FAKULTÄT FÜR !NFORMATIK

Faculty of Informatics

# Appropriate obfuscation of location information on an application level for mobile devices

## Empower user to regain control of their location privacy

DIPLOMARBEIT

zur Erlangung des akademischen Grades

**Diplom-Ingenieur**

im Rahmen des Studiums

**Software Engineering & Internet Computing**

eingereicht von

**Christoph Hochreiner**
Matrikelnummer 0726292

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Privatdoz. Dipl.-Ing. Mag.rer.soc.oec. Dr.techn. Edgar Weippl
Mitwirkung: Univ. Lektor Dr.techn. Markus Huber, MSc

Wien, 03.03.2014

(Unterschrift Verfasser) (Unterschrift Betreuung)

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.ac.at

# Erklärung zur Verfassung der Arbeit

Christoph Hochreiner
Jungerstraße 16, 4950 Altheim

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

| | |
|---|---|
| (Ort, Datum) | (Unterschrift Verfasser) |

# Acknowledgements

I want to thank my supervisors Edgar Weippl and Markus Huber for their great support and the opportunity to write my master thesis at Secure Business Austria.

In case the self treatment methods proposed by Upper [59] failed to work, I could always count on my colleagues: Lukas, Felix, Christoph and Peter who helped me on the quest to get rid on any writing block and provided valuable feedback as well as ideas for my work.

The deepest gratitude goes to my parents, who made my study possible and supported me during the entire process.

Last but not least I want to thank Edith, who was always there for me and showed me, that there is also a world besides my thesis.

# Abstract

Mobile devices, like smartphones and tablets, find their way into more and more households and are currently used by over one billion people. While these new mobile devices introduce new possibilities for their users, they also reveal several security and especially privacy challenges. This work focuses on the sensitive location privacy, which is increasingly violated by the extended use of location aware applications. Those applications make use of the location provided by the mobile device to either offer context sensitive services or to monetize the personal information. The goal of this thesis is to provide theoretical as well as practical approaches, to support user to regain their power over their location privacy without limiting the usage of location aware applications.

The first challenge of this thesis deals with mitigation approaches that are used to reduce the entropy of location information and improve the personal privacy. So far, there are no recommendations of obfuscation algorithms, which can be used for specific applications to improve the privacy while maintaining the quality of service provided by the applications. The second aspect evaluates, how the obfuscation algorithms can be deployed to modern mobile operating systems. Currently these systems only provide very limited support in terms of location privacy and they provide no functionalities to implement the privacy preserving mechanisms. Consequently it is necessary to find alternative approaches to deploy location obfuscation mechanisms.

The research approach for this thesis was based on the constructive research. The process started with an intense literature research, which was used to retrieve state-of-the-art obfuscation techniques. Subsequently techniques were classified based on a qualitative evaluation to create an algorithm recommendation for location aware applications. For the interception of the location information, two different prototypes were designed and implemented. These two approaches were then evaluated against each other as well as against alternative already existing privacy improvement frameworks.

The first contribution of this thesis is a new classification scheme for location obfuscation algorithms, which includes the plausibility aspects of obfuscated locations for the first time. Based on this classification, a mapping between commonly used location aware applications and appropriate obfuscation algorithms is suggested. The second major contribution is the improvement of an existing location information interception framework and an implementation of a novel approach, to improve the location privacy. These two prototypes resolve several issues of previous interception approaches and can be used to support the user to regain power over their personal location privacy.

# Kurzfassung

Smartphones sowie Tablet-PCs finden immer mehr Anklang und werden derzeit von mehr als einer Milliarde Menschen verwendet. Diese Geräte ermöglichen es, neuartige Anwendungen, wie zum Beispiel standortbezogene Dienste zu entwickeln, bringen aber auch neue Herausforderungen im Hinblick auf die Privatsphäre mit sich. Das Ziel dieser Arbeit ist es, sowohl theoretische, als auch praktische Lösungen zu entwickeln, damit standortbezogene Dienste wie bisher genutzt werden können, die Privatsphäre der Benutzer im Bezug auf ihre Standortdaten jedoch verbessert wird.

Die Arbeit behandelt einerseits die Verschleierung der Standortdaten und andererseits die Möglichkeiten, diese in bereits bestehende Betriebssysteme einzubauen. Für viele Anwendungen, welche standortbasierte Daten verarbeiten, ist eine Näherung anstatt der tatsächlichen Ortsangabe ausreichend. Aus diesem Grund ist es möglich, die tatsächliche Ortsangabe zu verschleiern, ohne dass das Ergebnis der Anwendung verändert wird. Bis jetzt kann man in der Literatur zwar verschiedene Verschleierungsalgorithmen finden, jedoch gibt es keine Vorschläge, für welche Anwendungen diese Ansätze geeignet sind. Ein Ziel dieser Arbeit ist es, diesen Umstand zu ändern und eine Lösung für dieses Problem vorzustellen. Da kein mobiles Betriebssystem Verschleierungen unterstützt, müssen Wege gefunden werden, wie man die Ortsangaben verändern kann, sodass die Anwendungen nur noch eine Näherung der Ortsinformation erhalten.

Am Anfang wurde eine ausgiebige Literaturanalyse durchgeführt, um alle Verschleierungsalgorithmen zu sammeln. Auf Basis dieser Literaturanalyse wurde eine qualitative Bewertung der verschiedenen Verschleierungsansätze vorgenommen und eine Empfehlung für verschiedene Anwendungsfälle erstellt. Weiters wurden zwei Prototypen entwickelt, welche anschließend sowohl untereinander, als auch im Bezug auf bereits bestehende Lösungen verglichen wurden.

Im Zuge der Arbeit wurde ein neuartiger Klassifizierungsansatz entwickelt, welcher erstmals auch die Plausibilität der verschleierten Standortinformation analysiert. Auf Basis dieser Klassifizierung wurde eine Empfehlung von geeigneten Verschleierungsansätzen herausgearbeitet, welche von den Prototypen sofort umgesetzt werden kann. Durch die Entwicklung der beiden Prototypen wurde die Nutzerfreundlichkeit für einen bereits bestehenden Lösungsansatz deutlich verbessert sowie ein grundsätzlich neuer Lösungsansatz vorgestellt. Diese beide Prototypen lösen viele der bestehenden Probleme und können potentielle Nutzer dabei unterstützen, die Hoheit über ihre Privatsphäre wieder zu erlangen.

# Contents

**1 Introduction** **1**
- 1.1 Motivation . . . 1
- 1.2 Problem description . . . 3
- 1.3 Research questions . . . 5
- 1.4 Methodology . . . 6
- 1.5 Structure of the thesis . . . 7

**2 Background** **9**
- 2.1 Location information . . . 9
- 2.2 Location retrieval . . . 10
- 2.3 Location accuracy . . . 13
- 2.4 Location granularity . . . 13
- 2.5 Location-based services . . . 15

**3 State of the art** **17**
- 3.1 Obfuscation methods and algorithms . . . 17
- 3.2 Mobile privacy preserving frameworks . . . 26

**4 Algorithm classification** **31**
- 4.1 Obfuscation algorithm assessment . . . 31
- 4.2 Classification of location obfuscation algorithms . . . 36

**5 Proposed obfuscation categories** **43**
- 5.1 Obfuscation categories . . . 43

**6 System design and Implementation** **47**
- 6.1 Initial situation . . . 47
- 6.2 Proxy based interception . . . 48
- 6.3 Operating system level interception . . . 55

**7 Evaluation and Results** **63**
- 7.1 Applicability of proxy based interception mechanism . . . 63
- 7.2 Applicability of system level interception mechanism . . . 66

**8 Discussion** **69**
8.1 State of the art obfuscation techniques . . . . . . . . . . . . . . . . . . . . . . . . 69
8.2 Countermeasures against location obfuscation frameworks . . . . . . . . . . . 71
8.3 Algorithm suggestion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 72
8.4 Comparison of different frameworks . . . . . . . . . . . . . . . . . . . . . . . . . 72

**9 Conclusion** **75**
9.1 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 75
9.2 Contribution . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 76
9.3 Future work . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 77

**A Evaluation of proxy based interception** **81**

**B Evaluation of system level interception** **85**

**Bibliography** **87**

**List of Figures** **92**

**List of Tables** **93**

CHAPTER 1

# Introduction

## 1.1 Motivation

Due to the rapid diffusion of smart devices, like smart phones or tablets, the amount of location aware applications increases rapidly. Location aware applications use the geographic location information of the device, to provide convenient services, like the retrieval of nearby Point Of Interests (POIs). According to a survey in March 2012 [39] almost sixty percent of all smartphone users use location based applications. The convenience aspects of these services are popular with a lot of users, which is the main reason for the increasing amount of such applications. Besides the retrieval of nearby POIs, location aware applications are often used to inform other people about the current location or popular places [45], to record sportive activities [3], to retrieve the local weather or simply to use the application for routing purposes [37]. Although these location aware applications are convenient for the users, they also interfere with the privacy of their users [46, 41]. These privacy issues range from unauthorized collection of location information, to the interpretation of the collected location information to derive behavioral patterns. Behavioral patterns reveal a significant amount of private information about the user, like daily habits, shopping preferences or working habits [25].

The majority of these users are not aware of their violated location privacy or they do not care about the implications of the privacy violations. In 2005, only about 25 % of users were concerned about the privacy implications, which might occur after disclosing their location information [36]. Other studies [37, 8, 11] also support these figures. Within the last years the sensitivity concerning location privacy has risen [26], nevertheless it remains on a problematic low level. The main reason for this increased sensitivity is that all major mobile operation systems implemented some kind of location privacy settings. Besides the unaware users, there are also privacy sensitive users, who want control about their location privacy or at least want to be informed on the location information which is transmitted to external services [11, 16]. The demand of control varies based on the type of the application respectively on the date or the time of the day. Typical examples for an increased requirement of location privacy are stays in a hospital or nightclub visits, while the location tracking in a foreign city on vacation causes less demand of location privacy in terms of behavioral analysis. Based on user studies, the most important realization is that any effective privacy preserving mechanism should provide an automatic mechanism. This mechanism should improve the privacy without any user interaction required. This default mode is necessary, because most people are not aware of potential negative consequences or they simply do not care. Additionally for some people the effort, which is required to manually adjust the privacy configuration for applications, does not pay for the increased location privacy. Besides the automatic default mode there should be an advanced mode, which allows advanced users to tweak the default settings according to their requirements, as found out in [11].

Beresford and Stajano [13] where among the first, who understood this problem and they proposed a simple mitigation strategy. Since their publication, there have been several different researchers, who tried to solve this problem on an architectural level [55, 10, 35, 7] or to provide usable mitigation solutions, like obfuscation algorithms [4, 40, 10, 19, 57]. Although there have been numerous projects and publications, there is no effective solution, which allows the user to use location aware applications while maintaining the privacy of the user. Most solution approaches are either too complicated for the daily use or they provide too little configuration possibilities to cater the requirements of the users.

## Motivational example

The following example illustrates several problems, a privacy sensitive user runs into, when he wants to use different location aware applications for different scenarios. To access different location aware services, the user uses a smartphone, with multiple location aware applications installed. The installed applications range from routing applications, weather retrieval applications, applications which list nearby cash points and several location-based games. Each of these different applications requires a different level of location granularity to provide the desired service. The user wants to reduce the entropy of the location information as much as possible while maintaining the quality of service of the application. Some of the applications require the same level of location granularity, which should be clustered into the same category. The most important advantage of such categories is the reduced configuration effort to maintain the privacy policies.

The actual location obfuscation should be carried out on the fly and has to be transparent to the user. The transparency aspect is important, because the user does not care, how the location information is obfuscated or how big the derivation of the actual location information is, as long as his privacy is maintained and he can use the location aware services.

Additionally to the demand for privacy, the user also desires that his obfuscation actions cannot be detected by the service. The main reason for this request is, that some applications, especially location-based games, block players who try to fake the location information. In location-based games, the usage of fake location information compromises the game balance and therefore any modification to the actual location information might lead to an exclusion of the player. Although the main incentive of the user is to improve his privacy, the obfuscation of the location information might be interpreted as malicious by the service and the user can be blocked from the game or a service.

In this scenario, the user is a curious user, who regularly installs different new location aware applications. A manual assignment of newly installed applications to their categories is not feasible, because the configuration requires a significant amount of work and is prone to be forgotten. Therefore the location privacy solution should implement some kind of algorithm, which automatically assigns newly installed applications to appropriate categories. In the case that the automatic initial assignment is not appropriate, the user requires the functionality to reassign the application to another category.

## 1.2 Problem description

In the area of location privacy there are several problems, which have to be solved to provide an effective solution to maintain the location privacy of users, who use location based applications or services. Although the major problem of location privacy is the awareness of the users, this work only targets the technical and theoretical aspects of mitigation approaches. Solutions for these technical or theoretical problems might improve the awareness of users, but they are primarily designed to help users, who are unaware of the potential negative consequences.

### No granularity possible

The most important problem with modern mobile operating systems, like Android or IOS is, that they only provide limited configuration options for the user. These configuration capabilities only allow the user to either globally disable the location retrieval for all applications or to disable it only for single applications. The settings do not allow the user to set different levels of granularity for different applications. These different levels are required to maintain the location privacy. Although these additional settings improve the location privacy for users, they also introduce additional complexity to the settings preferences. This additional complexity is an issue for mobile operation system vendors, because they try to cater the requirements of the majority of their users and not of a few ones, like the privacy sensitive ones. To compensate this additional complexity, the developers are challenged to implement more efficient and intuitive settings, like the possibility to cluster single applications into categories and assign the settings to these categories. Another approach is to implement an algorithm, which automatically assigns newly installed applications to a category respectively a granularity level. This algorithm has to be based on solid default values and the user can refine these values afterwards to his personal requirements, if necessary.

### Classification of obfuscation algorithms

Within the last decade different location obfuscation algorithms have been proposed (see 3.1). Although there is a large number of different location obfuscation algorithms, there are hardly any comparisons or classifications of these algorithms. Existing classifications [60, 2] only provide a coarse qualitative classification (see 4.2). Currently there is no classification or evaluation, which proposes appropriate location obfuscation algorithms to different use case scenarios. The first step here has to be a qualitative evaluation of the most common location obfuscation algorithm and an analysis of typical use case scenarios of location based applications. Based on this evaluation and the analysis, appropriate algorithms should be suggested for the different use case scenarios. A further step towards a solid classification is a quantitative evaluation of this proposed assignment. In such a quantitative evaluation, the proposed algorithms are evaluated by a large number of users regarding their applicability. The most important aspect of this applicability is, whether the obfuscation maximizes the privacy of the user while maintaining the quality of service of the application. The result of the qualitative as well as the quantitative evaluation are a solid foundation for any automatic assignment algorithm, which assigns newly installed applications to an appropriate location granularity level respectively obfuscation algorithm.

### Plausibility

The aspects concerning the plausibility of obfuscated location information is not prominent in the area of location obfuscation algorithms. Most location obfuscation algorithms are only designed to ensure the privacy of the users, but they do not evaluate whether the newly generated location information is plausible. This plausibility analysis covers, whether the location is actually plausible or whether a sequence of different locations is plausible. Up to now, the plausibility of artificially generated locations was not important for location obfuscation scenarios. With the rise of location aware games, where the location is a constituent element in the game, the developers start to implement plausibility checks, to detect unnatural and therefore problematic location changes of players. The basic assumption for such plausibility changes is, that a user cannot dissolve at one location and turn up on another location within a fraction of a second. Besides the game industry, the traces of location information are also important for behavioral analysis of customers. More and more shopping malls use the location information of their customers to detect popular places or stores within their shopping malls [49]. This behavioral analysis rely to a large extend on recorded tracks of location information. These tracks are provided by applications, which are installed on the mobile devices and record all movements of their owners. The correctness of this information is crucial for any behavioral analysis and therefore the provided information is often also checked for its plausibility. With the rise of these applications for behavioral, the development of plausibility checks becomes more important, which was originally only relevant for in attack scenarios [40].

To respect these additional external requirements for location obfuscation algorithms, the plausibility component has to be evaluated and included in any future evaluation respectively classification of obfuscation algorithms.

### Non invasive solutions

Most solution approaches, which try to improve the location privacy for users either require an unauthorized modification of the mobile operation system [1, 55, 12, 32] or they have to be implemented by the application developer [10, 35]. Both approaches are not feasible respectively possible for normal users. Any modification to the mobile operating system requires advanced technical skills. Additionally the owner of the mobile device might loose the warranty for the device as soon as the operation system contains any unauthorized modifications. The second approach, the implementation of a privacy preserving framework [35], is desirable from a data officials point of view, but no developer wants to intentionally decrease the entropy of the location information on purpose. Any decrease of the location information can potentially decrease the quality provided by service of the application. This potential decrease regarding the quality of service is not acceptable for service providers.

The invasive aspects of current solutions require that any effective solution has to be implemented by the vendor of the mobile operation system, like Google of Apple. The only alternative is a non-invasive solution, which obfuscates the location information outside of the mobile device transparently for the user. These non-invasive systems can be integrated within the mobile operation systems by already implemented means, like a proxy for the system.

## 1.3 Research questions

Based on the problem description, several research questions occur, which are listed in this section. The goal of this work is to provide an answer to each of these questions and figure out further problem definitions. These further problem definitions can be used as a basis to further improve the location privacy for the user in succeeding projects.

#### RQ1: Which location information obfuscation techniques for mobile applications are state of the art?

The first research question triggers a literature research to list different proposals and implementations, which are designed to improve the location privacy. Although the goal is to identify different location obfuscation techniques for mobile applications, also alternative obfuscation technologies should be analyzed and evaluated whether they can be applied to the mobile application domain.

#### RQ2: Are there any countermeasures against location obfuscation frameworks implemented by mobile application developers?

The goal of this research question is to collect different attacks against location obfuscation algorithms. The focus is on attacks, which reverse the application of an algorithm to obtain the actual location and those who detect the application of an obfuscation algorithm.

#### RQ3: Which location obfuscation algorithms are suited for typical mobile applications?

This research question is based on the result of the two previous research questions. The goal of this research question is to provide a list of appropriate location obfuscation algorithms for typical mobile applications. The selection of these algorithms is based on the minimal required level of location granularity and the robustness of the algorithm against attacks.

#### RQ4: Are there any practical interception mechanisms in state of the art mobile operating systems for location information?

This research question evaluates different interception possibilities, which can be used by an external location privacy improving solution to modify the location information, with the goal to reduce the entropy of the reported location information. These interception mechanisms are necessary to design location obfuscation frameworks, which obfuscate the location information according to the privacy policies provided by the user.

**RQ5: How does the implemented frameworks compare to other proposed or existing solutions regarding functionality, efficiency and usability?**

One of the results of this work should be a framework, which is designed to improve the location privacy and resolve existing problems. The result of this research question describes, how the new framework improves the situation for the user and how it compares to other already existing or proposed frameworks.

## 1.4 Methodology

The research approach for this work is based on the constructive research [38]. This approach was extended to match the requirements, which were posed by the research questions. The approach is structured into eight sequential steps, which are listed in this section.

**Literature research**
The succeeding phase consists of an extensive literature research, to gather state of the art research results in the area of location privacy, especially for mobile devices. This literature research is divided into two parts. The first part covers the research on location obfuscation algorithms, which are used to obfuscate the actual location information. The obfuscation improves the privacy of the user by providing similar artificial location information. The second part is dedicated to find already existing or proposed location obfuscation frameworks, which are used to apply location obfuscation algorithms. The latter part is required to investigate, if there are already implementations, which solve the previously stated problems or if there is still room for improvement. The outcome of this phase is a detailed list of state of the art location obfuscation algorithms as well as a list of location obfuscation frameworks.

**Classification of location obfuscation algorithm**
Based on the literature research, a classification of location obfuscation algorithm is conducted. This classification is required to implement appropriate obfuscation algorithms for different use case scenarios of location aware applications. The classification is based on a qualitative assessment that evaluates different properties of the algorithms.

**Prototypical implementation**
The prototypical implementation phase is divided into three parts. In the use case definition part, typical use case scenarios for Location Based Service (LBS) are defined, which are later used to evaluate the implemented prototypes. In this phase, suitable location obfuscation algorithms are assigned to the different use case scenarios. The result of the use case definition phase is used, to derive additional requirements for future frameworks.

The prototype concept part is based on the literature research of location obfuscation frameworks. In this phase, the results of the requirements gathering phase, based on the use cases, and the outcome of the literature research are combined to design a prototype. The prototype should cover all requirements and solve the problems, which were detected in the requirements definition phase. The goal of this phase is to provide a concept, which considers all requirements. If there are any conflicting requirements, more than one prototype has to be designed.

Based on the concept part, the prototypes, which cover most of the requirements, are implemented. The implementation phase is based on a cyclic development approach [58], where the implementation is continuously validated against the requirements.

**Prototype evaluation**
The evaluation covers a functional evaluation of the implemented prototypes as well as an evaluation against real world scenarios. The latter evaluation is designed to validate the recommended algorithm assignment regarding the effectivity to improve the privacy while maintaining the quality of service.

## 1.5 Structure of the thesis

The remainder of this thesis is structured as follows. Chapter two contains all related Background information and chapter three provides an overview about State of the art obfuscation algorithms as well as privacy preserving frameworks (see 3). The previous chapter about state of the art obfuscation algorithms is foundation for the assessment of the obfuscation algorithms and the classification of these algorithms in the fourth chapter (see 4). The fifth chapter (see 5) provides an overview about the most important use cases scenarios and suggests an appropriate algorithm for each of these scenarios. The System design and Implementation chapter (see 6) describes the default architecture (see 6.1) for modern mobile operating systems as well as the two implemented interception frameworks, namely the proxy-based interception framework (see 6.2) as well as the operation system level interception framework (see 6.3). This chapter describes and explains the design decisions as well as important implementation details for these frameworks. The succeeding chapter (see 7) provides the setup as well as the results for the evaluation of the two implemented frameworks. These results as well as the proposed algorithms in the fifth chapter are discussed in the eight chapter (see 8). The Discussion chapter provides an answer for all research questions, which were stated in the introducing chapter. The last chapter (see 9) concludes the results of the thesis, highlights the contribution of this thesis and points further research challenges, which should be tackled in the future. The Appendix of this work provides detailed information about the evaluation, which are already aggregated in the Evaluation and Results chapter.

CHAPTER 2

# Background

## 2.1 Location information

The definition of location information in this context describes the geographic location of the user. This geographic location is retrieved by location retrieval systems, which are implemented in mobile devices. These location retrieval systems derive the geographic location based on external reference points such as geostationary satellites or mobile communication infrastructure, like the Global System for Mobile Communications (GSM) infrastructure. A short overview about different location retrieval techniques can be found in the location retrieval section 2.2

The default representation of the location information in this work is the geographic representation based on longitude and latitude, e.g. (48.2084671, 16.3730908) for the center of Vienna. In this work it is assumed that the geographic location of the user and the geographic location of the technical device, which is used to retrieve the location information, is identical.

There are two different types of location information that are used throughout this work:

**Actual location information**
This type represents the actual location information that is derived from the mobile device. The location information might differ from the real geographic location, due to negative external effects (see Section 2.3), but this is the most accurate location information that can be passed on to any service respectively location aware application.

**Generated location information**
The second type describes the location information that is reported to a service. This location information has been modified by an obfuscation algorithm, which reduces the accuracy of the generated location information compared to the actual location information. This accuracy reduction can be either done on a geographic level or on a time based level. In the first case, the algorithm translocates the actual location to an artificially generated one. The time-based approach decouples the actual location and the time when the actual location is recorded. The

actual location is reported with some delay, which improves the privacy of the user in some scenarios.

## 2.2 Location retrieval

There are multiple approaches that can be used by mobile devices to retrieve the location information. The selection of the chosen approach depends on the required precision as well as on other external influences like the weather or available telecommunication infrastructure. The following section provides a short overview about the most commonly used location retrieval mechanism, which are often combined to improve the precision as well as the accuracy of the location information.

### Satellite based positioning system

Satellite based position systems are terminal based approaches. In terminal based approaches, the location is calculated on the mobile device and the user can decide whether the location should be transmitted to external services or not. In such a scenario, the external infrastructure, like satellites, only provide the information about their position and their local time, but they are not capable to locate the user.

The most popular positioning system, is the Global Positioning System (GPS) system [33], which was developed by the U.S. Department of Defense. The core of the GPS system consists of at least 24 satellites, which orbit the earth. These satellites are equipped with an atomic clock and a transmitter, which continuously broadcasts the current location of the satellite and the current local time. To calculate the actual position of a mobile device, the mobile device requires a GPS receiver, which processes the transmitted messages. The mobile device then derives the actual information, based on the transmitted messages and some initial information about the orbits of the satellites. The receiver requires the location and time information of at least four satellites, to calculate the actual geographic location. The accuracy of the location information correlates positively to the amount of visible satellites for the receiver.

Besides the plain GPS approach, there is also the assisted GPS approach [22], where the receiver improves the location information, which is calculated by the GPS system, with the location information retrieved from other location retrieval systems, like those described below. In this setup, the actual location is calculated based on these multiple location information and the result is generally more accurate than the plain GPS location information. The major downside of this assisted GPS approach is that the mobile device can also be located by external users, like in the network based location approaches (see 2.2).

In addition to the dominant GPS, there are also other satellite based positioning systems, like the Russian implementation GLObal NAvigation Satellite System (GLONASS) or the European implementation GALILEO, which are designed in a similar manner [34]. This work will only uses the GPS implementation, but these facts can also be applied to the alternative satellite based location retrieval systems.

### Network based approach

The telephone network based approach makes use of the GSM respectively Universal Mobile Telecommunications System (UMTS) base stations to derive the relative position from the mobile device to the Base Transceiver Station (BTS). The BTS has a fixed position and therefore it can be used to derive the actual location of the mobile device. In this scenario, the location retrieval is normally carried out within the network infrastructure and does not require any additional interaction with the mobile device, besides being logged into the network. From a privacy point of view, this system has a major disadvantage, because the mobile device can be located without the approval of the user. There are different techniques that can be used to calculate the position of the mobile device. These different techniques provide different levels of accuracy.

**Cell identification**
The cell identification approach is the conceptually easiest, but also the least precise positioning method. It requires only one BTS to retrieve the location of the mobile device. Mobile phone networks are divided into different cells, which are operated by different BTS. One BTS can operate one or more cells. The cell identification algorithm takes the cell identifier of the currently used cell and looks up the location of the corresponding BTS [23]. This location is then reported to the initiator of the location information request.

Figure 2.1 shows a typical scenario for the application of the cell identification approach. The user is located in the cell with the identifier C2, whereas C2 is assigned to the black BTS. To retrieve the location of the user, the algorithm looks up the location of the black BTS and returns its location to the user. The accuracy of this approach depends on the cell size, which can range from 250 meter in cities up to 35 kilometer in rural areas. The accuracy can be improved, by also considering the period of time, which is required to send a message from the BTS to the mobile device. The combination of the BTS location and the period of time can be used to estimate the distance between the mobile device and the base station and restrict the location to a segment of the cell.

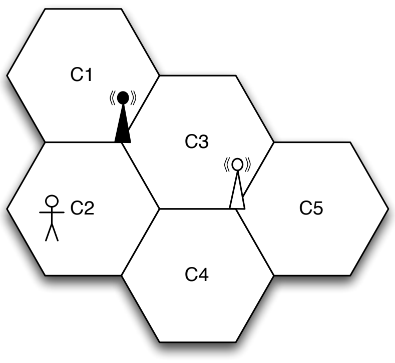


**Figure 2.1:** Cell indentification (adapted from [23])

**Angle of arrival**
The angle of arrival approach is based on triangulation and requires at least three BTS. These BTS must be able to measure the angle under which they receive the incoming signal from the mobile device [23]. The location retrieval procedure starts by emitting a signal to the nearest three BTS, who record the angle of the incoming signal. Based on the three angles, they then calculate the point-of-intersection, which is then reported as the location of the mobile device, as one can see in Figure 2.2. Besides the initial signal, which is emitted on a regular basis, this approach is completely network based and there is no interaction with the mobile device required. The initial signal can be an announcement by the mobile device or a request to process a call.

This positioning approach works better within rural areas, where it is more likely to have a clear line-of-sight among the mobile device and the BTS, than in urban areas. In urban areas this approach is not feasible, because it suffers from multi-path propagation. Multi-path propagation describes the problem where different subjects, like buildings, bend the signal and therefore the incoming angle is not valid any more.

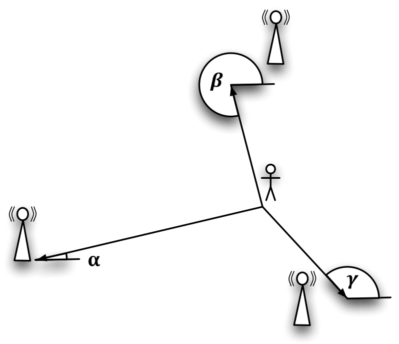


**Figure 2.2:** Angle of arrival (adapted from [23])

## Fingerprinting approach

The fingerprinting approach is an implementation of a pattern matching approach. The mobile device tries to match the actual location, defined by the signal strength of BTS or Wi-Fi access points, with similar setups stored in a database [44, 6]. This approach is divided into two phases, namely training, and positioning. The training phase is used to build a reference database of location fingerprints. These fingerprints consist of a location, the signal strength of nearby access points, and identifiers of these access points. In the positioning phase, the mobile device, which wants to obtain the location information, creates a fingerprint based on all available access points and compares the fingerprint with those stored in the database. The location of the fingerprint that correlates most with the submitted fingerprint is very likely the location of the mobile device. Compared to the other positioning approaches, this one can also be used indoors.

| | Urban | Suburban | Rural |
|---|---|---|---|
| Satellite based | 17 | 10 | 5 |
| Cell identification | 247 | 1062 | 1042 |
| Angle of arrival | 242 | 894 | 3362 |
| Fingerprinting | 101 | 255 | 243 |

**Table 2.1:** Mean deviation of actual geographic location in meter

This location retrieval approach is privacy friendly, assumed the positioning phase is only carried out on the mobile device. If all operations are carried out on the mobile device, the user can decide, whether he wants to reveal his location to a service or not.

## 2.3 Location accuracy

Most location information services like GPS or other cell based location retrieval mechanism cannot provide the exact geographic location due to negative external influences, like atmospheric effects, clock errors or multi-path errors. The inaccuracies have been analyzed by several surveys [43, 56], which compare different positioning approaches for cellular networks. The deviation to the actual geographic location ranges from five meter up to more than three kilometers. The values provided in their surveys provide an overview about the mean deviation in meter. Table 2.1 summarizes the most important location retrieval techniques and their deviations in different environments. Although the most commonly used location derivation mechanism (GPS) provides the best results, it is also not capable of providing the exact location information. In the best case scenario, when there are no additional negative inferences, the deviation accounts for at least five meter [61]. Based on this imperfection, most services or applications already assume that the location information is not absolutely correct. They assume that the actual location of the device is within a small circular shape around the given location information. In order to compensate these small location inaccuracies, most services implement mitigation algorithms, which balance out this imperfection to provide a stable service.

## 2.4 Location granularity

Besides the standard location information representation given by coordinates, there are also other representations, which translate the coordinates into concrete places, like cities or countries. The concept of location granularity describes different levels of granularity, which can be expressed by different location information representations. The most precise location specification approach are geographical coordinates, represented by longitude and latitude. Other approaches implement semantic aspects, which divide the landscape into different granular partitions [15]. These partitions can have an arbitrary shape respectively size and can be used to provide a semantic description for a specific location. Bittner [14] provides a taxonomy for granular partitions, which originate from the geographic domain. In his taxonomy, he provides

| *Granularity* | *Example* |
|---|---|
| continent | Europe |
| country | Austria |
| state | Upper Austria |
| city | Vienna |
| district | Vienna, Innere Stadt |
| street | Vienna, Karlsplatz |
| building | administration building / Vienna, Karlsplatz 13 |
| room | seminar room / Vienna, Karlsplatz 13 - room 101 |

**Table 2.2:** Examples for selected location granularity sizes

different partition approaches. He mentions a categorical coverage based on the soil, like obstructed areas as well as cadaster, which are based on ownership and political properties.

The concrete information of geographic entities is often stored in a standardized model, which is based on the Ressource Description Framework (RDF). The Open Geospatial Consortium (OGC) maintains several standards which are used to store the geographic information [30] respectively to retrieve specific information [48, 9]. These standards are implemented into commercial geographic solutions, like Google Maps and may also be used to perform a coordinate transformation into a spatial element, as it is provided by the Google GeoCoding API [29].

This work uses the location granularity only in combination with the political and ownership based partitioning approach. The different partitions are used to describe geographical related areas. The commonness of these areas is, that a service always returns the same result for each coordinate within this geographically related area. The most coarsely partition approach is to divide the landscape into very large entities, which are called continents. These continents can then be broken down into different countries, states, districts and cities. This downsizing can be carried on down to single rooms, like seminar rooms, which can also be seen as distinct partitions. The location granularity concept describes the degree of approximation to the actual location.

The selection of the appropriate partition depends on the use case as well as on the technical respectively organizational possibility to distinguish the different partitions. The Table 2.2 provides a list of different partition sizes with a typical example. While the first four partition sizes can be identified with only one name, the remaining ones require additional information to avoid ambiguities. The smaller the partition size gets, the more information is required to describe the partition.

## 2.5 Location-based services

A LBS is a service which makes use of geographic locations to improve the value provided by a service [51, 5]. Such LBSs can be found within different domains, ranging from emergency systems, routing systems to surveillance systems which track people or goods. The integration of geographic information often improves the usability and value for the user of existing services or creates new services, like routing services or location aware recommendations.

CHAPTER 3

# State of the art

## 3.1 Obfuscation methods and algorithms

The following section lists different location obfuscation algorithms, which provide different characteristics in term of privacy and plausibility of the generated location information. It describes the basic concept of the location obfuscation algorithm, lists known weaknesses respectively attacks and potential application scenarios.

The literature provides several classification approaches for obfuscation algorithms, like their use case scenario [2], different protection goals [60] or whether there are any external services required, to implement the location obfuscation algorithm [32]. The approach chosen for categorizing this listing of algorithms is based on the amount of user, which is required to generate an obfuscated location. It also clusters different algorithms, which are based on similar principles. Most of the mentioned algorithms do not require any additional participants, besides the user, which makes them more practicable for real world scenarios. For the sake of completeness, this work also deals with the most important multi user algorithm, the k-anonymity algorithm, which is often mentioned in the literature, but has hardly any practical relevance.

### Deactivation

The deactivation approach informs all services or applications, which request location information, that there is no location information provider available or the location information provider is deactivated. On the one hand, this obfuscation approach is the only approach, which unrestrictedly preserves the privacy of the user, but on the other hand, this algorithm renders all location-based applications useless. This approach is often used as the default approach, when the user does not require any location-based services and wants to improve his privacy for all applications, which may run in the background and request the location information.

### Fake location information

The computationally easiest approach to obfuscate the location information is to choose a new arbitrary location information. This arbitrary information is not connected with actual location information and it cannot be used to derive the actual geographic location. There are different approaches that can be used to generate fake location information. The selection of a specific approach depends on the use case scenario, which is defined by the user.

#### Random location information

The easiest approach is to generate geographic locations based on a random number generator. This basic approach can be improved by implementing constraints, which restrict the generated locations. These constraints can either be static or dynamic. The static constraints ensure, that the location is within a feasible area, like the home country of the user, while the dynamic constraints ensure a plausible sequence of several location information. The dynamic constraints are derived from feasible movement patters. These feasible movement patterns are based on all possible transportation means in a specific area and restrict the potential area for new random locations within a specific time frame. By restricting the generated location information to these movement patters, unrealistic sequences of location information, like different generated locations in the center of different continents within one hour can be circumvented. Such a sequence is not possible, because currently no transportation mean exists, which provides such a movement speed. Additionally a person cannot dissolve in one location and appear in another one within a fraction of a second.

The generation of plausible routes in not trivial, as the project by Krumm [42] shows. The goal of the project was, to generate realistic routes based on probabilistic models. The routes look very similar to normal recorded routes. They simulate a realistic driving speed and incorporate some GPS noise that is normal when using an actual GPS system.

#### Chosen location information

Another approach is to select the fake location on purpose. This can be done by either entering arbitrary geographic coordinates or by selecting the desired location on a map. The chosen location information approach can be used, to gain profit by pretending to be on a specific location. The arbitrary location selection is handy for location-based games, like Ingress [1]. In such games the user can achieve advantages over the competitors, when he can simulate location changes or pretend to be at multiple locations at once.

The most common implementation approach for this algorithm is to provide a map, where the user can choose the desired location. While selecting the location, the user can already ensure the plausibility of the selected location.

[1] `http://www.ingress.com` [Online; accessed 30-November-2013]

### Spatial Obfuscation

The basic principle for spatial obfuscation is to use the real location information and perform a spatial translocation according to a given algorithm. There are different types of algorithms, which cater to different requirements. These requirements range from increased location privacy, precision required by the service and adjustability to robustness against countermeasures.

The spatial obfuscation algorithms are vulnerable to inference attacks. The work by Krumm [40] provides a good introduction to inference attacks, which can be used to revert the usage of the obfuscation algorithm.

The further section mentions four algorithms that improve the privacy by increasing the inherent location inaccuracy.

#### Enlarging the radius

The most common approach for spatial obfuscation is to augment the area, which is used to generate a new location information [4]. Due to the fact that the original location information is not absolutely accurate. It is already assumed by many services or applications that the retrieved location is within a small circular area around the actual location. This small area is represented by the circle with radius R1 in Figure 3.1. The enlarging of the radius algorithm increases the radius to R2. Due to this augmentation of the radius, the surface, which contains the generated location, is increased. The generated location is represented by a randomly picked location within the augmented surface.

In practical scenarios the radius R1 is rather small, depending on the technology used to determine the location. The length of the radius ranges from five meter in the GPS case up to one kilometer in the cell based setting (see 2.3). The radius R2 depends on the desired level of privacy respectively the accuracy that is required for the service to perform adequately. The selection of the radius is based on the minimal required accuracy of the LBS. This means that the user selects the largest possible radius for an application, which at the same time represents the highest possible level of privacy provided by this algorithm.

The goal of this algorithm is to obfuscate the actual location or route within the artificially increased area. The obfuscation works very well, when only a few location information are transmitted to the service and the service is not able to perform an interference attack [40]. Assumed that this algorithm uses a true random algorithm to select the generated location from the surface, one can observe that a set of different generated location information will be equally distributed over the circular shape. When an attacker obtains multiple locations for a static actual location, he can retrieve the actual location with an inference of the equally distributed generated locations.

#### Shifting the center

The shifting the center algorithm translocates the actual location information by a constant factor. This algorithm maintains the relative movement information of the user, but it obfuscates the actual location of the user. There are two different possibilities of how to shift the location information.

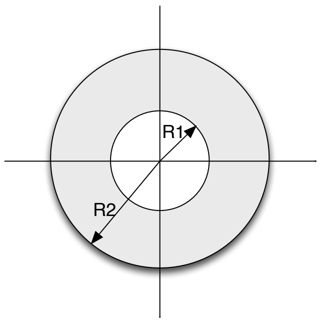


**Figure 3.1:** Obfuscation by enlarging the radius (adapted from [4])

**Local shift**
The local shift operation is displayed in Figure 3.2, where the actual location is shifted by a fixed factor and the shifted location is then reported to the service. In the local shift scenario the generated location is generally within at least 100 m and at most 10 km. This local shift scenario can be useful when one wants to use location aware services, but do not want to reveal the actual location.

**Arbitrary shift**
The second approach is to choose an arbitrary starting location on a map. The algorithm then calculates the distance between the actual location and the selected point on the map. This distance is then added to the actual location for every location retrieval request. This algorithm can be used when the user wants to pretend to take a walk one the beach instead of the actual walk at home.

**Double obfuscation**

Double obfuscation algorithm describes the combination of the enlarging the radius algorithm and the shifting the center algorithm. An application of these two approaches can be seen in Figure 3.3. The double obfuscation algorithm enlarges the area and then shifts the surface, which contains the generated location information. The generated location information is then a randomly chosen point within the shifted and enlarged area.

**Discretization to points on a grid**

The discretization to points on a grid approach is a very easy approach in terms of implementation. In this approach the actual location information is mapped to a grid, as described in [40]. An example application of this algorithm can be seen in Figure 3.4, where the black dots represent the grid and the red dots represent actual locations. The actual locations are mapped to the closest point on the grid and therefore the reported locations get more inaccurate for the service and the privacy of the user is improved.

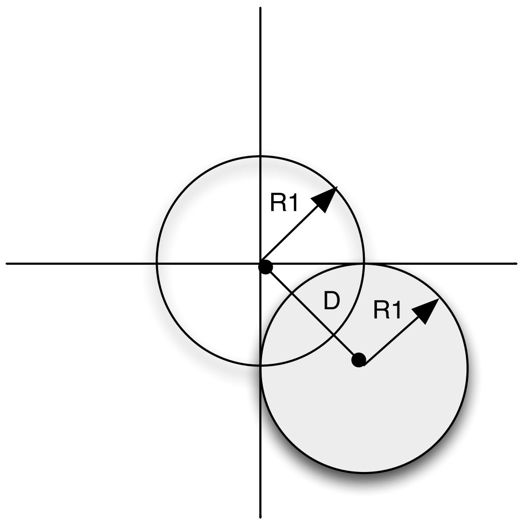


**Figure 3.2:** Obfuscation by shifting the center (adapted from [4])

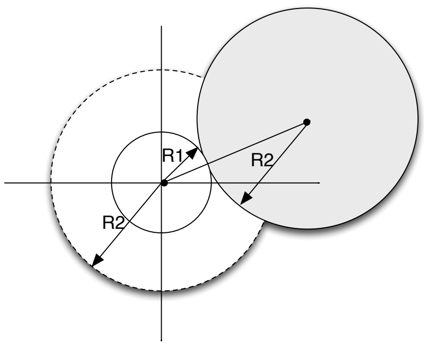


**Figure 3.3:** Combination of enlarging the radius and shifting the center (adapted from [4])

The easiest implementation of this algorithm is to truncate a defined amount of figures at the end of the coordinates. This truncation process automatically aligns the generated locations to grid. The level of obfuscation can be controlled by the amount of digits truncated from the actual coordinate, e.g. 48.1989, 16.3699 is transformed to 48.19~~89~~, 16.36~~99~~. The more digits are truncated, the more wide-meshed the grid becomes.

## Time based obfuscation

Time based obfuscation reduces the quality of a continuous set of location information. A typical location based application requests the location information on a regular basis with only a short timeframe between two requests. The time based obfuscation algorithm introduces a random

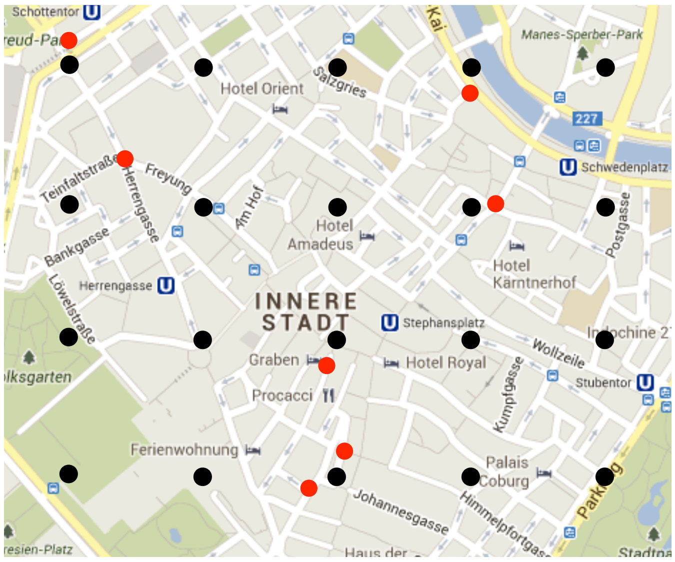


**Figure 3.4:** Discretization of location information to a grid

time component between two location information reports. The actual location is still recorded on a regular basis, but the service waits for the random time span until it submits the data to the service. In Figure 3.5 the first line represents the timeline for the requests of the application respectively the actual locations represented by the captions L1 to L10. The second timeline demonstrates the reporting events of the random response algorithm, where the response to the application is plotted. Another time-based approach extends the time span between two updated geographic locations in order to introduce a kind of discretized pattern. This approach is very similar to the grid based one. The service does not recognize the extended time span, because it receives the location information on a regular basis, but the information itself stays the same until the extended timespan is over. This algorithm is shown by the third timeline in Figure 3.5, where the algorithm reports the location on a regular based, but the location information is updated only once in three times.

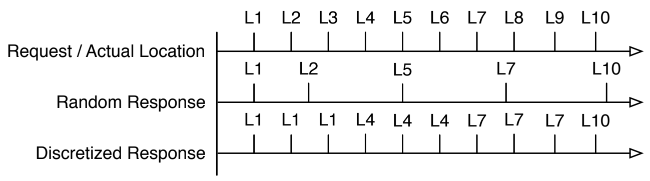

Request / Actual Location
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10
Random Response
L1 L2 L5 L7 L10
Discretized Response
L1 L1 L1 L4 L4 L4 L7 L7 L7 L10

**Figure 3.5:** Time based obfuscation

Both approaches reduce the usefulness of the meta-information, like the movement speed or the length of a stay at a specific location. The random response approach creates a forged trace while the discretized one reveals too little information to calculate a useable trace. The meta-information is normally calculated from a set of location information and can be very useful to generate behavior pattern of users. In a mall scenario, these behavioral patterns can be used to describe some shopping preferences and habits of the user, like the length of a stay in a specific shop. Assumed that the time-based obfuscation is applied, these recorded behavior patterns are useless, because they do not represent the actual behavior of the user.

## Semantic obfuscation

The semantic obfuscation algorithm creates privacy compatible location information based on semantic information of the area where the user is located. This semantic information ranges from basic information like lakes, rivers or mountains to very concrete information like POI that describe sights or infrastructure, like hospitals or tube stations. Although these algorithms provide more privacy and the generated location information is more robust to plausibility checks, there is one major downside of this kind of obfuscation approaches, namely the required semantic information. It is often hard to obtain recent semantic information about an area, because this information is expensive or not available. To resolve this issue, there are some projects, which gather geographic information, like the OpenStreetMap Project[2]. This project gathers all kind of geographic relevant information, connects it with semantic information and provides them under the Creative Commons Attribution-ShareAlike 2.0 (CC-BY-SA)-license. Although this project is already very mature, there are several white areas on the map that are not indexed or contain semantic information. Due to the lacking information, the user cannot apply any semantic based obfuscation algorithm for an actual location within these white areas. Besides the open projects, there are also commercial ones, like the reverse Geocoding API by Google [29], which provides a better quality for some areas.

### Landmarks

One of the most obvious semantic location obfuscation approaches is to map the actual location to a POI. This obfuscation approach can be seen as an extension to the one, which discretizes the points to a grid [40]. In this case, the algorithm replaces the static and therefore often meaningless points with POI. These POI can either be infrastructural entities, like stops for the public transport system or cultural ones, like churches or statues. These POI are locations, which attract a large number of people and therefore the entropy of a location information that is reported from such a location is lower than one reported from an actual location. The actual location of a user is normally a very distinct one, which is not accessed by many people. Besides the increased privacy aspect, there is also a second major advantage compared to the grid based one. This algorithm creates plausible geographic locations, which make it hard for services to detect the application of this algorithm. In terms of implementations, there are already ready to use solutions available like the reverse Geocoding API by Google [29]. The Application Programming Interface (API) of the geo-location service allows the developer to retrieve a human readable address for a given set of geographic coordinates, whereas the developer can limit the granularity of the human readable address. This service only implements political semantic aspects, like streets or districts and therefore only allows a coarse mapping of the actual location to a POI.

[2]`http://www.openstreetmap.org` [Online; accessed 30-November-2013]

There are basically two parameters that can be used to adjust the level of privacy for this algorithm. The first parameter is the density of the used POI. This algorithm can either use unevenly distributed POI, like sights or more regular distributed POI, like bus stops. While the first approach increases the privacy due to the larger amount of people near this location, the second approach provides a better plausibility, especially in terms of the movement speed. The second type of parameter is the location granularity. Here the user can adjust the level of the geographic segmentation that could be on the level of countries, counties, districts or streets. The latter aspect is described in the work by Bellavista [10], where they provide a concrete example for these different parameters. On a more fine-grained level, the user might choose between the tube stations or the bus stations, whereas the first one provides a better level of privacy due to the more coarse mashed net.

### Avoid areas

There are two types of information that can be concluded based on location information. The first type describes the actual location information and movement patterns. This type can reveal common locations a user visits or general behavioral patterns, but there are no semantic interpretations about locations. The second type combines the location information with additional semantic information. This can be the location information of buildings, which might reveal sensitive information about the user, like hospitals, religious buildings or night clubs [19]. This combination can reveal diseases, the users belief or not generally accepted leisure-time activities.

The approach of avoiding such sensitive areas, is the basic motivation for the PROBE obfuscation framework proposed by Damiani et al. [18]. This framework categorizes the surface into three categories. The first category describes all public accessible areas, like streets or parks that are not considered as sensitive regions. The second category contains all privacy sensitive regions like hospitals or religious buildings, whereas the user can add additional areas that the user considers as sensitive. The last category describes all unreachable areas, like military areas or lakes, where it is forbidden to use boats. Based on this categorization, the map is then segmented into these three different areas and a customized map is generated for every user. The actual obfuscation is then carried out based on these generated maps. The generated location has to be within a public accessible area and not within the sensitive respectively the unreachable ones.

This obfuscation algorithm transforms the actual location information into plausible generated location information that reduces the possibility to leak sensitive information. The plausibility of the location information is guaranteed due to the black list of not accessible areas.

There are several extension possibilities to this approach. The most obvious one is the introduction of time based preference lists. These different lists can cover the different privacy requirements that are required when somebody is at work or somebody is at home. In the work scenario there might be more areas that are marked as sensitive, like shops or cafes that are near the work location. These locations are sensitive locations for a working user during working hours, because the user does not want to reveal a coffee break or similar activities.

### Mediated obfuscation by a trusted third party

All obfuscation approaches presented above, can be executed on the location aware device and do not require any external services or information, apart from the semantic information, that theoretically could also be stored on the device.

#### k-Anonymity

The basic concept of k-anonymity [57] can also be applied to the area of location obfuscation. The goal of k-anonymity is to create a set of the size k with individual information pieces, where an information is indistinguishable among all other k-1 elements. Although the elements are indistinguishable, the information itself remains valid and useful. The work by Gedik and Liu [28] applied this concept to location privacy. They propose a process to preserve the privacy as well as a practical implementation for this process. The process starts with a user recording the actual location information and transmitting the location information to a Trusted Third Party (TTP). The TTP collects the information pieces from different users and waits until there are enough users to perform the permutation of the location information. Once either the time runs out or enough users are available, the service randomly assigns the location information of one user to another user and assures that no user is assigned their original location information. As soon as the messages are permutated, they are forwarded to the services that already wait for the location information.

The threshold value that describes when there are enough users for the permutation, differs due to local and timely constraints. In the city this number is higher than on the countryside, because on the countryside there are less people who can use the service. The same applies for the time aspect. In the night there will be less people using the service. A real world implementation has to consider both aspects. It has to implement a suitable threshold value and a timer, which limits the maximum amount of time that is used to wait for more users.

This location obfuscation approach ensures that submitting the location information from another user preserves the privacy of all participants of this swapping procedure. The major downside of this approach is, that there are a lot of people required, who use this obfuscation service in order to provide a decent level of privacy. A high level of privacy is only possible in areas with a high users density, as in cities.

In areas with only a few people using the system, it is feasible to perform informed attacks [54]. These attacks are based on some information about the victim and therefore, the attacker can perform some kind auf plausibility attack on the provided locations and check, whether they are feasible. Another approach is to generate fake location information that can be detected afterwards and filtered out. This attack reveals the location of the victim, because all other locations are only padding to trigger the permutation process and are filtered out by the service. As soon a previously unknown location information appears, the attacker knows the location of the victim.

## 3.2 Mobile privacy preserving frameworks

With the goal to reduce the negative privacy impacts of location information, researches as well as software companies proposed and partly introduced privacy preserving-frameworks into their operating systems. These frameworks allow the user to control the transmission of his location information. This section provides an overview about different location privacy improving approaches for modern mobile operating systems. Due to the fact that the default implementations are often very minimalistic, several privacy extensions have developed by independent researchers. These additional extensions provide more privacy preferences respectively features for the user, than the default capabilities of the operating systems. Besides dedicated extensions to the existing mobile operating systems, there are also operating system agnostic approaches, which obfuscate the location information outside of the mobile device.

### Mobile operating systems

The two most important mobile operating systems are iOS, developed by Apple and Android, which is maintained by Google. In 2013 these two operating systems were used on over 90 % of the sold mobile devices [27]. This work only looks at the privacy implementations and privacy extensions for these two platforms, because the usage of all other operating systems, like Blackberry or Symbian, is either declining or they are still in the development phase, like the Firefox OS.

### iOS

Prior to the sixth major release of iOS (iOS 6), the operating system only allowed the user to enable or disable locations services for all applications. In iOS 6 this setting was improved, by allowing the user to enable the location service for each application individually. By default the location services are disabled for every application. Every application, which requests access to the location information, prompts the user to allow the application to access the location information. The user can then decide to access to the location provider for this application or stick with the default setting and deny it.

Besides the general settings, there are also some awareness features. Every time an authorized application obtains a location information, an icon in the status bar indicates, that the location API was queried. The user can then open the preferences application and check, which application requested the location information. This feature does not improve the privacy of the user, but it informs the user about issued location requests and therefore improves the awareness concerning the location privacy of the user. Due to the restrictive policies by Apple, it is impossible to alter the location information functionalities without rooting the device. Nevertheless there are some applications, like Protect My Privacy (PMP), which can be installed on rooted devices.

PMP is an application, which is designed to protect the user's personal information on iOS devices [1]. Besides the location information aspects, this application also improves the privacy settings for contacts or unique identifiers, which are used for advertisement purposes. Additional to the client application, the developers also maintain a server component, which anonymously gathers settings of users and generates usage statistics for research purposes. Based on these statistics, the server provides recommended settings for multiple applications. This application is only available on not authorized third party distribution channels, like the Cydia[3] store, because it does not meet the requirements to be admitted to the official App Store. The Cydia store provides similar capabilities than the App Store, but it enforces less strict development guidelines. To improve the location privacy, this application makes use of the already implemented location privacy settings and adds the functionality to provide a fake location for each application. By means of this extension, the user can submit any arbitrary location to disclose his actual location.

In terms of implementation aspects, this application adds an additional module to the operating system, which intercepts all personal information related data requests issued by applications. The module then generates an appropriate response based on the actual information provided by the operating system as well as the configuration provided by the user and returns the response to the application. This architecture ensures, that the applications cannot obtain any private information stored on the device, unless the user explicitly allows it. Some aspects, like the access to contacts, of this third party application have already been implemented in the latest release of iOS (iOS 7), but there are still deficiencies regarding the location privacy aspects.

### Android

The location privacy settings of current Android releases (Android 4.3) are very minimalistic. The privacy settings only allow the user to globally enable or disable the location services, but this is not possible for each application individually. Besides this global setting, the preferences also provide some functionality to set the location granularity for the applications. The user can choose, whether he wants to use either GPS - satellites respectively Wi-Fi and mobile network based location retrieval systems or both approaches to determine the location of the mobile device. Although this feature is only intended to conserve the battery it can also be used to simulate different levels of location granularity, by choosing the one of the different location retrieval approaches. The GPS approach in urban areas and the mobile network location in rural areas for example reduces the accuracy of the location information and therefore improves the privacy of the user. Besides the minimalistic location privacy settings, the operation system also implements an indicator, which visualizes the location requests. This indicator works very similar to the one implemented in iOS, but it lacks an overview about all applications, who requested the location provider within the last couple of hours.

Currently the only possibility to adjust the privacy settings is to install third-party privacy extensions, like PDroid or PrivacyGuard [55]. Some of these privacy extensions can be obtained from the Google Play Store[4], while others require root access to install them. Besides the additional configuration capabilities, some extensions, like MockDroid [12] implement a mocking functionality to fake locations. The extension proposed by Henne et al. [32] extend the concept of providing only fake location information. Instead of only providing two choices, namely deactivate the location information or provide fake location information, they implement additional

[3]`http://cydia.saurik.com` [Online; accessed 30-November-2013]

[4]`https://play.google.com/store` [Online; accessed 30-November-2013]

algorithms, which can be used to obfuscate the actual location information. These algorithms provide different levels of location granularity and the user can select different algorithms to adjust the granularity according to his requirements. This extension is very useful to empower the people to take care of their privacy, but for an average user, the configuration possibilities of this application might be confusing. The last two applications are based on Cyanogenmod [5], which is a developer friendly port of the Android code basis. Due to the fact, that these two applications heavily interfere with the internals of the mobile operating system, it is not trivial to deploy them on mobile devices.

### Alternative approaches

Besides the different privacy extensions, which can be installed on existing mobile operating systems, there are also alternative location obfuscation architectures. These proposed architectures are either based on theoretical concepts or provide a dedicated environment, which can be used to implement the location obfuscation. Although these architectures provide several feasible approaches, it is generally very hard to implement them on already existing operating systems, without completely revamping the existing architectures.

#### Proxy based

The work by Bellavista [10] proposes a proxy system which can be used to obfuscate the location granularity according to the requirements proposed by the user as well as the service. The proxy consists of a client proxy which is compulsory and a server proxy which is optional, as one can see in Figure 3.6. The client side proxy is deployed on a mobile device and the user configures this proxy by entering a location granularity level, which is acceptable for him. The server side proxy is deployed on the server, which provides the service, and the configuration describes the minimal level of location granularity that is at least required to provide the service. This architecture provides two different procedures to improve the privacy of the user. If the system only contains a client side proxy, the proxy reduces the entropy of the location information according to the configuration provided by the user and then transmits the location information to the server. In the second scenario, when the system consists of a client side proxy and a server side proxy, these two proxies open a secured connection and negotiate a location granularity level based on both provided configurations. The goal is to provide as little information as possible, while maintaining the quality of service. This system is a rather theoretical approach and the authors do not provide a suggestion, how this system can be embedded into currently existing mobile operating systems.

#### Framework based

The framework approach suggests a framework, which has to be provided by the mobile operating system and has to be used by the applications. Such frameworks introduce a more detailed location API than the currently provided ones, which can be used to negotiate the level of location granularity. The Confab toolkit [35] provides such a comprehensive framework, which can

[5] `http://www.cyanogenmod.org` [Online; accessed 30-November-2013]

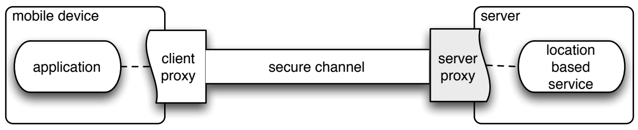


**Figure 3.6:** Proxy based architecture (adapted from [10])

be used by application developers to implement privacy-sensitive applications. The framework consists of a data model, which represents different privacy preferences issued by a user, like the location granularity and of a library, which has to be embedded into applications. The library provides different methods and policies, which can be used to implement the communication among clients and the server. Each component of the system has to implement the library in order to create a location privacy aware system.

### Trusted Third Party based

The TTP approach adds an additional external component to the already existing system. The existing system consists of applications, which are installed on mobile devices, and service providers, which are deployed on external servers. The TTP component is introduced between the application and the service to obfuscate the location information by means of an implemented location obfuscation algorithm. The Figure 3.7 shows the message flow of a location request issued by the server. While the flow of the request message is identical with one in conventional architectures, the response message is routed over the TTP component. This routing process is transparent for the server, so that the server does not discover the application of the obfuscation algorithm. There are several different proposals, which make use of the TTP concept, like the Caspar system [47] or the Privacy Grid [7]. Both systems implement the k-anonymity algorithm (see 3.1) and use similar architectures to preserve the privacy of the user. The major problem of the TTP approach is, that the user must trust the TTP. The TTP could be malevolent and store the location information provided by the user to generate user profiles or use them for other purposes. In such an attack scenario, the location privacy problem is only dislocated from the server to a TTP.

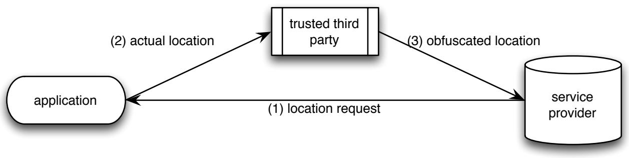


**Figure 3.7:** Message flow using a Trusted Third Party (adapted from [47])

### Summary

Although all of the presented location obfuscation architectures can improve the location privacy for the users, all of them also have some significant disadvantages. While the commercial approaches are intuitive and provide a high level of usability, they lack the functionality to assign different levels of location granularity for each application individually or at least assign a fake location for applications, which require location information. One of the most sophisticated third-party extensions is the selective cloaking extension by Henne [32]. This application allows enough configuration possibilities, but is lacking some features in terms of usability. These usability improvements should support the user either by selecting appropriate default algorithms or clustering similar applications to reduce the overall configuration effort. Besides the usability aspect, most of the applications are built on top of the Cyanogenmod and require a significant amount of effort to be deployed on mobile devices.

The alternative approaches suggest different promising ideas, but it is very hard to implement these ideas on state of the art mobile operating systems. The major problem of the framework approach is, that it has to be implemented by every application developers. Due to the fact that reduced location granularity does not provide any competitive advantage over other applications, most application developers do not care about location privacy and try to get as accurate location information as possible to either generate usage statistics or monetize the user profiles, which includes location profiles. The last approach, the TTP, provides a location obfuscation architecture, which can be added on top of the existing application infrastructure. The addition of a TTP does not require additional changes to the operating system, besides routing the traffic from the applications over the TTP component. Although this approach is promising in terms of implementation for already existing systems, the problem of trust is ultimately shifted from the application server to the TTP component. The major difference between the proxy and the TTP approach is, that the proxy approach is transparent for the applications while in the TTP scenario, the application makes active use of the TTP.

CHAPTER 4

# Algorithm classification

## 4.1 Obfuscation algorithm assessment

The following section provides an evaluation concerning the privacy as well as the plausibility aspects of the previously described obfuscation algorithms. These assessments establish the basis for the succeeding classification. The privacy aspect analysis evaluates behavioral privacy, location privacy and the possibility to avoid sensitive locations, whereas the plausibility aspect deals with the properties whether the location, the trace or the movement speed are plausible. A more detailed description of these categories can be found in the classification section (see 4.2).

### Deactivation

#### Privacy

The deactivation algorithm provides a high level of privacy for the user, because the application or service cannot obtain any location information from the location information provider. This approach, as well as all other location obfuscation approaches only applies to the installed applications on the device and not to the location capabilities of telecommunication providers, as described in Section 2.2.

#### Plausibility

The aspect of plausibility is not relevant for the deactivation approach, because the application or service knows that the user does not provide any location information.

### Random based location information

#### Privacy

Assumed that the algorithm uses a true random number generator, this algorithm does not leak any information about the actual location of the user. The most severe implementation problem of this algorithm, would be the usage of the actual location information as a starting point to derive the generated location information.

**Plausibility**
The fake location algorithm is vulnerable to plausibility checks. There are two different types of plausibility checks. The first is static and evaluates whether the location is feasible for the usage context of the service. The second is one is a dynamic approach that evaluates whether the sequence of given location information points is plausible. This check calculates the distance between two locations and the speed that is required to move between these two locations, within the timespan of the two location recordings. If the speed, which is required to travel between the generated locations, is higher than the highest possible movement speed, it is very likely that the generated location information are not genuine and the service can detect the usage of an obfuscation algorithm.

### Chosen location information

**Privacy**
The level of privacy for this algorithm is lower than the random location, because it is likely to choose a location, which is influenced by the actual location. Assumed that the application or service collects multiple coordinates provided by the user, it may conclude some information about the user, like the country or the city the user is living in.

**Plausibility**
The level of location plausibility of the generated location provided by the chosen location algorithm is stronger than the one generated by the random location algorithm. Assumed that the user chooses only plausible locations, it is hard to detect whether the location information is genuine or fake. This makes it almost infeasible for applications, which only request the location information once. Applications, which request the location information more than once, can check the movement plausibility. The level of the movement plausibility depends on the frequency, which is used by the application to retrieve the location information. Given that different geographic locations are reported multiple times during one session, it is easy detect whether the location information is fake or genuine. For the service provider chosen locations might look like the user is jumping between different locations, which may not be possible in such a short time. In such a situation, the generated location is not plausible at all.

### Enlarging the radius

**Privacy**
The level of privacy for this location-obfuscation algorithm depends on selected radius R2, whereas a larger radius increases the privacy, as one can see in Figure 3.1. Given that an attacker records a large set of location information from a victim, who remains on the same place for a long time, he can conduct an inference attack. A location inference attack flags all generated locations on a map and then selects the spot that lies in the center of all flagged locations. Due to the fact that all locations are equally distributed within the circular shape, it is very likely

that the actual location is in the center of all reported information. This attack is also feasible for a moving victim, but then it becomes harder to distinguish between actual movements of the victim and artificial random noise.

**Plausibility**
The plausibility of this obfuscation algorithm also depends on the selected radius R2. Assumed that the user chooses a small radius R2, the algorithm generates rather accurate location information, which are similar to the actual location and therefore plausible. When the radius R2 gets larger, the reported locations might not be as plausible as they should be. It may occur that the user seems to jump around unrealistically within the given area or reports locations that are normally not possible, like from military areas.

### Shifting the center

**Privacy**
The shifting the center algorithm maintains the relative speed between the single generated locations and provides a high level of privacy regarding the actual location of the user. There is one rather theoretical attack approach against this algorithm. This attack analyzes the walking or driving pattern by the user and possible areas to derives the actual location of the user [20]. Assuming the attacker has a detailed trace of information without any gaps, he can construct a trace. The attack then tries to match the trace with the map, near the generated location. If there is a match, he can deduce the actual geographic location of the user. This attack is feasible if the provided track is long enough and the generated location is near the actual location. Therefore this attack scenario is only feasible for the local shift algorithm.

**Plausibility**
Due to the plausible sequence of location information it is hard to detect whether the generated location information is genuine or not. There is one approach that requires semantic information about the area, like obstructed areas, rivers or not accessible areas like military areas. Given that the provided location information is fine grained enough to be projected on a map, one can try to check whether the sequence of location information is plausible, in terms of available streets. When the sequence of location information differs from the possible routes on the streets or the recorded trace goes through obstructed area, like private buildings, it can be assumed that the provided trace of location information is not genuine. This plausibility check can be applied in both the local shift and the arbitrary shift approach.

### Double obfuscation

**Privacy**
This approach increases the privacy for the user by summing up the advantages of the two previously described algorithms. The combination prevents the pattern matching attack very effectively, under the assumption that the increased area is large enough, to generate a random trace, which cannot be mapped to available streets. The generated trace consists of multiple

randomly distributed points, which cannot be used by the service provider to extract any personal information, besides an approximate actual location.

**Plausibility**
The plausibility of the combination of these two algorithms is weaker than for the shift algorithms, because the random element can render not feasible location information respectively traces of location information. To generate more plausible obfuscated tracks, one has to restrict the random locations to publicly accessible property. Depending on the level of sophistication, one can additionally limit the random location information to those points that are accessible within the time frame between the two location announcements. This algorithm then provides a feasible track of location information and it is hard to verify whether the location information is correct without any external context information, like semantic information.

### Discretization to points on a grid

**Privacy**
The level of privacy for this algorithm depends on the width of the grid, whereas a larger distance between the intersection points of the grid provides a better level of privacy. This obfuscation algorithm is mainly designed to absorb the movement information, like the actual route or the movement speed. Therefore this algorithm is suitable to maintain the behavioral privacy of the user.

**Plausibility**
Depending to the granularity of the grid and the movement speed of the user, it can be easy for a service to detect the usage of this algorithm. Assumed that the user chooses a wide-meshed grid and walks slowly by foot, the generated location will remain the same for a long time. After the long timespan, the generated location jumps to next point on the grid, within a not feasible timespan.

### Time based obfuscation

**Privacy**
This algorithm does not alter the actual locations reported by the user, but it obfuscated the movement behavior. Due to the delayed reporting of the locations, the service can only extract a randomized behavior pattern. Therefore this algorithm is very useful to be applied in behavioral sensitive scenarios, like within shopping malls, to obfuscate the personal shopping manners.

**Plausibility**
This algorithm creates plausible location information traces, because it only alters the movement speed between single generated coordinates. As long as the time spans between two generated coordinates are equal or longer than the least time required to move between these two coordinates, this algorithm provides plausible traces and the service cannot detect the application of this algorithm

### Landmarks

**Privacy**
This algorithm improves the privacy by relocating the actual location to nearby popular points of interests, which are also used by many other persons. After applying this algorithm, it generates a very common location information, that is exchangeable with those of other people, who either use this obfuscation algorithm or simply are currently standing at such locations and also use LBS. This common locations reduce the entropy concerning the individual information provided by the user.

**Plausibility**
Due to the fact that this algorithm only uses much frequented POI, it always generates plausible locations. The only weakness for this algorithm concerning the plausibility is the sequence of generated locations regarding the behavioral aspects. These algorithm jumps between the single POI, which might be suspicious for a service, similar to the one of the discretize to a grid algorithm.

### Avoid areas

**Privacy**
This obfuscation approach does not alter the location information, unless the user is visiting a sensitive location, like medical facilities or nightclubs. The algorithm hides such sensitive information, because they might be interesting for insurance companies and entrepreneurs or prevents any problematic check-ins on location-based social networking websites, such as Foursquare[1]. Besides the privacy improvements for sensitive areas, this algorithm does not improve the location privacy of the user in regular areas.

**Plausibility**
Due to the fact that this algorithm only alters a small subset of location information, it is very hard to detect the application of this algorithm. Therefore the few generated locations are very plausible.

### k-Anonymity

**Privacy**
The k-anonymity algorithm provides a good level of privacy, assumed that there are enough people using the service. If there are only a few people using this service, the level of permutation might be too low to provide the desired level of privacy. Therefore this approach is only suitable in crowded areas or for theoretical scenarios, but it is not useful in practical scenarios, due to the organizational overhead.

[1]`https://foursquare.com` [Online; accessed 30-November-2013]

**Plausibility**
The outcome of this algorithm is a set of location information, whereas the single location information is not connected. Due to this pertubation, this approach generates not plausible movement pattern, although the location information itself is plausible, because it is a real location reported by a user, who participates in the pertubation process. When the application or service only requests the location once, this algorithm generates plausible locations, but when there are multiple location queries, the movement plausibility might suffer.

## 4.2 Classification of location obfuscation algorithms

In order to compare different location obfuscation algorithms, it is necessary to create a formal model or extract essential characteristics, which can be used to evaluate the properties of algorithms. Due to the mixed nature of different obfuscation approaches respectively goals it is very hard to provide a common formal model for all algorithms. Such a formal model is only feasible for similar algorithms, like the relevancy notation for spatial algorithms [4], but it is hard to compare the effectiveness of location environment aware algorithms with unaware ones. Especially the introduction of semantic aspects makes it impossible to create a universal formal model. There are some projects, which try to quantify the effectiveness of different obfuscation approaches, based on their robustness against generic attacks. Shokri et al. [53] developed the Location-Privacy-Meter, which allows the user to implement different location obfuscation algorithms and evaluate them against a generic attack scenario. The evaluation is based on statistical methods and the results can be used to compare different algorithms. Besides the generic attack scenario, they also provide some specific attacks, like localization attacks, which can be used to evaluate algorithms for dedicated scenarios.

Besides the quantification and formalization approaches, there are also more abstract approaches to classify location obfuscation algorithms. The classification approach by Wernke et al. [60] classifies different obfuscation approaches based on their ability to guard protection goals against predefined attacks. The authors defined three basic protection goals: user identity, spatial information and temporal information, which have to be protected by the obfuscation approaches. These three goals are then evaluated against common attacks. Based on the evaluation they generate a matrix, which can be used to lookup the effectiveness of different algorithms. This approach only provides a very high level comparability among different algorithms.

Another approach was taken by Anderson et al. [2] who tried to evaluate the appropriateness of different obfuscation approaches for different use case scenarios. They cover anonymity, spatial obfuscation and temporal obfuscation, but also rather pragmatic aspects like classical security or the protocol, which is used to transmit the location information. To evaluate the approaches, they define four use case scenarios, which are very common in the area of LBS: point-of-interest, social networking, collaborative sensing and routing scenarios. These scenarios differ regarding to the amount of users and receivers as well whether they are used in an online or in an offline setting. The actual classification maps different location obfuscation approaches onto the generated matrix, which is defined by the protection goals and the scenarios. This matrix should help software engineers to select an appropriate algorithm for their use case.

Both classification approaches as well as the quantification approach only look at the privacy aspects of location obfuscation algorithm. Due to the improved capabilities regarding the validation of obfuscated locations on the service side, the selection of the appropriate algorithm depends on the plausibility of the generated location information. The classification approach provided by this work, takes both aspects, namely the privacy as well as the plausibility one into account, to provide a better basis of decision-making for software engineers, who implement privacy aware software solutions. The goal of such privacy preserving solution is that the usage of any obfuscation algorithm cannot be detected by the service.

## System model

The system, which is used for this classification approach, consists of two components, namely a location aware mobile device and a LBS provider. The mobile device is able to retrieve the actual location and can execute arbitrary LBS on the device, such as routing services or location based recommendations. The service provider offer location aware services and use the submitted user related information to either improve the service or try to monetize the location information by generating profiles about users. The monetization is carried out by either targeted advertising or by selling the user profiles to a third party. The goal of the service provider is to only record valid location information to create valid profiles. Therefore he has employed plausibility checks, to detect not genuine locations respectively location tracks. The user on the other hand wants to use the LBS, but does not want to reveal any aspects of his personality or preference, like shopping preferences or often visited locations. To achieve the goal of the user, the location information has to be obfuscated as much as possible.

## Protection goals

The classification uses four different categories of protection goals. Two goals are applied from the user's point of view and two goals deal with implementation aspects, which are relevant for engineers who design privacy aware applications. The user oriented properties are divided into different subcategories, which refine the overall protection goal, whereas the developer oriented properties are rather binary decisions, which can be used to rule out not appropriate algorithm.

### Privacy

The privacy category describes, how the algorithm improves the privacy for the user, who uses different location aware services. Each of the three specific privacy properties described below, is summarized with one of the three states: high (H), medium (M) and low (L). The high state means that the algorithm handles the specific privacy aspects very well. The low state means that there are hardly any privacy related improvements compared to the actual location information. The medium option states, that there are some improvements concerning the privacy, but the algorithm is not suitable for privacy sensitive scenarios.

**Location privacy**
The location privacy category describes whether the algorithm is capable of generating an arbi-

trary location, which is not related to the actual location of the user. Depending on the use case scenario of the user or the desired level of privacy, different levels of privacy are possible. The possibilities range from completely unrelated locations to a fixed shift of the actual location.

**Behavioral privacy**
This category covers the privacy aspects of a sequence of location information, which are transmitted to a service within a short time. Based on these multiple location information, the service can derive a trace. This trace can be used by the service to derive several meta-information, like the movement speed or whether the user stayed on a specific spot for some time. The behavioral category assesses, whether the different algorithms are capable to obfuscate the behavior of the user and therefore replace the actual user specific movement pattern by a generic respectively random one.

**Avoidance of sensitive locations**
The last category of the privacy section describes, whether the algorithm can take care of sensitive locations, like hospitals or nightclubs. The major use case of this category is to provide a realistic trace of location information, but omit those locations, which might be problematic for the user. There is only one algorithm, which is explicitly designed to take care of such locations, but there are also other algorithms, which cover this requirement implicitly.

**Plausibility**

The plausibility category describes how feasible the generated traces are and whether it is possible to detect the usage of an obfuscation algorithm by the service. The category also consists of three subcategories, which can obtain one of the three states: high (H), medium (M) and low (L). The high state means that the generated location is very plausible and it is not likely that the service detects the usage of an obfuscation algorithm. On the other hand the state low denotes that the application of the obfuscation algorithm is obvious, whereas the middle state is be suitable for services that do not implement any sophisticated plausibility analysis for the received location information. The three subcategories describe different plausibility checks, whereas the latter categories are harder to comply than the first ones, because the succeeding subcategories are basically extension of the preceding ones.

**Plausible areas**
This category describes, whether the generated location is within a plausible area. The plausibility of an area depends on the service, but for normal scenarios it is save to say that any reported locations from the middle of the ocean, on top of high mountains or within military restricted areas are not plausible. This plausibility check can be used to detect arbitrarily generated locations based on a random number generator. The usage of this plausibility analysis is only possible, if the service is capable to query topological information from a knowledge base, which contains semantic information about an area. This semantic information are required to contain the composition of the ground as well as political borders and obstructed areas, like military ones.

**Plausible traces**
The second category handles a sequence of reported locations and checks whether the sequence is plausible. To perform this analysis, the plausibility services require a knowledge base, which contains all available transportation possibilities among different locations. This can be a map that provides the information about streets, flight schedules or ferry connections or any other knowledge base that contains such information. The easiest implementation of this plausibility analysis is the usage of a route planner, which checks if there is a direct connection between these two locations.

**Plausible movement speed**
The last plausibility category extends the previous category by additional context sensitive information, like the maximum speed for cars on a specific street, a flight schedule or even real time traffic information to detect, whether there are any traffic jams on the streets. The plausibility algorithm checks, whether it is possible to travel between two reported locations within the timeframe, they are reported in. If the reported timeframe is significantly smaller than the time needed between these two locations, it is obvious that the provided location information is not genuine.

#### Implementation

This category describes the ease of implementing the algorithm, whereas the ease is defined by the amount of code or whether there are any external resources required. Most algorithms are based on a simple spatial relocation, which do not require any external resources. Besides these simple algorithms there are also some algorithms that require location aware information, like nearby POI or sensitive locations, like hospitals. The most complex implementation scenario includes the collaboration with other users, which is for example required for the K-anonymity algorithm. The classification for this category also consists of three stages: high (H), medium (M) and low (L), whereas high classifies those algorithms that are very easy to implement and the other states are used to indicate the increasing required effort.

#### Infrastructure

The last category states two attributes of the algorithms, which might be relevant for software engineers when selecting a suitable algorithm. The first aspect is, whether there are any other users required who also use the algorithm. The second aspect describes whether the algorithm requires any external services, like a map provider or any other location sensitive knowledge bases. The classification options for this category are simple, namely yes (Y), when there are any other users or services required or no (N) if the algorithm does not require these two types of external interaction.

### Classification

This section provides the classification of the location obfuscation algorithm against the protection goals and the development properties, as one can see in Table 4.1.

| | Deactivation | Random location | Chosen location | Enlarging the radius | Local shift | Arbitrary shift | Double obfuscation | Discretization to grid | Time based obfuscation | Landmarks | Avoid areas | K-Anonymity |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Level of privacy* | | | | | | | | | | | | |
| location | H | H | H | H | M | H | M | L | L | M | L | M |
| behavior | H | H | M | M | L | L | H | H | H | H | L | H |
| sensitive locations | H | L | H | L | L | L | L | L | L | M | H | L |
| *Plausible locations* | | | | | | | | | | | | |
| plausible areas | - | L | H | H | M | M | M | H | H | H | H | H |
| plausible traces | - | L | L | M | M | M | M | M | H | M | H | M |
| plausible movement speed | - | L | L | L | H | H | L | M | H | M | H | M |
| *Implementation* | | | | | | | | | | | | |
| ease of implementation | H | H | H | M | H | H | M | H | H | L | M | L |
| *Infrastructure* | | | | | | | | | | | | |
| standalone solution | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | N |
| external services | N | N | Y | N | N | N | N | N | N | Y | Y | Y |

**Table 4.1:** Classification Table

The goal of this classification is to provide an overview about the effectiveness and applicability of location obfuscation algorithm in the area of tension between privacy aware users and services, which are based on alternative monetization schemes, like the selling of user profiles. This classification can also be used to select an appropriate location obfuscation algorithm for the different obfuscation categories described in the succeeding section (5.1).

The classification is based on the qualitative evaluation, which can be found alongside every algorithm, stated in the algorithms section (see 4). The entries in the Table 4.1 provide an indicator about the effectiveness of the algorithm concerning the stated protection goal. The developer-oriented entries in this table indicate the infrastructure requirements for implementing the described algorithms. They are intended to be used as a simple knock out criteria to easily eliminate algorithms, which are infeasible to implement due to potentially given resource restrictions.

The missing entries in the Table 4.1, in the deactivation column were omitted on purpose, because it is impossible to evaluate the plausibility of geographic coordinates, when there are none.

The diagram in Figure 4.1 provides an alternative representation of the classification. It accumulates the evaluation of the user oriented protection goals and can be used to get an overview about commonly used obfuscation algorithm and their privacy respectively plausibility assets.

The first stacked column represents the privacy aspect, while the second column represents the plausibility aspects. The different evaluation results from the classification in the table (H, M, L) are encoded by different heights, whereas H is represented by the largest block and L by the smallest one. This encoding allows any software engineer to denote privacy respectively plausibility friendly algorithms at a first glance. It also shows the tradeoffs by the different algorithms and can be used as a basis of decision making to select potential algorithms for a concrete use case scenario.

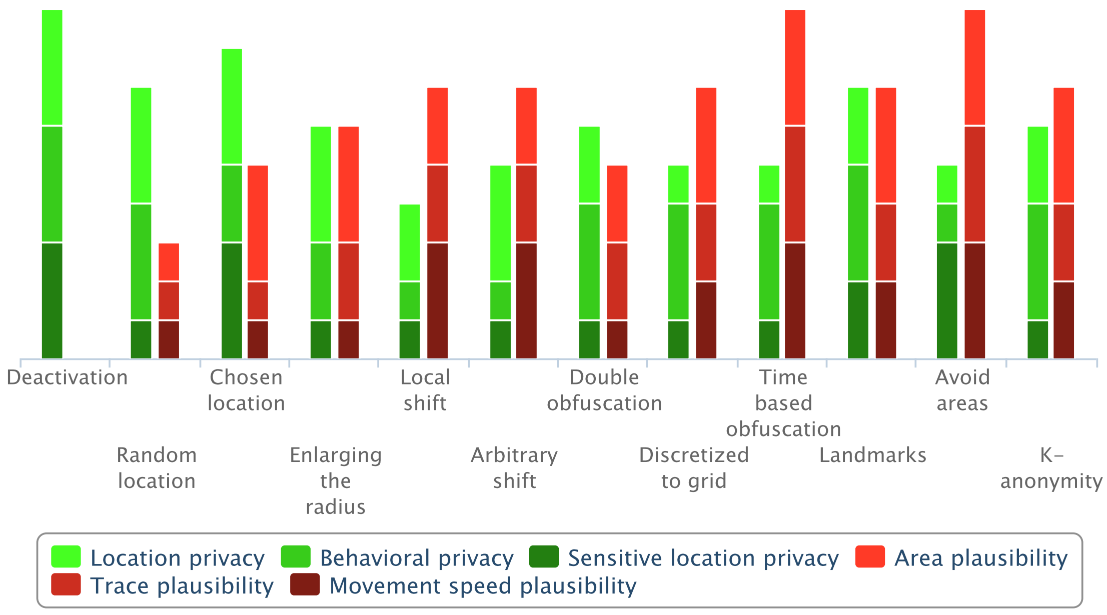


**Figure 4.1:** Aggregated privacy and plausibility of location obfuscation algorithm

CHAPTER 5

# Proposed obfuscation categories

## 5.1 Obfuscation categories

The major problem with state of the art implementations of obfuscation frameworks is, that these approaches either allow a very coarse approach, where one can only set the location obfuscation algorithm for all applications or a very fine-grained approach, where the user has to configure every application individually. Both approaches are not ideal for the user. On the one hand, the coarse grain approach is not applicable, when the user uses different location aware applications, which require a different level of granularity. On the other hand the fine-grained approach is not feasible, because it is too much work for most users and therefore they do not use it. This work proposes, that the solution for this problem should be something in between, where the user can cluster different applications into categories and assign specific location obfuscation algorithms for these categories. The assignment of applications to categories can either be carried out manually or automatically based on an existing knowledge base, like the Google Play Store[1].

This section proposes different categories of applications, which require the same level of location granularity, uses different scenarios to describe these categories and suggests appropriate location obfuscation for the categories.

### No obfuscation

The no-obfuscation category is a very basic category, which contains all applications that are useless as soon the location information is obfuscated. The emergency notification and the routing scenario describe two typical use cases, which are carried out by applications that are assigned to this category.

An emergency notification system is often implemented in cars, which automatically informs authorities or rescue workers that an accident occurred and transmits the location of the accident. Such systems can reduce the waiting period between an accident and the arrival of

[1]`https://play.google.com/store` [Online; accessed 30-November-2013]

help dramatically. In an emergency situation, the privacy aspects are only secondary. Such a situation requires that the reported location information is as accurate as possible to reduce the time, which is required to find the victim.

Routing applications are often used to obtain a direct route from one location to another one without prior knowledge of the route. The routing services are provided with the actual location and the target location. Based on these two locations and context information about routes, the service calculates a route. Once the user starts driving on this route, the current location is constantly updated and the routing service informs the user about the succeeding path. The information is presented to the user either by announcements, when the user has to turn left or right, or by plotting the route on a display. Routing systems are available for all kinds of transportation means, like ships, cars and even for pedestrians. These routing systems require a spatial as well as temporal accurate sequence of geographic location information to provide a useful service. It is not possible to obfuscate the location or the time property of the location information and maintain the quality of service, like accurate information when to turn left or right.

Due to the necessary requirements posed by the two scenarios in this category, this category does not allow any modification of the location information by an obfuscation approach.

### Minor local obfuscation

The minor location obfuscation category is designated to cluster different applications, which provide fine grained location aware recommendation services, like the retrieval of POI.

The location aware POI retrieval is a very common use case for people in foreign cities. The starting situation consists of a need, like to withdraw cash at an ATM, to refill gas at a gas station or to eat something at a restaurant, but the user does not know where to find such facilities. In this scenario, the LBS retrieves all nearby POI and presents them to the user. Depending on the type of the POI, an obfuscation algorithm with different parameters can be applied without reducing the quality of service. The search for an ATM requires very precise information, because the distance between two ATM is very short. Assuming if the location would be obfuscated too much, the returned list of the nearest ATM machines would differ from the result generated with the actual location. When the POI is a more distinct place, like a fine restaurant, the location information does not need to be as accurate as in the ATM scenario. The distance among these fine restaurants is significantly longer than for the ATM machines and a minor obfuscation of the actual location will not alter the order of the returned restaurants.

The service of applications in this category is robust to a small local location obfuscation, like the enlarging the radius algorithm (see 3.1), where the radius is augmented at most 300 meter in the ATM scenario and up to 1000 meter in the restaurant scenario.

### Fine-grained semantic obfuscation

The fine-grained semantic obfuscation category consists of all applications, which are used for social networks.

Some social media platforms allow the user to attach the location information to their posts or to provide a dedicated check-in functionality to announce that the user is currently staying in a specific location. These checkins are often made from public locations, like universities, nightlife locations or other cultural POI. Social networks often provide a symbolic naming functionality, where the submitted location coordinates are assigned to the nearest known POI. This nearby location is very likely the location, where the user also is located. Although the social networks retrieve the symbolic name for the location, they also store the exact location coordinates [50]. This additional information might reveal further details about the user, specifically within large locations, like a university. In order to increase the level of privacy, the user could also insert the symbolic name on his mobile device or provide a set of static coordinates, like the entry of the building. This decrease of accuracy does not reduce the quality of service for the user, but he does not submit the actual location information to the social networks.

The obfuscation of this category is predestined to be carried out with the landmarks obfuscation algorithm (see 3.1), whereas the algorithm uses a fine grained net of POI.

### Corse-grained obfuscation

The coarse-grained obfuscation category contains all applications, which do not require very accurate location information, like weather applications.

The goal of the weather retrieval is to obtain information about the weather situation in a specific area, like a city. In this use case scenario it is sufficient to submit any location within a specific border, like the city boundary. The result will always be the same, because most weather services implement an algorithm, which matches location to symbolic names, like cities or districts and then return the weather information for this symbolic name. This use case scenario allows the user to obfuscate the actual location without any degradation of the quality of service.

There are two different location obfuscation algorithms, which are suitable for this category. The first algorithm is the semantic based landmark algorithm (see 3.1), where the algorithm only uses one POI for each city. The second algorithm, which requires less infrastructure, is the algorithm that discretizes geographic coordinates to a grid (see 3.1). Due to the fact, that the latter algorithm requires fewer infrastructure, like a knowledge base of landmarks, it is proposed as the most appropriate algorithm for this category.

### Behavior maintaining obfuscation

The behavior maintaining obfuscation category is a very specific category, where the user wants to maintain the sequence of individual locations and the movement pattern, but he wants to obfuscate the actual location. This requirement can be found in sport applications, which are used to track sport activities, like running.

Modern pulse monitors often implement a GPS receiving component, which records a sequence of location information. This sequence of location information can be used to analyze the activity afterwards by the athlete respectively can be published on social networks. The most important information for any location information within this sequence is the accurate temporary and relative spatial information relating to the preceding geographic coordinate. The absolute location information might be nice, but it is not relevant when the athlete is only running on a flat environment. This may not apply to a hilly environment, because an athlete requires more energy to run up a hill than on a flat surface.

This sequence of location information can be obfuscated, as long as the algorithm maintains the relative time and distance changes. Although most pulse monitors are designed to work offline, there are some services where the user can upload the track to an online platform during the sportive activity to share the activity with friends or colleges. In such situations it might be desirable to obfuscate the actual location of the route and only want to provide the distance or the shape of the route of the run. The actual route can reveal the place of residence of the user, which is very likely to be the starting position.

The behavior maintaining requirement can be implemented by the shift algorithm (see 3.1) where the actual location is shifted for a fixed distance but the relative movement respectively movement-speed between two geographic coordinates is preserved.

### Selective obfuscation

The selective obfuscation category contains all applications, which only require location information within predefined areas, like the workplace or the home place.

One example for such an application is a location aware reminder. The basic use case for location aware reminders is, to remind the user, when he reaches a specific location or leaves one. A typical use case for such a location-based reminder is to remind the user to buy milk, when he leaves his work place. The location information of the user is constantly recorded to inform the user, when the specific location is reached or left. In this scenario it is sufficient to report only those coordinates to a server, which are near the defined locations and hide all other locations. The hiding of the location information between the relevant locations does not reduce the quality of service for the user, but it still improves the privacy of the user, because the service cannot track all movements made by the user.

The appropriate location obfuscation algorithm for this category is an inversion of the avoid areas (see 3.1) algorithm. The inversion of this algorithm only provides location information, when the user is within specific areas and suppresses it as soon the user leaves these locations.

### Deactivate location information

The last category contains all kinds of applications, which require unnecessary location requests. These requests are posed by different services, although they are not necessarily required by the application to provide the service. Very common examples for such services are games, which do not require the location information. They are only needed to generate usage statistics or to submit the location to advertisement providers. The user does not require these requests and they only decrease the privacy of the user.

In this scenario it is suitable to deactivate the location information or to provide fake coordinates, when a service insists on the information.

CHAPTER 6

# System design and Implementation

This work proposes two different solution approaches, which are designed to resolve the issues, which are stated in the problem description section (see 1.2). Due to the fact that it is impossible to resolve all issues with one location privacy framework solution, it is necessary to design and implement different approaches, which solve as much issues as possible.

The chapter starts with a description of the initial architectural situation. The initial situation represents a typical communication flow for modern mobile operating systems. After the description of the initial situation, two different approaches are presented. The first approach introduces an additional external component, which obfuscates the messages on the way from the mobile device to an external server. This approach can be deployed to all modern mobile operating systems, which provide the functionality to route the traffic over a proxy. The second approach introduces a modification to the operating system and can be deployed on Android based operating system.

## 6.1 Initial situation

All modern mobile operating systems implement a very similar architecture in terms of location information retrieval and communication with external servers. Figure 6.1 shows a typical workflow, where the user triggers a location based service from an external service provider within an application and the application presents the result to the user. The workflow only deals with the retrieval of the location information and the communication between the application and external servers. All other aspects, like the internal logic of the application are represented by the generic method call *createRequest*, because they are not directly relevant for the location privacy domain.

The user initiates the workflow by requesting some kind of location-based service from an application, which is installed on a mobile device. The mobile device issues a request to retrieve the location information from the location provider, like GPS or Wi-Fi. The location provider gathers the actual location information and returns it to the application. Based on the location information, the application then creates a request, which is transmitted to the dedicated server. The server performs different operations and generates a machine-readable result (e.g. Weather:Vienna:-1:celsius:snowing), which is then sent back to the application. At the end of the workflow, the application processes the machine-readable result and presents it to the user.

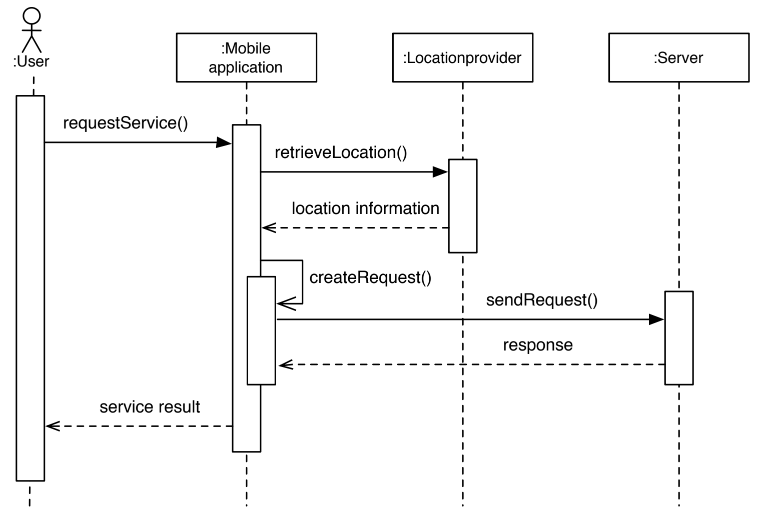


**Figure 6.1:** Sequence diagram of an uninfluenced information flow

## 6.2 Proxy based interception

The proxy based interception approach makes use of an external component, which intercepts all outgoing traffic from the mobile device. This external component acts as a proxy and is capable of extracting the payload from ordinary HyperText Transfer Protocol (HTTP) traffic as well as breaking Secure Sockets Layer (SSL) communication respectively Transport Layer Security (TLS) encryption to extract the payload. The payload contains all kind of information, like the location information of the user, which is transmitted from the mobile application to the dedicated server. The proxy filters the incoming payload based on a predefined rule-set and forwards all messages that contain location information on to an obfuscation service. The obfuscation service modifies the location information based on a predefined configuration and returns the message to the proxy, which passes it on to the originally targeted server. Due to this special treatment of outgoing messages, the server only receives that location information, which is conform with the privacy policy of the user.

The proxy based interception approach is actually a combination of the TTP and the proxy based approach. From the mobile device point of view, it is a TTP approach, but from the application point of view it is a proxy based approach, because the obfuscation is transparent for the applications.

## Architecture

The system design for the proxy-based interception requires two additional subsystems compared to the initial setup. These additional systems are deployed outside of the mobile device on dedicated hardware. The following two sections provide an overview about the modifications to the service request workflow and a short description of the additional required components.

### Workflow

The workflow generated by the proxy extension for a simple location based service-request and its corresponding result can be seen in Figure 6.2. The first part of the workflow is identical with the workflow in the initial situation (see 6.1). The user triggers a service request in a location based application. To perform this request, the application retrieves the actual location information from the location provider and creates a request, which is dedicated to be transmitted to the server. Up to this step, the two workflows are identical, but the succeeding ones differ from the original scenario.

When the application tries to send the request directly to the server, an interceptor intercepts the request. This interceptor analyses all messages, which are sent from the mobile device, to filter out messages, which are known to come from location based applications and contain location information.

If the interceptor finds such a message, he passes it on to the message utility component, which extracts the location information from the message. Beside the location information, this message utility component also extracts the unique Identifier (ID) of the application, which created the request. The unique ID and the location information are returned to the interceptor who passes them on to the obfuscation component.

The obfuscation component retrieves the obfuscation configuration from the database, based on the unique ID. This obfuscation configuration consists of the required algorithm and other obfuscation relevant properties. Based on the configuration, the obfuscation component then selects the designated location obfuscation algorithm and obfuscates the location information. The result of this obfuscation procedure, the obfuscated information, is then returned to the interception component.

Once more the interceptor makes use of the message utility and replaces the actual location information with the obfuscated one. The modified message now contains the obfuscated location information and can be passed on to the dedicated server. As soon as the server receives the message, he generates a result and encodes it in a machine-readable structure. This structure is then directly returned to the application. The application interprets the machine-readable result and represents the result to the user in a human-readable manner.

The workflow above describes a successful message interception. This is only possible, when the location based application and the structure of the messages, which contain location information, are registered in advance. For all other messages, the interception component performs a wild guess search, where the messages are scanned for the keywords *long* and *lat*, which are commonly used to transmit location information. Additionally the messages are also queried with the actual geographic coordinates. If any message contains these keywords, the interception utility stores the message as a basis for further signature creations.

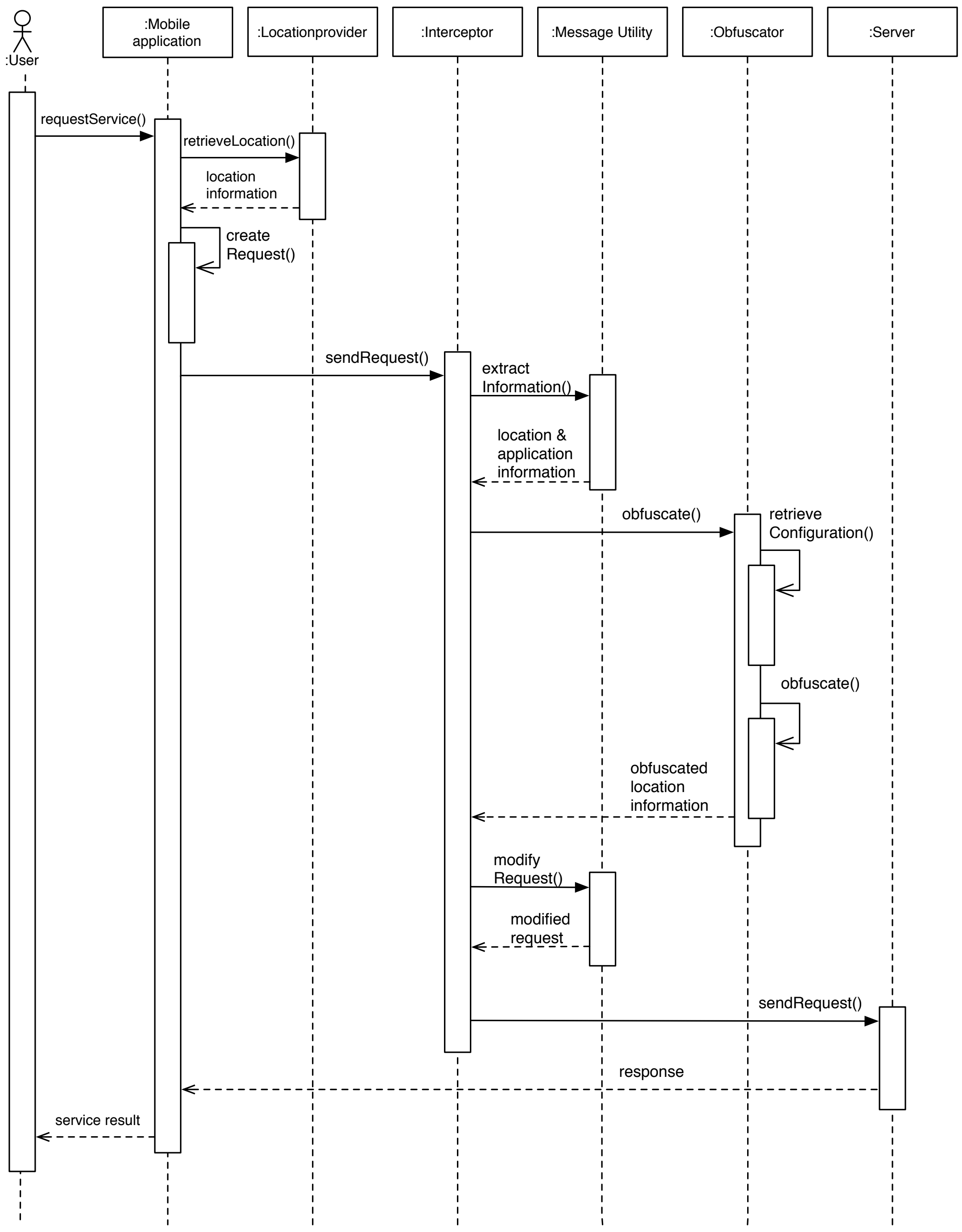


**Figure 6.2:** Sequence diagram for proxy architecture

## Components

The proxy based obfuscation approach requires additional components, which are shortly described in this section. Figure 6.3 represents the system architecture of this approach on a component level. The system architecture consists of three subsystems and the external servers, whereas two subsystems are only dedicated to the interception functionality. On the mobile device, only the application and the location provider are deployed. The interception system and the obfuscation system are deployed on a dedicated hardware, which is connected over the network with the mobile device and the server. This section describes the additional components in detail and explains their internal functionality.

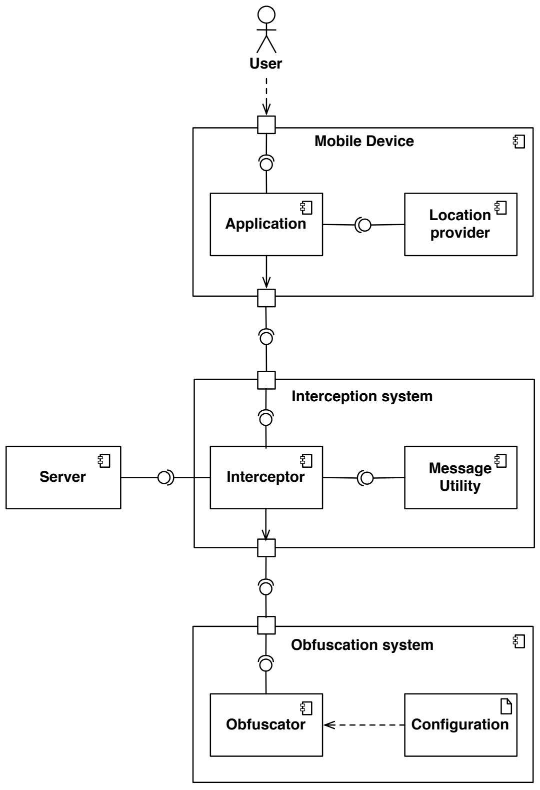


**Figure 6.3:** Component diagram for the proxy architecture

## Interceptor

The interception component implements the proxy principle [52], which gathers all outgoing HTTP traffic from the mobile device in a centralized location. This centralized location is capable of analyzing and modifying the gathered traffic. After the analysis and modification activities are carried out, the traffic is forwarded to the initially targeted servers. This prototype requires a proxy infrastructure, which is capable of handling HTTP as well as HyperText Transfer Protocol Secure (HTTPS) traffic. The handling of HTTPS traffic requires additional effort, because the proxy has to break the encryption of the message. This most common approach to break a secured communication, is a Man-In-The-Middle (MITM) approach, where the attacker introduces an additional certificate for the proxy [17]. In a scenario without the proxy, the application would encrypt the message with the key from the server, represented by the green lock in Figure 6.4. The reply from the server is encrypted with the key of the client, which is represented by the white lock. In the proxy scenario, a new key, which is derived from the proxy certificate, is introduced. The user mistakes the red key, which was introduced by the proxy as the genuine key of the server and encrypts the message with this key. The proxy then decrypts the message, optionally modifies it and encrypts the message again with the actual key. The server accepts the message and replies to the proxy by using the new key. When the proxy receives the reply, he re-encrypts it with the actual one of the application and returns it to the application. Although this attack is well known, a lot of applications do not implement certificate pinning and therefore their traffic can be intercepted and modified.

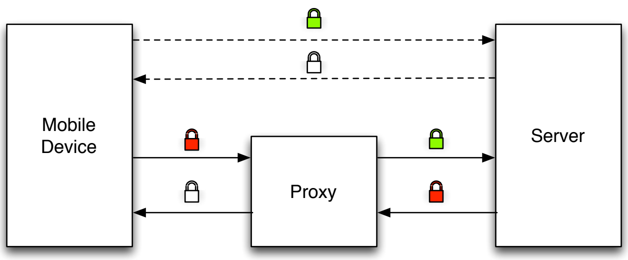


**Figure 6.4:** Certificate usage for a HTTPS interception approach

The implementation of this design approach uses the mitmproxy[1], which is capable of handling both HTTP and HTTPS traffic. Additionally this proxy also provides an API that allows a developer to integrate custom Python scripts. These Python scripts are called for every intercepted message and can be used to determine whether the message contains location information, to extract the location information and trigger the obfuscation process.

### Message utility

The message utility component is a collection of three different python scripts, which are capable of analyzing respectively modifying intercepted messages. Besides the three scripts, the message utility component also contains a list of different signatures. These signatures are used to associate messages with applications and extract the location information.

The first script analyzes new messages and tries to determine whether they contain any location information and if they originate from a known application. This detection is carried out based on the list of predefined signatures. A signature is a unique sequence of characters, which can only be found within the messages of one application, like a unique server Uniform

[1] `http://mitmproxy.org` Last accessed: 30.11.2013

Ressource Locator (URL). Such signatures can be found either in the URL, which is used to send the request, or within the body of the message.

Assuming that one of the signatures matches, the script returns the unique ID of the application, which sent the message. If none of the signatures match, the script conducts a search for the keywords *long* and *lat*, which are often used to encode location information. If these keywords can be found within the message, the message is stored for further analysis. In the course of the further analysis steps, the message is analyzed and a new signature can be added to the signature list, so that the message utility can correctly handle message from this application in the future.

The second script extracts the location information from the message. Based on the result of the first script, the second script applies regular expressions on the message to extract the encoded location information. This location information is then returned and passed on to the obfuscation component.

The last script is used to replace the original location information with the obfuscated one. This script simply looks for the actual longitude respectively latitude within the message and replaces the values with the obfuscated ones. After this replacement the script returns the modified message, which can then be passed on to the server.

The challenge for this component implementation is, that there is no standardized message structure. Every application generates an individual message and pursues a different approach to encode the location information. Therefore it is impossible to generate a general extraction or replacement rule for the location information. This is the reason that every application requires an individual set of signatures, which are used to extract the required information and replace the location information.

#### Obfuscator

The purpose of the obfuscation component is to transform the actual location information into a privacy friendly one. Therefore this component consists of several obfuscation algorithms and a set of configurations. The configurations define the assignment of applications to categories, the assignment of algorithms to categories, and the optional configuration parameters for the algorithms.

Whenever the obfuscation component retrieves an obfuscation request from the interception component, the obfuscation component performs the obfuscation procedure which is described in algorithm listing 1. The obfuscation request consists of the actual location information and an unique ID of the application. Based on the ID, the category and the configuration are retrieved from the configuration file. The configuration contains the assigned algorithm and optional parameters, which are then used to load the appropriate obfuscation algorithm. Once the algorithm is loaded, the parameters are set and the algorithm obfuscates the actual location information. The result of this procedure, the obfuscated location information, is returned to the interception component, to be inserted into the intercepted message.

The architecture of this component supports the addition and removal of new obfuscation algorithm during runtime. It is possible to simply copy a new compiled algorithm into the dedicated folder and the component loads it on runtime. This feature guarantees a high uptime of the obfuscation component, because it can be extended without any required reboots.

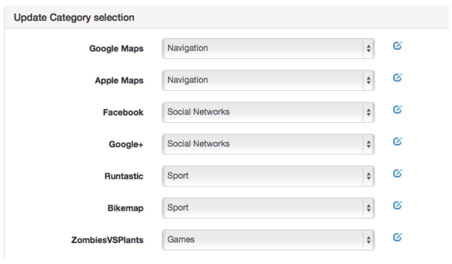
```
**Data**: actualLocation, uidOfApplication
**Result**: obfuscatedLocation
category $\leftarrow$ null
configuration $\leftarrow$ null
**if** realLocation $\neq$ *null* **then**
    location $\leftarrow$ actualLocation
    category $\leftarrow$ categoryFromConfigurationFile(uidOfApplication)
    configuration $\leftarrow$ configFromConfigurationFile(category)
    algorithm $\leftarrow$ configuration.getAlgorithm()
    algorithm.setProperties(configuration.getProperties())
    location $\leftarrow$ algorithm.obfuscate(actualLocation)
    return location
**else**
    return null
**end**
```

**Algorithm 1:** Obfuscation procedure for Proxy interception

### Configuration

The configuration component caters for two purposes. The first purpose is to store the assignment of applications, categories and other configuration parameter permanently. The configuration is stored in a JavaScript Object Notation (JSON) file, which can be updated externally during runtime. There are three possibilities to update the configuration. The first one is to replace the old JSON file with a new one. This approach can be used to import a new set of default assignments. The second one is to modify the JSON file with a text editor and the third one is to modify the configuration by means of the Graphical User Interface (GUI), which also represents the second purpose of this component.

The configuration GUI is an optional component, which helps the user to adjust the configuration and to export signatures for the message utility. All changes made within the GUI are stored within the JSON file so that the obfuscation component can retrieve the updated configuration. The user can assign applications to categories, as one can see in Figure 6.5 or create new categories.



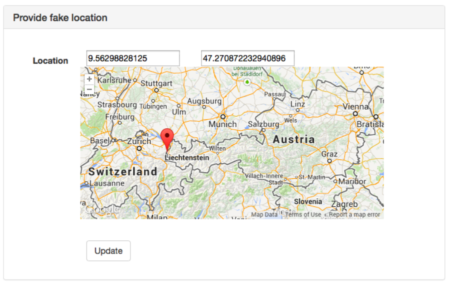
Update Category selection

| | |
|---|---|
| Google Maps | Navigation |
| Apple Maps | Navigation |
| Facebook | Social Networks |
| Google+ | Social Networks |
| Runtastic | Sport |
| Bikemap | Sport |
| ZombiesVSPlants | Games |

**Figure 6.5:** Category assignment with the configuration GUI

Additionally, the user can update the signatures and the regular expressions for an application within the GUI. These signatures are required for the message utility component, which classifies the messages and extracts the location information. Although the signatures are strictly speaking not necessary for the obfuscation process, they are also stored within the JSON file to maintain only one configuration source. The GUI also provides the functionality to generate Python code, based on the stored signatures, which can be directly imported into the message utility.

Besides the category and algorithm assignment, the GUI provides an easy to use interface to adjust parameters for the obfuscation algorithms. The Figure 6.6 shows for example the geo-location selection interface, which can be used to retrieve the coordinates, for the fake location algorithm. The user can drag the pin to the desired location and with a click on the Update element, the selected location is stored in the JSON file.


Provide fake location

Location: 9.56298828125 | 47.270872232940896

**Figure 6.6:** Algorithm adjustment with the configuration GUI

## 6.3 Operating system level interception

An alternative to the proxy based interception approach, is to intercept the location information on an operating system level and to modify it before it even reaches the application, which makes use of the information. This approach requires a modification to the operating system itself. It is only feasibly for operating systems, which provide the source code for the system, like Android. For closed source systems, like iOS, it is required to reverse engineer the source code and alter the system's binaries to apply such a modification. Any reverse engineering process is not only against the license agreements issued by the mobile operating system provider, but it also requires a lot of effort, as the work by Egele [24] shows. Egele implemented a system, which is designed to find data leaks on the proprietary iOS. To achieve the goal, he had to apply similar modifications to the operating system, which are also necessary for the interception of the location information. Due to the fact, that reverse engineering projects are not feasible to provide a stable interception mechanism, this work focuses only on the Android operating system. The actual modification of the operating system consists of a wrapper implementation for the location API, which modifies the location information based on the policy provided by the user. In the operating system level interception setup, the application only receives modified location information, which improves the privacy of the user.

**External influences**

The implementation of this prototype is based on the Android Location Privacy project [31, 32]. This project provides a framework with the ability to intercept calls to the location API and modify the location information based on a privacy policy. The framework already provides several location obfuscation algorithms and it is easy to add additional algorithms. The configuration of this framework is carried out on an application level. Every newly installed application is assigned to a global default obfuscation algorithm and the user has to assign a specific obfuscation algorithm to each application. The configuration is carried out by means of a graphical extension to the already installed settings application on Android and the configuration data is stored in an encrypted database, which is based on a SQLite implementation.

The authors made the source code publicly available[2] under the Apache License, Version 2.0. The implementation provided by this work adapted the source code to be compatible with CyanogenMod version 10.2, which is based on Android 4.3 (Jelly Bean), modified some parts of the framework and implemented additional components. A detailed description of the changes can be found in the components section (see 6.3).

## Architecture

The architectural design for the system level interception includes three additional components compared to the initial setup. All of these additional components are deployed on the mobile device and the message flow gets more complex than in the initial setup (see 6.7). The following two sections describe the extensions of the required workflow and provide a short description for the additional components.

### Workflow

This section provides a detailed textual description for the sequence diagram, which can be seen in Figure 6.7. The trigger for this modified workflow is the same as the one in the initial setup (see 6.1). The user requests a location based service from an application, which is deployed on the mobile device. To provide this service, the application tries to retrieve the location information from the location provider. As a result of the modification to the mobile operating system, the interception component, intercepts the location request and stores the unique ID of the requesting application internally. The interceptor then issues a new location information request and retrieves the actual location information from the location provider. The result of this newly issued request and the unique ID of the application are then forwarded to the obfuscation component.

The obfuscation component is designed to modify the actual location information according to a default privacy policy or a privacy policy issued by the user. To simplify the required configuration effort for the user, the obfuscation component clusters applications into distinct categories. When the obfuscation component retrieves a new application ID, the application ID is submitted to a category detection component, which assigns the application to one of the existing categories, based on an application category knowledge base. The category assignment

[2]`https://github.com/bhenne/android-location-privacy` Last accessed: 30.11.2013

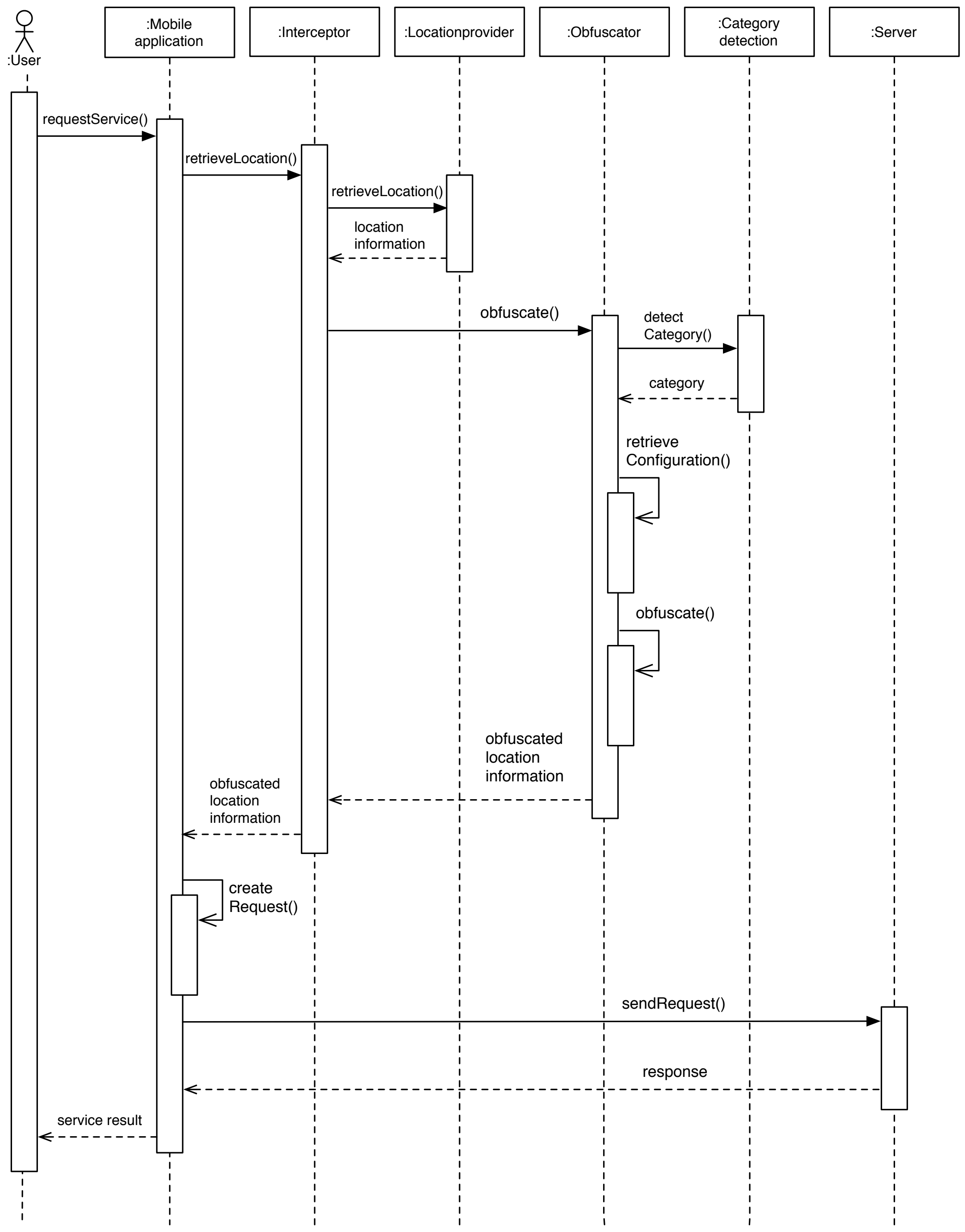


**Figure 6.7:** Sequence diagram for system the interception architecture

is then stored into the internal database. This assignment procedure is only carried out once for every application.

Based on the category assignment, which was either retrieved from the category detection component or from the internal database, the obfuscation component retrieves the obfuscation configuration for the specific application from the internal database. This configuration data contains the assigned obfuscation algorithm and optionally additional parameters, like the obfuscation radius or fake location information, which are required to obfuscate the actual location information.

The actual location obfuscation is carried out, by calling the obfuscation component with the retrieved configuration and the actual location information. Based on the configuration and actual location information, the obfuscation component generates a new location information. This generated location information is then returned to the interception component, which returns it to the issuing application.

The remaining part of this workflow is the same as for the initial setup. The application processes the generated location information and generates a request, which is transmitted to the server. The server then generates a machine-readable result, which is interpreted by the application and the application returns the information to the user.

The execution of this workflow is carried out transparently for the user. This means that the user does not notice any differences, between the workflow in the initial situation and the workflow for the system level interception. The requested result is also identical, assuming that an appropriate location obfuscation algorithm was selected.

Besides this major workflow, there are also other less important activities, which are required to use this system in a real world environment. These additional activities are executed, when the user wants to update the privacy policy by modifying obfuscation parameters or to reassign single applications to another category.

## Components

The component diagram in Figure 6.8 provides an overview about all additional components, which participate in the service retrieval process (see Figure 6.7). All of the additionally required components are deployed on the mobile device. This section provides a short description about these additional components, the design decisions and an internal process description.

### Interceptor

The interception component is designed to intercept all location information requests issued by applications. This functionality is implemented by replacing the actual location provider with a wrapper for the actual location provider. The interfaces of the location provider and the wrapper are identical, so that installed applications do not recognize the replacement.

Whenever an application issues a location information request, the wrapper receives the location request. The wrapper then retrieves the actual location information from the location provider itself and triggers the obfuscation procedure of the obfuscation component. The structure of the result provided by the obfuscation component is identical with the structure of the

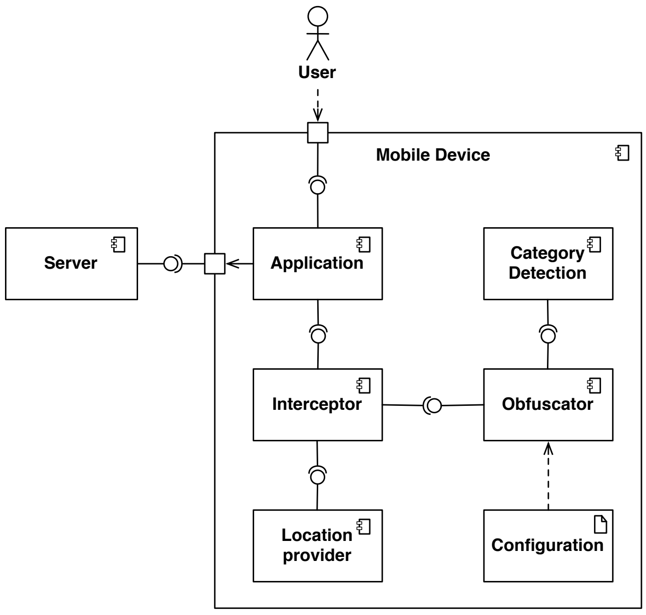


**Figure 6.8:** Component diagram for system the interception architecture

actual location information, so that the wrapper can simply return the obfuscated location information to the application. From the application point of view, the only difference between a location information request on a genuine operating system and a modified one is the execution time. The additional obfuscation activities require slightly more execution time. The amount of time, which is required to obfuscate the location ranges from 0.0008 seconds up to one second. This additional amount of time is acceptable for a location based service request. The actual required amount of time differs as a result of the complexity of the obfuscation algorithm. While some algorithms, like the deactivation algorithm (see 3.1) or the fake location algorithm (see 3.1) are very fast, other algorithms, like the geocoding algorithm (see 3.1) requires external services and needs additional time to communicate with these services.

The implementation of the interception component is identical with the interception component used in the Android Location Privacy project.

### Obfuscator

The obfuscation component provides the functionality to obfuscate a location, based on the actual real location and the unique identifier of an application. The capabilities of the obfuscation component are represented by the algorithm listing 2, which demonstrates the core functionality of the obfuscation component in pseudo code. The core functionality only shows configuration retrieval and the actual obfuscation. Additional caching aspects, which improve the performance of the component, are omitted in this listing for the sake of readability. They only improve the performance of the component, but are not essential for the functionality.

At the beginning of every obfuscation procedure, the obfuscation component is triggered by the interception component and obtains the actual location information and the unique ID of the application. If the actual location information does not represent a valid location, the obfuscation procedure terminates and returns an empty location element. If the location information represents a valid location, the procedure tries to obtain the category assignment for the application from the database. Assuming that this is the first time, an obfuscation procedure is triggered for the given application, this application is not assigned to a category. In this case, the obfuscation component triggers the category detection component to retrieve the appropriate category for the application. The retrieved category is then stored in the database to speed up future obfuscation procedures. After obtaining the category assignment for the application, the procedure retrieves the obfuscation configuration from the database. This configuration consists of an algorithm assignment, optional parameters for the algorithm, and whether the obfuscation is enabled for the category. When the obfuscation is enabled, the procedure retrieves the algorithm, adjusts the algorithm properties and obfuscates the location. The output of this obfuscation procedure is then returned to the interception component. The result can be the actual location information, if the obfuscation was disabled or the obfuscated location information.

```
Data: actualLocation, uidOfApplication
Result: obfuscatedLocation
category ← null
configuration ← null
if realLocation ≠ null then
    location ← actualLocation
    category ← categoryFromDB(uidOfApplication)
    if category = null then
        category ← retrieveCategory(uidOfApplication)
        storeCategory(uidOfApplication, category)
    end
    configuration ← configFromDB(category)
    if configuration.get(status) = true then
        algorithm ← configuration.getAlgorithm()
        algorithm.setContext(frameworkContext)
        location ← algorithm.obfuscate(actualLocation)
    end
    return location
else
    return null
end
```

**Algorithm 2:** Obfuscation procedure for the system level interception

In the Android Location Privacy project, the location privacy manager represents this component and the functionality of the two components is very similar. The modifications are limited to the introduction of the category structure and the obfuscation procedure for this prototype obtains the configuration of the assigned category instead of the individual configuration of an application.

### Category detection

The category detection is a new module, which does not exist in the underlying implementation. The task of the category detection component is to automatically assign a newly installed application to an appropriate existing category. The automatic category assignment is based on the category classification of the Google Play Store[3]. On the Google Play Store every application is assigned to a category, like Transportation, Social, Weather, Sports and many more. This category assignment is used, to assign the application to an appropriate granularity category.

The actual retrieval of the category assignment from the Google Play Store, is not trivial, because Google does not provide any API to access these category assignments. The implementation of this component retrieves the information by using a web scraper to access the dedicated website of an application and extracts the actual information with a predefined regular expression.

Although this approach works as intended, it has one major downside. Every time Google updates the layout structure of their Google Play Store, it is very likely that the regular expression has to be adapted according to the new structure. This approach is not completely satisfying, but it is the only possible approach, unless Google provides an API, which allows developers to retrieve meta information for applications.

### Configuration

The configuration component consists of two parts. One part covers the backend functionality of storing category assignments and the configuration of these assignments, while the other part provides a GUI for the user, who wants to adjust the configuration. The backend is built on top of a SQLite database, whereas the content of the database is secured by an additional encryption layer, which implements Advanced Encryption Standard (AES) encryption and several helper methods to provide an easy interaction with the GUI.

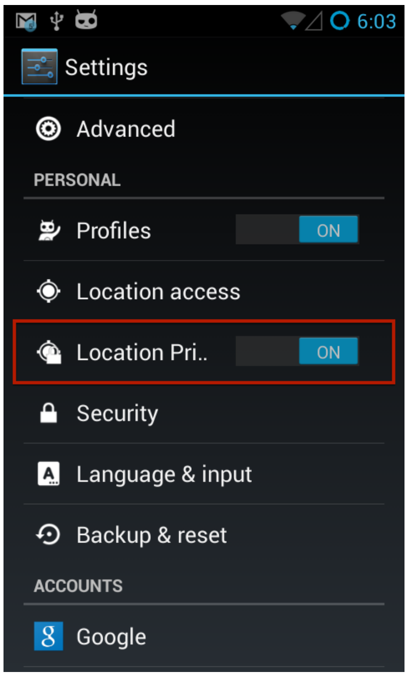
The configuration GUI is embedded within the default settings application, as one can see in Figure 6.9. On the highest level of the settings application, the user can enable or disable the location interception, by adjusting the switch.

With a tap on the entry in the settings menu, the user enters the configuration menu, where he can find a list of all currently installed location based applications and two menu entries. The first entry opens the category configuration (see Figure 6.10), where the user can create new categories or assign a new obfuscation algorithm to a category. In the category configuration menu, the user can also enable or disable single categories by setting the switch to the desired position. The configuration of this switch is then respected by the obfuscation component.

The second entry provides a GUI to configure the default algorithm. The default algorithm and the default category represent two fallback settings, which are used when it is not possible to assign the application to an appropriate category or the user has not defined an algorithm for a category. The last screenshot (Figure 6.11) provides an overview about the configuration capabilities of an exemplary algorithm. Although each algorithm already provides reasonable default parameters, the user can adjust these parameters according to the personal requirements.

[3]`https://play.google.com/store` Last accessed: 30-November-2013

The configurations are stored individually for each category, which means that the user can use the same algorithm with different parameters for different categories.



6:03

Settings

Advanced

PERSONAL

Profiles ON

Location access

Location Pri.. ON

Security

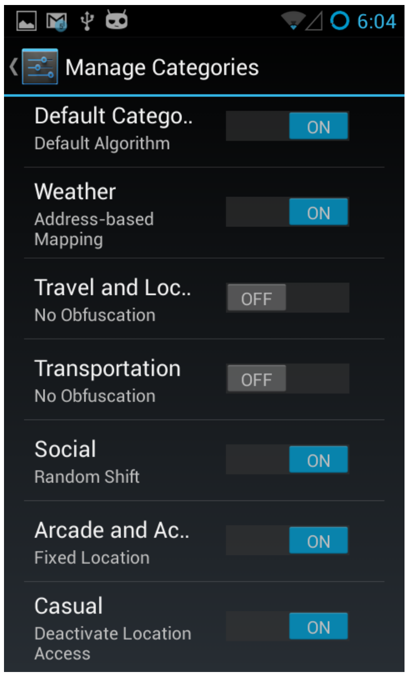
Language & input

Backup & reset

ACCOUNTS

Google

**Figure 6.9:** Configuration integration



6:04

Manage Categories

Default Catego.. ON
Default Algorithm

Weather ON
Address-based Mapping

Travel and Loc.. OFF
No Obfuscation

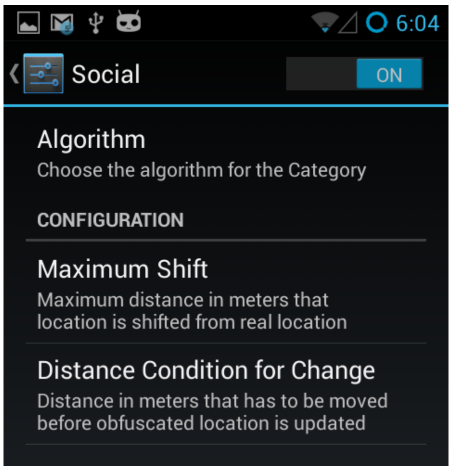
Transportation OFF
No Obfuscation

Social ON
Random Shift

Arcade and Ac.. ON
Fixed Location

Casual ON
Deactivate Location Access

**Figure 6.10:** Configuration of Categories



6:04

Social ON

Algorithm
Choose the algorithm for the Category

CONFIGURATION

Maximum Shift
Maximum distance in meters that location is shifted from real location

Distance Condition for Change
Distance in meters that has to be moved before obfuscated location is updated

**Figure 6.11:** Individual configuration for the algorithms

CHAPTER 7

# Evaluation and Results

## 7.1 Applicability of proxy based interception mechanism

The evaluation of the proxy based interception mechanism is based on an analysis of the top 100 free applications of the Google Play Store in December 2013. The goal of the evaluation is, to determine how many applications use location information and whether the proxy interception mechanism is applicable for widespread applications.

### Test setup

The hardware components for this evaluation consisted of a Nexus S device and two Raspberry Pi Model B as one can see in Figure 7.1. The Nexus S was operated with CyanogenMod 10.2[1] and the Internet Protocol (IP) address of the Raspberry Pi, which acts a a proxy server, was added to the proxy settings, on the Nexus S. Besides the modification to the proxy settings and the installation of the proxy certificate, which is required to intercept HTTPS traffic, no additional modifications to the operating system were made. On the first Raspberry Pi (Proxy Pi), an instance of the mitmproxy[2] and the message utility component was deployed on top of a stock Raspbian operating system[3]. This Proxy Pi acted as the proxy server, respectively interceptor for the messages. The second Raspberry Pi (Obfuscator Pi) conducted the obfuscation operations and stored the configuration. Therefore the configuration GUI and the obfuscation service were deployed on a Jetty[4] instance, which was also running on the Raspbian operating system.

These three hardware components were placed in the same Local Area Network (LAN) and every component had access to the Internet, although the communication of the Nexus S was routed through the Proxy Pi.

[1]`http://www.cyanogenmod.org` Last accessed: 30.11.2013
[2]`http://mitmproxy.org` Last accessed: 30.11.2013
[3]`http://www.raspbian.org` Last accessed: 30.11.2013
[4]`http://www.eclipse.org/jetty/` Last accessed: 30.11.2013

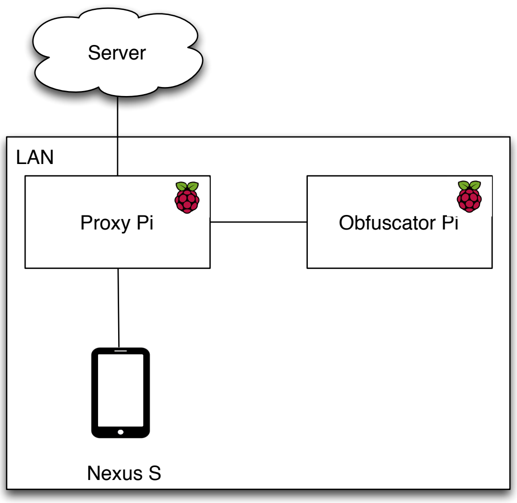


**Figure 7.1:** Setup for evaluation of the proxy based interception

### Execution of the evaluation

The evaluation consisted of three phases. In the first phase, all top 100 free applications were installed on the device. The entire list of the installed applications can be found in Table A.1. That table contains the names of all applications as well as the detailed figures of the evaluation. In the course of the installation phase, all applications, which do not request access to the location API were noted.

The succeeding phase was used to identify whether the applications transmit the location information to an external server and if so, how this location information is encoded. To identify the location information usage of the application, the application was started and interaction events were triggered. The proxy recorded all messages, which were sent since the start of the application until the application was terminated. The message utility analyzed the messages, whether they contain the keywords *long* and *lat* or the current coordinates. If these keywords were present, the message and the request URL of the message were stored in the configuration. This basic heuristic offers a suitable approach to identify the majority of all messages, which transport location information. The messages of the remaining applications were either identified by querying the messages with the actual coordinates respectively by manual analysis. In the following, unique signatures based on the application target hostnames were created. The stored messages were then analyzed and unique signatures for the application detection and the location information extraction were defined and stored within the configuration. This process was executed for all applications which requested access to the location API.

In the last phase, the applications were started again and the same events as in the first phase were triggered. During this phase it was evaluated, whether the signatures worked as intended or whether any countermeasures against the modification of messages by a proxy were deployed by the application developers. Additionally it was evaluated, whether the service works as intended or if the selected location obfuscation is too strong and the returned result is not usable.

## Evaluation results

Among the 100 applications, there were 43 applications, which requested access to the location API, but only 31 of them actually transmitted the location information to the server. This means that there are twelve applications, which request access to the location API, but actually do process the location information.

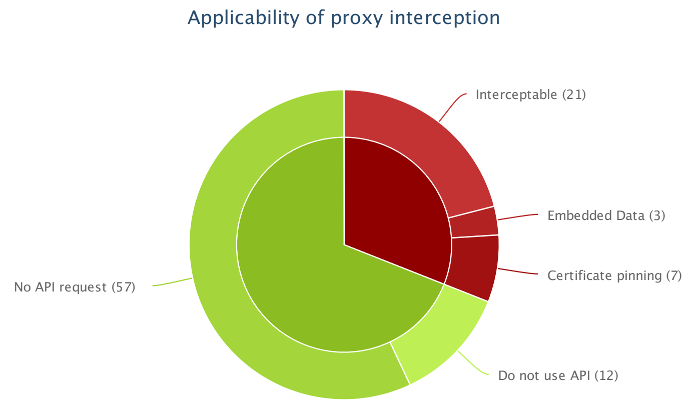


**Figure 7.2:** Usage of location information and applicability of proxy interception

Among the remaining 31 applications, which transmit the location information to an external server, only half of them actually provide location-based services. The other half only uses the location information either for statistical purposes or as a parameter for ad networks. The latter one is a very common use case within the analyzed applications.

Concerning the application of the proxy interception approach, there are three result categories. The majority consisting of 21 applications, worked as intended. This means, that the proxy could intercept the location information, modify it, pass the modified message on to the server and the application receives a correct reply. For 10 applications this was not possible. The majority of the failing applications deployed a certificate validation procedure, like certificate pinning, which checks if the correct certificate of the server respectively client was used. In these cases the HTTPS interception approach is not possible and the location information cannot be modified with this approach. Besides the certificate check, there was another problem, which averted the modification of location information for outgoing content. Three applications provided photo enhancement features and the possibility to upload these enhanced photos to an external server. In this case, the location information is stored within the Exchangeable Image File Format (EXIF) information of the image file. The current implementation is not capable of modifying this information and therefore the proxy based interception approach is not applicable for those applications. This limitation however is rather a implementation issue than a conceptual one.

The evaluation of these applications has shown, that eight out of 21 applications only use the location information for advertisement purposes. Therefore they transmit the location information, besides other privacy relevant information, to an external server. There are only a couple of different advertisement servers, which are used by the majority of the applications. Due to the limited amount of different advertisement providers, a couple of signatures can be defined, which are applicable for multiple applications. These signatures are only triggered when advertisement servers are called and therefore the application does not influence the structure of the messages going to an advertisement server. Under the assumption, that all applications, which use the location information to obtain advertisements, are assigned to the same category, these generic signatures can be deployed without any side effects. For further iterations of this interception mechanism it might be useful to compile a list for commonly used advertisement providers, which can be shipped as an initial configuration.

## 7.2 Applicability of system level interception mechanism

The evaluation of the system level interception approach consists of two parts. The first part covers a functional evaluation of the extended interception mechanism. In the second part, the applicability of this interception mechanism was evaluated against two real world scenarios. These two scenarios cover static as well as dynamic aspects of LBS.

### Test setup

The evaluation was executed on a Nexus S with a patched version of Cyanogenmod 10.2[5] installed. The patches deploy the system level interception as well as the configuration capabilities to the device. For the static scenario, the application *Raiffeisen Meine Bank*[6] was used. The recording of the movement scenario was executed with the application *My Tracks*[7].

### Execution of the evaluation

In the static scenario, the user wants to retrieve the nearest ATM, but does not want to reveal his actual location. To evaluate this scenario, the application *Raiffeisen Meine Bank* was used to retrieve a list of nearby ATM. The application was assigned to a category, which uses the enlarging the radius algorithm (see 3.1). This algorithm increases the radius of possible generated locations up to 250 meter. At the beginning of the evaluation, the location obfuscation was disabled and the actual location information was used, to retrieve the nearest five ATM. For the succeeding retrievals, the obfuscation was enabled and the retrieval was executed ten times. Each time the obfuscation component generated a new location information, which was transmitted to the service provider. The nearest five ATM of these generated locations were also recorded. The list of all recorded ATM can be found in Table B.1.

The movement scenario consists of the recording of a running activity. These recordings are composed of multiple single location information. These locations are assigned to a time

[5]`http://www.cyanogenmod.org` Last accessed: 30.11.2013

[6]`https://play.google.com/store/apps/details?id=com.isis_papyrus.raiffeisen_pay_eyewdg` Last accessed: 30.12.2013

[7]`https://play.google.com/store/apps/details?id=com.google.android.maps.mytracks` Last accessed: 30.12.2013

stamp and then stored within a storage component. Based on the location information and the timestamps, the route can be reconstructed respectively analyzed any time after the information was recorded. For this evaluation the application *My Tracks* was assigned to a category, which uses the shifting the center algorithm (see 3.1). The algorithm shifts the actual information ten kilometers, so that the reported location is located within a park. The location of this park does not contain any location sensitive information. During the run, the application recorded a sequence of obfuscated location information and stored them within the application. The run was repeated on the next day for the same route to record the actual route. These two routes can be seen in Figure 7.4, which contains two screenshots of the application *My tracks* for each recording.

### Evaluation results

In the course of the evaluation of the two scenarios, the already existing obfuscation functionality, provided by the basic project, as well as the usability improvements were evaluated. The additional usability improvements, which automatically cluster different applications into categories, as well as the already existing obfuscation functionality work as intended. This functional evaluation was conducted by installing the two used applications and the category assignment algorithm assigned them correctly to their designated categories.

The results of the static scenario show, that the lists of the nearest ATMs overlap, although the location was modified. In the static scenario, the location was modified ten times and each modified location was submitted to the service. The generated locations are flagged in Figure 7.3 with red pins, while the mauve one represents the initial location. The Table B.1 shows the five nearest ATM machines for every location, which were returned by the service. The result shows, that for all but one location, at least two of the nearest ATMs were selected, which are also the nearest ones for the initial situation. All other ATMs listed in the table are within maximum 500 meter of the initial location. Given that the ATM density is very high in the center of an urban area and that the nearest ATM machines are overlapping for most locations, it can be said, that it is feasible to use an enlarging the radius algorithm with the radius of 250 meter and still get the same or a very similar result set while improving the personal location privacy dramatically.

The goal of the movement scenario is to evaluate a situation where the user wants to obfuscate the actual location, but wants to maintain the relative location changes. The result of this scenario consists of two sequences of location information. The first one describes the actual location while the second one describes the obfuscated one. While the first one can be used for private analysis, that latter one is suited to be published on social networks. As the Figure 7.4 shows, the shapes of the two tracks are identical, but the location of the shape is shifted for the fixed distance. The obfuscated track is located within a recreational area and seems to be a cross-country run, which is equally feasible compared to the actual one, which takes place on streets. Due to the feasibility of the obfuscated track, the obfuscated track can also be posted on social networks to brag about the sportive activity without revealing the actual starting location. The starting respectively terminating location is often the actual living address of the user and therefore it is a privacy sensitive location, which should be kept private.

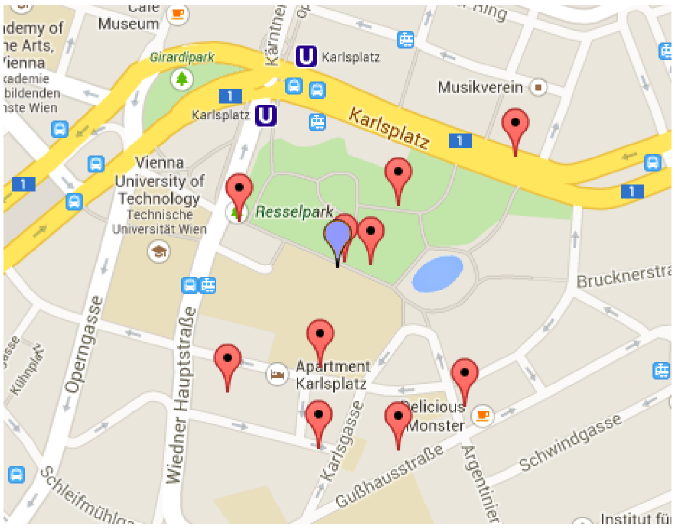


**Figure 7.3:** Extending the radius obfuscation applied ten times to one location

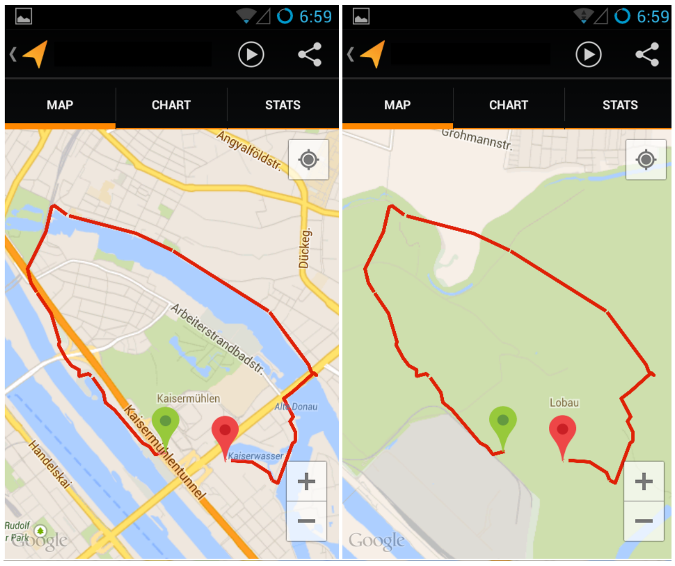


**Figure 7.4:** Shifting the center obfuscation applied to a sequence of coordinates (original track on the left, obfuscated one on the right)

CHAPTER 8

# Discussion

In this section the major findings of this thesis are discussed. The findings range from the rather theoretical classification of the algorithms to the evaluation of the practical implementations of the interception mechanism. The goal of this section is to provide an answer to all the research questions, which are posed in the introducing chapter (see 1.3). Further this chapter discusses consequences of the findings as well as limitations to the currently implemented approaches.

## 8.1 State of the art obfuscation techniques

Based on the first research question (RQ1), an analysis of state of the art obfuscation techniques was conducted. The literature research has shown, that the obfuscation aspects have to be divided into two parts. The first part covers the interception of the location information and the second part deals with the actual obfuscation of the location information. In an ideal scenario, the first part should be needless, because the mobile operation system developers should already provide such functionality. This functionality can be used to obfuscate the location information according to the privacy policy issued by the user. Due to the lack of such capabilities, additional effort is currently required to obfuscate the information without the knowledge of the mobile operation system provider or even the mobile operation system.

**Information interception**
There are several approaches, which can be used to modify the location information. Each of these different approaches caters to different requirements. Some approaches, like the framework approach or the trusted third party approach require the application developer to implement the obfuscation framework within the application. They rely on the assumption, that application developer care about the location privacy of the user, however most approaches modify the location information without the notification of the application developer.

These approaches intercept the location information at some point between the location information provider on the mobile device and the application respectively on the way to the server, which makes use of the information. An interception at an early stage, for example between the location information provider and the application requires modifications to the operation system itself. These modifications are only feasible for research purposes, but they are not feasible to be deployed on mobile devices for everyday usage, because they require a lot of knowledge to be deployed. Nevertheless, the most important advantage of this approach is, that the application never receives the actual location information and the interception system does not require any external components.

Another approach is to intercept the information on the way between the application and the server, which uses the location information. From the user's point of view, the setup process for this interception mechanism is easy, because it is only necessary to set up a proxy for the mobile device and add the certificate of the proxy to the system. Beside the easy setup procedure, this interception approach is operation system agnostic. That implies, that the proxy based interception approach can be deployed on modern operation systems without any additional technical modification required. The major disadvantage of this approach is, that this approach is not applicable to all applications due to countermeasures deployed by the application developers and it requires a lot of configuration effort to be set up by the developer.

The best solution for the interception problem, would be that mobile operation system developers either provide an API, which can be used to register obfuscation algorithms or an GUI, where the user can configure already implemented algorithms. This approach is very similar to the system level interception approach, with the only difference that the operation system provider officially supports the interception. Until this official interception capabilities are introduced, the proxy interception mechanism is the only feasible approach, because the setup process does not require any advanced technical skills.

**Location obfuscation**

For the actual obfuscation of the location information, there are several obfuscation algorithms available. These algorithms range from very simple approaches, which perform a spatial transformation of the location information up to very complex algorithms. The complex algorithms often include semantic information about the environment or try to generate plausible tracks [21]. While the simple algorithms normally only care about the privacy of the users, some of the complex algorithms also consider plausibility aspects. They make it harder to detect the usage of an obfuscation algorithm.

A very common problem for obfuscation algorithm is, that they are seldom tested within real world environment. Most of these algorithms were designed on a theoretical foundation, which is only based on assumptions. These algorithms are only evaluated within purposely constructed scenarios. These scenarios are only designed to support a positive evaluation of the proposed algorithm. There are no large-scale surveys, which evaluate the applicability of different obfuscation algorithms in the daily use. Without such an evaluation it is hard to assess the applicability and feasibility of the algorithms regarding their capabilities in a real world scenario. Currently it is only possible to select potential algorithms based on theoretic models.

Due to this lack of any practical experience with obfuscation algorithms, it is hard to define state of the art algorithms, because simple algorithms, which might be older, may even be better suited, than complex ones. This work performs a qualitative evaluation of several algorithms and tries to select appropriate algorithms based on this evaluation. This selection can be considered as a selection of state of the art algorithms, which are applicable for a large range of scenarios and not only edge cases, like in some research papers that propose new algorithms.

## 8.2 Countermeasures against location obfuscation frameworks

Based on the second research question (RQ2), different countermeasures against existing location obfuscation frameworks were evaluated. Each of the two categories introduced in the previous section has to deal with possible countermeasures.

**Information interception**
The first type of countermeasures targets the interception of the location information. These countermeasures in this area are designed to prevent respectively detect any modifications to messages, which are sent over an untrusted network, like the Internet. They check the genuineness of SSL certificates or deploy proprietary communication protocols. The countermeasures are not specifically designed to protect the location information. They are rather very general mechanisms, which were designed to secure the communication between two parties over an untrusted network. The implications of these countermeasures are only applicable for the proxy interception mechanism. For all other approaches there are no countermeasures from an application point of view.

**Location obfuscation**
The more interesting countermeasures can be found in the second area, which actually target the obfuscation itself. There are two types of attacks, which can be applied against an obfuscation algorithm.

The first type describes a plausibility check, which evaluates whether the generated location information is actually plausible. For static scenarios, where only one location information is transmitted to a service, there are only very limited plausibility checks. The static plausibility analysis can only check, whether the generated location is within a realistic area and not an unrealistic one, like in the middle of an ocean or within a restricted area. For movement scenarios, which consist of a sequence of location information, it is easier to examine the plausibility. The easiest approach here is to submit the reported locations to a routing algorithm and check whether the sequence of location information is feasible with known transportation means. If the distance between two reported locations is too long for a short timespan, it is very likely that the sequence of location information is not genuine.

The second type of countermeasures describes actual attacks against the obfuscation algorithm. Once the attacker figures out, that the sequence of location information is not plausible, he can apply different mechanisms, to derive the actual location information. One approach is to try to derive the actual location information by reducing artificial interference, introduced by the enlarging the center algorithm. Another approach would be a mapping of the track on a map to reverse the effects of the shifting the center algorithm. Currently there are only known countermeasures for spatial obfuscation algorithm, but further research will also reveal weak points for other obfuscation approaches.

| *Category* | *Algorithm* |
|---|---|
| no obfuscation | no obfuscation |
| minor local obfuscation | enlarging the radius |
| fine grained semantic obfuscation | landmark |
| coarse grained obfuscation | discretization to points on a grid |
| behavior maintaining obfuscation | shift the center |
| selective obfuscation | inversion of avoid areas |
| deactivate location information | deactivation |

**Table 8.1:** Recommended location obfuscation algorithm

## 8.3 Algorithm suggestion

One goal of this thesis was to provide a suggestion of appropriate location obfuscation algorithm for different applications (RQ3). To perform this recommendation, the applications were categorized into different use case categories, which require a different level of granularity. Based on these different categories, a mapping between the categories and the appropriate algorithms was set up. This algorithm assignment can be seen in Table 8.1.

The algorithm recommendation was carried out based on a qualitative evaluation of different obfuscation algorithms. The downside of such an exclusive qualitative recommendation is, that these algorithms are only evaluated in theory or within small, and therefore not representative scenarios. Due to this weak basis of decision-making, this recommendation can only be used as a starting point for further recommendations. Any further evaluations should also contain empirical studies, where users use these algorithms on a daily basis to assess their applicability. The practical evaluation of the different algorithms should also incorporate different configurations for these algorithms, like different radii, different grid widths or time constants. Based on the empirical analysis, this recommendation should be revisited and if necessary updated.

## 8.4 Comparison of different frameworks

The following section compares the capabilities of different location privacy improving frameworks as well as the default capabilities of the latest versions of the Android and the iOS operating system (RQ5). While the default implementations do not require any interception mechanism, they are essential for all other obfuscation approaches. Therefore this section also discusses, whether there are any feasible interception mechanism in modern mobile operating systems (RQ4). When one compares the location privacy related capabilities of Android and IOS, one can see that iOS provides more configuration options and privacy preserving capabilities than Android. On Android it is only possible to enable or disable the location provider for all applications, while for iOS it is possible to enable the location provider for each application, which requests location information, individually. Beside the individual configuration possibilities, one can enable or disable the location provider globally for all applications on iOS. The major advantage of these default capabilities is, that the user is not required to install any third party tools or has to configure traffic routing paths, which is represented in the Table 8.2 by the *no system modification* capability.

Beside the two operating systems, the application PDroid was selected as a representative tool for this comparison. PDroid is a typical third party interception mechanism, which allows the user to deactivate the location information for particular applications or to fake the location information for them. Beside PDroid, there are several other applications with similar capabilities for Android and IOS. The major improvement provided by all these third party applications is, that they provide the functionality to spoof the location information and transmit it to the applications. The downside for this application, as for all other applications, which modify the operating system, is, that the functionality is not implemented natively and with every update of the operating system, the modifications to the mobile operating system might be invalidated. Beside the functionality to provide fake location information, these applications do not provide any other capabilities, which solve the problems stated in the problem description section.

The first research project, which introduced the concept of location granularity, was the selective cloaking prototype. This prototype introduced the possibility to assign different location obfuscation algorithms to applications, to cater to the different levels of the minimal required location granularity. By adding the location granularity, the different levels of location accuracy, which are required by the applications to provide the desired service, were respected.

The remaining two prototypes are those, which were implemented for this thesis. The system level interception prototype is an extension to the selective cloaking one. The functionality was extended by adding the possibility to cluster different applications into categories and by implementing a categorization procedure to automatically categorize newly installed applications. The clustering capabilities were introduced, to improve the usability of this prototype, which is required to improve the acceptance by the users. With the clustering capabilities in place, it is very easy to update location privacy configuration for a large set of applications at once, instead of updating them for each application individually one after another. The second addition, the automatic categorization functionality, is also designed to improve the acceptance by the users. This automatic categorization capability categorizes all newly installed applications into predefined categories. For the user there is no need to modify this automatic configuration, unless the automatic assignment was flawed or the user has a special location policy, which differs from the default settings.

The last framework in this comparison is the proxy based interception approach. The proxy based interception approach provides very similar obfuscation capabilities than the system level interception one in terms of categorization and configuration. Due to the complex categorization procedure, which requires the definition of new signatures, this prototype does not provide any automatic category assignment procedures. Beside the lack of the automatic categorization capabilities, the obfuscation capabilities are also not applicable to all applications. Some applications deploy countermeasures against tampering the communication between the application and an external server. The unique selling proposition of this prototype is, that it behaves similar to the default capabilities of the mobile operating systems. This framework does not require any modifications of the mobile operating system and can be deployed to all modern mobile operating systems. Additionally this interception mechanism is operating system agnostic, which means, that it can be used with Android and iOS respectively other mobile operating systems

| | *Proxy based interception* | *System level interception* | *Selective cloaking* | *PDroid* | *IOS 7* | *Android 4.3* |
|---|---|---|---|---|---|---|
| Disable location information provider | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Disable location information for selected applications | ✓ | ✓ | ✓ | ✓ | ✓ | |
| Spoof location information | ✓ | ✓ | ✓ | ✓ | | |
| Granular obfuscation | ✓ | ✓ | ✓ | | | |
| Applicable for all applications | | ✓ | ✓ | ✓ | ✓ | ✓ |
| No system modification | ✓ | | | | ✓ | ✓ |
| Clustering of applications | ✓ | ✓ | | | | |
| Automatic categorization of applications | | ✓ | | | | |

**Table 8.2:** Comparison of different location obfuscation Frameworks

like Blackberry or Symbian.

When summarizing the comparison of the different frameworks, one can recognize, that the system level interception framework provides the largest set of capabilities in terms of location privacy and usability. Therefore this framework is the most promising one, unless the mobile operating system cannot be modified. In this case, the proxy based interception mechanism is the only approach, which can be used to redeem the lack of default functionality provided by the mobile operating systems.

CHAPTER 9

# Conclusion

## 9.1 Conclusion

In the last couple of years, the scientific community has proposed several obfuscation algorithms. The design focus of these algorithms was only put on improving the privacy of the user, but other aspects like maintaing the quality of service for applications or a plausibility evaluation of the algorithms were neglected. While the privacy aspect is crucial for any obfuscation algorithm, the other two aspects should not be neglected in future research projects. Assumed that an algorithm provides a very good level of privacy, but makes the usage of location based application impossible, it cannot be used in a real world scenario. Additionally the researchers, who design new algorithms, should also focus more on the evaluating the algorithm against attacks, which are designed to detect the application or even reverse the usage of the algorithm.

One goal of this thesis was to propose appropriate obfuscation algorithms for different applications. In the literature, only a few classifications of obfuscation algorithms can be found. These classifications evaluate the different algorithms on a very abstract level, like their general area of application. They might be suitable for a coarse partition of algorithms, but they are not suited to be used for a recommendation of algorithms. Therefore this work analyzed different algorithms and tried to extract different capabilities of the algorithms. The extraction of the capabilities is not trivial, because the algorithms are either designed to work with different geographic notations or are designed to improve the privacy for very specific use cases. Thus the classification provided by this work shows, how the algorithms perform for different scenarios. The result of this classification can then be used to propose the desired appropriate mapping between location based application categories and location obfuscation algorithms.

During the evaluation of the location obfuscation frameworks, also two of these algorithm recommendations were evaluated. In two scenarios, the proposed algorithms improved the location privacy for the user without reducing the quality of service provided by the applications. This positive evaluation results are a small indicator, that the assignment of location obfuscation algorithms is also feasible for other applications. To prove the proposed algorithm assignment, this evaluation is way to little and only a large empirical study can verify the algorithm recommendations. Such an empirical analysis would likely highlight shortcomings of the qualitative survey and reveal additional research issues.

Besides the obfuscation algorithms, also several obfuscation frameworks and proposed architectures were evaluated. Most of the currently existing frameworks are either not applicable to real world scenarios or the user interface is to so bad, that hardly anybody can use the solutions. The major part of these frameworks originate from research projects and the only goal of these research projects was to provide a proof of concept, how the location privacy could be improved, without considering any external aspects like application developers or the user of location based applications. In the course of this work, two different prototypes were implemented, with the goal to design privacy preserving solutions, which can be deployed in real world scenarios. The first design approach extends an already existing research prototype with usability features, like clustering capabilities and a simple automatic category assignment algorithm. This prototype can be applied for all applications, which can be installed on mobile devices and can be used as a blueprint by mobile operation system developers to implement privacy preserving capabilities to their systems. The downside of this first prototype is, that it requires modifications to the operating system. These modifications are feasible for research projects, but not for the average user. To solve this problem, a second prototype was designed, which can be added to any modern operating system. This framework intercepts the location information on the way between the application and an external server, to modify the location information. This approach has also several downsides. First, this framework is not applicable to all applications and second it requires dramatically more administrative effort than the first prototype. As a conclusion it can be said, that the first prototype is a feasible blueprint for mobile operation systems and until these capabilities are added natively to them, the second prototype is a feasible workaround for privacy sensitive users.

In the past, a lot of effort has been put in the area of location privacy preserving frameworks on a theoretical level. Currently most theoretical aspects are supposed to be solved and it is time to evaluate these theoretical concepts in real world scenarios. After a positive evaluation, they should be implemented into mobile operation systems to improve the privacy of the users.

## 9.2 Contribution

This thesis provides several contributions. The first contribution is an analysis concerning the plausibility of generated location information. Such a plausibility analysis is lacking for most obfuscation algorithm proposals. Therefore this thesis provides an initial plausibility assessment for the most important obfuscation algorithms. Besides the plausibility analysis, also the privacy improvement capabilities of obfuscation algorithms were evaluated, based on predefined criteria, with the goal to make the algorithms comparable.

The second contribution consists of a new proposed classification scheme for location obfuscation algorithms. This classification scheme is based on the privacy improvement and plausibility capabilities provided by the algorithms for specific scenarios. These specific scenarios represent real world scenarios, which are very common for the usage of location based applications. This classification makes the different algorithms comparable for non domain experts. Former classification schemes only provided a high level classification of algorithms, which are useless for engineers, who want to select an appropriate algorithm. The classification scheme is based on very simple categories and can be used to evaluate also other algorithms, besides the ones evaluated for this work.

The third contribution is an evaluation of existing and proposed location privacy frameworks. The analysis of these frameworks has shown, that each framework has its own disadvantages. Based on this evaluation two approaches were selected and were used to implement more advanced location privacy preserving frameworks as the fourth contribution. The first prototype was built on top of an already existing prototype and it was improved with several usability extensions. The second prototype is based on a theoretical concept, which was implemented from scratch. These two prototypes were evaluated with real world scenarios and the results of this evaluation confirmed that almost all of the existing problems were solved with the implemented prototypes.

## 9.3 Future work

During this work, several pursuing research questions and tasks occurred. The most important future research task is, to perform a large-scale empirical analysis of all currently existing location obfuscation algorithms and the proposed assignment of algorithms to application categories, which is provided by this work. This empirical analysis either confirms the qualitative analysis, which was conducted in the course of the thesis or reveals issues with the assignment respectively the algorithms itself. The revealed issues can then be used to improve existing algorithms, to revise the algorithm proposal and the initially proposed parameters for the algorithms or to develop new obfuscation algorithms.

Currently the proxy based framework is only capable of modifying location information, which are encoded within text messages. Therefore this framework is not capable of obfuscating the location information, which is encoded within the EXIF information of images. This deficiency should be solved by implementing an additional module for the proxy interception framework, which is capable of extracting and modifying EXIF information.

The evaluation of the proxy based implementation has shown, that the location information is often only used for advertisement purposes. Different applications often use the same advertisement providers, with identical message structures. This fact can be used to define signatures for these advertisement services, which can be added as a default set of signatures to the proxy framework.

For most users, the privacy preferences change during the daily routine. They often have different requirements during their working hours, than in their leisure time. The fact is currently neglected by all privacy preserving frameworks. A future improvement for these frameworks should be the inclusion of such context sensitive aspects, like daytimes or locations into their privacy policies.

# List of Acronyms

| | |
|---|---|
| **AES** | Advanced Encryption Standard |
| **API** | Application Programming Interface |
| **BTS** | Base Transceiver Station |
| **CC-BY-SA** | Creative Commons Attribution-ShareAlike 2.0 |
| **EXIF** | Exchangeable Image File Format |
| **GLONASS** | GLObal NAvigation Satellite System |
| **GSM** | Global System for Mobile Communications |
| **GPS** | Global Positioning System |
| **GUI** | Graphical User Interface |
| **HTTP** | HyperText Transfer Protocol |
| **HTTPS** | HyperText Transfer Protocol Secure |
| **ID** | Identifier |
| **IP** | Internet Protocol |
| **JSON** | JavaScript Object Notation |
| **LAN** | Local Area Network |
| **LBS** | Location Based Service |
| **MITM** | Man-In-The-Middle |
| **OGC** | Open Geospatial Consortium |
| **PMP** | Protect My Privacy |
| **POI** | Point Of Interest |

| | |
|---|---|
| **RDF** | Ressource Description Framework |
| **SSL** | Secure Sockets Layer |
| **TLS** | Transport Layer Security |
| **TTP** | Trusted Third Party |
| **UMTS** | Universal Mobile Telecommunications System |
| **URL** | Uniform Ressource Locator |

APPENDIX A

# Evaluation of proxy based interception

The evaluation of the proxy base interception approach is an analysis of the 100 top free applications on the Google Play Store [1]. The list was retrieved on the 19.12.2013 and contains all kinds of application. The table listed below lists for all applications whether they request access to the location provider, whether they actually request location information from the provider and if the proxy interception approach works for this application or not. The checkmarks indicate that the column description applies for the application, which is listed in the row. At the bottom of the table, one can find the sum of all checkmarks of this column.

| Application | Location Data Request | Location Data Usage | Interception works | Interception does not work |
|---|---|---|---|---|
| WhatsApp Messenger | ✓ | | | |
| Facebook Messenger | ✓ | ✓ | ✓ | |
| Facebook | ✓ | ✓ | | ✓ |
| Quizduell | | | | |
| Traffic Racer | | | | |
| Ski Challenge 14 | | | | |
| Skype - free IM & video calls | ✓ | | | |
| Angry Birds Go! | ✓ | | | |
| Farm Heroes Saga | | | | |
| Candy Crush Saga | | | | |
| Shazam | ✓ | ✓ | ✓ | |
| Viber | ✓ | | | |
| Overall $\sum$ | 43 | 31 | 21 | 10 |

[1] `https://play.google.com/store/apps/collection/topselling_free` Last accessed: 19.12.2013

| Application | Location Data Request | Location Data Usage | Interception works | Interception does not work |
|---|---|---|---|---|
| Ball Travel 3D Full Version | | | | |
| Simple mp3 Downloader | | | | |
| willhaben.at | ✓ | ✓ | ✓ | |
| ÖBB Scotty | ✓ | ✓ | ✓ | |
| Instagram | ✓ | ✓ | | (✓) |
| Hay Day | | | | |
| Bitstrips | | | | |
| Tiny Flashlight + LED | | | | |
| Stickman Soccer | | | | |
| Subway Surfers | | | | |
| Pou | | | | |
| My Talking Tom | | | | |
| Google Translate | | | | |
| Egg Baby | | | | |
| Adobe Reader | | | | |
| Hill Climb Racing | | | | |
| Spotify | | | | |
| AnitVirus Security | ✓ | ✓ | ✓ | |
| ORF Ski Alplin Weltcup | | | | |
| Ultimate Puzzle | | | | |
| PicsArt - Photo Studio | ✓ | ✓ | | (✓) |
| Angry Birds | ✓ | | | |
| 4 Bilder 1 Wort | | | | |
| Mein A1 | | | | |
| The Simpsons$^{TM}$: Tapped Out | | | | |
| Snail Bob 2 | ✓ | ✓ | ✓ | |
| GMX Mail | | | | |
| Temple Run 2 | | | | |
| Escape Action | | | | |
| Despicable Me | | | | |
| Google Earth | ✓ | ✓ | ✓ | |
| Clash of Clans | | | | |
| QuickCheck | ✓ | ✓ | | ✓ |
| LOVOO - Live Dating & Friends | ✓ | ✓ | ✓ | |
| Overall $\sum$ | 43 | 31 | 21 | 10 |

| Application | Location Data Request | Location Data Usage | Interception works | Interception does not work |
|---|---|---|---|---|
| Amazon DE | ✓ | | | |
| 3Kundenzone | ✓ | | | |
| eBay | ✓ | ✓ | ✓ | |
| Mountain Climb Race 2 | ✓ | ✓ | ✓ | |
| Tango Messenger | ✓ | ✓ | ✓ | |
| Super-Bright LED Flashlight | ✓ | ✓ | ✓ | |
| Barcode Scanner | | | | |
| Shpock: mobile yard sale | ✓ | ✓ | ✓ | |
| BADLAND | ✓ | | | |
| Elfyourself by Officemax | | | | |
| Night Vision Spy Camera | ✓ | | | |
| qando | ✓ | ✓ | ✓ | |
| Snapchat | | | | |
| Dropbox | | | | |
| Firefox Browser | | | | |
| for Android | ✓ | ✓ | ✓ | |
| ZEDGE$^{TM}$ | | | | |
| Papa Pear Saga | | | | |
| Flashlight HD LED | | | | |
| Block Gun 3D: Ghost Ops | ✓ | | | |
| Wondershare PowerCam | ✓ | ✓ | | (✓) |
| Moy - Virtual Pet Game | | | | |
| Clean Master (Cleaner) - FREE | | | | |
| Geizhals Preisvergleich | ✓ | ✓ | | ✓ |
| ErsteBank/Sparkasse netbanking | ✓ | ✓ | | ✓ |
| Raiffeisen Meine Bank | ✓ | ✓ | ✓ | |
| Castle Clash | | | | |
| Mobile Security & Antivirus | | | | |
| Truck Parking 3D | | | | |
| WhatsApp Wallpaper | | | | |
| Tuneln Radio | ✓ | ✓ | ✓ | |
| LEO dictionary | | | | |
| Stocard - Loyalty Cards | ✓ | ✓ | | ✓ |
| Dumb Ways to Die | | | | |
| Overall $\sum$ | 43 | 31 | 21 | 10 |

| Application | Location Data Request | Location Data Usage | Interception works | Interception does not work |
|---|---|---|---|---|
| Talking Santa | | | | |
| Pet Rescue Saga | | | | |
| Mobogenie Market | | | | |
| Apple Shooter | | | | |
| Fruit Ninja Free | ✓ | ✓ | ✓ | |
| AutoScout24 - - used car finder | ✓ | ✓ | ✓ | |
| Hofer | ✓ | | | |
| ORF TVthek: Video on demand | | | | |
| Doodle Jump | | | | |
| ES File Explorer File Manager | | | | |
| wetter.com | ✓ | ✓ | | ✓ |
| Amazon | ✓ | | | |
| Solitaire | | | | |
| Where is that? | | | | |
| Speedtest.net | ✓ | ✓ | ✓ | |
| MX Player | | | | |
| Battleship 2 | | | | |
| My Baby (Virtual Pet) | | | | |
| Twitter | ✓ | ✓ | | ✓ |
| Empire: Four Kingdoms | | | | |
| Logo Quiz - Guess the logo | ✓ | ✓ | ✓ | |
| Overall $\sum$ | 43 | 31 | 21 | 10 |

**Table A.1:** Evaluated applications for proxy based interception

APPENDIX B

# Evaluation of system level interception

The Table B.1 represents the nearest five ATM machines related to a given location. The first column represent the original location and the the other ten columns represent obfuscated locations. These locations were obfuscated using the enlarging the radius algorithm with the radius increased to 250 meter. The names of the nearest banks, where the ATM are located, were retrieved from the *Raiffeisen Meine Bank* application [1]. The valued stated in the interception of the coordinates and the name of the bank represent the distance between the location and the ATM machine according to the application. The distance values represent the distance in kilometers.

| | 48.19900,16.36994 | 48.19962,16.37084 | 48.19802,16.36967 | 48.1972,16.37084 | 48.19763,16.37182 | 48.20011,16.37256 | 48.19904,16.37042 | 48.19777,16.36829 | 48.19946,16.36847 | 48.19907,16.37004 | 48.19722,16.36965 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Unicredit Bank Austria AG<br>1040 Wien, Wiedner Haupstr. 11 | 0.2 | | 0.2 | 0.3 | 0.3 | | 0.2 | 0.1 | 0.1 | 0.2 | 0.2 |
| RLB NOE-WIEN AG<br>1040 Wien, Gusshausstrasse 21 | 0.2 | 0.3 | 0.2 | 0.1 | 0.0 | | 0.2 | 0.3 | 0.3 | 0.2 | 0.2 |
| HYPO NOE LANDESBANK<br>1040 Wien, Operngasse 21 | 0.3 | | 0.3 | 0.4 | | | | 0.2 | 0.2 | 0.3 | 0.3 |
| RLB NOE-WIEN AG<br>1010 Wien, Kaerntner Ring 2 | 0.3 | 0.3 | | | | | 0.3 | | 0.3 | 0.3 | |
| RaiffeisenLandesbank Noe-Wien<br>1030 Wien, Landstr. Hauptstr. 2 | 0.4 | | | | | | | | 0.3 | | |

[1] `https://play.google.com/store/apps/details?id=com.isis_papyrus.raiffeisen_pay_eyewdg` Last accessed: 30.12.2013

| | 48.19900,16.36994 | 48.19962,16.37084 | 48.19802,16.36967 | 48.1972,16.37084 | 48.19763,16.37182 | 48.20011,16.37256 | 48.19904,16.37042 | 48.19777,16.36829 | 48.19946,16.36847 | 48.19907,16.37004 | 48.19722,16.36965 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| RLB NOE-WIEN AG<br>1010 Wien, Schwarzenbergpl. 16 | | 0.3 | | | 0.4 | 0.1 | 0.3 | | | | |
| RaiffeisenLandesbank Noe-Wien<br>1010 Wien, Schwarzenbergpl. 3 | | 0.3 | | | 0.4 | 0.2 | | | | | |
| Unicredit Bank Austria AG<br>1010 Wien, Kaerntner Ring 1 | | 0.3 | | | | | | | | | |
| Erste Bank<br>1040 Wien, Wiedner Hauptstr. 20 | | | 0.3 | 0.3 | 0.4 | | | 0.2 | | | 0.2 |
| Unicredit Bank Austria AG<br>1040 Wien, Wiedner Hauptstr. 65 | | | 0.4 | 0.4 | | | | | | | 0.4 |
| Vakifbank International (Wien)<br>1010 Wien, Kaerntner Ring 18 | | | | | | 0.2 | | | | | |
| Denizbank AG<br>1010 Wien, Kaerntner Ring 14 | | | | | | 0.2 | 0.3 | | | 0.3 | |
| Unicredit Bank Austria AG<br>1010 Wien, Schubertring 14 | | | | | | 0.2 | | | | | |
| BAWAG P.S.K.<br>1040 Wien, Naschmarkt-MA 59 | | | | | | | | 0.4 | | | |

**Table B.1:** Distance to nearest five ATM machines for a given location

# Bibliography

[1] Agarwal, Yuvraj and Hall, Malcolm. ProtectMyPrivacy: Detecting and Mitigating Privacy Leaks on iOS Devices Using Crowdsourcing. *MobiSys*, 2013.

[2] Andersen, Mads Schaarup and Kjærgaard, Mikkel Baun. Towards a New Classification of Location Privacy Methods in Pervasive Computing. In *Mobile and Ubiquitous Systems: Computing, Networking, and Services*, pages 150–161. Springer, 2012.

[3] Anderson, Ian and Maitland, Julie and Sherwood, Scott and Barkhuus, Louise and Chalmers, Matthew and Hall, Malcolm and Brown, Barry and Muller, Henk. Shakra: tracking and sharing daily activity levels with unaugmented mobile phones. *Mobile Networks and Applications*, 12(2-3):185–199, 2007.

[4] Ardagna, Claudio Agostino and Cremonini, Marco and Damiani, Ernesto and di Vimercati, S De Capitani and Samarati, Pierangela. Location privacy protection through obfuscation-based techniques. In *Data and Applications Security XXI*, pages 47–60. Springer, 2007.

[5] Axel, Küpper. Location-Based Services: Fundamentals and Operation. *John Wiely & Sons*, 2005.

[6] Bahl, Paramvir and Padmanabhan, Venkata N. RADAR: An in-building RF-based user location and tracking system. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 775–784. IEEE, 2000.

[7] Bamba, Bhuvan and Liu, Ling and Pesti, Peter and Wang, Ting. Supporting anonymous location queries in mobile environments with privacygrid. In *Proceedings of the 17th international conference on World Wide Web*, pages 237–246. ACM, 2008.

[8] Barkhuus, Louise and Dey, Anind K. Location-Based Services for Mobile Telephony: a Study of Users' Privacy Concerns. In *INTERACT*, volume 3, pages 702–712. Citeseer, 2003.

[9] Robert Battle and Dave Kolas. GeoSPARQL: Enabling a Geospatial Semantic Web. *Semantic Web Journal*.

[10] Bellavista, Paolo and Corradi, Antonio and Giannelli, Carlo. Efficiently managing location information with privacy requirements in wi-fi networks: a middleware approach. In *2nd International Symposium on Wireless Communication Systems*, pages 91–95. IEEE, 2005.

[11] Benisch, Michael and Kelley, Patrick Gage and Sadeh, Norman and Cranor, Lorrie Faith. Capturing location-privacy preferences: quantifying accuracy and user-burden tradeoffs. *Personal and Ubiquitous Computing*, 15(7):679–694, 2011.

[12] Beresford, Alastair R and Rice, Andrew and Skehin, Nicholas and Sohan, Ripduman. MockDroid: trading privacy for application functionality on smartphones. In *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications*. ACM, 2011.

[13] Beresford, Alastair R and Stajano, Frank. Location privacy in pervasive computing. *Pervasive Computing, IEEE*, 2(1), 2003.

[14] Bittner, Thomas and Smith, Barry. A taxonomy of granular partitions. In *Spatial Information Theory*, pages 28–43. Springer, 2001.

[15] Bittner, Thomas and Smith, Barry. A theory of granular partitions. *Foundations of geographic information science*, pages 117–151, 2003.

[16] Brush, AJ and Krumm, John and 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*, pages 95–104. ACM, 2010.

[17] Callegati, Franco and Cerroni, Walter and Ramilli, Marco. Man-in-the-Middle Attack to the HTTPS Protocol. *Security & Privacy, IEEE*, 7(1):78–81, 2009.

[18] Damiani, Maria L and Bertino, Elisa and Silvestri, Claudio. PROBE: an obfuscation system for the protection of sensitive location information in LBS. *TR2001-145, CERIAS*, 2008.

[19] Damiani, Maria L and Bertino, Elisa and Silvestri, Claudio. Protecting location privacy through semantics-aware obfuscation techniques. In *Trust Management II*. Springer, 2008.

[20] Damiani, Maria Luisa and Bertino, Elisa and Silvestri, Claudio. Protecting location privacy against spatial inferences: the PROBE approach. In *Proceedings of the 2nd SIGSPATIAL ACM GIS 2009 International Workshop on Security and Privacy in GIS and LBS*, pages 32–41. ACM, 2009.

[21] Di Pietro, Roberto and Mandati, Roberto and Verde, Nino Vincenzo. Track me if you can: Transparent obfuscation for Location based Services. In *IEEE 14th International Symposium and Workshops on World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–9. IEEE, 2013.

[22] Djuknic, Goran M and Richton, Robert E. Geolocation and assisted GPS. *Computer*, 34(2):123–125, 2001.

[23] Drane, Christopher and Macnaughtan, Malcolm and Scott, Craig. Positioning GSM telephones. *Communications Magazine, IEEE*, 36(4):46–54, 1998.

[24] Egele, Manuel and Kruegel, Christopher and Kirda, Engin and Vigna, Giovanni. PiOS: Detecting Privacy Leaks in iOS Applications. In *NDSS*, 2011.

[25] Farrahi, Katayoun and Gatica-Perez, Daniel. Daily routine classification from mobile phone data. In *Machine Learning for Multimodal Interaction*, pages 173–184. Springer, 2008.

[26] Fisher, Drew and Dorner, Leah and Wagner, David. Location Privacy: User Behavior in the Field. In *Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices*, pages 51–56. ACM, 2012.

[27] Gartner. Worldwide smartphone sales in November 2013. `http://www.gartner.com/newsroom/id/2623415`, 2013. [Online; accessed 30-November-2013].

[28] Gedik, Bugra and Liu, Ling. Location privacy in mobile systems: A personalized anonymization model. In *Proceedings. 25th IEEE International Conference on Distributed Computing Systems*, pages 620–629. IEEE, 2005.

[29] Google Inc. The Google Geocoding API. `https://developers.google.com/maps/documentation/geocoding/`. [Online; accessed 30-November-2013].

[30] Gröger, Gerhard and Kolbe, Thomas H and Nagel, Claus and Häeferle, Karl-Heinz. OpenGIS city geography markup language (CityGML) encoding standard. *Open Geospatial Consortium Inc. Reference number of this OGC project document: OGC 12-019*, 2012.

[31] Henne, Benjamin. Android Location Privacy - A location obfuscation framework for Android. `http://bhenne.github.io/android-location-privacy/`. [Online; accessed 30-November-2013].

[32] Henne, Benjamin and Kater, Christian and Smith, Matthew and Brenner, Michael. Selective cloaking: Need-to-know for location-based apps. In *Eleventh Annual International Conference on Privacy, Security and Trust (PST)*, pages 19–26. IEEE, 2013.

[33] Hofmann-Wellenhof, Bernhard and Lichtenegger, Herbert and Collins, James. Global Positioning System. Theory and Practice. *Global Positioning System. Theory and practice., by Hofmann-Wellenhof, B.; Lichtenegger, H.; Collins, J.. Springer, Wien (Austria), 1993, 347 p., ISBN 3-211-82477-4, Price DM 79.00. ISBN 0-387-82477-4 (USA).*, 1, 1993.

[34] Hofmann-Wellenhof, Bernhard and Lichtenegger, Herbert and Wasle, Elmar. *Gnss: Global Navigation Satellite Systems: Gps, Glonass, Galileo, and More*. Springer, 2008.

[35] Hong, Jason I and Landay, James A. An architecture for privacy-sensitive ubiquitous computing. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 177–189. ACM, 2004.

[36] Junglas, Iris A and Spitzmuller, Christiane. A research model for studying privacy concerns pertaining to location-based services. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, pages 180b–180b. IEEE, 2005.

[37] Kaasinen, Eija. User needs for location-aware mobile services. *Personal and ubiquitous computing*, 7(1):70–79, 2003.

[38] Kasanen, Eero and Lukka, Kari. The constructive approach in management accounting research. *Journal of management accounting research*, (5):243–264, 1993.

[39] Kessinger, Kristen and Gellman, Marv. Geolocation Use and Concerns Survey. *Tech. rep. ISACA*, 2012.

[40] Krumm, John. Inference attacks on location tracks. In *Pervasive Computing*, pages 127–143. Springer, 2007.

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

[42] Krumm, John. Realistic driving trips for location privacy. In *Pervasive Computing*, pages 25–41. Springer, 2009.

[43] Lakmali, BD Shashika and Dias, Dileeka. Database correlation for GSM location in outdoor & indoor environments. In *Information and Automation for Sustainability, 2008. ICIAFS 2008. 4th International Conference on*, pages 42–47. IEEE, 2008.

[44] Li, Binghao and Salter, James and Dempster, Andrew G and Rizos, Chris. Indoor positioning techniques based on wireless LAN. In *LAN, First IEEE International Conference on Wireless Broadband and Ultra Wideband Communications*. Citeseer, 2006.

[45] Lindqvist, Janne and Cranshaw, Justin and Wiese, Jason and Hong, Jason and Zimmerman, John. I'm the mayor of my house: examining why people use foursquare-a social-driven location sharing application. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2409–2418. ACM, 2011.

[46] Minch, Robert P. Privacy issues in location-aware mobile devices. In *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*. IEEE, 2004.

[47] Mokbel, Mohamed F and Chow, Chi-Yin and 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*, pages 763–774. VLDB Endowment, 2006.

[48] Perry, Matthew and Herring, John. OGC GeoSPARQL - A Geographic Query Language for RDF Data. *OGC Implementation Standard, ref: OGC 11-052r4*, 2012.

[49] Popa, Mirela and Rothkrantz, Leon and Yang, Zhenke and Wiggers, Pascal and Braspenning, Ralph and Shan, Caifeng. Analysis of shopping behavior based on surveillance system. In *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, pages 2512–2519. IEEE, 2010.

[50] Ruiz Vicente, Carmen and Freni, Dario and Bettini, Claudio and Jensen, Christian S. Location-related privacy in geo-social networks. *Internet Computing, IEEE*, 15(3):20–27, 2011.

[51] Schiller, Jochen and Voisard, Agnès. *Location-based services*. Elsevier, 2004.

[52] Shapiro, Marc. Structure and encapsulation in distributed systems: the proxy principle. In *icdcs*, pages 198–204, 1986.

[53] Shokri, Reza and Theodorakopoulos, George and Le Boudec, Jean-Yves and Hubaux, Jean-Pierre. Quantifying location privacy. In *Security and Privacy (SP), 2011 IEEE Symposium on*, pages 247–262. IEEE, 2011.

[54] Shokri, Reza and Troncoso, Carmela and Diaz, Claudia and Freudiger, Julien and Hubaux, Jean-Pierre. Unraveling an old cloak: k-anonymity for location privacy. In *Proceedings of the 9th annual ACM workshop on Privacy in the electronic society*, pages 115–118. ACM, 2010.

[55] Stach, Christoph and Mitschang, Bernhard. Privacy Management for Mobile Platforms–A Review of Concepts and Approaches. In *14th International Conference on Mobile Data Management (MDM)*, volume 1, pages 305–313. IEEE, 2013.

[56] Sun, Guolin and Chen, Jie and Guo, Wei and Liu, KJ Ray. Signal processing techniques in network-aided positioning: a survey of state-of-the-art positioning designs. *Signal Processing Magazine, IEEE*, 22(4):12–23, 2005.

[57] Sweeney, Latanya. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[58] Takeuchi, Hirotaka and Nonaka, Ikujiro. The new new product development game. *Harvard business review*, 64(1):137–146, 1986.

[59] Dennis Upper. The unsuccessful self-treatment of a case of "writer's block" 1. *Journal of applied behavior analysis*, 7(3):497–497, 1974.

[60] Wernke, Marius and Skvortsov, Pavel and Dürr, Frank and Rothermel, Kurt. A classification of location privacy attacks and approaches. *Personal and Ubiquitous Computing*, pages 1–13, 2012.

[61] Wing, Michael G and Eklund, Aaron and Kellogg, Loren D. Consumer-grade global positioning system (GPS) accuracy and reliability. *Journal of Forestry*, 103(4):169–173, 2005.

# List of Figures

2.1 Cell indentification (adapted from [23]) . . . . . . . . . . . . . . . . . . . . . . . . 11
2.2 Angle of arrival (adapted from [23]) . . . . . . . . . . . . . . . . . . . . . . . . . . . 12

3.1 Obfuscation by enlarging the radius (adapted from [4]) . . . . . . . . . . . . . . . . 20
3.2 Obfuscation by shifting the center (adapted from [4]) . . . . . . . . . . . . . . . . . 21
3.3 Combination of enlarging the radius and shifting the center (adapted from [4]) . . . 21
3.4 Discretization of location information to a grid . . . . . . . . . . . . . . . . . . . . 22
3.5 Time based obfuscation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
3.6 Proxy based architecture (adapted from [10]) . . . . . . . . . . . . . . . . . . . . . 29
3.7 Message flow using a Trusted Third Party (adapted from [47]) . . . . . . . . . . . 29

4.1 Aggregated privacy and plausibility of location obfuscation algorithm . . . . . . . 41

6.1 Sequence diagram of an uninfluenced information flow . . . . . . . . . . . . . . . . 48
6.2 Sequence diagram for proxy architecture . . . . . . . . . . . . . . . . . . . . . . . . 50
6.3 Component diagram for the proxy architecture . . . . . . . . . . . . . . . . . . . . . 51
6.4 Certificate usage for a HTTPS interception approach . . . . . . . . . . . . . . . . . 52
6.5 Category assignment with the configuration GUI . . . . . . . . . . . . . . . . . . . 54
6.6 Algorithm adjustment with the configuration GUI . . . . . . . . . . . . . . . . . . . 55
6.7 Sequence diagram for system the interception architecture . . . . . . . . . . . . . 57
6.8 Component diagram for system the interception architecture . . . . . . . . . . . . 59
6.9 Configuration integration . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 62
6.10 Configuration of Categories . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 62
6.11 Individual configuration for the algorithms . . . . . . . . . . . . . . . . . . . . . . . 62

7.1 Setup for evaluation of the proxy based interception . . . . . . . . . . . . . . . . . 64
7.2 Usage of location information and applicability of proxy interception . . . . . . . . 65
7.3 Extending the radius obfuscation applied ten times to one location . . . . . . . . . 68
7.4 Shifting the center obfuscation applied to a sequence of coordinates (original track on the left, obfuscated one on the right) . . . . . . . . . . . . . . . . . . . . . . . . 68

# List of Tables

2.1 Mean deviation of actual geographic location in meter . . . . . . . . . . . . . . . . . 13
2.2 Examples for selected location granularity sizes . . . . . . . . . . . . . . . . . . . . 14

4.1 Classification Table . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 40

8.1 Recommended location obfuscation algorithm . . . . . . . . . . . . . . . . . . . . . 72
8.2 Comparison of different location obfuscation Frameworks . . . . . . . . . . . . . . 74

A.1 Evaluated applications for proxy based interception . . . . . . . . . . . . . . . . . . 84

B.1 Distance to nearest five ATM machines for a given location . . . . . . . . . . . . 86