{\text{k,attack}} \ \ : \ \ P_k(\phi_k) > Pr[\phi_{\text{k,attack}} \leq \phi_{\text{k}}]; \tag{4.3}$$

finally, the precision ϕ_k of each imprecise position p^{i+1}_k derived by share fusion after obtaining the minimized set S^{i+1}_{opt} has to be the same as the precision of the corresponding imprecise position p^i_k constructed from the original set of shares S^i:

$$\forall S^{i+1}_k \in S^{i+1}_{opt},\ \ S^i_k \in S^i \ \ : \ \ \phi_k(p^{i+1}_k(S^{i+1}_k)) = \phi_k(p^i_k(S^i_k)) \tag{4.4}$$

In other words, S^{i+1} and S^i should differ in as few shares as possible, i.e., in S^{i+1} as many shares as possible should be reused from S^i.

Note that we do not assume that an MO's complete trajectory is available. We consider only the close sequential position updates. Hence, we cannot apply statistical analysis of the past positions and the respective parameters such as speed, and therefore we do not consider approaches for preserving privacy of a complete trajectory (see Section 2.7.7). Those approaches are independent from the method of share updating, i.e., they consider MO's position as a single entity and not as a combination of shares. Our goal is only to reduce the number of shares to be sent during neighboring consecutive updates without violating the precision levels and their probabilistic guarantees provided by the share generation algorithm.

4.3. Position Sharing Update Approaches

Usually, one position change corresponds to one update message from the MO to the LS, and one message from the LS to an LBA. Our basic approach allows for sharing user's position among multiple non-trusted LSs, but the price for that is the increased communication overhead. The reason for the increased overhead is that after each location update, an MO must send n messages with new position shares to n different LSs, while an LBA must receive k messages from k LSs in order to obtain the position of the precision level k that it is authorized to know. Thus, our goal is to send a smaller number of messages than n after each position change.

Depending on the movement scenario, different location update approaches can be beneficial. The key factor here is the relation between the distance traveled between two consecutive updates and the radii of obfuscation circles.

First, we consider *continuous* position updates, i.e., updates which are close to each other with update intervals of up to 15 seconds. The examples of such position update scenarios are navigation, way finding and tracking (Figure 4.5a).

Second, we consider *sporadic* position updates, i.e., updates which are distant from each other with update intervals from several minutes to more than one hour. Such updates would occur as a result of point-of-interest queries from the LBAs to the LSs (Figure 4.5b). Sporadic updates are hardly predictable, not frequent, and the distances between them are usually large.

In both cases described above, we assume that only a limited movement history is known and that no complete trajectory (i.e., movement history) is available.

Next, we will present different approaches suited for different relations of the travelled distance to the radius provided by the position share, namely, for (a) *continuous* position

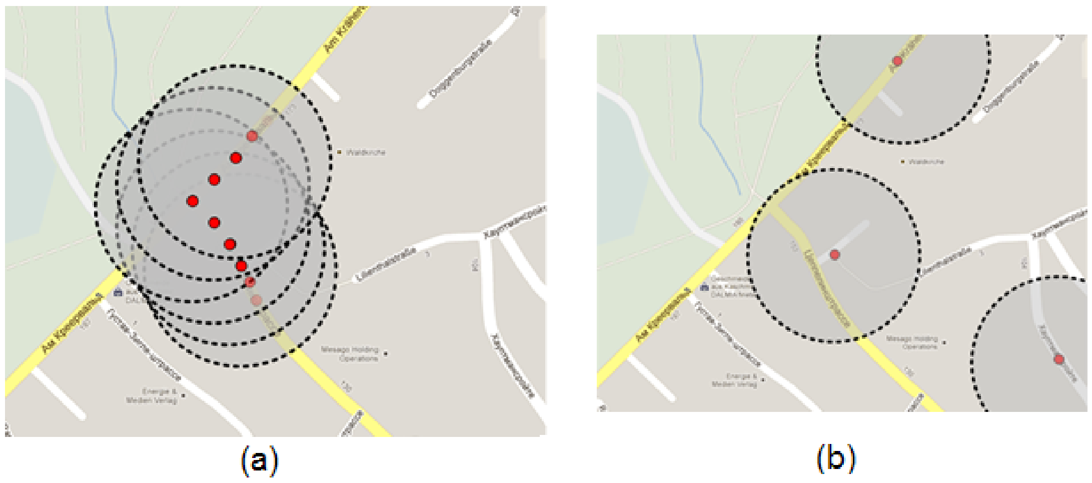


Figure 4.5.: Location updates optimization: (a) navigation scenario; (b) point-of-interest queries scenario

updates, (b) *sporadic* position updates and (c) *combined* position updates. Finally, we will integrate these approaches into a holistic location update algorithm.

In the following, we illustrate our location update approaches with the obfuscation circles based on the OSPS-ASO “a-priori” share generation and fusion. However, our location update approaches are also applicable to other share generation and fusion algorithms.

4.3.1. Position Sharing Update Approach 1: PSUA1

If the MO moves insignificantly or it does not move at all between two consecutive updates and remains inside circle c_k (and does not stay in c_{k+1}), then only the shares $s_{k+1} \ldots s_n$ have to be recalculated and sent to the LSs. For example, Figure 4.6 shows that the MO's movement does not intersect the innermost circle c_{n-1}. Therefore, only share s_n must be updated.

The pseudocode for our first position sharing update approach denoted as PSUA1 is presented in Algorithm 12. In order to find the last non-affected by the movement precision level k, we calculate the distance between the newly updated position π_{i+1} and the centers p_i^k of circles c_k generated for the previous position π_i (line 2). If this distance is smaller than the radius r_k, the user remained inside the circle c_k:

$$\text{distance}(\pi_{i+1}, p_i^k) \leq r_k \tag{4.5}$$

This condition needs to be checked for each circle starting from the smallest one c_{n-1}, until the condition is met or the master share's radius r_0 is exceeded.

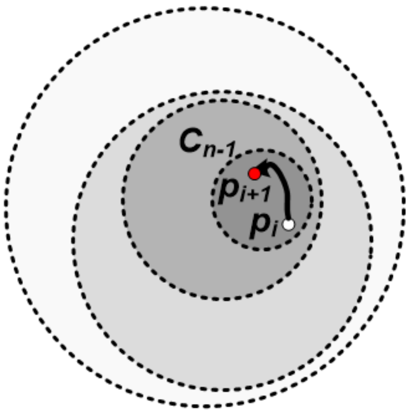


Figure 4.6.: PSUA1: little movement of MO; MO remains inside the innermost circle c_{n-1}

Algorithm 12 Location Update Algorithm: PSUA1

```
1: function $update\_shares\_1(\vec{\pi_i}, \vec{\pi_{i+1}}, n, \vec{s_0} \dots \vec{s_n})$
2: while distance$(\vec{\pi_{i+1}}, \vec{p_i^k}) \leq r_k$ do
3:     $k \leftarrow k - 1$
4: end while
5: $\vec{s_{k+1}} \dots \vec{s_n} \leftarrow$ regenerate_shares$(\vec{\pi_{i+1}}, k+1, n, \phi_{min}, \Delta_\phi)$
6: send$(\vec{s_{k+1}} \dots \vec{s_n})$
```

Having determined k, we can calculate the total number of update messages for PSUA1. Regarding the communication between the MO and the LSs, $n-k$ shares have to be re-generated (line 5) and re-sent (line 6) in PSUA1, The resulting saving rate is:

$$R^{MO-LS}_{\text{PSUA1}} = \frac{k}{n} \tag{4.6}$$

The number of saved messages is $N^{MO-LS}_{\text{PSUA1}} = k$.

It is more difficult to calculate the communication between LSs and LBAs, since it depends on the number of LBAs (which cannot be affected by the MO) and the required precision level. As defined in our problem statement (Section 4.2), the position update can be either queried by LBAs, or triggered by LSs for subscribed LBAs. We introduce the probability of an LBA receiving an update of position p_j, i.e., of getting shares $s_0 \dots s_j$ as $P_{upd}([s_0 \dots s_j])$; we will give an estimation of the P_{upd} probabilities later in Section 4.3.4 and Section 4.6.1. Thus, the saving rate in communication between LSs and the LBAs can be expressed as follows:

$$R^{LS-LBA}_{\text{PSUA1}} = \frac{\sum_{j=0}^{k} P_{upd}([s_0 \dots s_j])}{\sum_{j=0}^{n} P_{upd}([s_0 \dots s_j])} \tag{4.7}$$

The absolute number of saved messages is defined as the number of LBAs n_{LBA} multiplied by the sum of the update probability of the refinement shares which were not sent due to the optimization:

$$N^{LS-LBA}_{\text{PSUA1}} = n_{LBA} \cdot \sum_{j=0}^{k} P_{upd}([s_0 \dots s_j]) \tag{4.8}$$

4.3.2. Position Sharing Update Approach 2: PSUA2

If the MO moves fast or the update rate is very low, the new MO's master share can be located completely outside the previous master share, as depicted in Figure 4.7.

The condition of having no intersection between two consecutive master shares is:

$$\text{distance}(\pi_{i+1}, p_i^0) > 2 * r_0 \tag{4.9}$$

The main idea of our second position update approach (PSUA2) is that under the conditions of Equation 4.9, we can recalculate and update only the master share while keeping the refinement shares unchanged.

The pseudocode for PSUA2 is presented in Algorithm 13. If the condition of Equation 4.9 is met (line 2), only a new master share has to be generated (line 3) and sent (line 5) to the corresponding LSs, while the refinement shares $s_1 \dots s_n$ will remain the same without causing any inconsistency during their fusion. This is preserved by the fact that the shares are relative shift vectors, while the absolute coordinates are only contained in the master share s_0.

Algorithm 13 Location Update Algorithm: PSUA2

```
1: function $update\_shares\_2(\vec{\pi_i}, \vec{\pi_{i+1}}, n, \vec{s_0} \dots \vec{s_n})$
2: if distance$(\vec{\pi_{i+1}}, \vec{p_i^0}) > 2 * r_0$ then
3:     $\vec{s_0} = \vec{\pi_{i+1}} - \vec{\pi_i}$
4: end if
5: send$(\vec{s_0})$
```

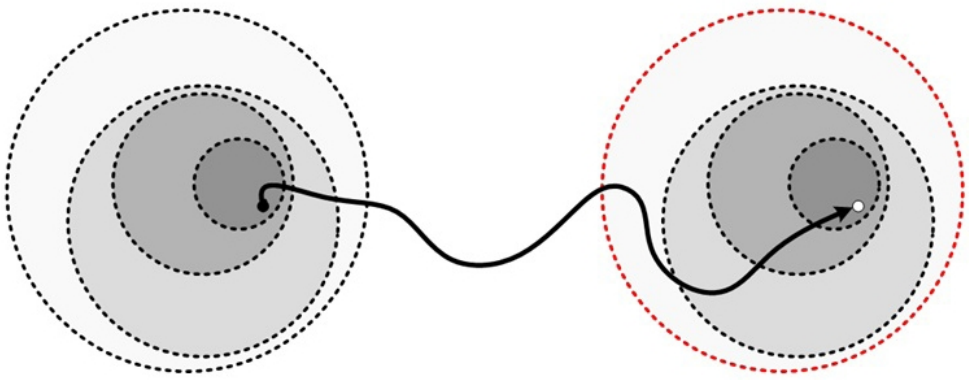


Figure 4.7.: PSUA2: large movement of MO; two consecutive master shares do not intersect

Next, we analyze the communication costs for PSUA2. Since only one share has to be updated, the number of sent messages between MO and LS is $N_{\text{PSUA2}}^{MO-LS} = 1$, and the saving rate is:

$$R_{\text{PSUA2}}^{MO-LS} = \frac{n-1}{n}, \tag{4.10}$$

whereas the message saving rate between LSs and LBAs is:

$$R_{\text{PSUA2}}^{LS-LBA} = \frac{\sum_{j=1}^{n} P_{upd}([s_j \dots s_n])}{\sum_{j=0}^{n} P_{upd}([s_j \dots s_n])} \tag{4.11}$$

The absolute number of messages between LSs and LBAs equals the number of LBAs, so $N_{\text{PSUA2}}^{LS-LBA} = n_{LBA}$, since the master share has to be re-sent to each LBA.

Note that since $N_{\text{PSUA2}}^{MO-LS} = 1$, PSUA2 can be considered as the optimal approach for the case when LS-LBA communication cost is ignored, e.g., when $P_{upd}(\dots)$ is negligibly small or cannot be estimated.

4.3.3. Position Sharing Update Approach 3: PSUA3

In special cases where neither PSUA1 nor PSUA2 is applicable, we apply non-optimized location updating called PSUA3. In PSUA1 and PSUA2, we considered that the new (obfuscated) position lies either completely inside the old master share or completely outside it. The next version of position sharing update approach (PSUA3) represents an intersection of two consecutive share sets, i.e., a situation when both conditions of Equation 4.5 for PSUA1 and Equation 4.9 for PSUA2 are not satisfied. This causes a degradation of privacy through reduction of the obfuscation area[1], i.e., it is more likely that the MO's actual position lies within the intersection of circles.

The pseudocode for PSUA3 is presented in Algorithm 14. All shares have to be re-generated (line 3) and sent (line 5), so that the total number of sent messages (without any messages

[1]For example, *Maximum Velocity Attack* [GDSB09] could be applied: if an attacker gets two successive master shares and their timestamps, he or she can predict a travelled distance d_{max} after a certain time.

saved) is:

$$N_{\text{PSUA3}} = n + (n_{LBA} \cdot \sum_{j=0}^{n} P_{upd}([s_j \dots s_n]) \tag{4.12}$$

Algorithm 14 Location Update Algorithm: PSUA3

```
1: function $update\_shares\_3(\vec{p_i}, \vec{p_{i+1}}, n, \vec{s_0} \dots \vec{s_n})$
2: if $\forall k \in \{1 \dots n-1\}$ : (distance($\vec{\pi_{i+1}}, \vec{p_i^0}$) $\le 2 * r_0$) & (distance($\vec{p_{i+1}}, \vec{p_i^k}$) $> r_k$) then
3:     $\vec{s_0} \dots \vec{s_n} \leftarrow regenerate\_all\_shares(\vec{p_{i+1}}, n, \phi_{min}, \Delta_\phi)$
4: end if
5: send($\vec{s_0} \dots \vec{s_n}$)
```

4.3.4. Estimations of Efficiency

Now, we will perform a preliminary analysis of the gain in efficiency that the two main proposed approaches can provide: PSUA2, corresponding to the location update strategy for sporadic updates, and PSUA1, corresponding to the location update strategy for frequent updates.

The total communication cost in both cases includes: (a) the cost of communication between the MO and the LSs; (b) the cost of communication between the LSs and the LBAs.

Consider a situation in which the LSs push location updates to an LBA subscribed to the given MO, when the observed positions of that MO are changing. Why is then one s_n update better than one s_0 update? Assume the following situation: $n = 4$; LBA has k shares, each k with equal probability 20%:

$$P_{upd}([s_k \dots s_n]) = 20\%, \quad k = 0 \dots n \tag{4.13}$$

For such a case, Figure 4.8 shows the rough estimation of LS-LBA communication cost depending on k, so that the privacy level k here corresponds to the number of shares being obtained by the LBA from k LSs.

If we only take into account communication between the mobile user and the LSs, we can state the following: The usage of PSUA2 will lead to the reduction of messages by $1 - 1/n = (n-1)/n$. For example, $n = 5$ leads to reduction by 80% (cf. Figure 4.8).

The possible limitations for PSUA2 include the situation when a new set of LSs is selected

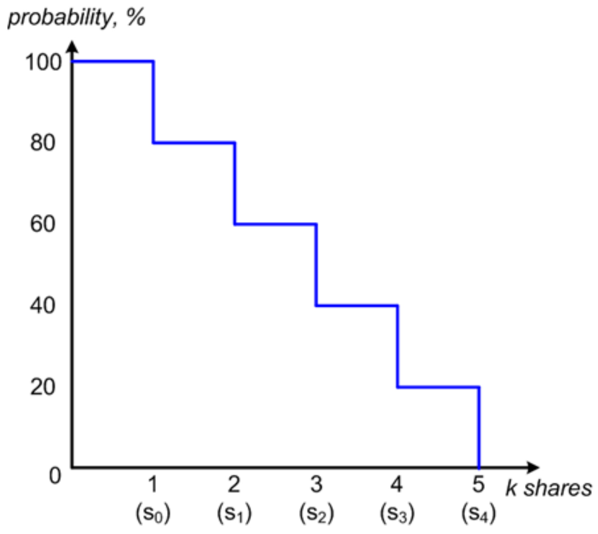


Figure 4.8.: Location updates optimization: estimation of communication cost LS-LBA

or the access rights are changed. In addition, the set of the MO's position shares ages and becomes more vulnerable with time. Using PSUA1, an ideal case we can achieve for MO-LS communication is the same message reduction as with PSUA2; in the worst case, the resulting reduction will be only $1/n$.

4.4. Optimized Location Update Algorithm

As we have shown, there are two optimized approaches for sending location updates (PSUA1 and PSUA2) and the non-optimized update approach (PSUA3). Since PSUA1 and PSUA2 produce varying numbers of messages in different cases, our goal now is to formally distinguish these cases and achieve the total communication cost as $N = min(N_{\text{PSUA1}}, N_{\text{PSUA2}})$.

Note that PSUA1 and PSUA2 cannot be further optimized, because the number of updates from MO to LSs triggered by PSUA2 is 1, which is the minimal number of messages which makes a position update possible. The number of updates from MO to LSs triggered by PSUA1 is $n-k$, since the precision levels $c_0 \dots c_k$ are not affected. If we send even one share less, it would mean that no correct refinement of precision levels possible at least for one of the levels $k+1 \dots n$ and thus the precision and privacy guarantees are violated.

Thus, we only need to select one of the approaches (PSUA1 or PSUA2), as it is not possible to further reduce the amount of messages within the approaches themselves. The main challenge is to find the condition defining the point where PSUA2 is better than PSUA1 in terms of the total number of messages required for location updates (including MO-LS and LS-LBA communication).

Therefore, before sending a location update, the MO determines the best way to do the update – namely, which one of the two optimized approaches can be applied and which one produces the minimal number of messages. To determine the break-even point between PSUA1 and PSUA2, we equate the absolute costs of MO-LS and LS-LBA for PSUA1 and PSUA2:

$$N_{\text{PSUA1}}^{MO-LS} + N_{\text{PSUA1}}^{LS-LBA} \stackrel{?}{=} N_{\text{PSUA2}}^{MO-LS} + N_{\text{PSUA2}}^{LS-LBA} \tag{4.14}$$

Next, the MO checks whether PSUA1 or PSUA2 should be applied for the kth precision level requested by the LBA for the given location update.

$$k + n_{LBA} \cdot \sum_{j=n-k+1}^{n} P_{upd}([s_j \dots s_n]) \stackrel{?}{=} 1 + n_{LBA} \tag{4.15}$$

Then, we check the conditions for PSUA1 (Equation 4.5) and PSUA2 (Equation 4.9). If they are satisfied, an optimized location update can be applied; otherwise, the naïve approach (PSUA3) should be used. In the end, we achieve a situation, such that any further reduction of update messages would reduce the delivered position precision, while any increase of update messages would increase the communication overhead without improving the pre-defined precision. The pseudo-code for the general location update algorithm is presented in Algorithm 15. Note that the find_optimal_k(...) function in line 2 finds k according to Equation 4.15.

As an example, we show the point where PSUA2 becomes more beneficial than PSUA1, if assumed that $n = 10$, $k = 1 \dots 10$, $n_{LBA} = 10$ (Figure 4.9). The number of required messages is heavily dependent on k. Another important factor is whether the updated share is the master share (PSUA2), which should be sent to every LBA after each update, or the updated shares are k refinement shares, which need to be re-sent in the case of PSUA1. The communication cost required by PSUA2 is constant, with only one message between the MO and the LSs, and the whole set of 10 (n) messages between LSs and LBAs. The cost of PSUA1 is lower for small k values, but it increases rapidly with both MO-LSs and LSs-LBAs communication increasing together with k. Thus, we can say that PSUA1 is more efficient

Algorithm 15 Location Update Algorithm

```
1: function $update\_shares(\vec{\pi_i}, \vec{\pi_{i+1}}, n, \vec{s_0} \dots \vec{s_n})$
2: $k \leftarrow$ find_optimal_k$(\vec{\pi_i}, \vec{\pi_{i+1}}, n, \vec{s_0} \dots \vec{s_n})$
3: if distance$(\vec{\pi_{i+1}}) < k * r_0/n$ then
4:     update_shares_1$(\vec{\pi_i}, \vec{\pi_{i+1}}, n, \vec{s_0} \dots \vec{s_k})$
5: else
6:     if distance$(\vec{\pi_i}, \vec{\pi_{i+1}}) > 2 * r_0$ then
7:         update_shares_2$(\vec{\pi_i}, \vec{\pi_{i+1}}, n, \vec{s_0} \dots \vec{s_k})$
8:     else
9:         update_shares_3$(\vec{\pi_i}, \vec{\pi_{i+1}}, n, \vec{s_0} \dots \vec{s_k})$
10:    end if
11: end if
```

for a number of refinement shares that does not exceed 4 out of 10, while PSUA2 is more beneficial for $k \geq 4$ (cf. Figure 4.9).

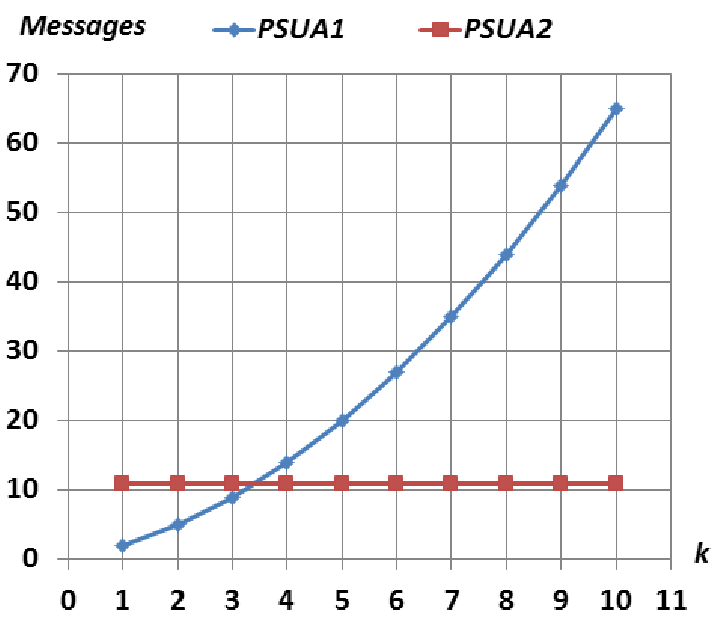


Figure 4.9.: The intersection of the curves is the point where PSUA2 starts to be more beneficial than PSUA1; the communication cost values are calculated for $n = 10$, $k = 1 \dots 10$, $n_{LBA} = 10$

4.5. Security of Location Updates

Thus far, we have extended our basic approach with a temporal dimension by considering multiple consecutive location updates and reduced the required communication overhead. Now, we consider the influence of location update algorithms and multiple updates on privacy.

4.5.1. Challenges of Consecutive Updates

The privacy guarantees provided by the basic position sharing approach introduced in Chapter 2 are valid for isolated position updates. In this chapter, we have already presented a location update algorithm which optimizes the communication overhead considering the last two consecutive position updates on-the-fly. However, the problem is that by acquiring a larger sequence of updates, an adversary is able to gain more information about the current MO's position than was originally intended to be revealed by the MO. The well-known adversarial method of gaining additional knowledge through consecutive updates is so-called *Maximum Velocity Attack* [GDSB09]. An adversary can apply it in cases where the approximate velocity of the MO is known, along with the MO's previous positions. Such information can either be inferred from previous updates, or based on observed statistics of similar MOs.

Knowing the time that has elapsed between the last updates and the maximum velocity of the MO, the maximum distance of the target MO from the last position can be calculated. Then an attacker can find the area that is reachable from the last known position in the elapsed time, given the maximum MO speed.

In Figure 4.10, we show the previous MO's update (obfuscation circle) on the left and the current update on the right. The larger circle's radius r_{max} is equal to the maximum distance that this MO could have traveled since the previous location update during the period of time between the two updates. Assuming the maximum possible velocity to be known, the adversary can determine the area covering the maximal possible traveled distance. The radius of the circle is the product of the maximum velocity and the time difference between the updates:

$$r_{max} = (t_i - t_{i-1}) * v_{max}, \tag{4.16}$$

$$\text{where}: \quad v_{max} = \frac{\text{distance}(p_{i-2}, p_{i-1}) + 2 * r_k}{t_{i-1} - t_{i-2}} \tag{4.17}$$

The area covered by the circle with r_{max} includes every point within the maximum reachable distance from every point in the last known location. By intersecting this area with the new obfuscation circle (center in p_i^k) provided by MO as the next update, the adversary can get an area that might be less than or equal to the area of the new update. Thus, we can check whether the current user's position is located in the overlapping area of both the circles, as shown in Figure 4.10. If the intersection is less than the area of the new update, then the attacker can conclude that the target MO is within the intersected area and not in the area outside of the intersection. Thus, the precision achieved is higher than the precision that was intended by the mobile user. In this case, the location privacy level of the MO is compromised by velocity based linkage attacks. Note that in this case, no additional information besides the positions and their corresponding timestamps is used.

4.5.2. Secure Location Updates

The *first privacy requirement* corresponding to our problem statement (Equation 4.4) is that the position update optimization must not reduce the obfuscation area, i.e., cause an

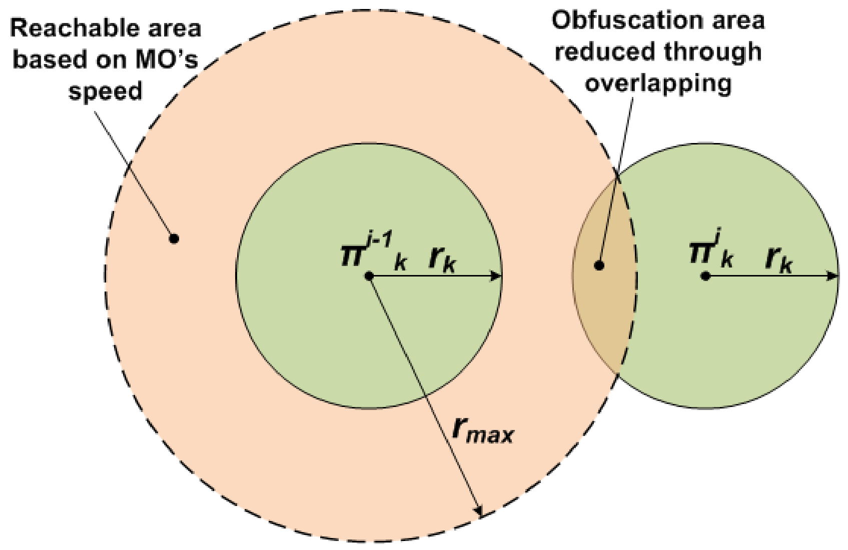

Reachable area based on MO's speed
Obfuscation area reduced through overlapping
π^{j-1}_k
r_k
π^j_k
r_k
r_{max}

Figure 4.10.: Velocity-based linkage attack: reachable area based on MO's speed and obfuscation area reduced through overlapping

undesired increase in position precision ϕ_k. Under this condition, we can state that the proposed location update algorithms do not reduce the number of shares, i.e., do not change the precision level available to the authorized LBAs. In other words, the smaller number of shares sent from the MO to the LS does not affect the number of shares provided to LBAs. Therefore, no change in precision occurs.

Until now, we have only considered the locations of updates. The problem is that if at least two previous consecutive updates with their timestamps are known, the obfuscation area can be reduced by applying the Maximum Velocity Attack described in the previous section, which limits the maximal possible travelled distance by the MO since the last update based on the MO's estimated maximal speed. To prevent this attack, we must omit the location update if the next obfuscation circle intersects with the maximal movement boundary. This is a well-known counter-measure against such an attack, described by Wernke et al. [WDR13]. We applied a similar defense against the Maximum Velocity Attack as already presented in the literature [GDSB09, WDR13]. We accomplish this by skipping position updates if they undermine the location privacy.

The general idea is that we must skip a position update if we see that the execution of the next update will make it impossible to satisfy the privacy requirement using spatial obfuscation alone. By skipping the scheduled update, we increase the area reachable from the last update, as it is illustrated in Figure 4.11. The new reachable area is depicted as an extended area resulting from increased time delay after an update was skipped. Obviously, the overlapping area between the new larger shared circle with r_{max} and the reachable area after skipping an update is larger than before. Thus, we achieve higher privacy by introducing more position imprecision.

We say that an update is secure if any point of the current c_k^i is reachable from the previous c_k^{i-1}. The time between two previous consecutive updates is used to determine the MO's maximal speed, and the reachability area for the next MO's position is then determined by using this speed estimation. We calculate the maximal possible distance that an MO can travel after the last update π_i with the interval $t_i - t_{i-1}$ as:

$$r_{max} = \frac{(t_i - t_{i-1}) \cdot (\text{distance}(\pi_{i-2}, \pi_{i-1}) + 2 * r_0)}{t_{i-1} - t_{i-2}} \tag{4.18}$$

Algorithm 16 shows how we make the decision whether to send the current update or skip it for security reasons. We assume that we know two previous updates in addition to

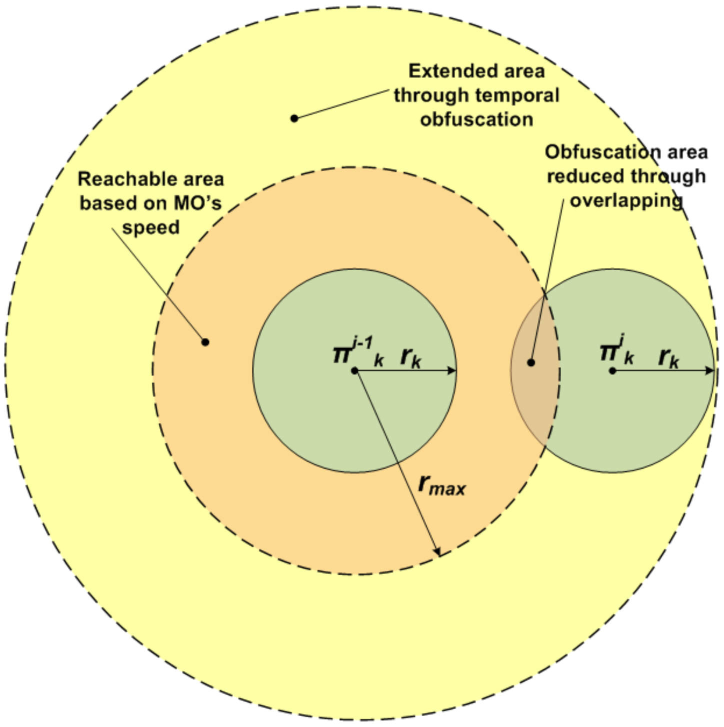

Extended area through temporal obfuscation
Obfuscation area reduced through overlapping
Reachable area based on MO's speed
π^{i-1}_k
r_k
π^i_k
r_k
r_{max}

Figure 4.11.: Skipped update's effect: extended area

the current update; this is the minimum number required to estimate the MO's speed. We assume that the current obfuscation level is k and it remains the same for all three of the updates.

Algorithm 16 Secure Location Update

```
1: function $update\_position(t_i, t_{i-1}, t_{i-2}, S_i, S_{i-1}, S_{i-2})$
2: $v_{max} = (\text{distance}(\pi_{i-2}, \pi_{i-1}) + 2 * r_k)/(t_{i-1} - t_{i-2})$
3: $r_{max} = (t_i - t_{i-1}) * v_{max}$
4: if $r_{max} > \text{distance}(\pi_{i-2}, \pi_{i-1})$ then
5:     send_to_LSs($s^i_0 \dots s^i_k$)
6: else
7:     skip_update($S_i$)
8: end if
```

First, we calculate the reachable radius r_{max} (lines 2-3). If this distance completely covers the obfuscation circle c^i_k of the current update s^i_k, we send the location update as usual (lines 4-5). If c^i_k is intersected and therefore reduced, we skip this update (lines 6-7).

The example in Figure 4.12 shows three consecutive updates, where any point within the last update's obfuscation circle c^i_k is reachable from the previous obfuscation circle c^{i-1}_k.

After applying this principle to the two scenarios tested in the previous section, we obtained

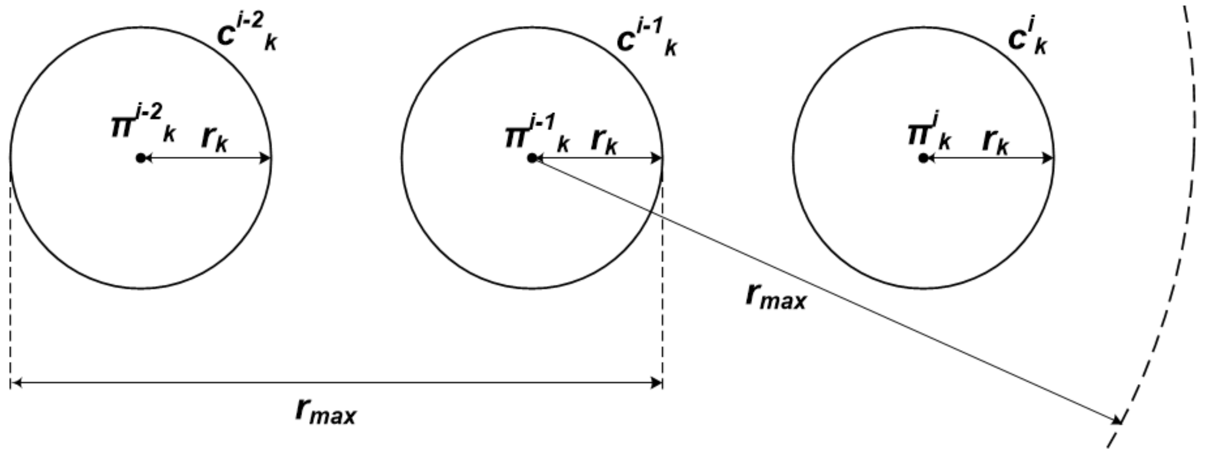


Figure 4.12.: Maximal movement boundary: estimation shows that any point within the obfuscation circle c_k^i is reachable from the previous obfuscation circle c_k^{i-1}

the following results. Since the updates are close to each other, 95-98% of updates are secure in the case of the continuous location updates scenario and 75-83% are secure in the case of the sporadic updates scenario. The reason for such high rates is that the estimated speed in the second case is usually low due to the long delays between updates, while the master share radius r_0 is larger.

The *second privacy requirement* corresponding to our problem statement (Equation 4.3) is the probabilistic metric $P_k(\phi_k)$. Having the separate location updates represented by the known obfuscation circle c_k^i, the remaining shares $s_k^i \dots s_n^i$ and the corresponding refined obfuscation circles $c_k^i \dots c_n^i$ within c_k^i remain unknown for an attacker, even under the assumption that he or she knows the share generation algorithm. The randomness of share generation is preserved by the unchanged share generation algorithms (see Algorithms 2-8), so that the probabilistic guarantees of privacy levels $P_k(\phi_k)$ are the same as in the basic position sharing approaches.

4.5.3. Further Privacy Challenges: Discussion

Additionally, it would be possible to consider a pdf based on the movement correlation instead of one based on intersection of the binary movement boundary. A similar idea of applying the probabilistic prediction based on linear dead reckoning was proposed by Zhou and Chirikjian [ZC03]. After determining such a pdf based on the previous updates record, it will be necessary to convolute the share generation pdf and the pdf of the next predicted position. However, such an approach requires analysis of the trajectory correlation pattern and is beyond the scope of this work.

Next, we consider the situation when an attacker knows the indices of the updated shares even without having their content. We analyze whether he or she can gain additional knowledge about π by considering such a situation with regard to PSUA1, PSUA2 and PSUA3 (selected according to Algorithm 15).

In PSUA2, only the master share is being updated by the MO. Since the master share s_0 is known to everyone according to the basic assumptions of the position sharing approach, the knowledge that it was updated does not affect privacy.

In PSUA3, all the shares $s_0 \ldots s_n$ are updated, therefore, knowing their indices also does not provide any new information. An attacker can only come to the conclusion that the MO did not move further than $2 * r_0$ from his or her previous position. This geometrical area is large enough, especially if taken into account that the previous MO's position remains unknown to the attacker as well.

In PSUA1, the set of updated shares can be different than the set of k shares that an attacker is already authorized to know. Whereas the precision levels are revealed in ascending order starting from p_0, the shares in PSUA1 are updated in descending order starting from s_n. We illustrate this principle in Figure 4.13, which shows that if $n = 10$, up to 3 shares can be updated in terms of PSUA1 (according to the calculations presented earlier in Figure 4.9). The gray area indicates the limits of the MO's movement according to an update of the last three shares s_8, s_9, s_{10}.

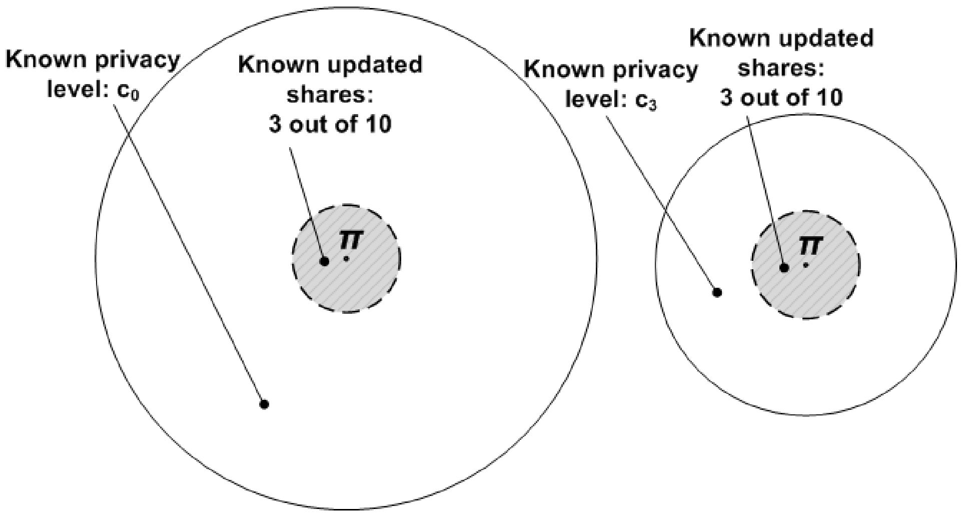

Known privacy level: c_0
Known updated shares: 3 out of 10
π
Known privacy level: c_3
Known updated shares: 3 out of 10
π

Figure 4.13.: Knowing indices of updated shares in PSUA1: examples for $n = 10$, 3 last shares are updated.

However, note that the location of π is unknown to an attacker, and therefore the location of the gray area is also unknown to him or her. Assuming that the previous trajectory is not known, or that it is not correlated, the basic property of the position sharing approach is that π is distributed uniformly within the obfuscation area of c_k known to the attacker. Thus, the knowledge about the indices of the updated unknown shares does not increase the probability of π being located within any sub-area of c_k. Since the precise user position is distributed uniformly within c_k (whether the previous one π_i or the next one π_{i+1}), even if π_i and π_{i+1} are highly correlated, an attacker cannot know exactly where they lie within c_k. This statement is correct if we use OSPS-ASO "a-priori", OSPS FSO or CSPS share generation as the basic location privacy approach. However, the problem of knowing the indices of updated shares is another argument against use of the "a-posteriori" share generation algorithm version; the reason is that one of the points π_i, π_{i+1} is likely to be in the center of c_k.

The speed-based movement restrictions do not restrict the obfuscation area in this case either, since the start and the end of the MO's movement are imprecise. Thus, the only privacy-critical issue is that by knowing the previous updates in PSUA1, an attacker may guess the length and the direction of the movement *within* the kth circle of π_{i+1}. This is the situation where updates of shares corresponding to kth (or less than kth) privacy level show a movement pattern, and only the unknown shares ($k+1$th or further) are updated. Figure 4.14 shows how trajectory analysis combined with knowledge of the indices of updated shares may help to predict the location of π_i within c_k. In such a case, since the MO's further movement is predictable, there will be a higher probability that π is located on the direction-dependent (here: right) side of c_k. However, it is important that the whole c_k area is still reachable since π remains unknown (note that in Figure 4.14 the exact location of π is depicted only for illustrative purposes). This case is beyond the scope of this work, since it is related to the analysis of trajectory patterns. We refer to the work of Riaz et al. [RDR15], where the corresponding analysis was conducted using a system model similar to our system model and having applied the principles of distributed position sharing.

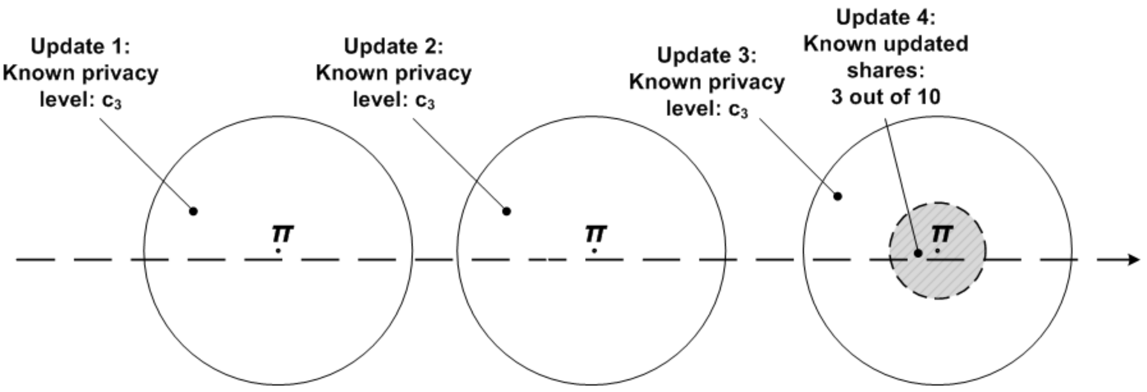

Update 1: Known privacy level: c3
Update 2: Known privacy level: c3
Update 3: Known privacy level: c3
Update 4: Known updated shares: 3 out of 10
π
π
π

Figure 4.14.: Knowing indices of updated shares and the trajectory pattern: c_0 is updated during the first 3 updates, while the 4th update affects only the last three shares (s_{10}, s_9, s_8); $n = 10$

4.6. Evaluation

Next, we present the evaluation of our optimized share placement algorithm. We start with an evaluation of the communication cost of the algorithm, before we compare the achieved probabilistic guarantees of precision levels provided by our approach to the ones of the basic approach.

4.6.1. Evaluation Setup

We evaluated our location update algorithms by using the open database of real location data called GeoLife [ZXM10]. The GeoLife data sets include daily routine trajectories such as the route to and from work, or hiking and biking trips. Most trajectories were recorded at intervals of 1-5 seconds or 5-10 meters (cf. Figure 4.15).

We selected two datasets which represent different scenarios: continuous updates (with update intervals of up to 15 seconds) and sporadic updates (with update intervals from several minutes to more than one hour). Then we evaluated the location update behavior and measured the rate of message reduction.

For the evaluation, we assumed that the probability for each precision level ϕ_i of MO's position being required for the LBAs is equal (see Section 4.3.4, Figure 4.8). If the probability of getting various precision levels is not equal, its function will remain similar to the one shown in Figure 4.8, i.e., it will be decreasing, with first shares always requested more often than the latter shares. The exception would be extreme cases such as those where only precision p_{const} is always required. Thus, in order to reduce the communication cost, the

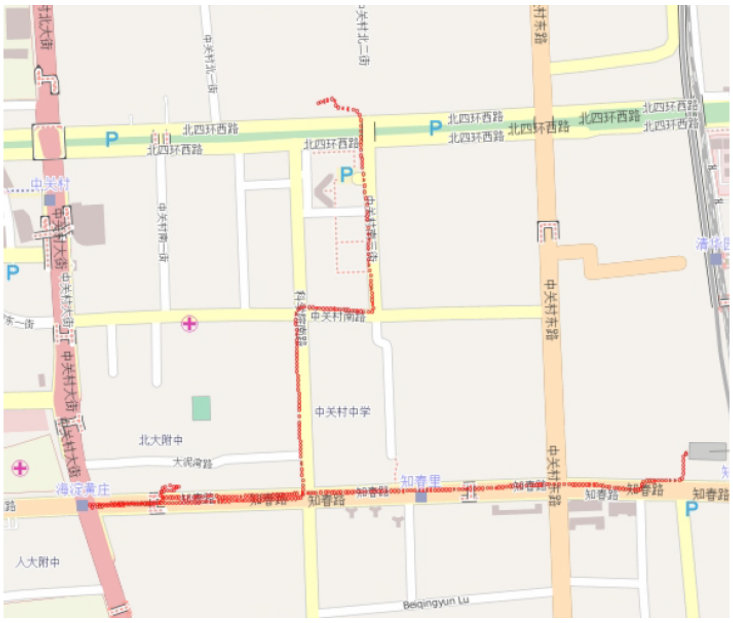


Figure 4.15.: An example of Geolife location data with position updates marked by red dots; small time intervals (1-5 seconds) and small distances (5-10 meters) between updates [Hae12]

latter shares should be updated if possible.

In this case, the number of messages sent to the LSs required for the LBAs linearly increases for each subsequent requested precision level ϕ_i. For example, to get the lowest precision, an LBA needs to obtain only a single share s_0 (1 message), while the highest precision requires obtaining the complete set of shares from s_0 to s_n (n messages). In this latter case, the probability of getting a share s_i linearly decreases for each subsequent i:

$$P_{upd}([s_i]) = 1 - \frac{i}{n} \tag{4.19}$$

4.6.2. Communication Cost after Reduction of Updates

First, we evaluate the number of messages sent through continuous updates for the number of LSs $n = 5$ and the number of LBAs $n_{LBA} = 5$. Note that the GeoLife data set represent very frequent position updates with only tens of meters distances between the updates. Therefore, in order to make possible selection of both PSUA1 and PSUA2, we selected radii of obfuscation circles comparable to the distances between the updates: radii $r_0 = 5$ m (Figure 4.16) and $r_0 = 50$ m (Figure 4.17). The horizontal axis depicts the sequential numbers of position updates. For the given setup, the naïve approach would generate 20060 messages. The optimized approach generates 11520 for $r_0 = 5$ m and 3856 messages for $r_0 = 50$ m, which equals a saving rate of 42.6% and 80.8% correspondingly. The PSUA2 is often selected in Figure 4.16 due to the small radius, while using a larger radius, as shown in Figure 4.17, only PSUA1 is selected.

Next, we analyze the scenario of sporadic updates for radii $r_0 = 50$ m (Figure 4.18) and $r_0 = 100$ m (Figure 4.19), where the naïve approach would generate 420 messages; $n = 5$; $n_{LBA} = 5$. The most commonly selected solution is PSUA2. Sometimes PSUA3 is selected due to master share intersections with $r_0 = 100$. As a result, the optimized location update algorithm generates 154 for $r_0 = 50$ and 224 messages $r_0 = 100$ (a saving rate of 63.3% and 46.7% respectively).

We also analyzed the interdependency between the saving rate and the radius r_0 of the master share (see Figure 4.20). For continuous (frequent) updates, a larger r_0 usually causes fewer updates, since the movements are smaller compared to the obfuscation circles. For sporadic updates, this tendency is similar in the long term, yet for a significant range of r_0 values (for the given data set: between 0 and 300 m), a smaller r_0 leads to less intersections and, therefore, to a smaller number of update messages.

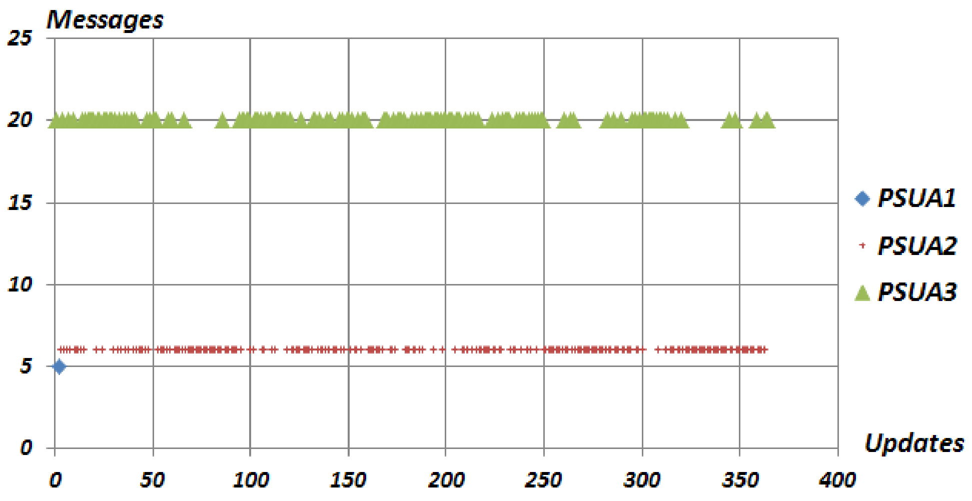


Figure 4.16.: Continuous location updates with radius $r_0 = 5$ m; $n = 5$; $n_{LBA} = 5$

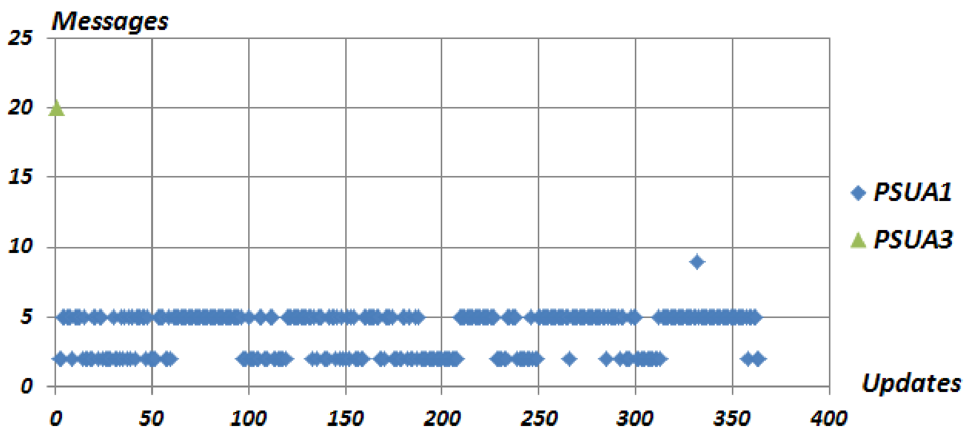


Figure 4.17.: Continuous location updates with radius $r_0 = 50$ m; $n = 5$; $n_{LBA} = 5$

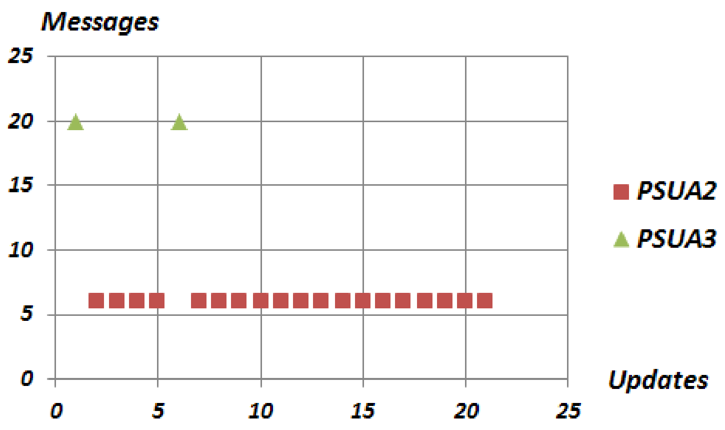


Figure 4.18.: Sporadic location updates with radius $r_0 = 50$ m; $n = 5$; $n_{LBA} = 5$

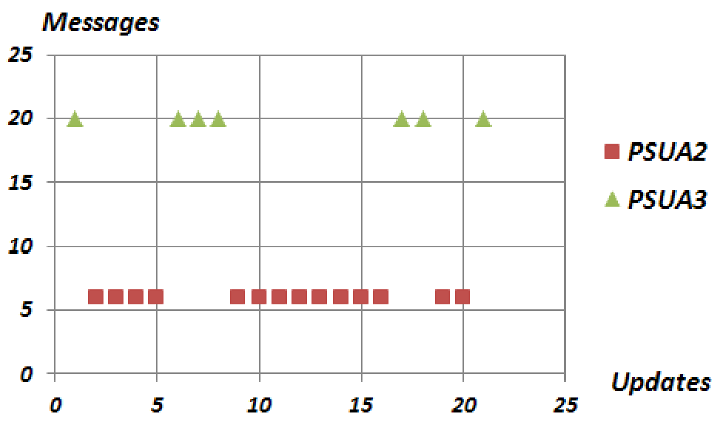


Figure 4.19.: Sporadic location updates with radius $r_0 = 100$ m; $n = 5$; $n_{LBA} = 5$

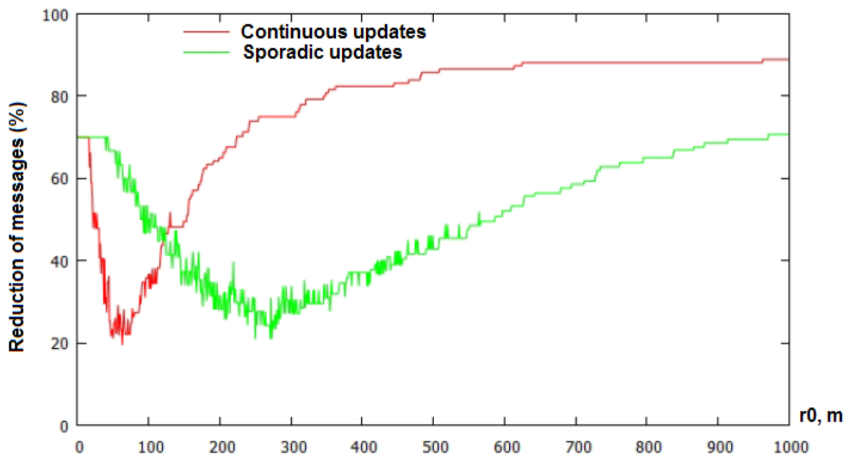


Figure 4.20.: Saving rate (reduction of update messages, %) depending on radius r_0 of the master share; $n = 5$; $n_{LBA} = 5$ [Hae12]

4.6.3. Probabilistic Guarantees of Privacy Levels after Position Update Optimization

Next, we analyze the probabilistic guarantees of privacy levels after the location update algorithms have been applied (Figure 4.21). The first (dashed) curve represents the probabilistic privacy guarantees of the OSPS-ASO “a-priori” share generation algorithm as measured in Section 2.5.3 and shown initially in Figure 2.22b. The second (solid) curve represents the probabilistic privacy guarantees of the same share generation algorithm after PSUA1 approach was used in order to reduce the number of updates. Note that we do not need to evaluate the effect of PSUA2, since it does not affect the generation of the refinement shares.

In order to simulate the effect of PSUA1 on the share generation algorithm, we re-generated $\{k+1\dots n\}$ shares after selecting each vector set and relocating π within the respective limits, where k was selected randomly in $\{1\dots n-1\}$. We made 100 runs of the Monte Carlo method for $n = 5$, same as in Section 2.5.3 (with r_0 represented as 1.0, since the absolute radius value has no influence on the $P_{k,10\%}$ values as we do not consider map knowledge in this comparison). We can see that the resulting probability values $P_{k,10\%}$ are very close for both OSPS-ASO “a-priori” with and without PSUA1, while the small deviation is due to probabilistic nature of the Monte Carlo method. These results show that the re-generation of a subset of shares does not change the stochastic properties of a share generation algorithm, if the re-generation was done according to the same algorithm which was applied to generate the initial set of shares.

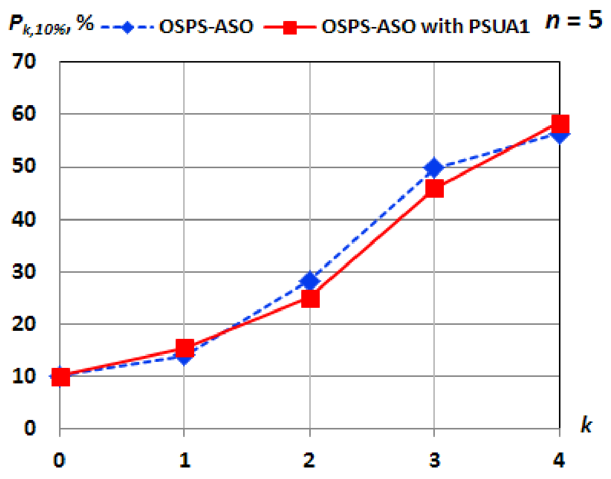


Figure 4.21.: Probabilistic guarantees of privacy levels $P_{k,10\%}$ of OSPS-ASO “a-priori” with and without PSUA1 applied; $n = 5$, 100 runs of the Monte Carlo method

4.7. Conclusion

In this chapter, we have extended our main position sharing approach in order to address further real-world challenges. The approach was adapted to the scenario of multiple consecutive updates, as opposed to the scenario of single isolated snapshot updates assumed in the previous chapters.

We have optimized the communication cost of location updates by sending only a subset of k shares out of total number of shares n, while preserving the required level of position obfuscation. The main factors contributing to the cost reduction are: 1) the radii of the obfuscation circles, 2) number of privacy levels n, and 3) the travelled distance between the last two updates. In most cases, we can achieve significant communication cost reduction by sending fewer messages than through the naïve basic position sharing approach.

To have a comprehensive determination of communication cost, we take into account the messages sent from MOs to LSs together with the messages sent from LSs to LBAs. The proposed location update algorithm selects the optimal strategy by analyzing the current distance and obfuscation parameters. In the evaluation section, we showed that the proposed location update approach minimizes the communication cost by saving up to 80% of messages without reducing the user’s location privacy.

Chapter 5 Conclusion

In this section, we summarize our work and outline possible directions for future research that could further improve our position sharing approach.

5.1. Summary

Both personal privacy and cyber security are currently attracting increasing attention, and location privacy is one of the things of most concern to many people.

In this work, we have presented a novel approach for preserving the location privacy of a mobile user of location-based services. The main idea of our approach is to distribute position information among n location servers of independent service providers. In order to hide the user's exact location, the user's position information is represented as an obfuscation circle. The size of the obfuscation circle defines the user's location privacy level, which can be changed if location-based applications are provided with additional pieces of information called position shares. The position shares are represented as randomly generated shift vectors. To increase the precision of the user's position, the position shares shift the center of the current obfuscation circle and reduce the circle size.

In our approach, we distinguish such steps as share generation, share update and share fusion. We proposed four different versions of the share generation algorithm: three of them apply to the open space scenario (i.e., without map knowledge available in the system), while the fourth approach modification is designed to preserve the desired privacy levels assuming that map-based knowledge is available to an adversary.

We have analyzed the probabilistic guarantees of the user's privacy levels provided by our share generation algorithms by using probabilistic privacy metrics. We measured probabilities of each precision level that can be gained by an adversary, assuming that he or she already knows k shares $(0 \leq k \leq n)$. Among the open space based share generation algorithms, the first version (OSPS-ASO “a-posteriori”) provides equal but high predictability of the target user’s position for each k, while the second algorithm (OSPS-ASO “a-priori”) guarantees high security for smaller k’s but increases the probability of a user’s precise position being guessed for higher k values. The third algorithm (OSPS-FSO) provides a close to uniform distribution of probability of the user being located within each kth obfuscation circle. However, this uniform distribution of probability is only possible if the order of share fusion is pre-defined and fixed (unlike in the first two share generation algorithm versions).

On the basis of our third share generation algorithm for open space, we have proposed an algorithm for constrained space (CSPS). This algorithm withstands an attacker’s use of map knowledge that could significantly reduce the original obfuscation area, i.e., location privacy level. We overcome this problem by increasing the radii of the obfuscation circles far enough to attain the desired size of the resulting obfuscation area.

We also improved our approach with regard to the placement of shares among the location servers by taking into consideration the trustworthiness of the servers. The availability of information about the servers’ trustworthiness allows mobile users to balance risk by placing more position information on more trusted servers.

Finally, we proposed an optimized location update algorithm, which reduces communication cost in the case of continuous position updates by up to 70-80%. This cost reduction is achieved by selecting the minimal required subset of shares that can be updated without affecting the precision levels and privacy guarantees of the user’s position information.

We can summarize the most significant properties of the proposed position sharing approach as follows:

- Graceful degradation of privacy levels in cases where some position shares became unavailable;
- No need for a trusted third party;
- Flexible management of privacy levels by mobile users.

5.2. Outlook

There are still some aspects of our position sharing approach that could be improved. Possible improvements and future research steps include the following:

The algorithm versions with adaptive increase of the obfuscation circle's radius (OSPS-FSO and CPSP) can be easily modified to provide k-anonymity guarantees by adjusting the obfuscation area not only until it achieves a required size but until it covers k users. The precondition here is the availability of information about all the neighboring users. Such precondition contradicts our current system model, which assumes that the position shares are generated by the MO with no global view of the system.

Boolean values assigned to map regions in CSPS can also be replaced by a probability estimation in the range of [0;1], if a database with the corresponding statistics is available. Such region-specific probabilities could be based on a concrete database or on general probabilistic estimations. However, this will not significantly change the principles of our adjustment algorithm. A challenge would be to consider different maps, with low and high percentages of areas where the user can be located, and special cases. In other words, diverse levels of privacy sensitivity can be considered for different map regions. The theoretical basis for formal definition and processing of a non-binary map representation is given in [ACG09]. An example of map representation with diverse probabilities assigned to map regions and objects is shown in Figure 5.1.

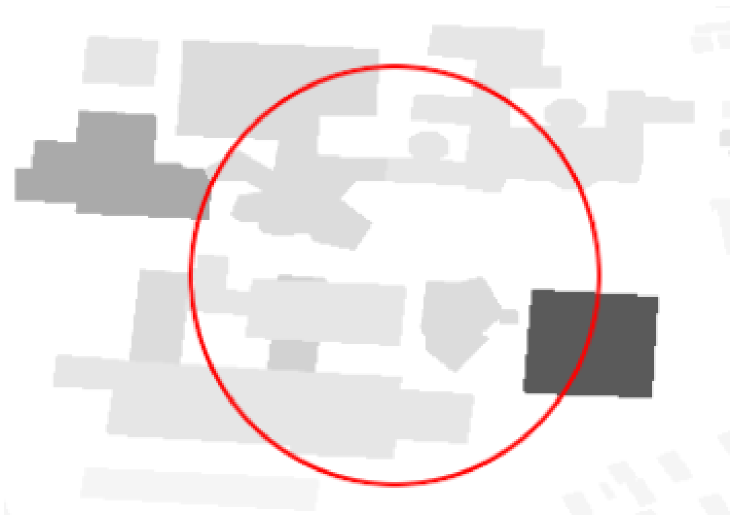


Figure 5.1.: Map representation with different probabilities assigned to different map objects [Pau11]

Also, the user may want to secure not only snapshot and continuous positions but a complete published trajectory. This is a challenge, since we would need to hide the start point and the end point as well as to obfuscate the whole route.

Furthermore, if we assume that additional contextual information such as user's interests, advanced history of events and traces are known to an attacker, the appropriate countermeasures would need to be applied.

It is also possible to use the "a-priori" and "a-posteriori" methods in a random way, for example, by adding a random variable to select which of those two methods is used for each position update (50% each on average). This would significantly undermine an attacker's ability to compute the pdf while preserving the ASO (any share order) property.

Bibliography

[ABN08] Abul, Osman ; Bonchi, Francesco ; Nanni, Mirco: Never Walk Alone: Uncertainty for Anonymity in Moving Objects Databases. In: *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, 2008, S. 376–385 (cited on pages 86, 87, and 184)

[ACD+07] Ardagna, C.A. ; Cremonini, M. ; Damiani, E. ; De Capitani di Vimercati, S. ; Samarati, P.: Location Privacy Protection Through Obfuscation-based Techniques. In: *Proc. of the 21st Annual IFIP WG 11.3 Working Conference on Data and Applications Security* Bd. 4602, 2007, S. pages 47–60 (cited on pages 16, 81, and 183)

[ACG09] Ardagna, Claudio A. ; Cremonini, Marco ; Gianini, Gabriele: Landscape-aware location-privacy protection in location-based services. In: *Journal of Systems Architecture* 55 (2009), April, S. 243–254. – ISSN 1383–7621 (cited on pages 88, 163, and 184)

[Alb03] Albrecht, Peter: Risk Based Capital Allocation / Sonderforschungsbereich 504, Universitat Mannheim. 2003 (03-02). – Forschungsbericht. – . (cited on pages 116 and 117)

[Ama14] Amazon Web Services LLC: *Amazon*. http://aws.amazon.com, October 2014 (cited on page 18)

[AP98] Arora, Shalini ; Puri, M. C.: A variant of time minimizing assignment problem. In: *European Journal of Operational Research* 110 (1998), Nr. 2, S. 314–325 (cited on page 117)

[BBC14] BBC: *Q and A: NSA's Prism internet surveillance scheme*. http://www.bbc.com/news/technology-23051248, October 2014 (cited on page 17)

[BKS10] Brush, A. J. B. ; Krumm, John ; Scott, James: Exploring end user preferences for location obfuscation, location-based services, and the value of location. In: *Proceedings of the 12th ACM international conference on Ubiquitous computing*. New York, NY, USA : ACM, September 2010 (Ubicomp '10). – ISBN 978–1–60558–843–8, 95–104 (cited on page 16)

[BLPW08] Bamba, Bhuvan ; Liu, Ling ; Pesti, Peter ; Wang, Ting: Supporting Anonymous Location Queries in Mobile Environments with PrivacyGrid. In: *Proceeding of the 17th international conference on World Wide Web (WWW '08)*. New York, NY, USA : ACM, 2008. – ISBN 978–1–60558–085–2, S. 237–246 (cited on page 79)

[BMW+09] Bettini, Claudio ; Mascetti, Sergio ; Wang, Xiaoyang S. ; Freni, Dario ; Jajodia, Sushil: Anonymity and Historical-Anonymity in Location-Based Services. In: *Privacy in Location-Based Applications*, 2009, S. 1–30 (cited on pages 91, 92, and 184)

[BS04] Beresford, Alastair R. ; Stajano, Frank: Mix Zones: User Privacy in Location-aware Services. In: *PerCom Workshops*, 2004, S. 127–131 (cited on pages 73, 74, and 183)

[Bun14] Bundesministerium für Verkehr und digitale Infrastruktur: *Telekommunikationsgesetz*. http://www.bmvi.de/SharedDocs/DE/Anlage/Digitales/telekommunikationsgesetz-2012.pdf, October 2014 (cited on page 17)

[CB04] Chun, Brent N. ; Bavier, Andy: Decentralized Trust Management and Accountability in Federated Systems. In: *Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 9 - Volume 9*. Washington, DC, USA : IEEE Computer Society, 2004 (HICSS '04). – ISBN 0–7695–2056–1, S. 90279.1– (cited on page 18)

[CC05] Chan, Chao-Wen ; Chang, Chin-Chen: A scheme for threshold multi-secret sharing. In: *Applied Mathematics and Computation* 166 (2005), Nr. 1, S. 1–14 (cited on page 89)

[CM11] CHOW, Chi-Yin ; MOKBEL, Mohamed F.: Trajectory privacy in location-based services and data publication. In: *SIGKDD Explorations* 13 (2011), Nr. 1, S. 19–29 (cited on page 90)

[CML06] CHOW, Chi-Yin ; MOKBEL, Mohamed F. ; LIU, Xuan: A peer-to-peer spatial cloaking algorithm for anonymous location-based service. In: *GIS '06: Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*. New York, NY, USA : ACM, 2006. – ISBN 1–59593–529–0, S. 171–178 (cited on pages 76, 77, and 183)

[CZBP06] CHENG, Reynold ; ZHANG, Yu ; BERTINO, Elisa ; PRABHAKAR, Sunil: Preserving User Location Privacy in Mobile Data Management Infrastructures. In: *6th Workshop on Privacy Enhancing Technologies* Bd. 4258/2006, Springer Berlin / Heidelberg, 2006, S. 393–412 (cited on pages 81, 82, and 183)

[DBS10] DAMIANI, Maria L. ; BERTINO, Elisa ; SILVESTRI, Claudio: The PROBE Framework for the Personalized Cloaking of Private Locations. In: *Transactions on Data Privacy* 3 (2010), August, Nr. 2, 123–148. http://dl.acm.org/citation.cfm?id=1824401.1824404. – ISSN 1888–5063 (cited on pages 88, 89, and 184)

[DF03] DOBSON, J. E. ; FISHER, P. F.: Geoslavery. In: *Technology and Society Magazine, IEEE* 22 (2003), Nr. 1, 47–52. http://dx.doi.org/10.1109/mtas.2003.1188276. – DOI 10.1109/mtas.2003.1188276 (cited on page 17)

[DK05] DUCKHAM, Matt ; KULIK, Lars: A Formal Model of Obfuscation and Negotiation for Location Privacy. In: *Proceedings of the International Conference on Pervasive Computing (Pervasive 2005)*, 2005, S. 152–170 (cited on pages 82, 83, and 183)

[DK06] DUCKHAM, Matt ; KULIK, Lars: Location privacy and location-aware computing. In: *Dynamic and mobile GIS: investigating changes in space and time* Bd. 3, CRC Press, Boca Rator, FL, 2006, S. 35–51 (cited on page 17)

[DND07] DJORDJEVIC, Ivan ; NAIR, Srijith K. ; DIMITRAKOS, Theodosis: Virtualised Trusted Computing Platform for Adaptive Security Enforcement of Web Services Interactions. In: *ICWS*, IEEE Computer Society, 2007, S. 615–622 (cited on page 22)

[DP12] DINI, Gianluca ; PERAZZO, Pericle: Uniform obfuscation for location privacy. In: *Proceedings of the 26th Annual IFIP WG 11.3 conference on Data and Applications Security and Privacy*. Berlin, Heidelberg : Springer-Verlag, 2012 (DBSec'12). – ISBN 978–3–642–31539–8, 90–105 (cited on pages 83, 84, and 183)

[DSR11] DÜRR, Frank ; SKVORTSOV, Pavel ; ROTHERMEL, Kurt: Position Sharing for Location Privacy in Non-trusted Systems. In: *Proceedings of the 9th IEEE International Conference on Pervasive Computing and Communications (PerCom 2011)*. Seattle, USA : IEEE, March 2011, S. 189–196 (cited on pages 18, 20, 21, 28, 32, 34, 49, 94, 96, 97, 110, 114, and 181)

[DWSR10] DÜRR, Frank ; WERNKE, Marius ; SKVORTSOV, Pavel ; ROTHERMEL, Kurt: Towards a Position Sharing Approach for Location-based Services. In: *Proceedings of the W3C Workshop on Privacy for Advanced Web APIs*, Online, July 2010, S. 1–3 (cited on pages 20 and 97)

[Eck87] ECKHARDT, Roger: Stan Ulam, John von Neumann, and the Monte Carlo Method. In: *Los Alamos Science* (1987), S. 131–143 (cited on page 48)

[Egl90] EGLESE, R. W.: Simulated annealing: A tool for operational research. In: *European Journal of Operational Research* 46 (1990), June, Nr. 3, 271-281. http://ideas.repec.org/a/eee/ejores/v46y1990i3p271-281.html (cited on page 107)

[Ela14] ELASTICHOSTS LTD: *ElasticHosts Ltd*. http://www.elastichosts.com, October 2014 (cited on page 18)

[Esr98] ESR, I. ; ENVIRONMENTAL SYSTEMS RESEARCH INSTITUTE, INC. (Hrsg.): *ESRI Shapefile Technical Description*. . : Environmental Systems Research Institute, Inc., jul 1998. http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf (cited on page 57)

[Eyo08] EYOB, Ephrem: *Social Implications of Data Mining and Information Privacy: Interdisciplinary Frameworks and Solutions*. Hershey, PA : Information Science Reference - Imprint of: IGI Publishing, 2008. – ISBN 1605661961, 9781605661964 (cited on page 17)

[Fac15] FACEBOOK PLACES: *Facebook*. www.facebook.com/places, January 2015 (cited on page 14)

[Fou14] FOURSQUARE: *Foursquare*. www.foursquare.com, October 2014 (cited on page 14)

[Gam88] GAMBETTA, Diego: Can We Trust Trust? In: *Trust: Making and Breaking Cooperative Relations*, Basil Blackwell, 1988, S. 213–237 (cited on page 104)

[GDSB09] GHINITA, Gabriel ; DAMIANI, Maria L. ; SILVESTRI, Claudio ; BERTINO, Elisa: Preventing velocity-based linkage attacks in location-aware applications. In: *GIS '09: Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. New York, NY, USA : ACM, 2009. – ISBN 978–1–60558–649–6, S. 246–255 (cited on pages 85, 86, 87, 142, 147, 149, and 184)

[GG03] GRUTESER, Marco ; GRUNWALD, Dirk: 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)*. New York, NY, USA : ACM, 2003, S. 31–42 (cited on page 85)

[GHS08] GUTSCHER, Andreas ; HEESEN, Jessica ; SIEMONEIT, Oliver: Possibilities and Limitations of Modeling Trust and Reputation. In: *WSPI* Bd. 332, CEUR-WS.org, 2008 (CEUR Workshop Proceedings), S. 1–12 (cited on pages 101 and 104)

[GKS07] GHINITA, Gabriel ; KALNIS, Panos ; SKIADOPOULOS, Spiros: PRIVE: anonymous location-based queries in distributed mobile systems. In: *WWW '07: Proceedings of the 16th international conference on World Wide Web*. New York, NY, USA : ACM, 2007. – ISBN 978–1–59593–654–7, S. 371–380 (cited on pages 78, 79, and 183)

[GL05] GEDIK, Bugra ; LIU, Ling: Location Privacy in Mobile Systems: A Personalized Anonymization Model. In: *ICDCS*, 2005, S. 620–629 (cited on pages 76, 77, and 183)

[GL08] GEDIK, B. ; LIU, Ling: Protecting Location Privacy with Personalized k-Anonymity: Architecture and Algorithms. In: *IEEE Transactions on Mobile Computing* 7 (2008), January, Nr. 1, S. 1–18. `http://dx.doi.org/10.1109/TMC.2007.1062`. – DOI 10.1109/TMC.2007.1062. – ISSN 1536–1233 (cited on page 76)

[Goo14] Google Cloud Platform: *Google*. https://cloud.google.com/compute/, October 2014 (cited on page 18)

[Goo15] Google Now: *Google*. http://google.com/landing/now, January 2015 (cited on page 14)

[GS97] Grinstead, C. M. ; Snell, J. L.: *Introduction to probability*. American Mathematical Society, 1997 (cited on pages 62, 63, 64, and 183)

[Gut06] Gutscher, Andreas: Coordinate transformation - a solution for the privacy problem of location based services? In: *Proceedings of 20th International Parallel and Distributed Processing Symposium IPDPS 2006*, 2006, S. 7pp. (cited on pages 83, 84, and 183)

[Gut07] Gutscher, A.: A Trust Model for an Open, Decentralized Reputation System. In: *Proceedings of the Joint iTrust and PST Conferences on Privacy Trust Management and Security (IFIPTM 2007)*, 2007, S. 213–237 (cited on page 104)

[Gut09] Gutscher, Andreas: Reasoning with Uncertain and Conflicting Opinions in Open Reputation Systems. In: *Electronic Notes in Theoretical Computer Science* 244 (2009), August, 67–79. `http://dx.doi.org/10.1016/j.entcs.2009.07.039`. – DOI 10.1016/j.entcs.2009.07.039. – ISSN 1571–0661 (cited on page 101)

[Hae12] Haenle, Simon: *Location Update Algorithms for Position Sharing*, Universität Stuttgart, Fakultät Informatik, Elektrotechnik und Informationstechnik, Germany, Diplomarbeit, October 2012. – 1–92 S. (cited on pages 20, 129, 155, 158, and 186)

[HGH+08] Hoh, Baik ; Gruteser, Marco ; Herring, Ryan ; Ban, Jeff ; Work, Daniel ; Herrera, Juan-Carlos ; Bayen, Alexandre M. ; Annavaram, Murali ; Jacobson, Quinn: Virtual trip lines for distributed privacy-preserving traffic monitoring. In: *Proceeding of the 6th international conference on Mobile systems, applications, and services (MobiSys '08)*. New York, NY, USA : ACM, 2008. – ISBN 978–1–60558–139–2, S. 15–28 (cited on page 86)

[HGXA07] Hoh, Baik ; Gruteser, Marco ; Xiong, Hui ; Alrabady, Ansaf: Preserving Privacy in GPS Traces via Uncertainty-aware Path Cloaking. In: *CCS '07:*

Proceedings of the 14th ACM conference on Computer and communications security. New York, NY, USA : ACM, 2007. – ISBN 978–1–59593–703–2, S. 161–171 (cited on page 86)

[Hoy12] HOYO, Daniel del: *Probabilistic map representation using GeoTools*. Studienarbeit: Universität Stuttgart, Institut für Parallele und Verteilte Systeme, Verteilte Systeme, May 2012 (cited on page 20)

[Hu82] HU, T. C.: *Combinatorial algorithms*. Reading, MA : Addison-Wesley, 1982 (cited on pages 104 and 106)

[HU90] HOPCROFT, John E. ; ULLMAN, Jeffrey D.: *Introduction To Automata Theory, Languages, And Computation*. 1st. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 1990. – ISBN 020102988X (cited on page 105)

[Ins14] INSTAMAPPER LLC: *InstaMapper LLC*. http://www.instamapper.com/, October 2014 (cited on page 14)

[JI02] JØSANG, A. ; ISMAIL, R.: The beta reputation system. In: *Proceedings of the 15th Bled Electronic Commerce Conference*, 2002, S. 1–14 (cited on page 104)

[KBR05] KINATEDER, Michael ; BASCHNY, Ernesto ; ROTHERMEL, Kurt: Towards a Generic Trust Model - Comparison of Various Trust Update Algorithms. In: *iTrust*, 2005, S. 177–192 (cited on pages 101 and 104)

[KC01] KILLMANN, Frank ; COLLANI, Elart von: A Note on the Convolution of the Uniform and Related Distributions and Their Use in Quality Control. In: *Economic Quality Control* 16 (2001), January, Nr. 1, S. 17–41. http://dx.doi.org/10.1515/eqc.2001.17. – DOI 10.1515/eqc.2001.17. – ISSN 0940–5151 (cited on page 62)

[Kru09] KRUMM, John: A survey of computational location privacy. In: *Personal and Ubiquitous Computing* 13 (2009), August, Nr. 6, S. 391–399 (cited on pages 90 and 92)

[KV06] KORTE, Bernhard ; VYGEN, Jens: *Combinatorial Optimization: Theory and Algorithms*. 3rd. Germany : Springer, 2006 http://www.springer.com/math/numbers/book/978-3-540-71843-7 (cited on pages 104, 105, and 107)

[KVD04] Kotz, S. ; van Dorp, J.R.: *Beyond Beta: Other Continuous Families Of Distributions With Bounded Support And Applications*. World Scientific, 2004. – ISBN 9789812561152 (cited on page 63)

[KYS05] Kido, H. ; Yanagisawa, Y. ; Satoh, T.: An anonymous communication technique using dummies for location-based services. In: *Proceedings of the International Conference on Pervasive Services (ICPS '05)*, 2005, S. 88–97 (cited on pages 71, 72, and 183)

[LDR08] Lange, Ralph ; Dürr, Frank ; Rothermel, Kurt: Online trajectory data reduction using connection-preserving dead reckoning. In: *Proceedings of the 5th Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*. ICST, Brussels, Belgium, Belgium : ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008 (Mobiquitous '08). – ISBN 978–963–9799–27–1, 52:1–52:10 (cited on pages 132 and 185)

[LHW07] Lee, Jae-Gil ; Han, Jiawei ; Whang, Kyu-Young: Trajectory clustering: a partition-and-group framework. In: *SIGMOD Conference*, 2007, S. 593–604 (cited on page 86)

[LLLZ09] Lee, Ken C. K. ; Lee, Wang-Chien ; Leong, Hong V. ; Zheng, Baihua: OPAQUE: Protecting Path Privacy in Directions Search. In: *ICDE*, 2009, S. 1271–1274 (cited on page 86)

[LLV07] Li, Ninghui ; Li, Tiancheng ; Venkatasubramanian, S.: t-Closeness: Privacy Beyond k-Anonymity and l-Diversity. In: *Proceedings of the IEEE 23rd International Conference on Data Engineering (ICDE 2007)*, 2007, S. 106–115 (cited on page 80)

[LNR02] Leonhardi, Alexander ; Nicu, Christian ; Rothermel, Kurt: A Map-based Dead-reckoning Protocol for Updating Location Information. In: *Proceedings of the 2nd International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing (IPDPSWPIM 2002)*, 2002, S. 1–11 (cited on pages 133, 134, and 185)

[LR01] Leonhardi, Alexander ; Rothermel, Kurt: A Comparison of Protocols for Updating Location Information. In: *Baltzer Cluster Computing Journal*

(2001), January, 355–367. http://www.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=ART-2001-12&engl= (cited on pages 130, 131, 132, and 185)

[LSTL13] Li, Ming ; Salinas, Sergio ; Thapa, Arun ; Li, Pan: n-CD: A geometric approach to preserving location privacy in location-based services. In: *INFOCOM, 2013 Proceedings IEEE*, 2013. – ISSN 0743–166X, S. 3012–3020 (cited on page 82)

[LW66] Lawler, E. L. ; Wood, D. E.: Branch-And-Bound Methods: A Survey. In: *Operations Research* 14 (1966), Nr. 4, 699–719. http://dx.doi.org/10.2307/168733. – DOI 10.2307/168733. – ISSN 0030364X (cited on page 106)

[Mat00] Mathews, Paul: The Circular Normal Distribution. In: *Mathews Malnar and Bailey, Inc.* (2000), S. 1–20 (cited on page 64)

[Mau96] Maurer, Ueli M.: Modelling a Public-Key Infrastructure. In: *ESORICS*, 1996, S. 325–350 (cited on page 104)

[MBW+09] Mascetti, Sergio ; Bettini, Claudio ; Wang, Xiaoyang S. ; Freni, Dario ; Jajodia, Sushil: ProvidentHider: An Algorithm to Preserve Historical k-Anonymity in LBS. In: *Mobile Data Management*, 2009, S. 172–181 (cited on pages 80 and 85)

[MCA06] Mokbel, Mohamed F. ; Chow, Chi-Yin ; Aref, Walid G.: The new Casper: query processing for location services without compromising privacy. In: *Proceedings of the 32nd international conference on Very large data bases (VLDB '06)*, VLDB Endowment, 2006, S. 763–774 (cited on pages 75, 76, and 183)

[MDKG05] Marias, G.F. ; Delakouridis, C. ; Kazatzopoulos, L. ; Georgiadis, P.: 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)*. Washington, DC, USA : IEEE Computer Society, June 2005, S. 614–620 (cited on page 89)

[MFB+11] Mascetti, Sergio ; Freni, Dario ; Bettini, Claudio ; Wang, X. S. ; Jajodia, Sushil: Privacy in geo-social networks: proximity notification with untrusted service providers and curious buddies. In: *The VLDB Journal* 20 (2011), August, Nr. 4, S. 541–566. – ISSN 1066–8888 (cited on page 71)

[MKGV07] Machanavajjhala, Ashwin ; Kifer, Daniel ; Gehrke, Johannes ; Venkitasubramaniam, Muthuramakrishnan: L-diversity: Privacy beyond k-anonymity. In: *ACM Transactions on Knowledge Discovery from Data* 1 (2007), Nr. 1, S. 3. http://dx.doi.org/http://doi.acm.org/10.1145/1217299.1217302. – DOI http://doi.acm.org/10.1145/1217299.1217302. – ISSN 1556–4681 (cited on pages 79, 80, and 183)

[MN88] Mazzola, J. B. ; Neebe, A. W.: Bottleneck generalized assignment problems. In: *Engineering Costs and Production Economics* 14 (1988), Nr. 1, S. 61–65 (cited on page 117)

[Mok07] Mokbel, Mohamed F.: Privacy in Location-Based Services: State-of-the-Art and Research Directions. In: *MDM*, 2007, S. 228 (cited on pages 14, 91, and 181)

[MT90] Martello, Silvano ; Toth, Paolo: *Knapsack Problems: Algorithms and Computer Implementations*. New York, NY, USA : John Wiley & Sons, Inc., 1990. – ISBN 0–471–92420–2 (cited on pages 104 and 106)

[NWvL07] Neisse, R. ; Wegdam, M. ; van Sinderen, M.J. ; Lenzini, G.: Trust Management Model and Architecture for Context-Aware Service Platforms. In: *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS* Bd. 4804. Berlin : Springer Verlag, November 2007 (Lecture Notes in Computer Science), 1803–1820 (cited on page 104)

[OXL$^+$08] Ouyang, Yi ; Xu, Yurong ; Le, Zhengyi ; Chen, Guanling ; Makedon, Fillia: Providing location privacy in assisted living environments. In: *Proceedings of the 1st international conference on PErvasive Technologies Related to Assistive Environments (PETRA '08)*. New York, NY, USA : ACM, 2008. – ISBN 978–1–60558–067–8, S. 1–8 (cited on pages 75 and 183)

[Pau11] Paul, Andreas: *Visualisierung von Kartenobjekten mit GeoTools*. Studienarbeit: Universität Stuttgart, Institut für Parallele und Verteilte Systeme, Verteilte Systeme, June 2011 (cited on pages 20, 42, 163, 182, and 186)

[Pav08] Pavlovic, Dusko: Dynamics, robustness and fragility of trust. In: *CoRR* abs/0808.0732 (2008) (cited on page 117)

[PL11] PALANISAMY, Balaji ; LIU, Ling: MobiMix: Protecting location privacy with mix-zones over road networks. In: *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering*. Washington, DC, USA : IEEE Computer Society, 2011 (ICDE '11). – ISBN 978–1–4244–8959–6, S. 494–505 (cited on pages 73, 74, and 183)

[PRB08] PARESCHI, Linda ; RIBONI, Daniele ; BETTINI, Claudio: Protecting Users' Anonymity in Pervasive Computing Environments. In: *PerCom*, 2008, S. 11–19 (cited on page 72)

[Pri97] PRITCHARD, C.L.: *Risk management*. ESI International, 1997 https://books.google.co.in/books?id=ZJLuoq-xgvMC. – ISBN 9781890367060 (cited on page 117)

[Pri14] PRIVACY RIGHTS CLEARINGHOUSE: *Privacy Rights Clearinghouse*. http://www.privacyrights.org/data-breach, October 2014 (cited on page 17)

[PRRJ06] POTLAPALLY, N.R. ; RAVI, S. ; RAGHUNATHAN, A. ; JHA, N.K.: A study of the energy consumption characteristics of cryptographic algorithms and security protocols. In: *Mobile Computing, IEEE Transactions on* 5 (2006), Feb, Nr. 2, S. 128–143. http://dx.doi.org/10.1109/TMC.2006.16. – DOI 10.1109/TMC.2006.16. – ISSN 1536–1233 (cited on page 70)

[PSD15] PERAZZO, Pericle ; SKVORTSOV, Pavel ; DINI, Gianluca: On Designing Resilient Location-Privacy Obfuscators. In: *The Computer Journal* (2015), February. http://dx.doi.org/10.1093/comjnl/bxv009. – DOI 10.1093/comjnl/bxv009 (cited on page 83)

[PZ02] PENSKY, Marianna ; ZAYED, Ahmed: Density Deconvolution of Different Conditional Distributions. In: *Annals of the Institute of Statistical Mathematics* 54 (2002), Nr. 3, 701-712. http://EconPapers.repec.org/RePEc:spr:aistmt:v:54:y:2002:i:3:p:701-712 (cited on pages 65 and 66)

[Rac15] RACKSPACE US INC.: *The Rackspace Cloud*. http://www.rackspace.com, January 2015 (cited on page 18)

[RDR15] RIAZ, Zohaib ; DÜRR, Frank ; ROTHERMEL, Kurt: Optimized Location Update Protocols for Secure and Efficient Position Sharing. In: *Proceedings of the*

2nd International Conference on Networked Systems: NetSys 2015; Cottbus, Germany, March 9-13, 2015, IEEE Computer Society, March 2015, S. 1–8 (cited on page 153)

[RM03] Rao, Bharat ; Minakakis, Louis: Evolution of Mobile Location-based Services. In: *Communications of the ACM* 46 (2003), December, Nr. 12, S. 61–65. – ISSN 0001–0782 (cited on page 14)

[RPB08] Riboni, Daniele ; Pareschi, Linda ; Bettini, Claudio: Privacy in Georeferenced Context-aware Services: A Survey. In: *Proceedings of the 1st International Workshop on Privacy in Location-Based Applications*, 2008, S. 151–172 (cited on page 71)

[Sch11] Schembera, Björn: *Platzierungsoptimierung für vertrauliche Verwaltung der verteilten Positionsinformationen*, Universität Stuttgart, Fakultät Informatik, Elektrotechnik und Informationstechnik, Germany, Diplomarbeit, May 2011. – 1–104 S. (cited on pages 20 and 99)

[SDFMB08] Solanas, Agusti ; Domingo-Ferrer, Josep ; Martínez-Ballesté, Antoni: Location Privacy in Location-Based Services: Beyond TTP-based Schemes. In: *PiLBA*, 2008, S. 127–131 (cited on pages 90, 91, and 184)

[SDR12] Skvortsov, Pavel ; Dürr, Frank ; Rothermel, Kurt: Map-aware Position Sharing for Location Privacy in Non-trusted Systems. In: *Proceedings of the 10th International Conference on Pervasive Computing (Pervasive 2012)*. Newcastle, UK : Springer, June 2012, S. 388–405 (cited on pages 19, 20, 21, 94, 96, 97, 110, and 114)

[SGI09] Shankar, Pravin ; Ganapathy, Vinod ; Iftode, Liviu: Privately querying location-based services with SybilQuery. In: *UbiComp*, 2009, S. 31–40 (cited on pages 72, 73, and 183)

[Sha79] Shamir, Adi: How to share a secret. In: *Communications of the ACM* 22 (1979), Nr. 11, S. 612–613. – ISSN 0001–0782 (cited on page 89)

[SNE06] Steiniger, Stefan ; Neun, Moritz ; Edwardes, Alistair: *Foundations of Location Based Services Lesson 1 CartouCHe 1- Lecture Notes on LBS, V. 1.0*. 2006 (cited on page 14)

[SO09] SINGHAL, Anoop ; OU, Xinming: Techniques for enterprise network security metrics. In: *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*. New York, NY, USA : ACM, 2009 (CSIIRW '09). – ISBN 978–1–60558–518–5, 25:1–25:4 (cited on page 104)

[SSDF08] SOLANAS, Agusti ; SEBÉ, Francesc ; DOMINGO-FERRER, Josep: Micro-aggregation-based heuristics for p-sensitive k-anonymity: one step beyond. In: *Proceedings of the 2008 international workshop on Privacy and anonymity in information society (PAIS '08)*. New York, NY, USA : ACM, 2008. – ISBN 978–1–59593–965–4, S. 61–69 (cited on page 80)

[Ste63] STEPHENS, M. A.: Random Walk on a Circle. In: *Biometrika* 50 (1963), Dezember, Nr. 3/4, 385+. http://dx.doi.org/10.2307/2333907. – DOI 10.2307/2333907. – ISSN 00063444 (cited on page 66)

[Sto01] STORER, J.A.: *An Introduction to Data Structures and Algorithms*. Birkhäuser Boston, 2001 (Progress in Computer Science and Applied Logic Series). http://books.google.de/books?id=S-tXjl1hsUYC. – ISBN 9780817642532 (cited on page 106)

[TA10] TALUKDER, Nilothpal ; AHAMED, Sheikh I.: Preventing multi-query attack in location-based services. In: *Proceedings of the third ACM conference on Wireless network security*. New York, NY, USA : ACM, 2010 (WiSec '10). – ISBN 978–1–60558–923–7, 25–36 (cited on page 78)

[The14] THE WASHINGTON POST: *Here's everything we know about PRISM to date*. http://www.washingtonpost.com/blogs/wonkblog/wp/2013/06/12/heres-everything-we-know-about-prism-to-date/, October 2014 (cited on page 17)

[TM08] TERROVITIS, Manolis ; MAMOULIS, Nikos: Privacy Preservation in the Publication of Trajectories. In: *9th International Conference on Mobile Data Management (MDM '08)*, 2008, S. 65–72 (cited on page 86)

[Tra14] TRACE4YOU: *FALCOM*. http://www.trace4you.com, October 2014 (cited on page 14)

[Usp37] USPENSKY, James V.: *Introduction to Mathematical Probability*. McGraw-Hill, 1937 (cited on page 63)

[VM03] VENTER, Gary G. ; MAJOR, John A.: Allocating Capital By Risk Measures: A Systematic Survey. In: *Guy Carpenter Views*, Guy Carpenter, August 2003, S. 1–7 (cited on pages 116 and 117)

[VMG+01] VIRRANTAUS, K. ; MARKKULA, J. ; GARMASH, A. ; TERZIYAN, V. ; VEIJALAINEN, J. ; KATANOSOV, A. ; TIRRI, H.: Developing GIS-supported location-based services. In: *Web Information Systems Engineering, 2001. Proceedings of the Second International Conference on* Bd. 2, 2001, S. 66–75 (cited on pages 14 and 16)

[WDR12] WERNKE, Marius ; DÜRR, Frank ; ROTHERMEL, Kurt: 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 2012)*. Lugano, Switzerland : IEEE, March 2012, S. 153–161 (cited on pages 89, 90, 96, 100, 101, 102, and 184)

[WDR13] WERNKE, Marius ; DÜRR, Frank ; ROTHERMEL, Kurt: PShare: Ensuring location privacy in non-trusted systems through multi-secret sharing. In: *Pervasive and Mobile Computing* (2013), Nr. 0, -. `http://www.sciencedirect.com/science/article/pii/S1574119213000229`. – ISSN 1574–1192 (cited on page 149)

[Wei02] WEICKER, Karsten: *Evolutionäre Algorithmen*. Stuttgart : Teubner, 2002 (cited on pages 107, 108, 118, and 184)

[Wes67] WESTIN, Alan: *Privacy and Freedom*. New York : New Jork Atheneum, 1967 (cited on page 16)

[Wik14] WIKILEAKS: *WikiLeaks Archives*. https://wikileaks.org/, October 2014 (cited on page 17)

[WL09] WANG, Ting ; LIU, Ling: From data privacy to location privacy. (2009), April, 217–247. `http://portal.acm.org/author_page.cfm?id=1326035`. ISBN 978–0–387–88734–0 (cited on page 90)

[WLFW06] WONG, Raymond Chi-Wing ; LI, Jiuyong ; FU, Ada Wai-Chee ; WANG, Ke: (alpha, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing. In: *KDD*, 2006, S. 754–759 (cited on page 80)

[WSDR14] WERNKE, Marius ; SKVORTSOV, Pavel ; DÜRR, Frank ; ROTHERMEL, Kurt: A Classification of Location Privacy Attacks and Approaches. In: *Personal and Ubiquitous Computing (Special Issue on Security and Trust in Context-Aware Systems)* 18 (2014), Nr. 1, 163–175. http://dx.doi.org/10.1007/s00779-012-0633-z. – DOI 10.1007/s00779–012–0633–z (cited on pages 20, 21, 71, 92, 93, 94, 95, 97, and 184)

[XCa14] XCALIBRE COMMUNICATIONS LTD: *Flexiscale*. http://flexiscale.com, October 2014 (cited on page 18)

[YJHL08] YIU, Man L. ; JENSEN, Christian S. ; HUANG, Xuegang ; LU, Hua: SpaceTwist: Managing the Trade-Offs Among Location Privacy, Query Performance, and Query Accuracy in Mobile Services. In: *Proc. of ICDE*, IEEE Computer Society, 2008. – ISBN 978–1–4244–1836–7, S. 366–375 (cited on pages 84, 85, and 183)

[YS02] YU, Bin ; SINGH, Munindar P.: An Evidential Model of Distributed Reputation Management. In: *In Proceedings of First International Joint Conference on Autonomous Agents and Multiagent Systems*, ACM Press, 2002, S. 294–301 (cited on page 104)

[ZC03] ZHOU, Yu ; CHIRIKJIAN, Gregory S.: Probabilistic models of dead-reckoning error in nonholonomic mobile robots. In: *In Proc. IEEE Int. Conf. Robotics and Automation (ICRA*, 2003, S. 1594–1599 (cited on page 151)

[ZH09] ZHANG, Chengyang ; HUANG, Yan: Cloaking locations for anonymous location based services: a hybrid approach. In: *GeoInformatica* 13 (2009), Nr. 2, S. 159–182 (cited on pages 76, 78, and 183)

[ZP07] ZOU, Joe ; PAVLOVSKI, Christopher J.: Towards Accountable Enterprise Mashup Services. In: *Proceedings of the IEEE International Conference on e-Business Engineering*. Washington, DC, USA : IEEE Computer Society, 2007 (ICEBE '07). – ISBN 0–7695–3003–6, S. 205–212 (cited on page 18)

[ZXM10] ZHENG, Yu ; XIE, Xing ; MA, Wei-Ying: GeoLife: A Collaborative Social Networking Service among User, Location and Trajectory. In: *IEEE Data Eng. Bull.* 33 (2010), Nr. 2, S. 32–39 (cited on page 154)

List of Figures

List of Tables

Appendix A

List of Selected Abbreviations

ABGAP	Agent Bottleneck Generalized Assignment Problem
ASO	Arbitrary Share Order
BRPP	Balanced Risk Placement Problem
CPU	Central Processing Unit
CSPS	Constrained Space Position Sharing
ESRI	Environmental Systems Research Institute
FSO	Fixed Share Order
GPS	Global Positioning System
HTTP	Hypertext Transfer Protocol
ILRQ	Imprecise Location-based Range Query
JSON	JavaScript Object Notation
LBA	Location-Based Application
LBS	Location-Based Service
LS	Location Server
MO	Mobile Object

- NP – Nondeterministic Polynomial time
- OS – Operating System
- OSPS – Open Space Position Sharing
- P2P – Peer-to-Peer
- pdf – Probability Density Function
- POI – Point of Interest
- PSUA – Position Sharing Update Approach
- QoS – Quality of Service
- RRS – Random Range Shifting
- SSL – Secure Sockets Layer
- TCP – Trusted Computing Platform
- TTP – Trusted Third Party
- TSP – Travelling Salesman Problem
- XML – Extensible Markup Language

Appendix B

List of Selected Notations

n	number of shares
m_0	total number of available LSs
m	number of selected LSs
L	total set of LSs
L'	set of selected LSs
p_i	obfuscated position of MO after obtaining ith share
p_i	in Chapter 3: risk level of ith LS
S	set of shares
S_n	set of n shares
s_i	ith share
n_i	number of shares assigned to $\mathrm{LS_i}$
ϕ	position precision
Δ_i^{ϕ}	precision increase provided by ith share s_i
π	precise position of MO
π_i	precise position of MO at the ith update
s_0	master share (share with minimal position precision i.e. maximal obfuscation)
π_{attack}	MO's position derived by an attacker
$P_{\text{k,attack}}$	probability of k LSs compromised by an attacker
P_k	acceptable probability of k LSs compromised