Preserving Users' Location Privacy in Mobile Platforms

Asma Salim Patel

Supervisors: Dr. Esther Palomar and Prof. Hanifa Shah

School of Computing and Digital Technology

Faculty of Computing, Engineering and The Built Environment

Birmingham City University

This thesis is submitted for the degree of

Doctor of Philosophy

May 2018

Dedicated to my father, mother, sister and brothers for their unconditional love and support.

Declaration

- No material contained in the thesis has been used in any other submission for an academic award.
- All sources used are appropriately acknowledged and that where the words of others are used these are clearly placed in quotation marks.
- I have published material relating to this research previously, and reference is made to any such publications in the thesis.
- This thesis does not exceed the 80000 word length.
- I have specified in acknowledgments all the main sources of help and support.

Signed: ______________________________

Date: ______________________________

Asma Salim Patel

June 2018

Acknowledgements

I am grateful to everyone who has contributed to the completion of my Ph.D. thesis in their own particular way. I want to take this opportunity to express my special thanks and appreciation for all the help I received.

First of all, I would like to thank my Ph.D. supervisor and director of studies, Dr. Esther Palomar, for her insightful suggestions, expert guidance and also personal support. She has always encouraged, helped and assisted me in this research, even while she was on her maternity leave. Ever since I have worked with her, she has always motivated me to bring my best to every task.

Secondly, I would like to express my sincere gratitude to Professor Hanifa Shah for believing in me and always willing to help. I want to thank her for many things, in particular, for creating the research environment within the faculty; providing the funding for my Ph.D. research; being supportive academically and emotionally through the difficult times that helped me stay focused and motivated to finish this research. I am also grateful to her for finding the time to comment and proofread my work at short notice. She has always been an inspiration to me.

A special acknowledgment to my colleague and my very dear friend, Sikander Khan, for being a source of positivity and encouragement during my research and in my life. I want to thank him for his constant motivation and providing a positive environment within the faculty and the University where I can forget my stress, refresh my mind and overcome challenges. His presence never made me miss my family and he always helped me stay cheerful.

In my Ph.D. journey, I was fortunately accompanied by my best friend, Salameh Abu Rmeileh, who is also a fellow PhD student at BCU. I am very grateful to him for being patient, caring and supportive in every aspect. He has been a source of knowledge with his commendable programming skills and computing competency, and I seek his advice on critical decisions.

I would also like to express my gratefulness to all my colleagues and research admin staff from the School of Computing and Digital Technology and the faculty of Computing Engineering and Built Environment at BCU. Due to the research environment created and sustained by Professor Shah, I have got to know many Ph.D. students, postdocs, researchers and professors who have helped me enhance my research. I would like to single out, Dr. Gerald Feldman. Thanks to all of them and my friends within the Faculty, the University and outside.

Since July 2017, I started working in the School of Computing and Digital Technology at Staffordshire University, and I would like to thank the school for their support and help.

And lastly, but most importantly, I express my deep gratitude to my family for their unconditional love and support. I want to thank: my sister, Asiya, for all the love and always staying late with me on phone calls during my difficult days; and both of my brothers, Wasim and Asif, for their love, support and encouragement. I am eternally indebted to my parents for giving me everything I need, and they have never asked for anything in return. Everything I achieved in my life, I owe it to my parents.

Abstract

Mobile and interconnected devices both have witnessed rapid advancements in computing and networking capabilities due to the emergence of *Internet-of-Things*, *Connected Societies*, *Smart Cities* and other similar paradigms. Compared to traditional personal computers, these devices represent moving gateways that offer possibilities to influence new businesses and, at the same time, have the potential to exchange users' sensitive data. As a result, this raises substantial threats to the security and privacy of users that must be considered. With the focus on location data, this thesis proposes an efficient and socially-acceptable solution to preserve users' location privacy, maintaining the quality of service, and respecting the usability by not relying on changes to the mobile app ecosystem.

This thesis first analyses the current mobile app ecosystem as to apply a privacy-by-design approach to location privacy from the data computation to its visualisation. From our analysis, a `3-Layer Classification` model is proposed that depicts the state-of-the-art in three layers providing a new perspective towards privacy-preserving location-based applications. Secondly, we propose a theoretically sound privacy-enhancing model, called `LP-Cache`, that forces the mobile app ecosystem to make location data usage patterns explicit and maintains the balance between location privacy and service utility. `LP-Cache` defines two location privacy preserving algorithms: on-device location calculation and personalised permissions. The former incorporates caching technique to determine the location of client devices by means of wireless access points and achieve data minimisation in the current process. With the later, users can manage each app and private place distinctly to mitigate fundamental location privacy threats, such as tracking, profiling, and identification. Finally, `PL-Protector`, implements `LP-Cache` as a middleware on `Android` platform. We evaluate `PL-Protector` in terms of performance, privacy, and security. Experimental results demonstrate acceptable delay and storage overheads, which are within practical limits. Hence, we claim that our approach is a practical, secure and efficient solution to preserve location privacy in the current mobile app ecosystem.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

APs	Access Points
API	Application Programming Interface
BSSID	Basic Service Set Identifier
EULA	End User License Agreement
GPS	Global Positioning System
IDE	Integrated Development Environment
IoT	Internet of Things
IPS	Inter-Process Communication
IP	Internet Protocol
LBS	Location-based Services
LPPM	Location Privacy-Preserving Mechanism
MAP	Mobile Application Providers
MNO	Mobile Network Operator

MPP	Mobile Platform Providers
OS	Operating Systems
PETs	Privacy Enhancing Techniques
PIR	Private Information Retrieval
PoIs	Point of Interests
QoS	Quality of Service
SSID	Service Set Identifier
TTP	Trusted-Third Party
UI	User Interface
URL	Uniform Resource Locator
WPS	WiFi Positioning Systems
PbD	Privacy-by-Design

Chapter 1

Introduction

1.1 Introduction

It is anticipated that the number of mobile devices can exceed the world's population by 2020 (Ericsson, 2017). The growth of the smartphone market is mainly driven by an increasing range of applications (also known as 'apps'[1]) and business models. The convenience of getting everything in one device and getting to enjoy wireless services on mobile devices[2] everywhere – like in home, hotels, shops, at work and universities – have led smartphone users to mobile phone addiction (Lee et al., 2014; Rozgonjuk and Elhai, 2018). These devices have become necessities in everyone's lives – such as to socialise with friends, to read news, to check weather updates, to find nearby point of interests (PoIs), to shop and for entertainment. In addition to this, governments' recent development strategies such as Digital Single Market/Gigabit Society (EU) (European Commission, 2017) and UK Digital Strategy (Gov. UK, 2017) will continue to raise this mobile trend into new forms of computing paradigm – such as Internet of Things (IoT), Connected Societies, Smart Cities or Smart Homes.

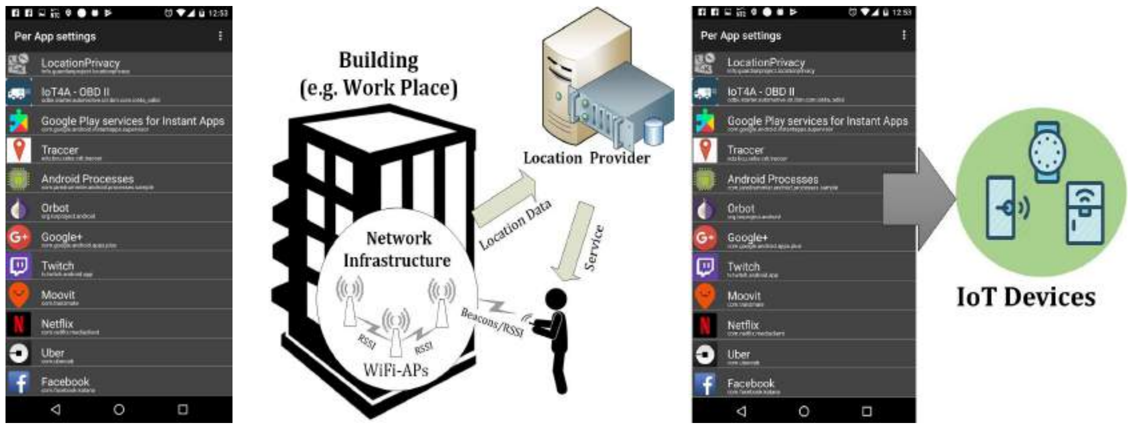
[1]Throughout this thesis, we use the terms LBS 'apps' and 'applications' interchangeably

[2]Throughout this thesis, we use the terms 'smartphones' and 'mobile devices' interchangeably

Recent forecasts on IoT devices suggest that the use of connected “things” (i.e., devices, sensors, and actuators) will rise to over 20-50 billion by 2020 (Ericsson (2017), Gartner (2017a), etc.). This means door locks, health and fitness devices, home appliances and cars are going to be connected to smartphones and each other in the near future. Such explosive growth of "connected things" and "smart devices and appliances" have created a demand for purpose-built mobile application development frameworks, e.g., Samsung Smartthings (2017) and Google Weave (2017)/ Android Things (2017). Moreover, the advancement in wireless and positioning technologies has leveraged context-aware mobile apps bringing tremendous opportunities for a whole new class of Location-based Services (LBS). For instance geo-marketing and geo-social networking, location-based games, and assisted eHealth represent a small subset of these opportunities (Pontes et al., 2012). This has enabled third-party developers to build apps that can constantly compute users’ sensitive data and pose serious threats to users’ privacy (Shklovski et al., 2014; Van Kleek et al., 2017). As a result, along with its obvious benefits, the smartphone has other effects that are not all that eminent and come at a considerable privacy cost. It is apparent that constant disclosure of the user’s exact locations, mobility and behaviour patterns could create severe threats in the long run such as damaging social status, theft and robbery, blackmailing, victims of frauds or physical violence. In this case, location privacy becomes a critical issue; certainly, a natural conflict arises when attempting to protect user privacy while building a system that allows flexible use of location information (Andrienko et al., 2013; Pennekamp et al., 2017).

1.2 Location-based Mobile Environments

In this section, we describe location-based mobile environments that can collect and potentially leak user’s sensitive data within the mobile app ecosystem, including user’s



(a) Mobile apps (b) Indoor localisation and businesses (c) IoT devices connected to mobile apps

Figure 1.1 Location-based mobile environments

mobility patterns, personal habits, behaviour and private locations. The advancements in mobile sensing technology and the use of smartphones can compromise the user's location privacy in three major scenarios: a) mobile apps, b) indoor sensing (i.e., indoor localising and businesses), and c) IoT apps (i.e., apps connected to IoT devices). We have given detailed descriptions of these three scenarios that can compromise the user's location privacy in the below sections.

1.2.1 Mobile Apps

The first scenario consists of a user's location data collected via mobile apps, which can be either system apps or third-party apps, running on the mobile device. Mobile OS (Operating Systems)[3] – such as `iOS` and `Android`– equipped with advanced positioning technology continuously collect users' location information and make that information easily accessible to third-party apps, which provide users with LBS. This attracts attention from adversaries that can range from the typical attacker on the network to app developers, malicious or not, and third-party library developers; hence, smartphone apps have become a serious threat to user's privacy (Shklovski et al., 2014; Van Kleek

[3]Throughout this thesis, we use the terms 'OS' and 'mobile platform' interchangeably

et al., 2017). The traditional approach to guarantee users' privacy has been based on the End User License Agreement (EULA) and other regulations or privacy policies, which do more to protect company's interests than to safeguard users' rights (Ozer et al., 2010). As a result, users aware of the privacy implications are reluctant to use apps and LBSs on their mobile devices (Ketelaar and van Balen, 2018; Muslukhov et al., 2012).

For example consider `Foursquare`, one of the most popular apps, which is completely location-based. To use the service, the user must turn on the location sharing settings on his/her device, and provide personal details for registration. At this stage, the user agrees to the terms and conditions via EULA stating "*By submitting any personal information, I consent to have my personal information transferred to and processed in the US, which I understand may have different data protection rules than my country*". This is also followed by another two EULAs: *(i) giving consent of access to the user's contacts, and (ii) to push notifications.* Some studies have already proven that users' sensitive information collected by `Foursquare`'s third-party location servers is shared with other services causing severe privacy leakage (Pontes et al., 2012). However, benefits of such apps for society are reasonably apparent, especially when they are associated with assistive healthcare systems and emergent IoT wearable technologies. Hence, privacy enforcement on smartphone apps still remains as an open issue (Pennekamp et al., 2017).

1.2.2 Indoor Sensing Environments: Localisation and Businesses

The second scenario that risks location and WiFi data collection via users' mobile devices consists of indoor positioning/localising environments (e.g., shopping malls, retailer shops, etc.). Traditional positioning systems (e.g., the Global Positioning

System *(GPS)* and cell-tower based positioning) have been inefficient when it comes to indoor positioning due to lack of signals and poor coverage. Compared to these, WiFi Positioning Systems (WPS) are nowadays considered as a very accurate method (up to 10m accuracy) for geolocation calculation (Skyhook, 2016). Location providers – e.g., Google Location Service (2016), Skyhook (2016), and Navizon (2016)– use enhanced WPS rather than GPS, primarily due to current smart mobile devices benefit from built-in WiFi clients that perform faster than most expensive traditional positioning technologies (e.g., GPS receivers). Therefore, advanced localising technologies – such as WiFi, Bluetooth, accelerometer or RFID sensors– are utilised to locate users accurately in indoor environments. Businesses based on such indoor sensing environments (e.g., free hotspots) can pose severe privacy threats, such as unauthorised tracking, profiling, and monitoring of users behaviours and movements.

1. *Indoor localising services* - Retailers and other businesses that require indoor localising services in their business operations liaise with the *location provider* to track consumers' movements in the indoor environments (e.g., stores, malls, airports, etc.). Indoor businesses models require dynamic and accurate device positions. Therefore, indoor WPS infrastructure is deployed to estimate users' real-time positions by means of the received signal strength *(RSS)* and known signatures of fixed WiFi Access Points (APs) – via *Triangulation*, *Trilateration*, *Centroid Localization* and *Proximity Estimation* positioning techniques (Kaemarungsi and Krishnamurthy, 2004; Pu et al., 2011). The location calculation is done outside of the user's device with the assistance of the data link layer identifiers[4] that are detected via beacons probes.

2. *Geo-locating services* - For indoor localising services, the fixed WiFi infrastructure may not be geotagged since it relies on received signals and on pre-defined network

[4]e.g., Media Access Control addresses (MAC addresses)

signatures of the indoor environment, i.e., the building's floor plan, levels in a shop or a mall. However, any location-based service request from a mobile app requires the user's geo-coordinates (i.e., latitude and longitude). Geo-coordinates can be generated within the mobile device using GPS receivers; however, *GPS* technology does not work well indoors and it is very slow at starting point calculation. To overcome these disadvantages, *GPS* technology is often combined with mapped WiFi APs and base station data to provide geolocation services (Jan et al., 2000). Therefore, the existing geolocation computation architecture to use location-based apps on smartphones comprises four main entities: smartphones with installed apps, app Provider, network infrastructure, and location Provider. The *location provider* maintains a database of surrounding network infrastructure (of a street, city or town), including WiFi APs, base stations, and IP addresses, which must be mapped to their exact geographical coordinates, also known as geo-coordinates. Every mobile device is deployed with an active probing process and WiFi APs continuously announce their existence in the way of network frames/beacons and transmit their Service Set Identifier (SSID) and Basic Service Set Identifier (BSSID)/MAC addresses. Location providers use these WiFi APs' identifiers to create network signatures and map them with geo-coordinates, also called *geolocation*. These geolocations are then used and shared by the smartphone apps via the standard application programming interface/API (Android, 2016) provided by the underlying mobile platform.

1.2.3 IoT Apps: Smartphone Apps Connected to IoT devices

The third scenario consists of smartphone apps that are connected to IoT devices. IoT frameworks for mobile platforms, e.g., Android Things (2017), leverage third-party developers to build smartphone apps that can easily integrate IoT devices such as body or health sensors, fitness trackers, home appliances, etc. IoT devices are lightweight low power devices; however, when connected to smartphones via specially crafted IoT apps, adversaries can easily get access to user's location data and mobility patterns that can leak users' sensitive information. According to Fernandes et al. (2016b), there are two categories of IoT software/app architecture: (1) Hub, and (2) Cloud. In both architectures, IoT apps running on the user's device are exposed to a slew of sensitive data from sensors and devices connected to the hub, as well as to the other remote cloud-based services. The adversary can easily program such apps and attempt to get access or leak sensitive data; hence, this demands mobile platforms to enforce privacy-preservation mechanism in the design of IoT app development frameworks (Fernandes et al., 2016a, 2017).

1.3 Research Challenges

This section highlights research challenges to preserve location privacy of mobile users.

Location privacy "*Location privacy is the right of individuals to decide how, when, and for which purposes their location information can be released to other parties*" (Duckham and Kulik, 2006). It is a particular type of information privacy that requires prevention of unauthorized parties from learning one's current or past location. Currently, approaches to privacy settings of user location on smartphones are based on a binary process[5]. Users are forced to rely on third-party service providers that in many cases continuously collect, use and share their location data, and in some cases even prompt the user to give away their position on page load (Almuhimedi et al., 2015;

[5]Data protection directives and acts (European Commission, 2016; IETF, 2017) across the globe state that personal data should not be disclosed or shared with third parties without consent from subject(s). Such a consent is typically obtained by mandatory acceptance of the conditions mentioned in the End User License Agreement (EULA), or through opt-out possibilities and other regulations (Michael and Clarke, 2013; Microsoft Phone, 2017a).

Muslukhov et al., 2012; Pennekamp et al., 2017; Shklovski et al., 2014). This attracts attention from adversaries that can range from the typical attacker on the network to app developers, malicious or not, and in-app (advertising) ad library developers. Such adversaries can either unnecessarily collect intrusive user information or allow third-party code of unknown source to execute within the installed app. In the past, several privacy threats have been identified due to smartphones' vulnerabilities, but most of the app developers and widely-deployed ad libraries do not adhere to these threats warnings and develop secure apps (Bettini et al., 2005; Liu et al., 2017). Moreover, it is unknown how the identified vulnerabilities change as apps get updated over a period of time. Since permission-based access controls cannot distinguish between actions performed by an ad library and those performed by its installed app, mobile platforms' security models provide little indication of the existence of privacy threats within any given app. As a result, users aware of the privacy implications are reluctant to use location-based apps or services on their mobile devices and are demanding effective privacy-preserving solutions (Ketelaar and van Balen, 2018; Muslukhov et al., 2012). Thus, both academia and industry agree on the urgent need of adopting a Privacy-by-Design (PbD) approach for the development of user-friendly and socially-accepted solutions to location privacy (ICO, 2018). Moreover, to avoid major privacy implications well-known companies (i.e., `Google` and `Microsoft`) have adopted a PbD approach that considers privacy as an integral part of the design process of their mobile products and services (Cranor and Sadeh, 2013; Gürses and del Alamo, 2016). As a result, PbD approach is applied at every stage of our research. This is to identify whether privacy can be an integral part of the design of mobile platforms, location-based apps or mobile environments that fall under the mobile app ecosystem.

Limitations of current proposals Several proposals have incorporate PETs (Privacy Enhancing Techniques) and other cryptographic schemes in the form of a middleware to address location privacy challenges (Gupta and Rao, 2017; Niu et al., 2015; Patel and Palomar, 2014; Wernke et al., 2014; Zhu et al., 2013). These proposals either claim to preserve location privacy during the LBS query formation or while sharing these queries with the service provider in different architectures and settings. Most of these techniques, however, rely on a series of theoretical assumptions such as the existence of a trusted infrastructure providing privacy protection, a group of similar app users being at the same time and same place, or data collection servers complying with location privacy regulations. These unrealistic assumptions and requirements result in critical limitations when applied to real scenarios. As a result, the lack of usability and deployment feasibility make these location privacy-preserving solutions impractical, hence, considered unsuitable for the existing mobile app ecosystem. An efficient and socially-acceptable solution requires to be practical that maintains the balance between the location privacy and Quality-of-Service (QoS).

Motivation To analyse users' location privacy preferences, we conducted a field study that will be followed by an in-depth secondary research of subsequent and recent literature (Almuhimedi et al., 2015; Liu et al., 2016; Shih et al., 2015; Wijesekera et al., 2017). For this, we designed a survey questionnaire (see Appendix A) and distributed it within the University and on social media platforms. In total, we surveyed 190 smartphone users. The survey result represents, 89.19% of the respondents expressed that they are concerned about their location privacy and care about who has access to their location information. Moreover, 91.89% users think granting permissions to apps on their device to access continuous and precise location can result in the violation of their privacy. Furthermore, 77.03% care about their privacy when using their mobile phones in indoor environments; whereas, 10.81% were unaware of such indoor businesses and sensing practices. 89.10% of the respondents are more concerned about their privacy while sharing their private locations – such as home and work– as compared to public locations. 82.18 % of the respondents rely on location result accuracy and location-based app functionality when they are anywhere outside or unknown places. Further, 78.92% users agreed that there is a need to for better privacy controls on their devices and are willing to install a location privacy enhancing tool on their devices.

1.4 Research Scope

Our research focuses on two major entities: smartphone apps and LBS providers, who are capable of causing location privacy threats. We are concerned that apps deliberately collect users' personal data, including location and other sensitive information as part of their operations. Furthermore, the current direct link of smartphones to the location provider and the continuous flow of LBS queries that include the device's exact geo-coordinates over the network create a serious risk to the protection of users' sensitive information. This is even more challenging, in the presence of a malicious location provider and via advanced network sniffing practices. The collected location information will enable curious and malicious adversaries to pose the following three types of fundamental location privacy threats (Fawaz and Shin, 2014; Wernke et al., 2014):

Tracking Threat can be caused if an adversary receives continuous location updates, which enables location and tracking the user in real-time. An adversary might also be able to track consistent mobility patterns or identify frequently traveled routes that can be used to accurately predict future locations or behaviour of the user.

Identification Threat can be caused even if an adversary sporadically access the user's locations and still be able to identify the user's frequently visited locations, such as home and work. An adversary can use these places as quasi-identities to detect the user's identity from anonymous location traces.

Profiling Threat can be caused even if the mobility traces might not include private places to detect the user's identity, but it can include places that an adversary can use to create the user's profile, for example specific health-care centers, religious places, or communal places and so on.

Hence, this thesis focuses on identifying an effective solution to mitigate these fundamental location privacy threats and improve security in the mobile app ecosystem.

Research Hypothesis

Our research hypothesis that will be tested states *we can bring practical, secure and efficient location privacy preserving solution for mobile users and environments.*

Aim and Objectives

This research aims to develop a location privacy-preserving model for mobile platforms that benefits both end users and service providers.
The objectives of this research are:

O1 To investigate and analyse state-of-the-art communication protocols, mechanisms, methods, and techniques used to provide location privacy in mobile platforms.

O2 To model, design and develop a PbD location privacy-preserving model for mobile platforms.

O3 To evaluate the proposed model to be deployed in the existing mobile app ecosystem.

1.5 Research Design/Methodology

To follow a sequence of activities, scope, objectives, and timescale, our research design consists of three main stages:

Stage 1 was established to accomplish the objective O1. The dominating methodology used in this package to define and monitor our research context is the *literature review*. We also conducted a *field study* using survey method to analyse mobile users' perspectives on the defined research problem. The final outcome of *Stage 1* is used to define our research hypothesis, to establish research challenges, to focus area for investigation, to identify possibilities of improvements and enhancements, and to compare the proposed model with the existing ones. Constant literature review and additional analysis was carried out to define and redefine (if required) the research context and the proposed model .

Stage 2 was determined to accomplish objective **O2**. *Stage 2* is the model development phase, which is divided into three sub-phases: theoretical modeling, design and requirements analysis, and prototype implementation. For modeling, both the theoretical methods and PbD principles were used to understand the complexity of privacy and security challenges into existing systems and the model development. To validate the developed theoretical model, we used rapid prototyping methodology (Jones and Richey, 2000) in the design, development, and implementation of the prototype system, which aims to be a practical proof-of-concept of the theoretical model. The exhaustive modeling and design phases within the development process helped us achieve a high-quality system at a relatively low cost of resources. Based on our study (in *Stage 1* and *Stage 2*), we also defined a privacy metric that will be used as the prototype evaluation criteria.

Table 1.1 Contributions of the thesis

Research Outcome	**Target**	**Type**	**Privacy Protection**	**Data Minimisation**
3-Layer Classification	LBSs in Mobile Environments (Smartphone Apps, Indoor Localisation and IoT apps)	State-of-the-Art Classification Model	Privacy Properties Metric	-
LP-Cache	Smartphone Apps, Indoor Environments and IoT Devices	Theoretical Model	Threat Mitigation-Tracking, Profiling and Identification	✓
PL-Protector	Smartphone Apps, Indoor Environments and IoT Devices	Middleware in a System	Threat Mitigation-Tracking, Profiling and Identification	✓

Stage 3 is assigned to accomplishing objective **O3**. The evaluation phase largely used the experimental method to identify the concepts that will facilitate in testing our research hypothesis. The empirical validation of the developed prototype was essential for testing whether the proposed model is feasible and can be deployed in the existing mobile app ecosystem. For data collection and analysis, we ran a series of tests using the prototype system to evaluate its functionality, performance, efficiency. We followed the privacy metrics (pre-defined in *Stage 2*) to assess the model's security and privacy features.

1.6 Contributions of the Thesis

A number of location privacy-preserving mechanisms (LPPMs) (Khoshgozaran et al., 2011; Patel and Palomar, 2014; Wernke et al., 2014) have been proposed to minimise the risk of major location privacy threats; however, they either lack deployment feasibility, usability, or require numerous changes to the existing mobile app ecosystem. This indicates that the state-of-the-art on LPPMs fails to bring a practical, secure and efficient solution to location privacy in the mobile app ecosystem. In order to effectively mitigate the location privacy threats, we need LPPMs, which provide theoretically and practically sound privacy enhancing systems and tools that are acceptable to end users and service providers. This thesis presents `3-Layer Classification`, `LP-Cache` and `PL-Protector` that contribute to a practical, theoretically sound, and usable location privacy solution. The main contributions of this thesis are summarised in Table 1.1.

1.6.1 3-Layer Classification

To guide researchers working towards location privacy, we present a new perspective of the state-of-the-art on LPPMs and literature findings in the form of the `3-Layer Classification`. This classification provides a brief description of all the protocols, mechanisms and interfaces covering from the application layer to the network layer. Also, we provide a comprehensive privacy analysis of all the classified LPPMs with respect to four privacy properties(Pfitzmann and Hansen, 2008): 1. Unlinkability, 2. Unobservability, 3. Anonymity and 4. Controlled information disclosure.

1. *Unlinkability* is defined as unlinkability of an PoI and the user.
2. *Unobservability* is defined as the state that whether specific PoI exist or not is indistinguishable.
3. *Anonymity* is defined as the state of being anonymous within a set of subjects.
4. *Controlled information disclosure* is providing users with controls to determine for themselves when, how, and to what extent information about them is communicated to others.

Along with this, our classification model embraces a holistic picture of research gaps, methods and implications to satisfy privacy properties and achieve privacy-preserving mobile LBSs.

1.6.2 LP-Cache

We provide detailed analysis of the current location computation process deployed in smartphones for running location-based apps, which can either be system apps or third-party apps. For this, we conduct a study of three major mobile platforms that span the domains of smartphones in order to analyse location privacy threats and security design issues, and to inform system's functionality goals to protect users from location-demanding apps. However, the outcome of this study indicates that the security models and the traditional approach to guarantee users' privacy on smartphones have already proved to be inadequate. Hence, in Chapter 3, we introduce a novel privacy-preserving model called *Location Privacy Cache* (`LP-Cache`) for mobile apps to overcome the shortcomings related to users' privacy during the location calculation process in mobile platforms. By making the user's device play a bigger role in the process, `LP-Cache` prevents users from relying on service providers' trustworthiness. The model applies a cache-based technique to determine the position of client devices by means of wireless APs and achieve data minimisation in the current ecosystem. The model also establishes enhanced location permission settings for the users while sharing their location information. Since `LP-Cache` is a theoretical model, we outline its possible implementation on different mobile platforms (e.g., `Android` and `iOS`), analyse the wireless data feasibility and usability and estimate cache storage requirements. The overall results demonstrate `LP-Cache`'s deployment viability in the mobile app ecosystem. To sum up, the main contributions of `LP-Cache` are as follows:

- A detailed evaluation of the current location computation process deployed in smartphones.
- A comprehensive definition of our model and its main components implementation requirements that are platform independent.

- The main benefits of `LP-Cache` are twofold:

  1. Provides personalised location privacy settings that control every installed app and private place distinctly and protects sharing of user's private location with third-party app providers.
  2. Minimises the amount of wireless access point data being shared within the current architecture for computing device's location by means of the minimum on-device caching mechanism.

- The feasibility and usability analysis demonstrate `LP-Cache`'s deployment viability in the mobile app ecosystem.

1.6.3 PL-Protector

Common approaches to privacy of user location on smartphones are based on two methods namely i) permission controls as a binary process[6] and ii) privacy policies[7]. In the former method, mobile operating systems implement permission-based access control for data sources (OS) and sinks (third-party apps and libraries); however, they do not control *flows* between the authorised sources and sinks. In the latter method, privacy policies are encoded in natural language and are directly enforced on users. Hence, users are forced to rely on third-party service providers that in many cases continuously collect, use and share their location data and, in some other cases, prompt the user to give away geolocation upon page loading (Almuhimedi et al., 2015; Felt et al., 2012; Muslukhov et al., 2012; Shklovski et al., 2014).

[6]Data protection directives and acts European Commission (2016); IETF (2017) across the globe state that personal data should not be disclosed or shared with third parties without consent from the subject(s). Such a consent is typically obtained by mandatory acceptance of the conditions mentioned in the End User License Agreement (EULA), or through opt-out possibilities and other regulationsMichael and Clarke (2013).

[7]A privacy policy specifies the privacy practices of an organisation, basically what kind of personal information is collected, the purpose and how the information will be used/shared.

To address this challenge, our PbD approach incorporates a new design principle and privacy policy recommendation that forces the mobile app ecosystem to make location data use patterns explicit while preventing all other sensitive data flows. We present the design and deployment of a middleware called *Private Location Protector* (`PL-Protector`), which implements our theoretical model (`LP-Cache`). `PL-Protector` implements `LP-Cache`'s two algorithms as LPPMs: LPPM-1) On-device Location Computation Mechanism, and LPPM-2) Personalised Location Permissions Mechanism on `Android` platform. Thus, `PL-Protector`'s LLPM-1 incorporates caching technique to determine users' geographical location in a privacy-preserving manner by means of wireless access points, and with minimum cache storage requirements. And LPPM-2 envisions beyond the simple grant/deny access method and provides the user with advanced mechanisms to decide the extent of disclosing location data with service providers. We implement our middleware on `Android 6`[8], and present its comprehensive evaluation in terms of performance, security and privacy analysis. Using `PL-Protector`, we conducted series of experiments with real apps from five popular and widely used categories of location-based services, e.g., instant messaging and navigation. Experiments demonstrate acceptable computational and development effort, memory consumption (up to 21.3 MB of RAM usage), delay overheads within practical limits (below 22 milliseconds), limited storage overhead (136 KB on-device storage for a period of 3 months). To assess privacy/data leakage within the collected datasets, we conduct security and privacy analysis. The test result confirms that `PL-Protector`'s LPPMs (1 & 2) significantly mitigated potential threats to the user's location data in the data link layer and the OS's middleware layer. The main contributions of `PL-Protector` are as follows:

[8]Please note: `Android 6` (or API level 23) was the latest version of the OS during the implementation phase of this research; however, to the extent of our knowledge, our middleware can similarly be implemented on later OS versions (i.e., API level > 23).

- Based on the analysis of existing mobile platforms and users' perspectives, we present enhanced design and implementation of our location privacy-enhancing middleware, which is a prototype developed to validate the theoretical model, `LP-Cache`. We have successfully implemented our middleware on `Android` platform to enforce the privacy rules over both the information and control flow occurring between the source (OS) and sink (apps).

- Through the implementation of our middleware, we demonstrate the deployment feasibility of a new series of privacy controls on a mobile platform to prevent private location disclosure during the formation of LBS queries. It only requires process isolation and IPC services from the underlying OS; thus, minimizing the requirements placed on the hardware/OS.

- The performance, security and privacy analysis evidences that our middleware mitigates critical location privacy threats at a tolerable loss of QoS and location-based apps functionality and at acceptable overheads on the underlying OS.

Hence, we claim that our proposal is a practical, privacy-enhanced, secure, and privacy efficient solution to location privacy in the mobile app ecosystem.

1.7 Organisation of the Thesis

The rest of the thesis is organized as follows:

Chapter 2 outlines the `3-Layer Classification` for privacy preservation in mobile LBS applications and reviews the state-of-the-art. It fully elaborates on each of the 3 layers: mobile platform (OS and Apps), query formation (LBS queries), and network communication (Privacy-preserving communication over the *Internet*). It critically reviews techniques, protocols, and mechanisms, and it then discusses identified research gaps in the field. Later, it provides privacy analysis of LPPMs in terms of satisfaction of the privacy properties. Followed by a review of related work on the proposed solution.

Chapter 3 describes and evaluates the current location computation process deployed in mobile app ecosystem. It presents the design and architecture of `LP-Cache` and fully elaborates on design decisions and attainable implementation. At the end, it presents the feasibility and usability analysis and deployment requirements.

Chapter 4 presents and describes a middleware, `PL-Protector`, which is an implementation of the proposed theoretical model. It describes the system model that includes `PL-Protector`, and it fully elaborates on its the design decisions, architecture, and implementation. Later, it overviews the system model such as system roles, threat, mobility, app usage and privacy model.

Chapter 5 describes the evaluation methods and goals. It presents validation and performance analysis of `PL-Protector`'s LPPMs in terms of cache storage estimation, communication, and computation overheads. It elaborates on the security and privacy analysis and that is followed by a comparative evaluation of our proposal with other related work. It concludes by highlighting users' perspective via the field study results.

Chapter 6 concludes the thesis, summarises the research contributions and suggests future work and research plans.

Chapter 2

State-of-the-Art and Related Work

Research findings from this chapter have been published in a conference paper [a]

[a]Patel, A., & Palomar, E. "Privacy Preservation in Location-Based Mobile Applications: Research Directions". *In: 9th International Conference on Availability, Reliability, and Security (ARES)*, pp. *227-233*, IEEE 2014.

2.1 Introduction

Mobile devices equipped with improved positioning technology (e.g., GPS, WiFi triangulation, IP address approximation, base station identification, other user provided geotagged information) have drastically increased the use of LBS apps. In addition, cheap data storage allows information collectors to constantly record daily activities of mobile device users (Roman et al., 2013). Nowadays, LBS applications are being leveraged by many companies because of the business growing around them, e.g. location-based games, geo-marketing, and social networking, to name a few. However, such applications can also be a serious threat to users' privacy (Liu et al., 2017; Shklovski et al., 2014). Certainly, a natural conflict arises when attempting to protect user privacy while building a system that allows for flexible use of location information (Andrienko et al., 2013; Wijesekera et al., 2017). Approaches to this challenge have applied cryptography and PETs (Gupta and Rao, 2017; Khoshgozaran et al., 2011;

Niu et al., 2015) as LPPMs to be part of infrastructures, systems or tools; however, they are still far from being deployable and socially-accepted.

Thus, we present a brief description of all the protocols, mechanisms and interfaces ranging from the application layer to the network layer in this chapter. To guide future research, a new perspective of the literature findings is proposed that highlights research gaps, methods, implications and satisfaction of the privacy properties. Our `3-Layer Classification` embraces a holistic approach towards privacy-preserving mobile LBS apps. This new perspective allows us to characterise and classify the protocols, to analyse the tradeoffs produced by different design decisions for location-based apps, and to link the various aspects and data flows at every stage of the service ranging from the users using the mobile device to the transmission of the user sensitive information within the query over the network.

The rest of the chapter is organised as follows. Section 2.2 outlines our classification model for privacy preservation of mobile LBS applications that consists of three layers. We fully elaborate on each layer: mobile platform, query Privacy and network communication in Sections 2.3, 2.4, and 2.5, respectively. In these sections, we highlight techniques, protocols, mechanisms, related challenges and provide privacy analysis of LPPMs in terms of satisfaction of the privacy properties. In Section 2.6, we present a summary of research gaps identified at each Layer. We then review related work for the proposed model in Section 2.7. Finally, Section 2.8 summaries the entire chapter.

2.2 Overview of 3-Layer Classification

According to Suikkola (2010), the success of any LPPM for LBS applications depends on critical factors such as the need for standardised interfaces, society-acceptance, to be practical, secure, and efficient. Therefore, to identify an effective solution, we must know who can officially collect such information? and how this information can violate users privacy? The architecture for using LBS on any mobile platform consists of four main entities: mobile device, app provider, 3. network infrastructure, and location provider. Based on this architecture, there are three main types of service providers: 1.) Mobile Network Operator (MNO), 2.) Mobile Platform Providers (MPP), and 3.) Mobile Application Providers (MAP).

For example, consider a mobile device with LBS app installed, the MNO collects the location information for providing signals; the MPP provides services to develop mobile apps and distribute on their official repositories; the MAP has to register with the MPP (e.g., `Android` or `iOS`) in order to publish their app on the official repositories. Alternatively, mobile apps can also be distributed via third-party app repositories (e.g. `Cydia` and `Amazon`).

Therefore, MNO/MPP/MAP are the main service providers, who can collect users location data and, in some cases, share for third-party use. But, studies presented so far on privacy preservation in mobile LBS (Mokbel, 2007; Shin et al., 2012; Wernke et al., 2014) have predominantly considered application-specific solutions and challenges – e.g., applying PETs or other cryptographic primitives to the LBS query formation or the LBS user identity – and they have not considered communication/network layer solutions. Network layer solutions depend on anonymous network communication techniques for location privacy; however, even if the data packet is encrypted, both the source and destination ip addresses located in the packet's IP header are still visible to an eavesdropper. Along with network layer privacy, the data link layer identifiers (e.g., MAC addresses found in beacons) can also affect application-specific location privacy solutions. In our classification, we examine the roles of these service providers based on the LBS application architecture to identify the most potential and promising directions to achieve location privacy for mobile users. Figure 2.1 depicts our classification model

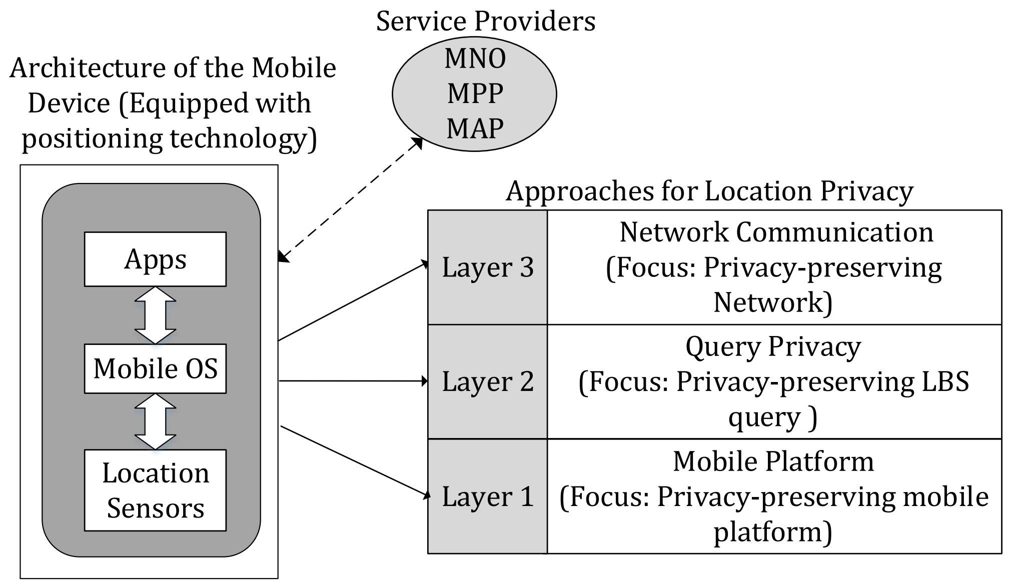

Service Providers
MNO
MPP
MAP
Architecture of the Mobile Device (Equipped with positioning technology)
Apps
Mobile OS
Location Sensors
Approaches for Location Privacy
Layer 3
Network Communication (Focus: Privacy-preserving Network)
Layer 2
Query Privacy (Focus: Privacy-preserving LBS query)
Layer 1
Mobile Platform (Focus: Privacy-preserving mobile platform)

Figure 2.1 Overview of the `3 Layer Classification`

consisting of three layers[1]: Layer 1. Mobile Platform- to provide privacy-preserving OS and apps, Layer 2. Query Formation - to provide privacy-preserving LBS query formation, and Layer 3.) Network Communication - to provide privacy-preserving network communication. To quantify the effectiveness of all classified LPPMs, we assess which and up to what extent they satisfy the four privacy properties (Pfitzmann and Hansen, 2008): unlinkability, unobservability, anonymity, and controlled information disclosure.

2.3 Layer 1: Mobile Platform

In Layer 1, we analyse privacy and security mechanisms integrated as LPPMs in mobile platforms by different MPPs. We highlight their location privacy risks, threats and then identify the research gaps. The two indicators to analyse location privacy risks in

[1]Please note: 3 Layers in our classification model are different from the standard OSI (Open Systems Interconnection) layers, which are part of the OSI reference model.

a mobile OS protection mechanism are: i) the number of identified vulnerabilities, and ii) the number of reported malware. The former indicates the number of weaknesses discovered in a platform that could potentially be used to compromise users' privacy. This case needs preventive systems to be implemented. The latter indicates the number of actual threats discovered using different types of detection systems and monitoring measures. Thus, LPPMs at this level are divided into two sub-domains: a) Mobile OS, and b) Mobile Apps and Repositories. MPPs use preventive mechanisms such as code-signing, encryption, and sandboxing to secure the platform; whereas, detection/monitoring mechanisms for securing regulating apps and repositories. We have given detailed descriptions of these security mechanisms in the sections below.

2.3.1 Mobile OS

According to Gartner (2017b), the worldwide sales of `Android`, `iOS` and other `OSs` hold 84.1%, 14.8 % and 1.1 % of smartphone OS market share, respectively. Existing literature confirms that attackers target users of popular MPPs to get the maximum amount of sensitive information. Clearly, the popularity battle is now between two MPPs: `Android` and `iOS`; however, we have given detailed descriptions of these two MPPs and `Windows Phone` below.

`Android` mobile OS is a modified version of the Linux kernel. Since it has an open source architecture, developers can use self-created certificates to self-sign their app code and publish their apps to the `Android`'s repository (i.e., it does not require any trusted certifying authority). Once the app is self-signed by the developer, `Android` security model then maps the signature of the developer with a unique ID of the app package and enforces signature level permission authorisation (Android, 2017). This openness of `Android` has attracted a large community of developers to work on its security challenges and extend the functionality of the OS. Therefore, `Android`'s dominance has continued to grow and it has the biggest user base. However, since there is no support for root certification authorities in `Android`, it preserves the anonymity of a potential attacker and makes it difficult to scrutinise/block apps with poor source origin, malicious code and code integrity protection. Thus, there is a high possibility that `Android` device users will easily be tricked by unsafe apps, or they will avoid the warning messages during installation or at runtime.

`iOS` mobile OS was developed based on OS X/Unix to support `Apple`'s mobile devices, such as `iPhone` and `iPad`. `Apple iOS` uses app-code singing, hashing, encryption and app-execution sandboxing to secure the platform. App-code signing process ensures execution of only allowed apps that are reviewed and distributed by `Apple` on users' devices. Encryption and hashing ensure the app code cannot be reverse engineered, and that only the authorised user can run the app in order to protect the financial gains and profits of app developers. Sandboxing ensures that an app does have unauthorised access other apps' data or stored file system. To prevent or detect malicious apps, `Apple` follows a vetting process that ensures all apps that are submitted for publication / distribution on their official repositories conform to `Apple`'s regulatory rules. However, `Apple` has kept this vetting/reviewing process and applied criteria unknown (Apple, 2016; Egele et al., 2011). The closed and proprietary nature of `iOS` do not take responsibility of any private data leakage, if the user breaks the security model (i.e., jailbreak) and installs the app from a third-party repository (e.g., Cydia). In spite of this, there have been cases where malicious apps have to be removed from the `App Store` after users' complaints. Since this vetting process is not well documented, it prevents the developers from understanding the flaws in the iOS security model (Wang et al., 2013). Whereas, the growing acceptance of open source platforms makes it apparent that smartphone users no longer assume the restricted and pre-defined terms and conditions are built to safeguard their privacy and security.

`Windows Phone` mobile OS is developed by Microsoft for smartphones, previously known as `Windows Mobile`. Similar to desktop versions of `Windows`, `Windows Mobile` was designed based upon the Windows CE kernel and is now superseded by `Windows Phone`. Third-party software development uses the Microsoft .NET Framework (e.g., C#) and developed Apps are packaged into XAP files, which is the Silverlight application package and users can purchase apps via the Windows Marketplace. Beginning 2015, `Windows Mobile/Phone` was in top 3 mobile OSs Gartner (2015); however, after 2015 it has been removed from the worldwide smartphone OS sales list due to the drastic decrease in its sales and end users. Therefore, to cope up with the competitive mobile OS market, `Microsoft` is constantly experimenting/changing the `Windows Mobile/Phone`'s security and business model. Starting Windows 10, `Microsoft` introduced the Universal Windows Platform (UWP, 2017), which aims to allow developers to create a single app package that can be installed on a wide range of devices. This opens new privacy and security challenge due to interoperability requirement.

Preventive mechanisms These mechanisms depend on cryptographic algorithms, digital signatures, and hash functions to assure fundamental security properties, i.e., confidentiality, authenticity or integrity. Both the closed and open source mobile platforms –i.e., `Android`, `iOS`, and `Windows Phone`– have similar security practice for approving any third-party apps to be distributed over their official repositories, however, the difference remains in their platform security policies. The common OS security model consists of three main techniques:

- *Sandboxes*: Sandbox approach (i.e. creating an environment in which the actions of a process are restricted according to a security policy) isolates one app from another for dynamic monitoring of third-party apps, so they are restricted from accessing files stored by other apps, cannot make changes to the device and have limited access to the OS (Android, 2017; Microsoft Phone, 2017b; Symantec, 2017). This is also a standard practice to prevent a malicious app from affecting another app running on the device.

- *Digital Signature or Code Signing*: Developer must digitally sign the app's installation package with a digital certificate issued by the MPP, and the identity of the developer is embedded into the app. The security model then maps the signature of the developer with a unique ID of the application package and enforces signature level permission authorization (Jain and Shanbhag, 2012). The digital signature process aims to guarantee that both the identity of the app and its developer are not modified or tampered with.

- *Permission-based access control*: In any permission-based mobile OS, apps can only access sensitive resources through the official APIs once the corresponding permissions declared in the system level configuration files (e.g. Android's manifest file and `Windows Phone`'s manifest editor) are granted and authorised by the user (Mylonas et al., 2011; Pennekamp et al., 2017). Mobile users can have two ways to grant permissions to apps: run-time (i.e., while the app is running), or one-time (i.e., once when they install/update the app). Once permissions are granted during installation, the one-time permission model does not allow the user to revoke/change resource access granted to the app at any point in time. Therefore, all the three popular mobile platforms have recently started adapting run-time permissions in their security models since it is more flexible (Android, 2017; Microsoft Phone, 2017b).

However, both the open and closed security models for mobile OSs are not effective in protecting the user's data privacy as well as security from installed apps. For instance, hackers or adversaries can use stolen identities to register with `Apple` (or `Android`) to post harmful apps (Symantec, 2017; Wang et al., 2013). Once the app is installed on the device, the user neither controls nor is prompted about any sporadic or continuous access to the OS resources or sensitive data leakage. This can result in serious location privacy threats that apps pose to mobile users. Mylonas et al. (2013) have tested location privacy attacks on mobile platforms – including `Android`, `BlackBerry`, `iOS`, `Windows Mobile`, and `Symbian`– and results demonstrate that they are vulnerable and require major fixes to mitigate location privacy threats. Similarly, `PiOS` (Egele et al., 2011) demonstrated several privacy threats within the `iOS` platform. In essence, even with the aforementioned security suite for mobile platforms, once the app is installed and user grants access to their location data, both open and closed mobile OS security models are vulnerable to privacy attacks. This is because OS security models focus mainly on companies' (or MAP) policies rather than securing their users' privacy rights. Consequently, malicious data sharing and location privacy attacks via apps are becoming very common no matter whether such apps are registered with official app repositories (e.g. `Apple store` or `Google market`) or third-party app repositories (e.g. `Cydia`, `Amazon AppStore`) (Liu et al., 2017; Zhou et al., 2012a). Thus, smartphone users and MPP are still vulnerable to such unauthorised data gathering and sharing, and code manipulation to get access to sensitive OS resources and information by malicious MAP.

2.3.2 Securing Mobile Apps and Repositories

With regards to location privacy of mobile users, mobile platforms are still vulnerable to location tracking attack and malicious code attack. We now analyse the incorporation of security and LPPMs into LBS apps running on users mobile devices and app repositories. To secure mobile apps and repositories, there are two main approaches: policies and regulations; detection/monitoring mechanisms.

Policies and regulations Privacy policy is a preventive mechanism established by regulatory bodies to monitor how an organisation handles any user information gathered in its business operations. It is a legal requirement that service providers' privacy policies must be defined in compliance with the latest privacy regulations. In addition, standards for location-based products and services exist in both the 3GPP (3GPP, 2017) and IETF (IETF, 2017) arenas. Although such privacy policies aim to be flexible and support the personalisation of privacy preferences, they only provide a deterrent against privacy violations. Therefore, if a third party decides to violate these norms, in spite of the risk of penalties, the user's privacy cannot be protected. Studies (Bettini and Riboni, 2015; Mokbel, 2007) on LBS commercial products and services state they do not comply with users' privacy laws and could be major privacy threats to their customers. Moreover, manual policy enforcement is expensive and thus is often ignored by service providers, particularly when users are unaware of their privacy rights and regulations. Hence, automating privacy policy enforcement has been the main objective of researchers working on privacy protection. We describe different LPPMs based on PETs and cryptography applied to automate privacy policies in Layer 2 and 3.

Detection/monitoring mechanisms If a malicious app developer evades preventive mechanisms (mentioned above), then detection/monitoring mechanisms are incorporated as additional security and privacy measure (Enck, 2011). Detection systems, e.g., malware detection and intrusion detection systems, can be deployed in two ways as host-based or cloud-based. The former is implemented on the device to identify malicious activities by installed apps and alert users or the MPP via the OS controls in real-time. The latter is used on remote apps distribution servers by MPP to monitor their official repositories regularly, to detect or remove any malicious app that is already published.

However, it has been repeatedly reported that many apps even on official repositories prompt for location settings on the device to be switched 'ON', although it is not needed at all in their operations. As a result, to improve mobile OS security models, many researchers (Burguera et al., 2011; Shabtai et al., 2012) are aiming to incorporate a collection of additional security tools that includes signature-based anti-virus, firewalling capabilities, harden access-control and sandboxing, and malware/intrusion detection systems.

Given its open source architecture and user base, comparatively, `Android` encountered a higher number of security and privacy challenges, such as one-time permission issue, easy user tracking, ease to reverse engineer and repackage apps. Hence, several platform-level extensions have been proposed for `Android` – such as `MockDroid` (Beresford et al., 2011), `TISSA` (Zhou et al., 2011), `AppFence` (Hornyack et al., 2011) and so on – that helped it to improve with the release of every OS version. Few prominent proposed app-specific and repository monitoring solutions for `Android` devices are as follows:

Host-based Host-based detection, as well as prevention mechanisms, are mainly based on `Android`'s permissions for addressing two main goals. The first goal focuses on detecting and alerting when a leakage is recognised by applying a re-delegation of permission mechanism (e.g., `AndroidLeaks` (Gibler et al., 2012), `TaintDroid` (Enck et al., 2014), and `Kirin` (Enck et al., 2009). The second goal empowers users with privacy controls that aim at enhancing permission settings of the OS

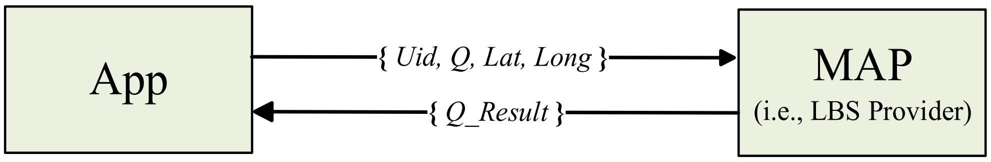


Figure 2.2 High-level communication schematic between the app (on user's device) and MAP (LBS providers), where *Uid* is the app user's identifier, *Q* is the LBS query sent to LBS provider, *Lat* and *Long* are the geo-coordinates of the mobile device.

at run-time (e.g. `TISSA` Zhou et al. (2011), `Locaccino` (Toch et al., 2010) and LP-Guardian (Fawaz and Shin, 2014)).

Cloud-based detection and prevention mechanisms that aim to improve app repository monitoring process have been proposed by researchers. For example, `DroidMOSS` (Zhou et al., 2012a) uses fuzzy hashing technique to match digital signatures and to detect repackaged Apps in third-party `Android` repositories. Whereas, `DroidRanger` (Zhou et al., 2012b) uses permission-based behavioral footprinting and heuristics-based filtering to detect repackaged apps in both, official and third-party, `Android` repositories to identify users' information leakage.

2.4 Layer 2: Query Privacy

Mobile apps share location information with MAPs in the form of LBS queries (see Figure 2.2). The continuous transmission of such queries to remote servers can allow attackers to gain access to the user's sensitive locations and mobility traces. Hence, in this layer, we analyse cryptographic and non-cryptographic techniques (e.g., PETs) applied as LPPMs into location-based system infrastructures, architectures, and applications for privacy-preserving query formation and processing. Existing surveys and taxonomies (Gupta and Rao, 2017; Khoshgozaran and Shahabi, 2010; Krumm, 2009; Wernke et al., 2014) on privacy preservation in LBS have only considered LPPMs and challenges at this Layer (i.e., query formation and processing). Unlike `3-Layer`

`Classification`, these surveys and taxonomies fail to characterise and classify the tradeoffs of LPPMs. Such tradeoffs are produced by different design decisions for location-based systems and link the control aspects and data flow at every stage of the service.

2.4.1 Non-cryptographic Techniques

For LBS query privacy, PETs that are non-cryptographic – such as k-anonymity, dummy locations, region cloaking, location perturbation and obfuscation, and mix-zone pseudonyms – have been established to be implemented in different location-based systems as LPPMs (Gupta and Rao, 2017; Khoshgozaran et al., 2011; Patel and Palomar, 2014; Wernke et al., 2014). These most commonly used non-cryptographic privacy techniques for LBS query formation and processing are briefly described as follows:

Dummy locations are generated and sent to LBS providers along with the actual position to hide users' location (Kido et al., 2005). This technique uses k-anonymity metric to measure users' location privacy. This approach can decrease the usability of actual data over dummy data, but the functional advantage of this approach is that the user is able to generate dummies without the need of TTP (You et al., 2007).

Cloaked region is created using spatial and temporal cloaking in order to satisfy incorporated privacy metric, i.e., k-anonymity, that maintains location privacy preservation (Gruteser and Grunwald, 2003; Sweeney, 2002). This technique can also be used with other PETs and privacy metrics such as l-diversity or t-closeness (e.g., `CliqueCloak` (Lee et al., 2011)). It is more suitable for LBS applications that require user's location information at a specific time. But, with continuous querying, the cloaked region becomes extremely large; and therefore, it fails to provide accuracy/usability (Pan et al., 2012).

Mix-zone technique refers to a service restricted area, where mobile users can change their pseudonyms so that the mappings between their old pseudonyms (i.e. entering the restricted area) and new pseudonyms (i.e. exiting the restricted area) are not disclosed (Beresford and Stajano, 2004). This technique is suitable for LBS applications that continuously track users' movements. It depends on geometric location transformation and uses privacy metrics to measure the level of achieved location privacy.

Location Perturbation and Obfuscation Obfuscation-based techniques perturb (i.e., alter or move from its original position) the actual location information while maintaining a binding with the user's identity. The idea of this approach is to utilise a log of historical user locations rather than the real-time location information to generate cloaked or obfuscated regions (Pingley et al., 2011; Quercia et al., 2011). Again, it can use k-anonymity as a metric to measure users' location privacy.

Query Privacy Metrics A number of metrics to measure query privacy in mobility sets that have been defined by taking into account the functionality of the proposed LPPM (Chen and Pang, 2012). However, amongst these, the largely adopted privacy metrics are k-anonymity and location entropy. More recently, owing to its formal privacy guarantees, the differential privacy (Andrés et al., 2013) notions have been applied to location-based applications and data mining practices as privacy metrics for handling and analysing private data. However, the differential privacy claims effectiveness after the private data is already collected by service providers (e.g., before releasing a dataset for public use); hence, it fails to guarantee privacy at runtime (i.e., during any private data collection). Definitions of the three popular privacy metrics are given below:

1. In *k-anonymity*, a number k is taken as the metric of users' query privacy, which is the size of the anonymity set of the issuer. This means, for a given query, a total of k locations are sent to the service provider, which is then unable to identify the user's real location with a probability higher than $\frac{1}{k}$.

2. *Location Entropy* the concept entropy of a random variable X is defined as $H(X)$= - $\sum_{x \in X} p(x) \cdot \log p(x)$, where X is the domain (all possible values) of X.

3. *Differential Privacy* (Andrés et al., 2013) is a privacy notion that is not based on a syntactic privacy notions (e.g., k-anonymity), but it relates uncertainty at an individual level to the noise or randomisation used in LPPMs. Formally, differential privacy is defined as follows:
   A randomized function K gives ε-differential privacy if for all data sets D and D' differing on at most one row, and all $S \subseteq$ Range(K),
   i.e., $Pr[K(D) \in S] \leq exp(\varepsilon) \times Pr[K(D') \in S]$.

Few authors have attempted to highlight various concepts of location privacy including user identity, LBS query types, query processing, privacy attacks and so on (Mokbel, 2007; Shin et al., 2012; Wernke et al., 2014). For example, for privacy preservation, Mokbel (2007) categorised LBS queries into three types: private queries over public data, public queries over private data, and private queries over private data. Whereas, Shin et al. (2012) categorised LBS queries privacy into two types: query privacy and location privacy. Wernke et al. (2014) states three different attributes of the location query: identity, position, and time. Furthermore, Chow and Mokbel (2009) described how these techniques can be used in different architectures including client-server, trusted-third party (TTP), mobile peer-to-peer and fully distributed. We refer the reader to the cited bibliography for details. However, all of them emphasize that the privacy of LBS query depends on two main factors: the protection of users' sensitive data within the query, and the unlinkability between the user's location and the query. Despite all their efforts made to guide future research in privacy-preserving LBS applications, the proposed LPPMs still lack standardisation, certainty, and are heavily centralised (i.e., rely on trusted location-based server/anonymisers). This makes privacy-preservation of LBS queries still an open issue that needs more attention.

2.4.2 Cryptographic Techniques

Cryptography-based LPPMs are predominantly used when location servers are not trusted. Example of cryptographic techniques used in mobile applications for LBS query privacy are given below:

PIR based techniques The Private Information Retrieval (PIR) protocol based LPPM uses standard public key cryptography to provide location privacy without the need for privacy metrics or TTP. The PIR protocol allows clients to retrieve the information privately from the server, without the server learning what information was requested by the client. This protocol provides maximum query privacy, but it causes high computational and communication overhead on the server, which makes it unfeasible to be practically implemented in LBS applications or systems (Ghinita, 2013; Papadopoulos et al., 2010).

Other encryption based techniques for query privacy Encryption based approach to query un-trusted servers are heavy-weight such as implementation of homomorphic, symmetric and asymmetric encryption (Hu et al., 2011; Wong et al., 2009). Compared to symmetric key encryption, asymmetric key encryption nodes are relatively much more expensive if used for continuous location querying, e.g., `LocX`, a location-based social application (Puttaswamy et al., 2012). Other cryptographic primitives like group signature, blind signature, and ring signatures have been used for anonymous user authentication or other access control operations to achieve privacy preserving communication network (Ren and Wu, 2010). However, in LBS applications, cryptographic primitives can add heavy computation and communication costs if used in continuous or spatial-temporal tracking queries.

In short, a cryptographic approach in LBS query handling is more suited for privacy-preserving, but involved cryptographic operations can incur high query processing, computational and communication overheads on the servers. As a result, this opens another research challenge to minimise cryptographic computation overheads on the location servers.

In summary, the main issue with applying non-cryptographic and cryptographic schemes for query formation and processing is that they rely entirely on the data collection servers to comply with location privacy. Moreover, LPPMs based on the above cryptographic schemes and PETs have been tested on service providers' data collection servers but neither are implemented on mobile platforms, nor on the actual app operation. Hence, they rely on theoretical assumptions - like trusted infrastructure to provide the privacy protection, requiring a group of similar app users to be at the same time and same place. Thus, the lack of usability and deployment feasibility are the two key constraints that hinder the adoption of any LPPM for query privacy in the mobile app ecosystem.

2.5 Layer 3: Network Communication

In location-based systems, a wireless network is used for calculating the user's device location by having network nodes communicating with each other. Hence, in Layer

3, we review LPPMs used to identify the possibility of improvements for mobile/LBS users in: 1. wireless networks and 2. communication protocols – including mixed networks, onion routing.

2.5.1 Wireless Networks

Wireless networks can leak a lot of sensitive user information, especially at the data link layer. Every mobile device that has been deployed with the active probing process, will transmit the identifiers of previously accessed wireless APs in plaintext, including surrounding cell-tower identifiers and other radio signal emitting devices. These identifiers and other unencrypted sensitive information can be easily intercepted and disclosed by eavesdroppers, leading to serious privacy attacks – such as building extensive user profiles. For this reason, it requires data link layer encryption algorithms to be in place; hence, several studies are attempting to minimise the security and privacy risks associated with active probing in wireless networks at the data link layer (Greenstein et al., 2008; Kim et al., 2014; Vratonjic et al., 2013).

Advertising businesses Businesses operating on wireless networks represent severe security and privacy risks, no matter whether it is indoor or online/global. Such businesses collect user's location and other sensitive information (i.e. personal habits, movement patterns, etc.) and pose the major privacy risks (Jan et al., 2000). Further, third-party ad-libraries used in mobile apps also collect private location information (Grace et al., 2012). This emerged as a new research challenge towards development of privacy-preserving advertising systems to avoid uncontrolled monitoring, e.g., *MobiAd* (Haddadi et al., 2010) and Privad (Guha et al., 2011).

Crowdsourcing Crowdsourced WiFi hotspot allows constant monitoring of users' accurate locations, their movements, and other personal data, in fact, the monitoring can be done secretively, without users necessarily being aware of it (Liu and Li, 2012). Even data received from mobile devices around the user can be gathered and analysed. Therefore, researchers and industrial groups are trying to balance the security, privacy and utility aspects of crowdsourcing methods to be applied in advanced positioning systems (Agarwal and Hall, 2013; Quercia et al., 2011). In particular, enhancing wireless network security and user privacy is critically essential for successful deployment of WiFi technology based positioning or user localising systems.

2.5.2 Network Communication

Existing network layer solutions mainly depend on anonymous network communication techniques, in which even if the data packet is encrypted, both the source and destination addresses located in the packet's internet protocol (IP) header are still visible to an eavesdropper (Erdin et al., 2015). Therefore, academia and industrial research communities are focused on designing and building privacy and security schemes as an infrastructure that runs on the top of the private IPs allowing users to communicate with each other without disclosing their network identifiers. These anonymous communication protocols and anonymity metrics are discussed below.

Anonymous communication protocols

Communicating over an anonymous network can provide LBS users with strong unlinkability, unobservability, and anonymity. Some of the successful anonymous routing protocols are Anonymizer (2016) and TOR (2016). Both of these protocols deal with anonymous service usage at the network layer while communicating over the Internet (i.e., the server can see the location data without knowing the identity of users). However, the TOR network creates a circuit with randomly selected TOR relays, which encrypts the original data from source to destination including the destination IP address. Another protocol called Crowds (Reiter and Rubin, 1998) is based on the idea of blending into a crowd (i.e., a user's request can either be submitted to the end location server or forward it to another randomly chosen router). As a result, no third-party knows the origin of the requested query. Whereas, Hordes (Shields and Levine, 2000), uses multicast routing to achieve anonymous communication over the Internet. However, these protocols have to be prevented from several security attacks such as the inference of the user location using the mobility trace without affecting the performance of the LBS applications. Amongst all the above protocols, only a few anonymising networks have been tested for the mobile Internet scenario.

Measuring Anonymity at Network Layer

Anonymity metrics, such as set size, k-anonymity, individual anonymity degree, and entropy, are used to measure anonymity in an anonymous network (Kelly et al., 2012). The literature suggets that both anonymous routing and anonymous communication protocols require further research to complement/ improve privacy preserving at the network layer; hence, they cannot work efficiently with LBS applications. Also, there are other cryptographic primitives that provide a high degree of anonymity, but cannot be applied to network communication due to high cost in terms of network traffic and processing, e.g. the dining cryptographers Chaum (1988); Krasnova et al. (2016).

Table 2.1 Critical comparison and privacy analysis of LPPMs at each layer. *Satisfaction of the four privacy properties (● = fully, △ = partially, ○ = not at all)*

Layers		Techniques	Drawbacks	Deployment	Unlink.	Unobserv.	Anonymity	Cont. Info. Disclosure
L1	Mobile OS	-Preventive mechanisms (sandboxes, digital signatures, permissions)	-Vulnerable to over privileged apps, unauthorised data gathering, colluding access attacks and malicious code attack	Apple (2016) Android (2017)	○	○	○	△
	App & Repository	-Redelegation of policies and regulations	-Vulnerable to malicious or curious service providers	Bettini and Riboni (2015)	○	○	○	△
		-Detection mechanisms (i.e., host and cloud)	-Needs to be installed separately, works after the data is leaked	Enck et al. (2014), Zhou et al. (2011)	○	○	○	△
		-App specific location sharing controls	-Requires user interference and app-code modifications	Toch et al. (2010)	△	△	○	●
L2	Non-Cryptographic LPPMs	-Dummy locations	-High dummies reduce usability	You et al. (2007)	●	●	●	○
		-Cloaked region	-Vulnerable to continuous querying	Pan et al. (2012)	●	●	●	△
		-Mix-zone	-Restricted to specific area	Beresford and Stajano (2004)	●	●	●	●
		-Location Perturbation and Obfuscation	-Constant data obfuscation reduces data usability	Pingley et al. (2011)	●	●	●	○
	Cryptographic LPPMs	-PIR	-High computational and communication overhead	Khoshgozaran et al. (2011)	●	●	●	●
		-Encryption	-Requires expensive encryption nodes, infeasible towards continuous querying	Puttaswamy et al. (2012)	●	●	●	●
L3	Wireless Networks	-Privacy preserving advertising and crowdsourcing	-Active probing process vulnerable to location privacy attacks	Haddadi et al. (2010),Grace et al. (2012)	●	●	●	○
	Network Communication	-Anonymous Communication protocols	-Expensive and vulnerable to internal attacks, high cost and communication overhead	TOR (2016)	●	●	●	○

2.6 Research Gaps

Standards for LBS applications exist in both the 3GPP (2017) and IETF (2017) arenas. The need to securely gather and transfer location information for LBS, while at the same time protecting the privacy of the users involved have been identified. However, industry and academia are making joint efforts to identify the requirements for LBSs in future large-scale community and better addressing end-user concerns. Apparently, to come up with an efficient socially-acceptable solution, all the above three levels must be considered. Current research demands LPPM that protects mobile user's privacy at every level and benefit both the end users and service providers. However, achieving such a solution is currently still an open research question and requires further investigation in this field. Table 2.1 illustrates critical comparison of most common approaches at each layer, and the privacy analysis that determines to what extent they satisfy the four privacy properties. We have analysed each layer's shortcomings but only Layer 2 approaches fully satisfy the privacy properties; however, they lack in deployment feasibility and usability. Based on the outcome of the study on state-of-the-art, we present a summary of research gaps identified at each layer.

- In Layer 1, there are two main research challenges for the location privacy issue: first, to improve mobile OS security models against location privacy attack, and second, to minimise/avoid unnecessary collection of users' personal information by service providers. Insights from mobile OS security models regarding location privacy attacks attract the attention for a more rigorous vetting process for regulation of third-party apps on MPPs' repositories. Since the permission-based OS controls are vulnerable to location privacy attacks, the mobile OS security models require additional system and/or application level solution against malicious code and location data leakage.

- In Layer 2, LBS query privacy depends on two main factors: the protection of users' sensitive data within the query, and the unlinkability between the user's location and the query. We distinguish two research challenges in this regard. First, to identify effective LPPMs to handle LBS query privacy before the private data is already collected by the service providers (i.e., not just to rely on service providers' trustworthiness). One of the potential approach (PbD) is to apply privacy techniques or metrics such as l-diversity, t-closeness and differential privacy on user's device in order to handle LBS query privacy at runtime; this will also bring data minimisation in the current mobile app ecosystem. Second, improving cryptographic approaches by reducing the computational and communication overheads on the location servers' side.

- In Layer 3, anonymous and unlinkable sharing of location information between the LBS applications and location servers over the network can be further investigated with the network layer's viewpoint. To achieve privacy-preserving network communication, privacy mechanisms should also be applied at data link layer to stop passive or active eavesdropping and improve anonymous communication protocols.

2.7 Related Work

Considering the above analysis on the state-of-the-art, we adopt the PbD approach aiming to fill most, if not all, of the aforementioned research gaps. Our approach aims to bring location privacy in mobile platforms that benefit both end users and service providers. Our research provides a novel theoretical model, `LP-Cache`, and its implementation, `PL-Protector`, on `Android` platform as a working system (i.e., a proof-of-concept / prototype). Our proposals incorporate a caching technique on the user's device to determine their geographical location in a privacy-preserving manner, and with minimum cache storage requirements. Hence, this section reviews relevant related work with regards to our research proposals. Considering `LP-Cache` and `PL-Protector`, we have divided the related work in two sections: cache-based LPPMs and other LPPMs applied on mobile platforms for users' location privacy preservation.

1. Cache-based LPPMs - As mentioned earlier, PETs and other cryptographic schemes have been proposed for the query privacy preservation between the app/LBS providers in the different architectures and infrastructures. Along with PETs, few authors have used caching technique to address location privacy challenges. For instance, MobiCaché (Zhu et al., 2013) applies k-anonymity for caching location-based queries. Similarly, Niu et al. (2015) attempts to improve k-anonymity based caching by adding dummy locations. However, both proposals require a trusted infrastructure to maintain location privacy. Similar to our approach, `Caché` (Amini et al., 2011) maintains an on-device cache, but it stores entire location-based queries contents and data-types to be re-used in future LBS queries that demands heavy cache storage requirement. Besides this, `Caché` also requires the willingness of app developers to modify the way their app access user's location data. All these cache-based systems, either intended to generalise or obfuscate the LBS query or minimise the number of queries sent to the app providers, but they do not provide privacy from WiFi content distributors. Moreover, their implementation either relies on theoretical assumptions or lacks deployment feasibility; hence, they cannot be incorporated in mobile platforms or in the ecosystem. Whereas, our proposals only cache private network fingerprints and mapped geo-coordinates, which significantly reduces the cache storage and memory requirement, and it considers installed apps as black boxes and does not require the app developer to modify the app code.

2. Other LPPMs for mobile platforms - A few studies have proposed static and dynamic monitoring systems to detect location privacy leaks in mobile platforms. The former method statistically analyses apps by creating permission mapping, generating call graphs and data flow analysis to report privacy leaks for further auditing, e.g, `AndroidLeaks` (Gibler et al., 2012) and `PiOS` (Egele et al., 2011) for `Android` and `Apple iOS`, respectively. The application of dynamic methods involves modification of the mobile platform. For example, `TaintDroid` (Enck et al., 2014) adds taint[2] tracking information to sensitive sources calls from apps, and it tracks location data flow as it generated through applications during execution. `MockDroid` (Beresford et al., 2011) relies on instrumenting `Android's` manifest permission mechanism to mock sensitive data from OS resource, including location data, which can affect apps' usability and functionality. Such dynamic monitoring methods can only be deployed on mobile devices that are rooted or jailbroken. Hence, in our research, we aim at providing a platform independent solution that is scalable and provides usable privacy. Our model not only monitors the location sources but also modifies, if required, the generated location based on pre-defined user permissions. Furthermore, the service called `Koi` (Guha et al., 2012) is cloud-based and demands numerous changes in the smartphone app ecosystem. The app developers are required to use a completely different location API, which implements a location comparison mechanism and matching criteria, in order to access the device location. Similar to our prototype implementation method, Fawaz and Shin (2014) rely neither on the adaptation of the app code modification nor on the existence of trusted infrastructure (e.g., TTP). However, it does not allow the user to control wireless and location data that is shared with the location provider and/or the app provider. This issue is mainly due to considering the location provider as the only source of the user location when developing location-based apps. Moreover,

[2]taint is a label that is used to detect when the user's sensitive data leaves the system via third-party applications.

Fawaz and Shin (2014) applies *indistinguishability* on the location data and PoIs that increases the computational overhead and memory consumption over time.

Besides, mobile devices do not only send vast amounts of location data to app providers but also to location providers creating different location privacy shortcomings (Almuhimedi et al., 2015; Shklovski et al., 2014). In this regard, limited work has been published on privacy preservation from the location provider's perspective (Damiani, 2011; Doty and Wilde, 2010). Damiani (2011) proposes a theoretical approach for privacy-aware geolocation-based web services to encourage further research to minimise the amount of location data being shared with the location provider. This is mainly due to the location provider being considered as the only source to get the user location when developing any location-based app. Our model minimises the process of wireless AP data collection by the WiFi content distributors or service providers, and we control information disclosure within the generated LBS query (e.g.,PoIs) since it will be sent to the third-party app provider. Moreover, we also implement a working system as a proof-of-concept that demonstrates how our model provides practical and scalable LPPMs, and it can automate recommended user policy to enforce location privacy in mobile platforms.

2.8 Summary

Secure gathering and transfer of location data via smart mobile devices while at the same time preserving users' privacy are concerning needs. Research and industry communities are making joint efforts to identify the requirements for LBS applications in future large-scale scenarios and addressing end-user concerns. Studies based on cryptographic schemes and PETs have been tested on service provider's data collection servers but neither are implemented on mobile platforms, nor via actual app operations.

In addition, lack of usability is one of the factors that hinders the adoption of existing privacy-preserving solutions. In this chapter, we have proposed a state-of-the-art classification, which embraces a holistic picture towards approaching privacy-preserving LBS for mobile users. We have identified three different layers: mobile Platform, query privacy, and network communication. We have described all the protocols, techniques and infrastructures ranging from the application layer to the network layer used for location privacy, and assess them against the satisfaction of the privacy properties. Hence, we claim our study provides a better-focused classification of the state-of-the-art to guide researchers in the field. Considering the literature findings, we have identified research gaps at each layer to come up with an efficient socially-acceptable solution. We then present related work with regards to our proposals that aim to bring practical, secure and efficient location privacy solution for mobile platforms in the smartphone app ecosystem and beyond. We elaborate on our model and its implementation in next chapters.

Chapter 3

LP-Cache: Location Privacy Preserving Model for Mobile Apps

Research findings from this chapter have been published in conference papers[a][b]

[a]Patel, A., & Palomar, E. "Protecting Mobile Users' Private Locations through Caching". *In: Lecture Notes in Communications in Computer and Info. Science (CCIS)*, Springer 2016.

[b]Patel, A., & Palomar, E. "LP-Cache: Privacy-aware Cache Model for Location-based Apps". *In: 13th Int. Conf. on Security and Cryptography (SECRYPT)*, pp. *183-194*, Springer 2016.

3.1 Introduction

In this chapter, we present an analytical study that was conducted to understand how the location calculation process works in smartphones and how apps access and collect users' location data. This study also analyses location privacy threats and challenges within the smartphone app ecosystem. By critically investigating the study outcome, we design, develop and propose a theoretically sound and platform independent model called *Location Privacy Cache* (`LP-Cache`). To encounter identified location privacy challenges and threats, `LP-Cache` makes the user device play a bigger role in the location calculation process. It applies a cache-based technique to determine the position of the client device by means of wireless access points and achieves data

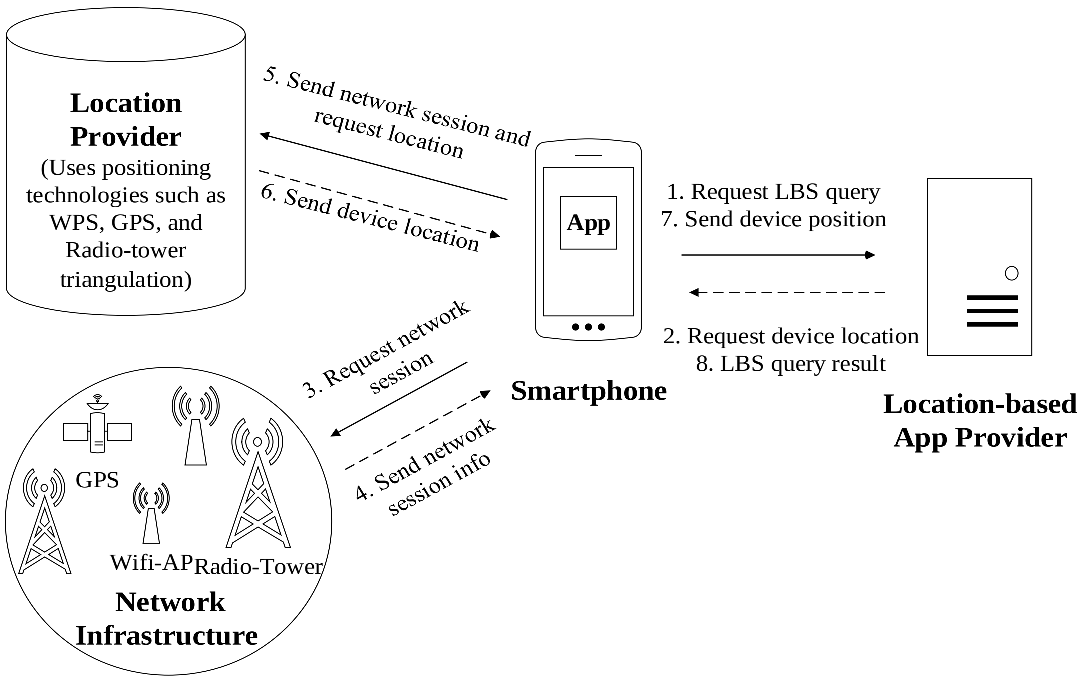

Location Provider
(Uses positioning technologies such as WPS, GPS, and Radio-tower triangulation)
5. Send network session and request location
6. Send device location
App
Smartphone
1. Request LBS query
7. Send device position
2. Request device location
8. LBS query result
Location-based App Provider
3. Request network session
4. Send network session info
GPS
Wifi-AP
Radio-Tower
Network Infrastructure

Figure 3.1 Current location computation architecture

minimisation in the process. It envisions beyond the simple grant/deny access method and provides the user with advanced permissions to decide the extent of disclosing location data with service providers. Several caching based solutions (Amini et al., 2011; Niu et al., 2015; Zhu et al., 2013) have been proposed to minimise the risk of the fundamental location privacy threats, but they lack deployment feasibility. They rely on unrealistic assumptions such as vast cache data storage requirements, or on the app developers modifying the code to incorporate their cached databases. Whereas, `LP-Cache` incorporates the caching technique in a privacy-preserving manner and, comparatively, with minimum cache storage estimation and is feasible without modifying installed apps.

The rest of the chapter is organized as follows. Section 3.2 outlines the current location computation process and its analytical evaluation. Section 3.3 presents the architecture of `LP-Cache`, and Section 3.4 fully elaborates on the design decisions and its possible implementation. We evaluate the feasibility and usability requirements in Section 3.5. Finally, Section 3.6 summarises this chapter.

3.2 Current Location Computation Process

In this section, we describe roles and processes involved in the current architecture for computation of the user's device location.

3.2.1 Current Architecture

The current location computation architecture (see Figure 3.1) to use location-based apps on smartphones comprises of four main entities: 1. Smartphones with installed apps, 2. App Provider, 3. Network Infrastructure, and 4. Location Provider. This architecture mainly relies on third-party location providers, e.g., Google Location Service (2016), Skyhook (2016), and Navizon (2016). The location provider represents the central database, which maps the received signatures of nearby wireless access points to their geo-coordinates and, hence, handles every geolocation request. Therefore, the location provider has constant access to the user's location as well as to the trajectory and mobility data. To respond to any location request, the location provider maintains a database of surrounding network infrastructure, including WiFi APs, cellular-towers, and IP addresses, which must be mapped to their exact geographical coordinates (i.e., geo-coordinates). Compared to GPS and cell-tower based positioning, WiFi Positioning Systems (WPS) are nowadays considered as a very accurate method for location calculation (Skyhook, 2016). Location providers rather use enhanced WPS than GPS, primarily due to the current smart-mobile devices benefiting from built-in WiFi clients that perform faster than most expensive GPS receivers. As a result, *beacons scanning* and *crowdsourcing* of WiFi APs (i.e., hotspots) are the two main positioning techniques used by location providers. This enables the service provider to get a user's precise locations at all times. Hence, effective privacy preservation measures are needed in the current process to mitigate location privacy threats. We need to consider, how WPS are deployed and used by location providers. WiFi APs continuously announce their existence in the way of network frames/beacons and transmit their Service Set Identifier (SSID) and Basic Service Set Identifier (BSSID)/MAC addresses. Location providers use these WiFi APs identifiers to create network signatures and map them with their respective geo-coordinates, also called geolocation. IEEE 802.11 states two standardised ways to collect beacons from WiFi APs: 1. Active scanning, and 2. Passive scanning. Location providers are capable of deploying systems with either active scanning, passive scanning, or both together. Location providers use three different ways to collect geolocation of WiFi APs:

1. *Statically* - they collect WiFi beacons by the so-called the *wardriving* process (Sapiezynski et al., 2015). Basically, they map the equipped vehicle's exact geo-coordinates along with the signal strength of the captured beacons from surrounded WiFi APs.

2. *Dynamically* - they can collect data from WiFi APs automatically once the user device uses location services, e.g. maps and navigation apps. The user device, as configured to be geolocated, acquires unique identifiers from the surrounding WiFi APs, even if the network is encrypted, and then sends it over to the location provider in order to perform geolocation calculation. The collected information is utilised to build and update the database autonomously, for example, by applying crowdsourcing WiFi hotspots (Zhuang et al., 2015).

3. *User input* - they encourage users to manually input the WiFi APs' information, i.e., BSSID and the geo-coordinates, into their databases, e.g., Skyhook[1] to register WiFi APs.

3.2.2 Evaluation of the Current Location Computation Process

To analyse location privacy risks and vulnerabilities in the smartphone app ecosystem, we evaluate the current location handling process.

Experimental setup In this evaluation, experiments were designed to understand whether there are any differences in the location calculation process on each of the three mobile operating systems/platforms. We first collected and investigated published policies and procedures, legal regulations, guidelines related to location privacy of mobile users. Along with the investigation of published resources, we conducted a series of experiments on different mobile devices installed with `Android`, `Windows Phone`, and `iOS` operating systems to categorise the data flow and control flow in the current location computation process. With the assistance of sniffers, such as Wireshark (2016) and tPacketcapture (2016), we captured and analysed sequence and location data transmission when using location-based apps. We have collected a number of sessions of location-based apps communicating different types of LBS query (e.g., Navigation and Friend Finder) with the app provider. Figure 3.2 shows the screenshot of captured packets: (a) app sent LBS query that includes device's geolocation to the server, and (b) after receiving a LBS query from the app, the server verifies the device's current location to send the appropriate response. We maintained 15 to 30 mins sessions of packet capture, to identify the sequence of the communication process, data flow,

[1]Submit a Wi-Fi Access Point. See http://www.skyhookwireless.com/submit-access-point (last access in March. 2016).

control flow, type of data shared between the OS, apps, app providers and any other third party involved in the entire process.

Listing 3.1 Structure of a WiFi AP object cell-tower object sent to the location provider

```
{"wifiAccessPoints": [{
  "macAddress/BSSID": "11:22:33:44:55:YZ",
  "signalStrength": 50,
  "age": 0,
  "signalToNoiseRatio":-60,
  "channel": 8
  {
  "macAddress/BSSID": "11:22:33:44:55:YZ",
  "signalStrength": 50,
  "age": 0,
  }
}]}
```

Observation. Based on the results, it was observed that all of the three mobile platforms display common patterns of location data retrieval. The user device collects the unique identifiers from the surrounding network along with GPS data; and sends it to the location provider to get the exact device location. Listing 3.1 and 3.2 show the structure of the WiFi and Cell-tower objects sent to the location provider.

Listing 3.2 Structure of a cell-tower object sent to the location provider

```
"CellTowers":[
{
    "cellId": 01,
    "locationAreaCode": 415 ,
    "mobileCountryCode": 310,
    "mobileNetworkCode": 410'
    "age": 0,
    "signalStrength": -60,
    "timingAdvance":15
}
]
```

Calculation[2] of the user device's actual position is then performed by the location provider who sends back a location object (see Listing 3.3) containing exact geo-coordinates.

Listing 3.3 Structure of the location object received from the location provider

```
{"location": {
  "lat": 54.0,
  "lng": -0.012,
  },
  "accuracy": 190.2,
}
```

In short, the app developer over any mobile platform can utilise this location object to get the user's geolocation with no need to focus on the details of the underlying location technology. However, it is alarming that all the LBS queries sent by an app to the remote servers of the app providers are unencrypted and the location data can be easily intercepted or monitored by eavesdroppers; this poses high risks to user identification, mobility tracking, and profiling threats. Figure 3.3 shows captured packet that includes the structure and attributes of a LBS query sent to the app provider's server. A detailed description of the process sequence follows.

Process Sequence. Note that, on a smartphone, location sharing service settings must be 'ON' while using any location-based app. Figure 3.4 shows location settings on (a) `iOS` and (b) `Android` devices. These location settings highlight that both of these mobile platforms heavily rely on WiFi data for position calculation, specifically in indoor environments. Therefore, if the location sharing is 'OFF', then the device prompts for changing the setting from 'OFF' to 'ON'; otherwise, the user cannot use location services on any apps. The user device collects the unique identifiers from the surrounding wireless access points along with GPS data, and sends these over to the location provider to get the exact device location. Listing 3.1 and Listing 3.2

[2]Location calculation is commonly based on positioning technologies such as WiFi Triangulation and Cell-tower Triangulation, GPS Mapping, etc.

show the structure of the WiFi and Cell-tower objects sent to the location provider, respectively. Calculation[3] of the user device's actual position is then performed by the location provider who sends back a location object (see Listing 3.3) containing exact geo-coordinates. At the user device, this location object (see Listing 3.3) is shared amongst installed apps as well as with the app provider who will transmit it as a LBS query via the standard API (Android, 2016). Moreover, this location object is also used to estimate PoIs of users' daily lives. Simple eavesdropping on this location object is a major threat to this architecture even if users put in place the corresponding location sharing preferences[4], which generally are highly context sensitive and use dependent (Xie et al., 2014). Figure 3.5 illustrates the sequence of processes and messages involved in the current location computation architecture:

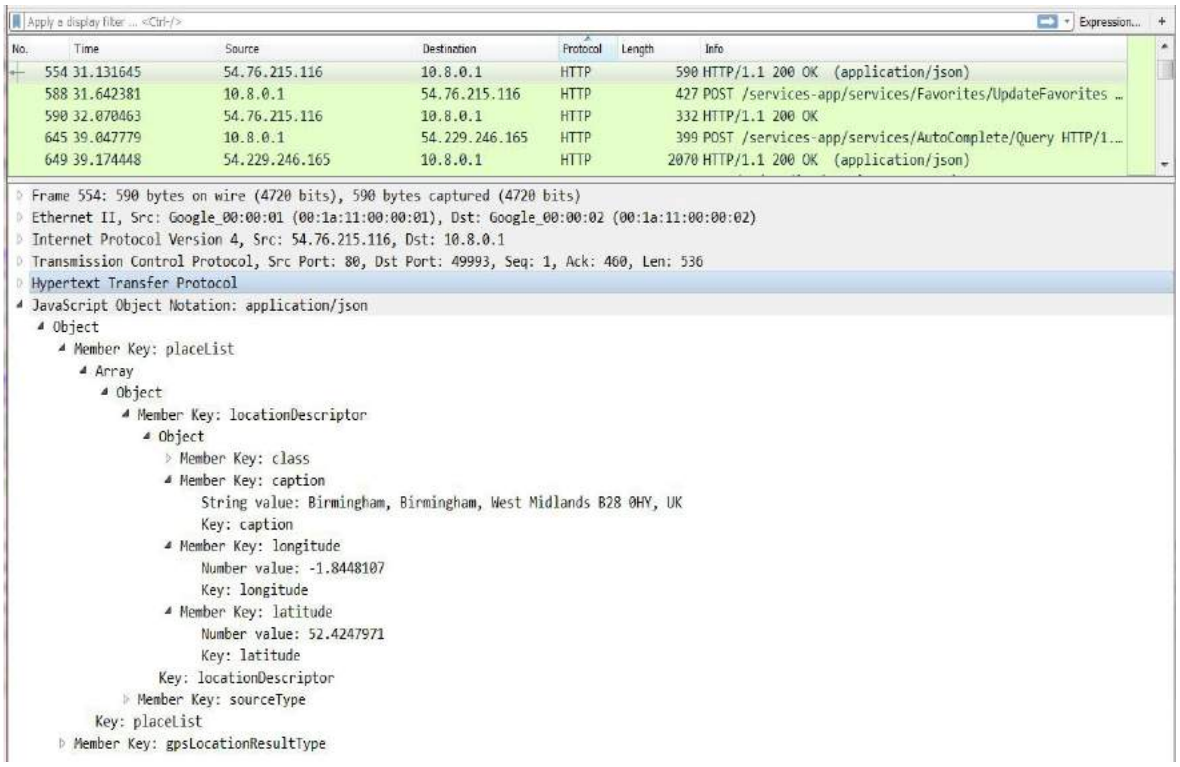
1. The app generates the LBS query, e.g., PoIs or nearest neighbor search, and sends it to the app provider. However, the app provider needs the client's location to tailor the service accordingly; and therefore, the LBS query must include the client's geo-cordinates.

2. To get the client's geo-coordinates, the app sends a request to the mobile OS via the standard location programming interface (i.e., location API).

3. The device then collects the surrounding network information via the built-in wireless client, cell-tower receiver, and GPS receiver. It sends the collected information to the location provider, which is considered as a trusted third-party. To get an accurate location, the location provider primarily prefer WPS mainly in dense urban areas and indoors, for they deploy many WiFi APs (Skyhook, 2016).

[3]using positioning technologies such as WiFi Triangulation and Cell-tower Triangulation, GPS Mapping, etc.

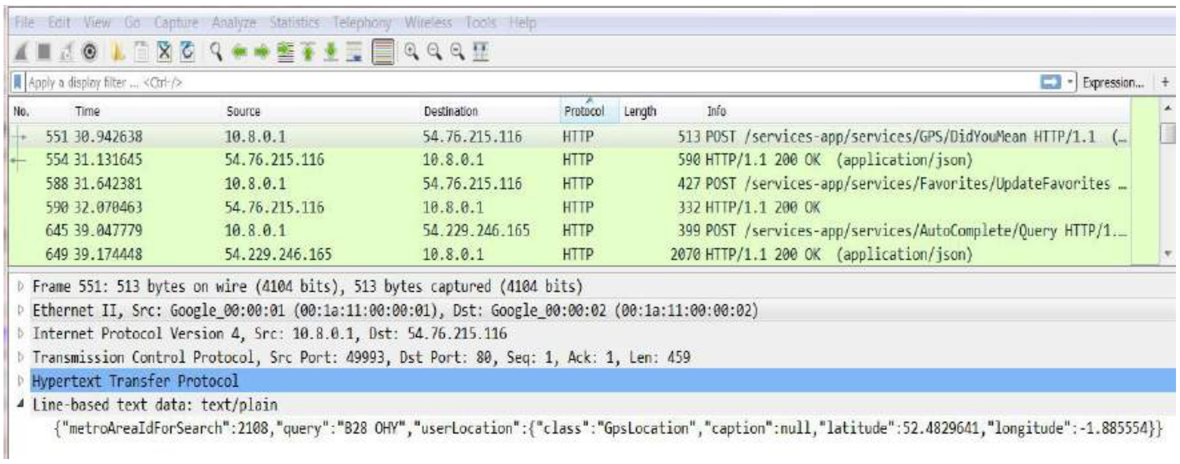
[4]Types and levels of controls for user location privacy settings depend on the OS and apps. In some cases, apps do not allow users to control others' access to their location data.



```
No.   Time        Source           Destination      Protocol  Length  Info
554   31.131645   54.76.215.116    10.8.0.1         HTTP       590 HTTP/1.1 200 OK  (application/json)
588   31.642381   10.8.0.1         54.76.215.116    HTTP       427 POST /services-app/services/Favorites/UpdateFavorites …
590   32.070463   54.76.215.116    10.8.0.1         HTTP       332 HTTP/1.1 200 OK
645   39.047779   10.8.0.1         54.229.246.165   HTTP       399 POST /services-app/services/AutoComplete/Query HTTP/1.…
649   39.174448   54.229.246.165   10.8.0.1         HTTP      2070 HTTP/1.1 200 OK  (application/json)
```

```
Frame 554: 590 bytes on wire (4720 bits), 590 bytes captured (4720 bits)
Ethernet II, Src: Google_00:00:01 (00:1a:11:00:00:01), Dst: Google_00:00:02 (00:1a:11:00:00:02)
Internet Protocol Version 4, Src: 54.76.215.116, Dst: 10.8.0.1
Transmission Control Protocol, Src Port: 80, Dst Port: 49993, Seq: 1, Ack: 460, Len: 536
Hypertext Transfer Protocol
JavaScript Object Notation: application/json
  Object
    Member Key: placeList
      Array
        Object
          Member Key: locationDescriptor
            Object
              Member Key: class
              Member Key: caption
                String value: Birmingham, Birmingham, West Midlands B28 0HY, UK
                Key: caption
              Member Key: longitude
                Number value: -1.8448107
                Key: longitude
              Member Key: latitude
                Number value: 52.4247971
                Key: latitude
            Key: locationDescriptor
          Member Key: sourceType
      Key: placeList
    Member Key: gpsLocationResultType
```

(a) App sent LBS query that includes device's geolocation to the server

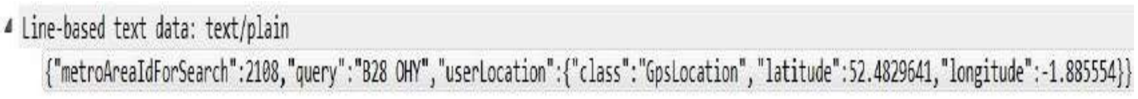

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...

```
No.   Time        Source           Destination      Protocol  Length  Info
551   30.942638   10.8.0.1         54.76.215.116    HTTP       513 POST /services-app/services/GPS/DidYouMean HTTP/1.1  (…
554   31.131645   54.76.215.116    10.8.0.1         HTTP       590 HTTP/1.1 200 OK  (application/json)
588   31.642381   10.8.0.1         54.76.215.116    HTTP       427 POST /services-app/services/Favorites/UpdateFavorites …
590   32.070463   54.76.215.116    10.8.0.1         HTTP       332 HTTP/1.1 200 OK
645   39.047779   10.8.0.1         54.229.246.165   HTTP       399 POST /services-app/services/AutoComplete/Query HTTP/1.…
649   39.174448   54.229.246.165   10.8.0.1         HTTP      2070 HTTP/1.1 200 OK  (application/json)
```

```
Frame 551: 513 bytes on wire (4104 bits), 513 bytes captured (4104 bits)
Ethernet II, Src: Google_00:00:01 (00:1a:11:00:00:01), Dst: Google_00:00:02 (00:1a:11:00:00:02)
Internet Protocol Version 4, Src: 10.8.0.1, Dst: 54.76.215.116
Transmission Control Protocol, Src Port: 49993, Dst Port: 80, Seq: 1, Ack: 1, Len: 459
Hypertext Transfer Protocol
Line-based text data: text/plain
  {"metroAreaIdForSearch":2108,"query":"B28 0HY","userLocation":{"class":"GpsLocation","caption":null,"latitude":52.4829641,"longitude":-1.885554}}
```

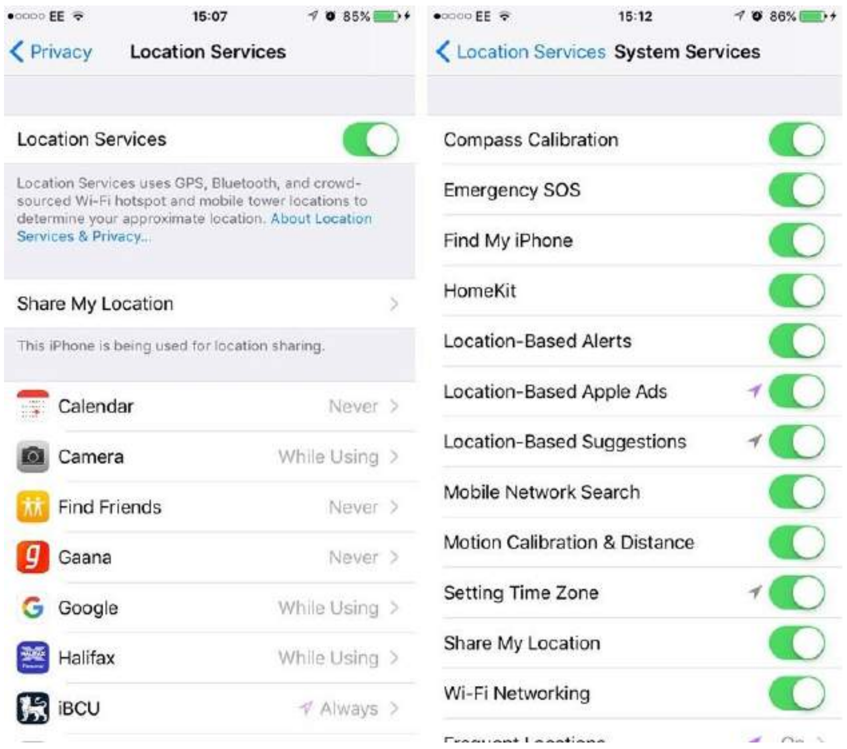
(b) After receiving a LBS query request from the app, the server is verifying device's current location to send appropriate response

Figure 3.2 Screenshots of captured packets using `tPacketCapture` tool



```
Line-based text data: text/plain
  {"metroAreaIdForSearch":2108,"query":"B28 0HY","userLocation":{"class":"GpsLocation","latitude":52.4829641,"longitude":-1.885554}}
```

Figure 3.3 A screenshot showing attributes of a LBS query sent to the app provider.



●○○○○ EE 15:07 85%

Privacy — Location Services

Location Services ✓

Location Services uses GPS, Bluetooth, and crowd-sourced Wi-Fi hotspot and mobile tower locations to determine your approximate location. About Location Services & Privacy...

Share My Location

This iPhone is being used for location sharing.

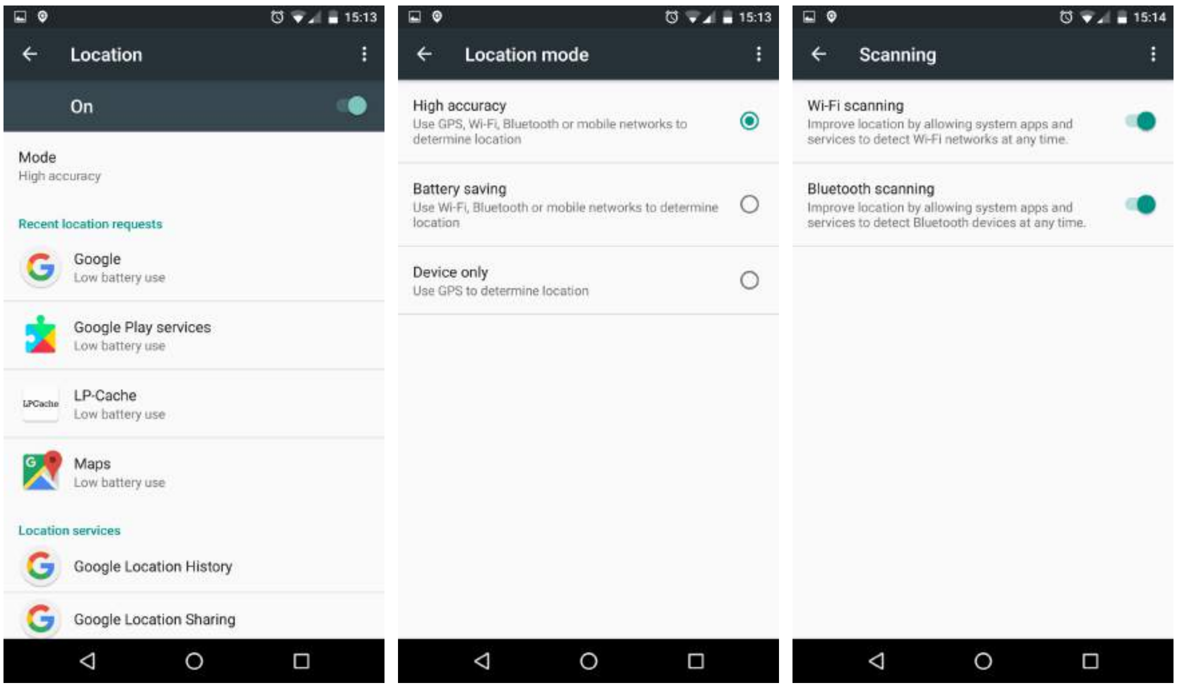
Calendar	Never
Camera	While Using
Find Friends	Never
Gaana	Never
Google	While Using
Halifax	While Using
iBCU	Always

●○○○○ EE 15:12 86%

Location Services — System Services

Compass Calibration	✓
Emergency SOS	✓
Find My iPhone	✓
HomeKit	✓
Location-Based Alerts	✓
Location-Based Apple Ads	✓
Location-Based Suggestions	✓
Mobile Network Search	✓
Motion Calibration & Distance	✓
Setting Time Zone	✓
Share My Location	✓
Wi-Fi Networking	✓

(a) iOS location settings and system services



15:13

Location

On ✓

Mode
High accuracy

Recent location requests

Google	Low battery use
Google Play services	Low battery use
LP-Cache	Low battery use
Maps	Low battery use

Location services

Google Location History

Google Location Sharing

15:13

Location mode

[✓ High accuracy] Use GPS, Wi-Fi, Bluetooth or mobile networks to determine location

Battery saving — Use Wi-Fi, Bluetooth or mobile networks to determine location

Device only — Use GPS to determine location

15:14

Scanning

Wi-Fi scanning ✓ — Improve location by allowing system apps and services to detect Wi-Fi networks at any time.

Bluetooth scanning ✓ — Improve location by allowing system apps and services to detect Bluetooth devices at any time.

(b) Android location settings, mode options and wireless scanning services

Figure 3.4 Screenshots of location settings and services on iOS and Android OSs

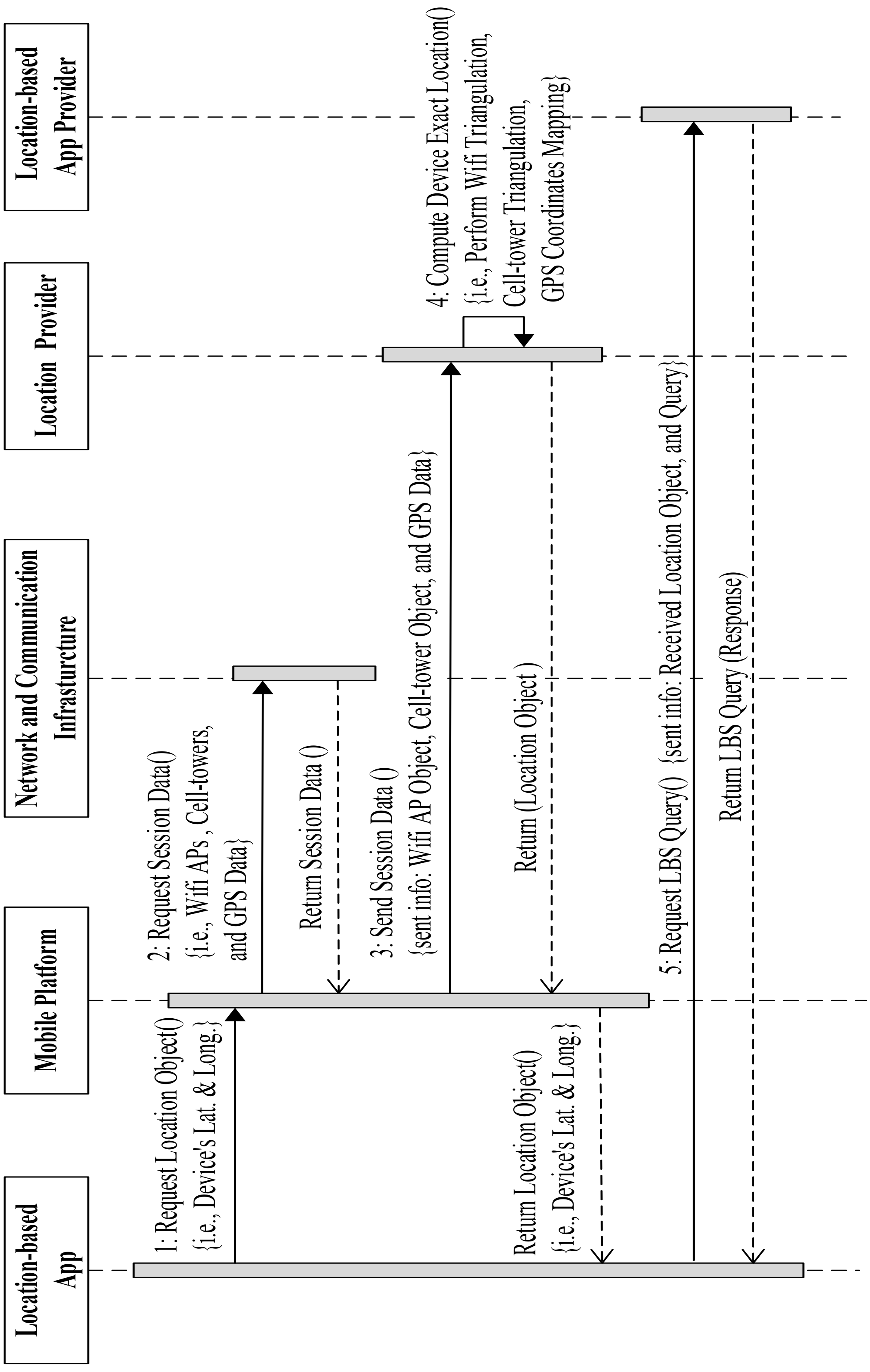

Location-based App
Mobile Platform
Network and Communication Infrasturcture
Location Provider
Location-based App Provider
1: Request Location Object() {i.e., Device's Lat. & Long.}
2: Request Session Data() {i.e., Wifi APs , Cell-towers, and GPS Data}
Return Session Data ()
3: Send Session Data () {sent info: Wifi AP Object, Cell-tower Object, and GPS Data}
4: Compute Device Exact Location() {i.e., Perform Wifi Triangulation, Cell-tower Triangulation, GPS Coordinates Mapping}
Return (Location Object)
Return Location Object() {i.e., Device's Lat. & Long.}
5: Request LBS Query() {sent info: Received Location Object, and Query}
Return LBS Query (Response)

Figure 3.5 Sequence diagram of current location computation process.

4. The location provider receives the network information, computes the geographical location and sends the geo-coordinates via location object (as shown in Listing 3.3).

5. Once the user device obtains the location object, it is sent to the app that includes the geo-coordinates in the LBS query via the standard location API Android (2016), and sends it to the app provider.

6. The app provider can then use the received location as part of their app's operations to send the corresponding reply to the LBS query.

3.2.3 Estimation of PoIs

A general LBS query consists of different attributes, e.g., LBS query {PoI, Latitude and Longitude, User-Info}, where included geo-coordinates estimate the device's geolocation and then generate the user's PoIs. Service providers use two categories of algorithms to estimate the user's PoI from collected smartphone data (Montoliu et al., 2013): 1. Geometry-based, and 2. Fingerprint-based.

Geometry-based algorithms use LBS queries (i.e., location data) to trace geo-coordinates, circles or polygons of regions to define the significant PoIs or private places where the user goes in real life.

Fingerprint-based algorithms obtain a list of places where the user goes, but provide no direct information about where the place is geographically located unless the fingerprint (or signature) of the observed fixed network infrastructure is mapped to the geo-coordinates. In general, PoI estimation techniques, which include fingerprinting-based algorithms, detect fixed radio/wireless environments that indicate a stay or frequent visits of the user's device for PoI estimation. Whereas, WiFi-based fingerprinting involves mapping of observed WiFi APs with signatures that are pre-stored on remote location servers to generate the list of PoIs.

In our research, `LP-Cache` uses fingerprinting to create private location database within the device instead of storing it on a remote location servers. This minimises the process of wireless AP data collection by the WiFi content distributors or location providers. In addition, `LP-Cache` controls information disclosure within the generated LBS query (e.g., PoIs and nearest neighbor) since it will be sent to third-party app providers.

3.3 LP-Cache Model

In this section, we describe fundamental elements of `LP-Cache`, including its threat model, design goals, architecture and main processes' sequence diagram. `LP-Cache`'s three main design goals are:

1. Data minimisation: It will reduce the data shared, transmitted and stored, and it also guarantees the minimum interaction with the location provider.
2. Practical and usable privacy: The user can set distinct privacy preferences for different apps and private places; the model works independently without the need for modifying the app's code; it cn be deployed in the current ecosystem without affecting LBS apps or OS functionality.
3. Efficiency: It will handle geographical location efficiently, i.e., computation intensity should not affect the apps' performance; the third-party app provider will not be able to infer the device's exact location without getting the user's consent; it will efficiently execute user's distinct privacy preferences for different apps and private places.

3.3.1 Threat Model

Apps deliberately collect the user's sensitive data, including location and other sensitive information as part of their operations. User tracking, identification and profiling (i.e. personal habits, movement patterns, etc.) are fundamental threats to location privacy (Fawaz and Shin, 2014; Felt et al., 2012; Wernke et al., 2014). Furthermore, the current direct link of smartphones to the location provider and the continuous flow of LBS queries that include the device's exact geo-coordinates over the network create a serious risk to the protection of users' sensitive information. This is even more challenging, in the presence of a malicious location provider and via advanced network sniffing practices.

`LP-Cache` computes the exact location within the user device, without service provider's involvement, and trusts the device on the storage of sensitive data. However, the user has still the option of giving consent for app providers or location provider to access their location. Mobile network providers might, however, collect users' location data via cellular clients. It is also excluded from our model the option of manually inserting the location data (e.g. street name, zip code, postcode) within LBS query.

3.3.2 LP-Cache Control Flow Architecture

Figure 3.6 depicts the block diagram for `LP-Cache`'s architecture; its main components are:

Permission Setter is the user interface (UI), which enables users to set and manage their private places and apply the personalised permissions when running installed location-based apps. Once received the user inputs, pre-set private locations are sent to the *Private Location Manager* module, and permissions are sent to the *Rule Mapper* Module.

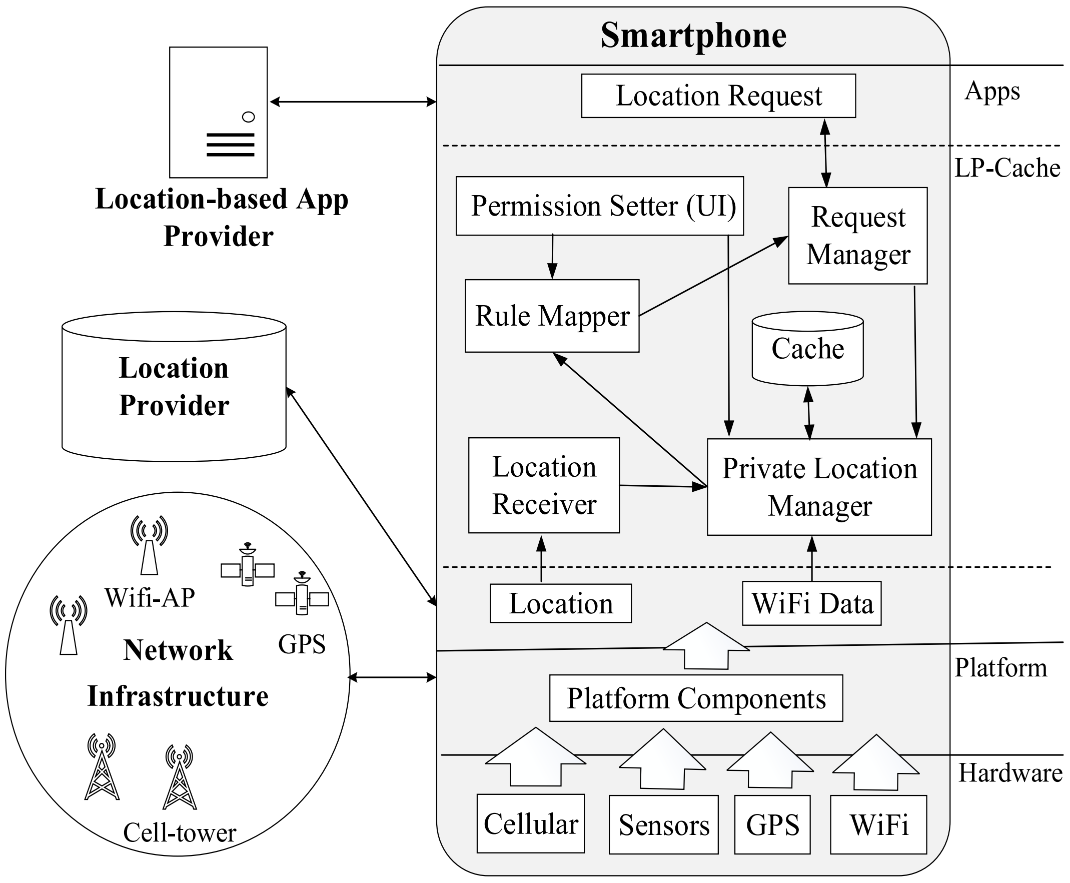


Figure 3.6 `LP-Cache` architecture.

Request Manager is responsible for intercepting the event of location access calls, and then lead the app's control flow to the *Private Location Manager* module. Besides, it will also be in control of receiving the processed user location (i.e., could be either anonymised or altered) from *Rule Mapper*, and then delivering it to the app in order to maintain every session's control flow.

Private Location Manager module's main task is to detect unique identifiers of the surrounding WiFi APs and compare them with the stored network fingerprints to determine whether the user is within the set of private places. User inputs from the *Permission Setter* will create network fingerprints for known private locations, which are then added or updated in the *Cache* database. Moreover, it maintains a binary flag to detect private places. In the case of a hit, the location data is sent to the *Rule Mapper.* Otherwise, the location is received by the *Location Receiver.* Whenever the *Private Location Manager* receives a new private place request, the received location is mapped to the detected network fingerprint and stored in the *Cache* database.

Rule Mapper dynamically collects and checks set permissions from *Permission Setter.* Once the representative location object is received from the *Private Location Manager*, it applies the user permissions to the location coordinates, alters them (if required), and outputs the processed location to the *Request Manager* module. If the flag is negative, then it forwards the exact location.

Cache is the established on-device cached database, and it is routinely queried by the *Private Location Manager* module, which can add, update and delete the cached location data. The locations in *Cache* are those which are to be protected, and they can also represent regions of space. Each entry is recorded along with a network fingerprint and geolocation that is acquired from the location provider.

Location Receiver module receives a location object, which includes the user device's geo-coordinates (as in Listing 3.3), from location providers and sends it over to the *Private Location Manager* for further processing.

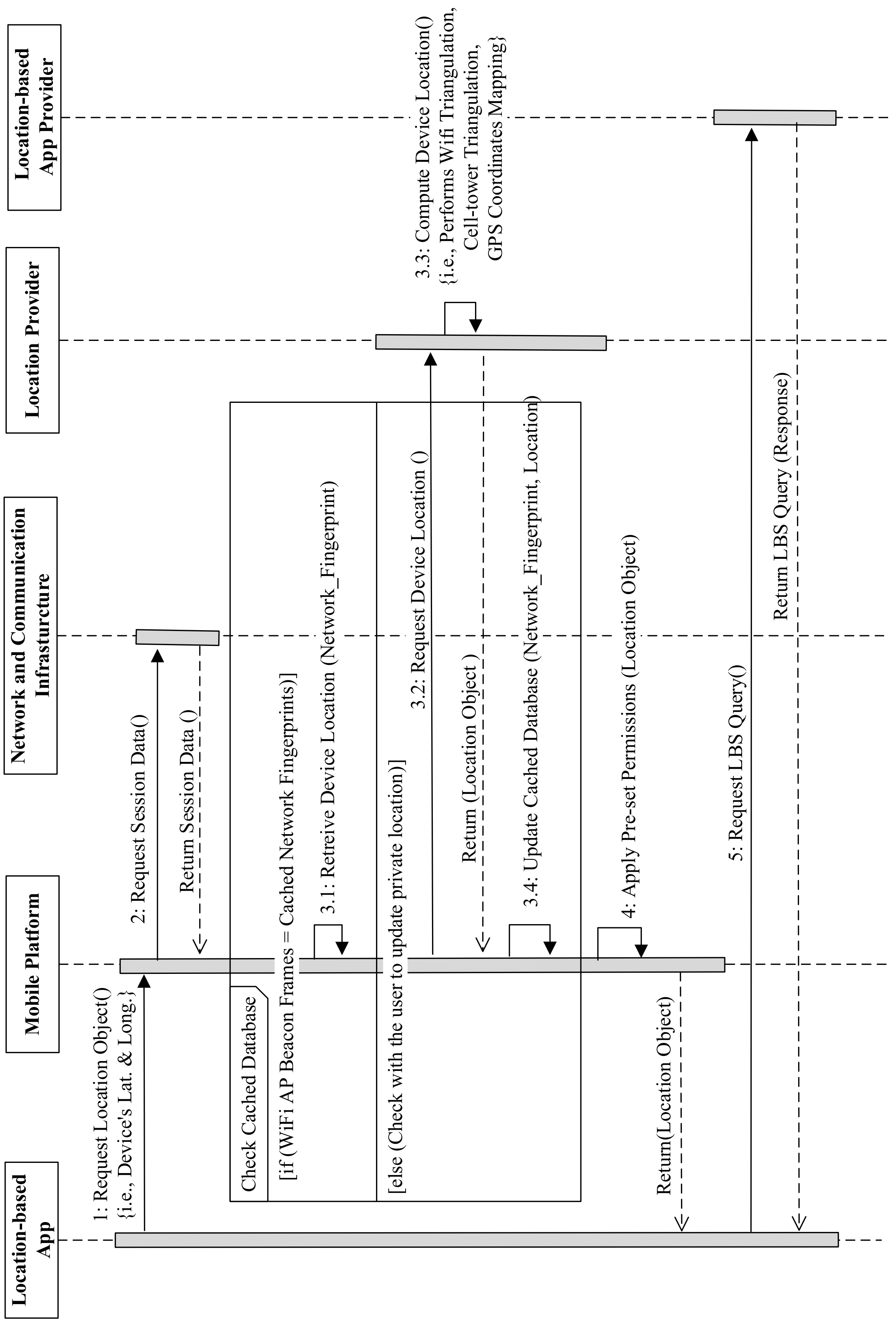

Location-based App
Mobile Platform
Network and Communication Infrastructure
Location Provider
Location-based App Provider
1: Request Location Object() {i.e., Device's Lat. & Long.}
2: Request Session Data()
Return Session Data ()
Check Cached Database
[if (WiFi AP Beacon Frames = Cached Network Fingerprints)]
3.1: Retreive Device Location (Network_Fingerprint)
[else (Check with the user to update private location)]
3.2: Request Device Location ()
3.3: Compute Device Location() {i.e., Performs Wifi Triangulation, Cell-tower Triangulation, GPS Coordinates Mapping}
Return (Location Object)
3.4: Update Cached Database (Network_Fingerprint, Location)
4: Apply Pre-set Permissions (Location Object)
Return(Location Object)
5: Request LBS Query()
Return LBS Query (Response)

Figure 3.7 Sequence diagram of location computation process using LP-Cache

3.3.3 Process Sequence

`LP-Cache` modifies the current location resource handling process; however, the involved entities (as in Section 3.2) remain the same. Figure 3.7 illustrates the sequence of processes and messages involved in `LP-Cache`:

1. At the event of the app requesting the device location, our service will intercept the request to get the location from the cache database instead of sending the request to the location provider.

2. Upon receiving the location request, our service will scan the surrounding network infrastructure.

3. Using observed network frames our service will execute as follows:

   (a) Our service compares the collected beacons with the stored network fingerprints to retrieve corresponding stored representative location coordinates.

   (b) In the case of an unmatched entry on the database, `LP-Cache` prompts the user two options either to input the location using UI, or to allow the query to be sent to the location provider that will calculate and send the current location coordinates. Note that this will only occur if the user has set the current location as private but the geo-coordinates are not cached.

   (c) The received location data for the encountered APs will be tracked within the local cache database for future use.

4. User location coordinates can be altered based on the privacy settings. `LP-Cache` provides three options for controlled information disclosure: (1) Adjust Location Granularity, (2) Obfuscate Location, and (3) No Change. The computed location is populated in the location object and sent to the app.

5. Once the app obtains the location object, it is then used by the app provider to send the corresponding reply to the LBS query via the standard programming interface/API (Android, 2016).

3.3.4 Example Scenarios

We have identified two different scenarios for using `LP-Cache` on a user's device as follows:

1. ***Bootstrapping scenario-*** *Bob* activates `LP-Cache` service and sets current location as home and apply permission setting. `LP-Cache` starts scanning the surrounding APs and creates a network fingerprint for *Bob*'s home, a private place that needs to be protected. Later, *Bob* decides to use `Facebook` app, which requests his current location. `LP-Cache` detects a location update call from `Facebook` app, only if per-app settings are not previously saved for `Facebook`, it prompts him to select one out of the three options for permission setting. Unless *Bob* changes the selected option via the user interface, the selected option will be saved and applied for future location calls from `Facebook` at *Bob*'s home.

2. ***Regular usage scenario-*** Assuming the bootstrapping process has been successful. When *Bob* uses any app at home, `LP-Cache` detects he is within a private palace (i.e., home) via the saved network signatures, retrieves the home location from the cached database, applies the privacy rule, and sends the processed location to any location requesting app.

3.4 Implementation Requirement

`LP-Cache` determines new design principle and privacy policy recommendation that forces the smartphone app ecosystem to make location data use patterns explicit, while preventing all other sensitive data flows. Further, one of the three design goals of `LP-Cache` is to be practical and usable. Therefore, primarily, `LP-Cache` performance evaluation focuses on two key aspects: 1). Implementation requirement analysis – to identify whether `LP-Cache`'s theoretical model can be practically implemented for performance evaluation and security analysis. and 2). Feasibility and usability analysis – to identify if this new approach feasible to be deployed. We describe `LP-Cache`'s implementation requirements in this section; whereas, its feasibility and usability analysis in the next section. `LP-Cache` orchestrates a mobile platform based location protection service to modify the location resource handling process. We give implementation possibilities on both iOS and Android platforms; however, it can also be implemented in similar ways on other permission-based mobile platforms. Compared to other mobile platforms, for instrumenting the `LP-Cache` implementation, `Android` is considered to be the best choice since it is open source and provides extra flexibility for testing, and it has the highest user base.

3.4.1 Bootstrapping

`LP-Cache` aims to protect the user's private places. Initially, `LP-Cache` does not have enough information to function since it is missing the two main required pieces of information, which are network fingerprints and geo-coordinates for the user's private places. `LP-Cache` cannot collect network fingerprints and geo-coordinates for private places at runtime, as by the time it has this information, other installed apps will also have access to it. Therefore, when `LP-Cache` first boots and before turning 'ON' location sharing settings, the user will have to do the initial setup, which includes enabling WiFi AP scanning, inputing geo-coordinates and setting privacy choices (see Section 3.4.4). In 2013, `Google` presented a new service API (also works on older `Android` versions) for location-based apps that allows developers to use the new and advanced *Location and Activity API*, i.e., they changed `Location Manager` to `Fused Location Manager`, hence combining sensors, GPS, Wi-Fi, and cellular data into a single API for location-based applications (Hellman, 2013). As a result, separating data from `GPS_PROVIDER` and `NETWORK_PROVIDER` is no longer straightforward. `LP-Cache` addresses this issue by preventing the app's location request to reach the `Fused Location Manager` that collects and sends the network session data to the location provider. Instead, the requested location is retrieved from the on-device cache, and then, it is sent to the requesting app (with privacy rules applied). Besides, geographic tool[5] can be incorporated in the `LP-Cache`'s UI to get the corresponding geo-coordinates. This will allow `LP-Cache` to achieve effective privacy without affecting location accuracy, at the same time, preventing the non-authorised sharing of device's exact location and network session data.

3.4.2 Mobile Platform

Considering existing mobile platforms, for performance evaluation, there are two possible methods of implementing `LP-Cache` location protection service: 1. app code modification and 2. platform modification. The first method requires modifying the app's location accessing interfaces and intercepting location updates before they reach the app provider. Whereas, the second method requires modifying the platform and changing the location data before reaching the app. Below we describe how both of these methods support the implementation of `LP-Cache` on a mobile platform.

1. **App Code Modification.** This comprises unpacking the app, rewriting the code to work according to the new rules, and then repackaging it again, e.g., Jeon et al. (2012). However, app repackaging changes the signature and stops future updates, and therefore, affects its functionality. Another way to modify

[5]LatLong is a geographic tool. See http://www.latlong.net/ (last access in March. 2016)

an app's location accessing interfaces is through the creation of an `Android or iOS` services and allowing apps to register with it. Both `Android and iOS` provide various APIs to communicate with the OS and allow apps to register a URL (Uniform Resource Locator) handler that allows them to communicate with each other using parameters. Then, apps can use the location data provided by this new service. This approach is easy to implement but relies heavily on app developers to modify their app's code, which is highly infeasible and unrealistic to be deployed in real-world scenarios. Nonetheless, this approach can be used as a simulated testing environment for any developed service of LP-Cache. For the on-device location computation and privacy analysis of `LP-Cache`, app code modification method will suffice. However, the developed service can then be added as part of the platform's system services. This requires system level permissions that can only be gained via rooting processes in `Android` or jail-breaking in `iOS`.

2. **Platform Modification.** This comprises customising the mobile platforms, i.e., gaining system level access to add new system level services or applications. `LP-Cache` requires the ability to intercept calls to official platform APIs that apps use to access private information such as user location. For the sake of experimentation, it is ideal to develop `LP-Cache` via platform modification without affecting apps functionality. However, platform modification opens other security and privacy challenges that are excluded from this research. Even after gaining system level access, compared to iOS, `Android` platform gives flexibility for inter-process communication and testing environments.

Here, we describe how `LP-Cache` can be deployed on both types of mobile platforms: a. `iOS` - closed platform and b. `Android` - open platform.

a. `iOS` - Closed Platform Focusing on `iOS`'s security model and privacy features. iOS uses code-signing by trusted third party, encryption and sandboxing to secure the platform. TTP based code-signing practice ensures only executables reviewed and distributed by Apple are allowed to run; however, this makes `iOS` platform extremely restrictive to implement `LP-Cache` without platform modification, i.e., jailbreaking. However, `iOS` provides various APIs to communicate with the OS and allow apps to register a URL handler that allows them to communicate with each other using parameters. Apps can access shared resources that provide users' private and sensitive data such as location with the help of well-defined `iOS` APIs. We can use such shared resources options to `LP-Cache`'s implementation on `iOS`.

b. `Android` - Open Platform One of the main tasks is to add a system service, where the class belongs to the location APIs; thus, it is placed in the `android.location package`, which detects private locations via wireless APs and can also be used by other components when calling context. In `Android`, a context allows an app to interact with the OS resources. Another task is to make `LP-Cache` communicate with location requesting apps. On `Android` there are two methods to access a user's location: 1) `Location Manager` Service (old), and 2) `Fused Location Manager` Service (new) that is a part of `Google Play Services`. Both methods require the app to request a callback function to get regular updates by registering a location listener. The app receives a new location object when a new location is available, the callback function is invoked. Modifying these two `Google` services is complicated, but there is a possibility to intercept the location object before it reaches requesting apps. We add a static context field to the location class, which is populated when the app is invoked; this enables us to know which app is currently requesting the location object, and also to communicate with the OS (Fawaz and Shin, 2014). The created location object

Algorithm 1 On-device Location Calculation Algorithm

```
Input: $n_x$: Network Frames
Output: $l_r$: Representative Location
 1: $n_x = 0$
 2: read $n_x$
 3: while $n_x \neq$ null do
 4:     if $n_x = n_i$ , $\forall\ i \in p$ then
 5:         (step 1) retrieve the corresponding $l_r$
 6:         add flag $f =$ (if private 1, else 0)
 7:         send $l_r$
 8:     else
 9:         (step 2) request $l_r$ from user or location provider
10:         set received $l_r$ to corresponding $p_i$
11:         update $c$
12:         send $l_r$
13:     end if
14: end while
```

has reference to the requesting app's context, and therefore, it can interact with our external service.

3.4.3 On-device Cache Database Creation

`LP-Cache` requires fixed wireless APs data (i.e., 802.11) to create the cached database of private locations - this process is called network fingerprinting. Initially, we decided to focus on WPS since it infers accurate user location. However, we can later include other fixed radio sources (e.g., cell-tower unique identifiers).

Network Fingerprinting. We can distinguish two main types of WPS techniques to determine the position of client devices in respect to APs (Bell et al., 2010): 1) Signal trilateration and 2) Fingerprinting. The former undertakes trilateration of Received Signal Strength (RSS), Angle of Arrival (AoA), and Time of Flight (ToF) from observed APs, and the later involves mapping observed APs signatures with a stored database. `LP-Cache` uses fingerprinting to create cached location database;

however, fingerprinting performance is highly related to the number of APs. Therefore, in Section 3.5 we have evaluated WiFi AP availability and consistency. The detected network management frames/beacons are mapped with the device's representative geolocation to create a network fingerprint, which is then stored in the local cached database, an example private location fingerprint is shown in Equation 3.1. Moreover, to reduce storage and computation overhead, our model only caches network fingerprints of private places (e.g., home, work, frequently visited places or particular stores), and it relies on user input for initial pre-setup. The user will have to select the option (via `LP-Cache` UI) to set current location as private place p_i, and then the fingerprint will be recorded. Later, the private place will be detected automatically with respect to observed beacons.

$$p_i = [n1], [n2], ..., [nx] \rightarrow [l_r] \tag{3.1}$$

where p_i represents ith private place IDs, $n_{1,2,...x}$ are the scanned beacons, and l_r is a representative location for that private place. WiFi AP beacon frame n consists of four attributes ⟨SSID, BSSID/MAC address, Signal-strength, and Timestamp⟩. The private representative location l_r consists of a tuple ⟨*Lattitude*, *Longitude*, and *Accuracy*⟩.

In `LP-Cache`, to set up network fingerprints at every private place, we measure the response rate as the ratio of detection count and the total number of scans for each beacon:

$$R_{l_c,x} = \frac{\sum_{i=1}^{n_{l_c}} b_{x,i}}{n_{l_c}}, b_{x,i} = \begin{cases} 1 & \text{if beacon } x \text{ found in } i\text{th scan} \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

Where $R_{l_c,x}$ is the response rate of beacon x at the current private location l_c and, n_{l_c} is the total scan count since the private place was entered. The detection count of each beacon is maintained to identify the frequently occurring beacons; and therefore, beacons with higher response rate are used to create the network fingerprint for the

Algorithm 2 Enhanced Personalised Permissions Algorithm

```
Input: $l_r$: Representative Location
Output: $l'_r$: Processed Location
 1: $u_p$ = User Input
 2: read $l$, $l_g$, $f$, $u_p$
 3: if $u_p$ = Adjust Granularity then
 4:     check granularity level $g_l$
 5:     truncate($l$, $l_g$)
 6:     replace $l$ to $l'$ and $l_g$ to $l'_g$
 7:     return $l'_r$
 8: else if $u_p$ = Obfuscate then
 9:     randomly generate angle $\theta$
10:     obfuscate($l$, $l_g$, $\theta$)
11:     replace $l$ to $l'$ and $l_g$ to $l'_g$
12:     return $l'_r$
13: else
14:     unchanged
15:     return $l'_r$
16: end if
```

current private place l_c. $R_{l_c,x}$ will be maintained in the `LP-Cache` database to update the response rate of every detected beacon during a specified time interval spent at private place l_c.z

On-device Cache-based Location Calculation Algorithm. The detailed steps of the privacy-aware geolocation calculation process are summarised in *Algorithm* 1. The surrounded beacons n_x are scanned and compared to the list of private WiFi fingerprints n_i to detect private place p stored in cached database c. Further, the representative l_r is altered based on set permissions (see Section 3.4.4).

3.4.4 Personalised Permissions for Location Sharing

A general LBS query consists of different attributes, e.g., LBS query {*PoI, Latitude and Longitude, User-Info*}, where included geo-coordinates estimate the device's geolocation. To satisfy one of the privacy properties called controlled information disclosure, we designed an enhanced permission mechanism to control these geo-coordinates before it is sent to app providers. When using `LP-Cache`, for every installed app and set private place, the UI provides three distinct privacy settings: (1) Adjust Location Granularity, (2) Obfuscate Location and, (3) No Change. In the first option, the geo-coordinate truncation method adjusts location precision level; in the second option, geo-coordinate transformation obfuscate the user's location; whereas, in the third option, the exact unchanged geo-coordinates are sent to the requesting app.

Enhanced Permissions Algorithm. Once `LP-Cache` receives an invoked location object l_r, it alters the location data according to the enhanced permission settings and returns processed location l'_r. The steps involved in enhanced permission mechanism are summarised in *Algorithm* 2, where u_p is the set permission, g_l is the adjusted location precision level, l is the latitude, and l_g is the longitude.

Geo-coordinates Truncation. The geographical coordinates are represented as latitude - "52°28'59.200" N and longitude - 1°53'37.0001" W, where the last digits specify more accurate geolocation. Geo-coordinate truncation method will enable us to adjust the location precision level, i.e., by removing last digits and rounding the location accuracy from street to city level or even more generalised. Generally, for any third party reuse, service providers or data collectors assure via the EULA that this method will be applied on the collected data since the truncated coordinates increase the ambiguity level (Aad and Niemi, 2010). On contrary, `LP-Cache` applies this method on the user device with the user's permission in order to minimise the user's sensitive data collection and privacy concerns.

Geo-coordinates Transformation. For privacy preservation, position transformation functions such as scaling, rotation and translation have been used by location data

Table 3.1 WiFi measurement dataset comparative summary.

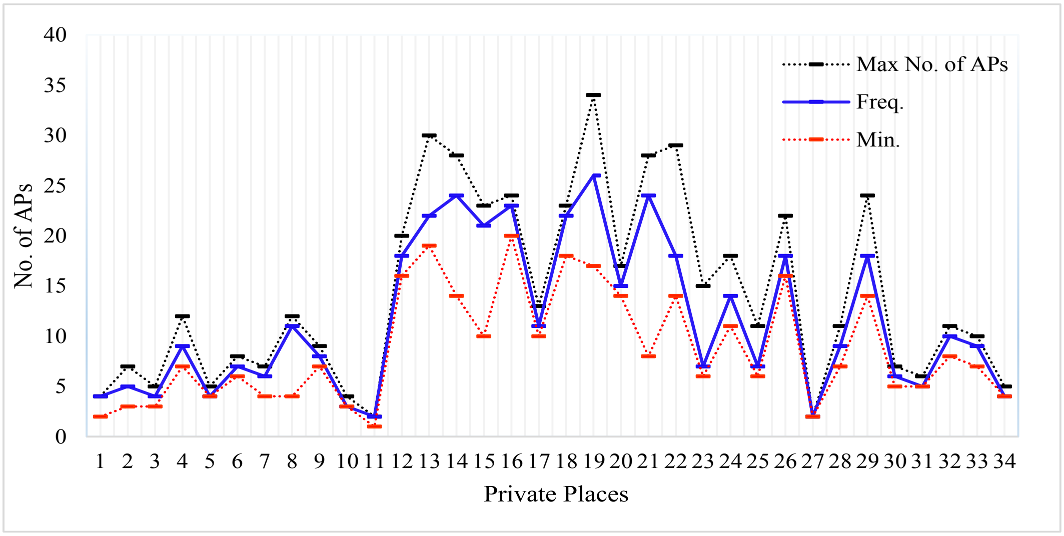
	1 Month	2 Month	Total scans.
Total number of scans	25480	21140	46620
Distinct private locations selected	34		% Change
Total APs detected	486	497	2.26%
Average APs detected	396	393	- 0.76 %

distributors or anonymisers (Lin et al., 2008; Wernke et al., 2014). In `LP-Cache`, we use geo-coordinate transformation on the device to obfuscate the user's private locations. Our service represents the geo-coordinate transformation using scaling and rotation, and denotes its parameters as a tuple $\langle s,\, \theta,\, (l,\, l_g)\rangle$, where s is the scaling factor, θ is the rotation angle, and $(l,\, l_g)$ are the original coordinates. The scaling factor range is decided based on the identified geographic poles (i.e., North, South, East and West). It applies Equation 3.3 to generate transformed or obfuscated geo-coordinates $(l',\, l'_g)$, where angle θ is randomly generated.

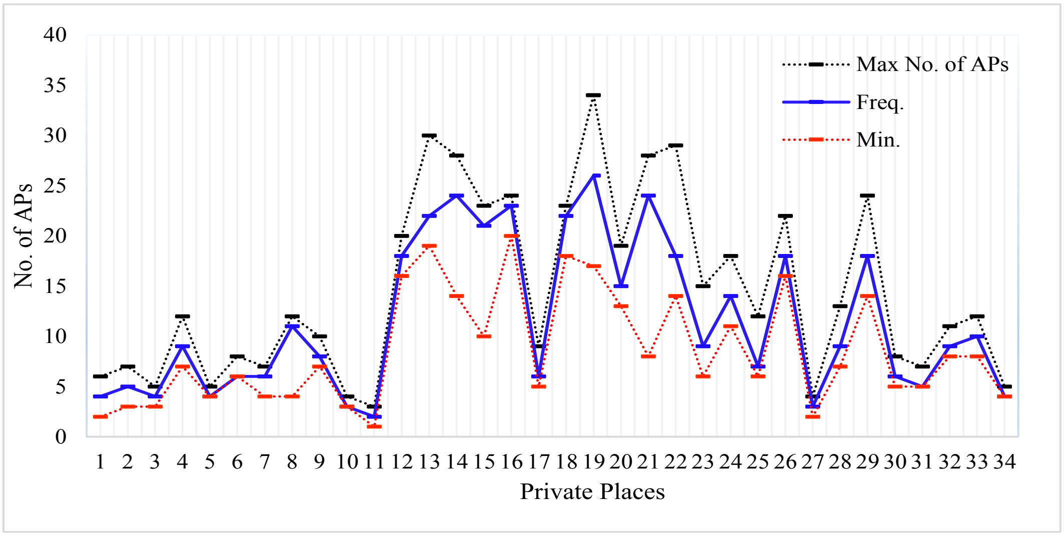
$$\begin{aligned} l' &= \theta(s.l) \\ l'_g &= \theta(s.l_g) \end{aligned} \tag{3.3}$$

3.5 Feasibility Analysis of `LP-Cache`

`LP-Cache`'s actual performance evaluation depends on the performance of locatxion-based apps. However, in this section, we present feasibility study of `LP-Cache` to assess the practicality of our theoretical proposal that depends on two main techniques: WiFi fingerprinting and caching. Hence, in the sections below, we analyse the feasibility of the WiFi fingerprinting method, estimation of cache storage requirements. And also present estimation of cache hits and cache misses to analyse the feasibility of cache update frequency and cache result accuracy.



(a) Measured density of detected WiFi APs at private places for a period of 1 month duration.



(b) Measured density of detected WiFi APs at private places for 2 months duration.

Figure 3.8 The comparative difference 1 month vs 2 months of detected WiFi APs density.

3.5.1 WiFi APs Availability and Consistency

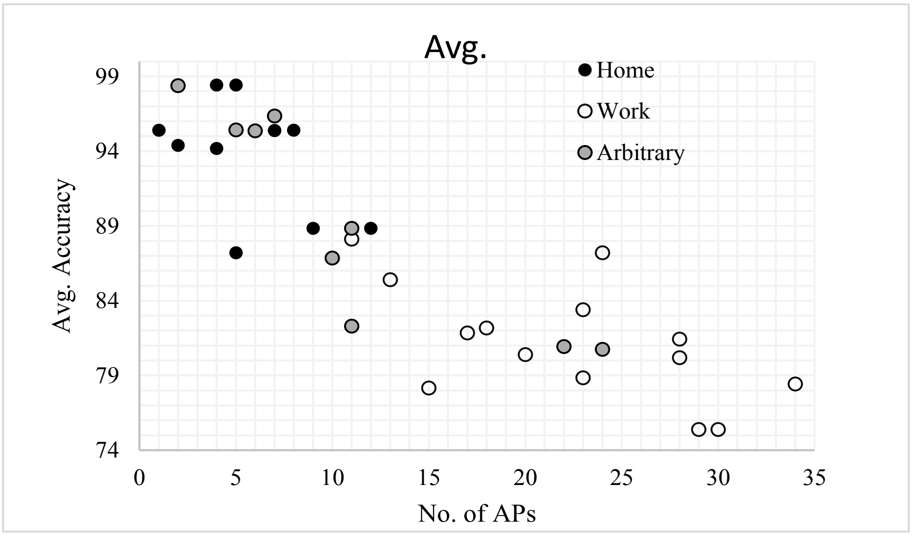
Here, we present the WiFi AP data availability and consistency to measure the feasibility of WiFi fingerprinting method to be included in `LP-Cache`'s implementation. We considered the sample size of 2 months for the WiFi APs dataset. We conduct a comparative study of the observed results from both 1st and 2nd month datasets to evaluate the scalability of the WiFi fingerprinting method. For the sake of comparision, we have maintained unique ID and sequence for all the selected 34 private places.

Experimental set-up. The experimental set-up to measure WiFi AP data availability and consistency consists of three steps.

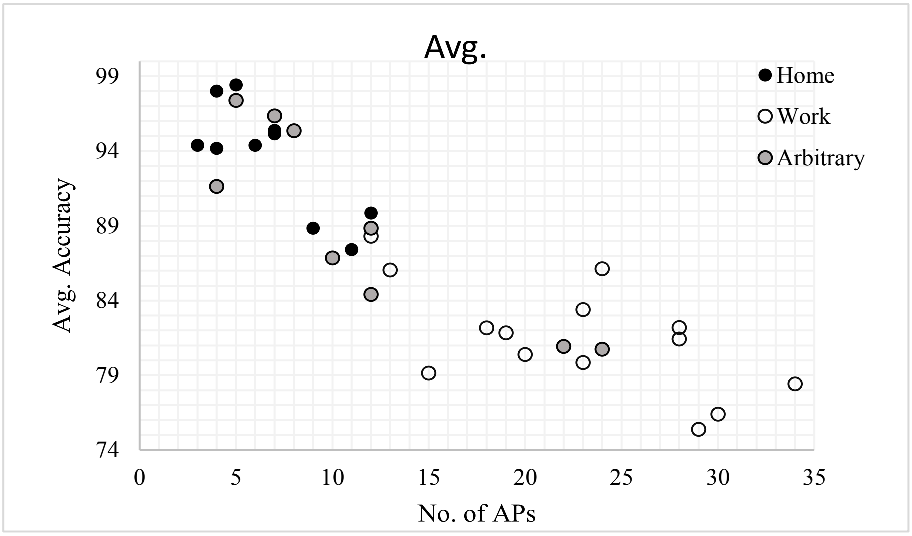
1. *Data collection.* We collected beacons from fixed WiFi APs using WiEye (2016) and NetworkInfoIi (2016) apps on `Android` smartphones that have 802.11a/b/g/n radio feature so they can operate in both 2.4GHz and 5GHz bands at 34 different private places for a period of two months.

2. *Location categorisation.* App users are more concerned about sharing their private locations (Almuhimedi et al., 2015); therefore, in our analysis, we selected three distinct categories of private places: 1. Home (i.e., residential place), 2. Work (i.e., commercial place), and 3. Arbitrary (i.e., any frequently visited place) to determine the categorical distribution pattern of WiFi APs.

3. *Data analysis.* We collected and statistically analysed the scanned WiFi AP data. Table 3.1 compiles the included sample size and the measured percentage changes; whereas, Figure 3.8 shows the relative difference between WiFi APs density, and Figure 3.9 depicts the relative average accuracy distribution pattern of detected WiFi APs for each category over the 2 months period.



(a) Average accuracy distribution pattern of detected WiFi APs at private places for 1 month.



(b) Average accuracy distribution pattern of detected WiFi APs at private places for 3 month.

Figure 3.9 The comparative 1 month vs 2 months average accuracy distribution patterns of detected WiFi APs at private places.

Observation 1. For each category of private places, experiments revealed the following:

Home The results demonstrate that Wifi APs are fixed and frequent and the difference between the number of constant beacons and minimum number of similar beacons is comparatively less, and therefore, it achieved the highest accuracy rate. Moreover, the ratio of SSID to BSSID is 1:1, i.e., one SSID (abc) has one BSSID (a0:12:b3:c4:56:78), this makes fingerprints distinct so improving the location detection performance.

Work This category has many fixed WiFi APs but with fluctuating signal strengths, and therefore, the sequence of available APs changes. However, the observed ratio of SSID to BSSID is many to one, i.e., one SSID has many BSSIDs; therefore, in this case, SSIDs along with BSSIDs can be used as unique identifiers to create a fingerprint to detect a private place dynamically.

Arbitrary In this category, the data collector could select any frequently visited locations, e.g., gym, shop, or friend's home. Figure 3.9 demonstrates that the outcome of this category is related to the other two categories, it either shows results similar to home or work.

The range of average accuracy for all the three categories of private places falls between 75% to 97%. Hence, it can be seen that smartphones regularly detect similar beacons at frequently visited places, for place detection at least one beacon should match with the stored fingerprints. Thus, the results demonstrates that WiFi fingerprinting can be effectively used as private place detection source in `LP-Cache`. Nonetheless, to achieve efficient capability for place recognition via beacons a *place discovering algorithm* like Kim et al. (2009) can be implemented (we suggest this as a future extension of the model).

Observation 2. Table 3.1 shows comparative analysis of WiFi APs data that has been scanned and collected during both 1st and 2nd month. Considering percentage changes, the number of total detected APs have increased by 2.26% and the number of frequently detected APs have remained almost similar, i.e, with a negligible difference of -0.76%. Equation 3.2 has been used statistically to identify frequently detected beacons while at a particular private place. Pre-set unique IDs and a sequence for all the selected 34 private places allowed us to measure the relative density and distribution pattern of the WiFi APs during both 1st and 2nd month. Figure 3.8 shows the relative difference between WiFi APs density, and Figure 3.9 depicts the relative average accuracy distribution pattern of detected WiFi APs for each category over the period of 2 months.

3.5.2 Estimated Storage Requirements

Location-based queries received from running applications or service providers includes several attributes and their data types require huge amount of storage. The cache-based systems (Amini et al., 2011; Zhu et al., 2013) apply caching techniques on location-based queries that result in vast amount of storage requirement. Whereas, `LP-Cache` does not store any location-based query attributes or contents received from running applications or service providers, instead it stores WiFi APs data and geo-coordinates of users' private locations. Moreover, the user's pre-set privacy rules are applied to the mapped geo-coordinates at runtime. As a result, comparatively, we claim `LP-Cache`'s on-device cache database does not have massive storage estimated requirements. Considering the 802.1 standard and datatypes sizes, Table 3.2 presents the field storage requirements in bytes and database components, where the network fingerprint table is a tuple of $\langle no.of beacons$, $beaconfield$, $counter\rangle$, and permission table is a tuple of $\langle location$, $placeid$, $accuracy\ counter$ (a), $no.of private places\rangle$. Moreover, Table 3.3 presents the measured changes in the 1st and the 2nd month of WiFi data collected at 34 distinct private places. The results indicate that the average change has increased by 2.26%. The cache storage of a total of 25.844 KB is needed that includes the sum of permissions and network fingerprints for 34 distinct private places. Therefore, it is anticipated that

Table 3.2 Estimated storage requirement

Field storage		**Size**	**Database Component**	**Size**
Beacon Field	BSSID/MAC SSID Place ID Timestamp/Age	6 bytes 32 bytes 3 bytes 8 bytes	Network Fingerprint Table	$= (beaconfield + counter) \times no.ofbeacons$
Location Field	Geo-coordinates Region	32 bytes 32 bytes	Location Permission Rule Table	$= (location + placeid + accu.\ counter) \times no.ofplaces$

Table 3.3 Comparative difference of monthly storage

Storage	**1 Month**	**2 Month**	**% Change**
Network Fingerprint	25272 bytes	25844 bytes	2.26% increase
Permissions	2380 bytes	2380 bytes	No change
Total Storage	27652 bytes	28224 bytes	2.07% total increase

the current mobile device internal storage capacity is sufficient for the required storage (Android, 2016).

3.5.3 Cache Hits and Cache Misses

For `LP-Cache`, accuracy and up-to-date database results are the main challenges, the three possible outcomes are given below:

1. *The location is cached and up-to-date* case comes with the positive result, and therefore, data can be used effectively.

2. *The location is cached but is out-of-date* case can occur if the network infrastructure changes, e.g., if a router is changed then the cache data needs to be updated. The overall results of Section 3.5.1 and Section 3.5.2 suggest that this case does not occur frequently. Nonetheless, for data accuracy a method that uses Equation 3.2 will be incorporated to detect and measure the occurrence of

such situations of cache misses at runtime. Moreover, the developed method can likewise be deployed to maintain *data freshness* and *data consistency*.

3. *The location is not cached* case occurs when the observed WiFi AP is not cached and/or mapped to the private locations, then our service will have to interact with the user to update the location cache. Besides, the response rate $R_{l_c,x}$ can be further extended to measure runtime occurrences of these outcomes.

3.5.4 Discussion

In this section, we discuss the potential benefits of `LP-Cache` and its implementation challenges.

Benefits `LP-Cache` lets users preserve their location privacy while at the same time using location-based apps effectively. It reduces the burden of location providers and app providers as the user can be in control of location sharing at his private locations. It, therefore, guarantees private location data minimisation in the ecosystem. Besides privacy preservation, it will also benefit the user who receives incorrect LBS query results because sometimes the location provider miscalculates the client's geolocation. Hence, we state `LP-Cache` when integrated in mobile platforms (i.e., `Android`, `iOS` or others) will benefit both end users and service providers.

Implementation Challenges The presented outline of `LP-Cache` implementation requires WiFi settings to be activated, but it does not mean that the user can only access the internet using WiFi connection. `LP-Cache` involves WiFi APs to detect the places, but the LBS query can be sent to the service provider by any means of communication. In the current proposal, an initial location request for a new private place is sent to the location provider, this might create infrequent information leakage; therefore, to overcome this challenge we are aiming to add a user-friendly option to enable users' to provide their geo-coordinates to `LP-Cache` instead of sending the request to the location provider. This will completely remove involvement of the location provider in our model, which aims at protecting private locations. Frequent WiFi scanning might affect the device's battery performance, but the continuous advancement in smartphones' hardware performance can prove `LP-Cache` to be more adaptable in the near future. However, even location providers' location services consume battery speedily; comparatively, WiFi scanning service needs less battery consumption. `LP-Cache` intercepts the app's control flow that can increase the computational overhead and cause a slight delay in getting the LBS query reply; however, our field study results state that users are comfortable to accommodate reduced functionality of their apps at private places, such as home and work.

3.6 Summary

In this chapter, we have presented a detailed analysis of the current location computation process and proposed a novel privacy-preserving model. Both the end users and service providers benefit from `LP-Cache` since the on-device caching technique works on the minimisation of the user's private location collection process. With the personalised permission mechanism, users can manage each app and private place distinctly and the on-device location calculation will reduce interactions with involved service providers. We have fully elaborated on `LP-Cache`'s design decisions and possible implementation. We have also analysed the feasibility and usability of `LP-Cache` to benchmark the WiFi APs availability and consistency. Following this, `LP-Cache`'s performance evaluation will be extended to analyse how frequently the cache needs to be updated and what the trade-offs are of the cache update frequency vs location privacy and accuracy in order to benchmark computational and communication overheads. For this, in the next chapter, we demonstrate the implementation of `LP-Cache` in the form of a middleware (`PL-Protector`) on `Android` platform. To measure the feasibility, usability and efficiency of our approach while interacting with different location-based apps, we plan to consider the fundamental caching based technical challenges – such as cache hits and cache misses, data freshness, data consistency, and estimated memory requirements – in the development and implementation of `PL-Protector` paying special attention to storage-efficient caching.

Chapter 4

PL-Protector: Implementation of LP-Cache as Middleware

Research findings from this chapter have been published in a conference paper [a] *and a journal article* [b]

[a] Patel, A.,& Palomar, E., "A middleware enforcing location privacy in mobile platforms". *In: 14th International Conference on Trust and Privacy in Digital Business (TrustBus)*, pp. *32–45*, Springer 2017.

[b] Patel, A., & Palomar, E. "A Practical Cache-based Location Privacy-enhanced Middleware for Mobile Users", Submitted to the *Journal of Network and Computer Applications*, Elsevier, Sep. 2017 (Under review).

4.1 Introduction

So far, findings from Chapter 2 and Chapter 3 indicate that privacy controls provided by both open and closed mobile OSs are inadequate. Instead, we need a solution that forces apps to make their location data use patterns explicit and enforces the declared privacy rules in the information flow, while preventing all other sensitive data flows within the underlying mobile platform. Hence, in this thesis, we develop and promote a new design principle and privacy policy recommendation that forces location privacy in the existing mobile app ecosystem. This chapter presents a middleware called *Private Location Protector* (`PL-Protector`), which implements the `LP-Cache` model introduced in Chapter 3, highlighting how we support our design goals mentioned in 3.3. In short, we implemented PL-Protector as a practical and functional proof-of-concept to test and validate our theoretical model. The design of PL-Protector implements the two location privacy-preserving algorithms, which are defined in the `LP-Cache` model, as LPPMs that includes: LPPM-1) On-device Location Computation Mechanism, and LPPM-2) Personalised Location Permissions Mechanism. Both of these LPPMs enable robust and efficient source (OS) to sink (3rd party app provider) flow control while sharing the user's sensitive locations. For instrumenting `PL-Protector`, we selected `Android` platform since it is open source. Nonetheless, our results can be extrapolated to other permission-based mobile platforms such as `iOS`. `PL-Protector` uses fingerprinting to create a private location database within the device instead of storing it on remote location servers. This minimises the process of wireless AP data collection by the WiFi content distributors or location providers. In addition, `PL-Protector` controls information disclosure within the generated LBS query (e.g., PoIs and nearest neighbour) since it will be sent to third-party app providers. We ran `PL-Protector` on a `Nexus 6P` with `Android 6.0` that acts as the platform's location privacy knob to minimise the risk of major location privacy threats. PL-Protector only requires process isolation and IPC (Inter-Process Communication) services from the underlying OS; thus, minimising the requirements placed on the hardware/OS.

The rest of the chapter is organized as follows. Section 4.2 overviews `PL-Protector`'s system model that includes system roles, threat, mobility, app usage and privacy model. Section 4.3 fully elaborates on the design decisions, architecture, and implementation of the middleware. Finally, Section 4.4 summarises this chapter.

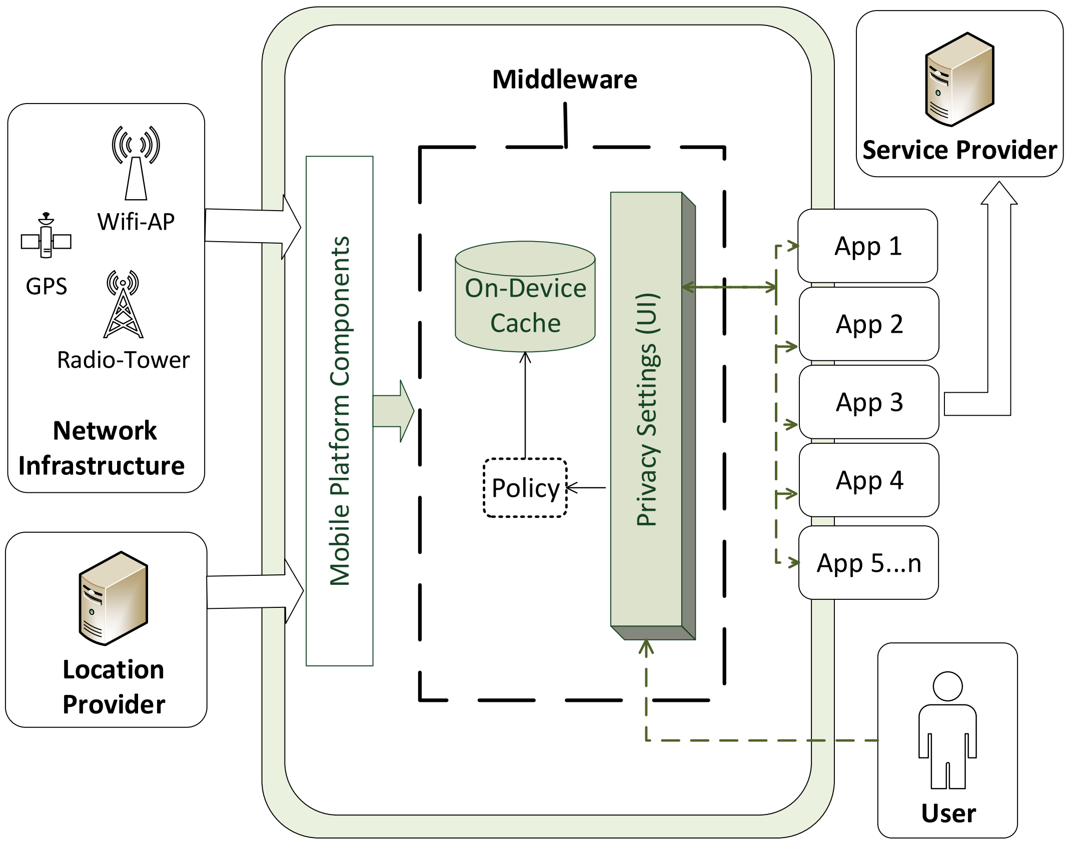


Figure 4.1 System model representing involved entities and their roles

4.2 System Model

We characterize `PL-Protector`'s system model considering system roles, the threat model and evaluation metrics for both app usage and privacy protection.

4.2.1 System Roles

`PL-Protector` modifies the current location resource handling process in mobile systems; however, the involved entities and their roles remain the same. `PL-Protector` follows the sequence of processes and messages as described in Chapter 3.3. Here we clarify the role of each entity by describing the overall communication procedure in presence of `PL-Protector` as follows (Figure 4.1).

1. *Middleware* determines two main components, namely, privacy settings and on-device cache to overcome the shortcomings related to user privacy within the existing mobile systems. It then functions according to the pre-set privacy rules that secure the calculation and transmission of sensitive location data.

2. *User* needs to set privacy choices using settings (via a UI) by marking sensitive locations as private, selecting the preferred level of privacy for every app and location distinctly. The user also has an option to manually map the location (i.e, geo-coordinate) with the observed network signature that will result in no involvement of the *location provider*.

3. *Service providers'* operations are unaffected, i.e., they can follow their standard process. However, *service providers* benefit from secure gathering and transfer of location data using mobile devices while at the same time preserving users' privacy since the *middleware* minimises sensitive location data flows and privacy concerns without affecting their app's operations or Quality of Service (QoS).

4. *Mobile platform*'s location access controls respond inadequately to major privacy threats (Almuhimedi et al., 2015; Fawaz et al., 2015); however, the *middleware* complements existing controls by better regulating the way apps and ad libraries access private location data at run-time.

5. *Network infrastructure* availability and consistency are extremely important to the performance of the included fingerprinting method, which requires fixed wireless APs[1] data to create the on-device cache of private locations.

6. In the case of an unmatched entry on the cached locations, only if the user has set the current location as private and do not want to input geo-coordinates

[1]Initially, we decided to focus on WiFi APs since they infer accurate user location. However, we can later include other fixed radio sources (e.g., Cell tower unique identifiers).

manually (via UI), the *location provider* is required to calculate and send the exact geo-coordinates (via location object, see Listing 3.3) for the first time.

4.2.2 Threat Model

We consider two different attack scenarios to `PL-Protector` mainly caused by the level of access to user location in and out of the device.

OS's Middleware Layer Threats. A series of attacks that operate at Android's middleware layer (Bugiel et al., 2013). In `Android`, the location resource and service availability can be abused or misused by both over-privileged and malicious 3rd party apps and libraries. The former can threaten user privacy by gaining unauthorized access to location, and other user sensitive information, that is not required for their operation. The underlying purpose lies in general in feeding advertisement libraries and, ultimately, exploiting the permissions of the host app (Backes et al., 2016). The later can leverage unauthorized location permissions for financial gains and leak users' mobility and behaviour information (e.g., unauthorized profiling). Moreover, the user's sensitive data can also be leaked due to inter-process unauthorised communication vulnerabilities that can be exploited as private/security attacks on the mobile OS, e.g., *Service* or *Activity* hijacking, *Broadcast Intent* theft on `Android`.

Privacy Threats. User tracking, identification and profiling (i.e. personal habits, movement patterns, etc.) are fundamental threats to location privacy (Wernke et al., 2014). Without PL-Protector, there is a continuous flow of LBS queries between user devices and location providers that include device's exact geo-coordinates and other sensitive information. This can leverage malicious misuse of location data, especially in the presence of a malicious location provider and via advanced network sniffing practices.

`PL-Protector` computes the exact location within the user device, without the service provider's involvement, whilst trusting the device on the storage of sensitive data. However, the user has still the option of giving consent for app providers and/or location providers to access location data. Mobile network providers might, however, collect user location data via cellular clients. We also exclude from our work, the option of manually inserting the location data (e.g., street name or postcode) within the LBS query.

4.2.3 Location-based Application Categories

Here we elaborate on the functionality of different mobile applications that can communicate with `PL-Protector` once installed on a mobile device. There are three categories of apps that can interact with `PL-Protector` to request user's location, as follows:

Category 1: Social Networking App. We have noticed that location-based apps from the social networking category have continuous access to user's location data. These apps continuously send location update requests to the OS. At any preset private location, `PL-Protector` operates in between the requesting app and the OS. This will require `PL-Protector` to able to reply to these location update requests in real-time without disrupting the app's operations. Listing 4.1 illustrates pseudo-code of an example social networking app interacting with `PL-Protector` that ensures whenever an app requests a location update at a private place, privacy policies are applied before sending the LBS query to service providers.

Category 2: Weather/Utilities App. We have noticed another category of location-based apps that sporadically send location update requests to the underlying OS. These apps are lightweight and, comparatively, do not involve frequent interactions (limited to once or twice a day) with `PL-Protector`. They follow the same interaction process as mentioned in Listing 4.1.

Category 3: Over privileged App. Apps from the over privileged category can continuously or sporadically send location update requests to the underlying OS. These apps are not location-based apps but have requested compulsory location access permissions from their users. `PL-Protector` restricts such apps from accessing user's private location by applying privacy rules on the location data. However, even in this case, the same standard interaction process (i.e., Listing 4.1) is followed by the app and `PL-Protector` but with different privacy rules being applied.

```
application  SocialApp
request  {LBS_Query.NearestNeighbour -> loc}
return flag(PrivLoc_recog(appContext, nBeacons));
if (flag = ON)
Location loc = extractCacheLoc(appContext, nBeacons);
void applyPolicy(loc);
Handle appSession = sessionHandler(appContext);
receive loc from \texttt{PL-Protector};
send request to service provider;
```

Listing 4.1 Pseudocode for a social networking app interacting with `PL-Protector` - get location data and send LBS query response.

4.2.4 Preliminaries

We now model the user mobility and app usage (specifically at private places) as a series of privacy evaluation metrics that will be used to validate `PL-Protector`'s working assumptions.

Mobility Model We formulate user's PoIs (e.g., Home and Work) as private places that the user frequently visit; hence, p_i represent ith private place identification, which is derived from a series of scanned beacons n_x and the representative location l_r for that private place, as shown in Eq. 4.1).

$$p_i = [n_1], [n_2], ..., [n_x] \rightarrow [l_r] \tag{4.1}$$

$$P_l = [p_i], [p_j], ..., [p_n] \tag{4.2}$$

At location p_i, the user can then visit a subset of private places $U_{p_i} \subseteq p_1, p_2, \cdots, p_x$ while running different LBS apps on the device. Hence, P_l is the total number of user's private locations (as given in Equation 4.2). PL-Protector relies on the user input to define the set of private places that are distinct for every user mobility profile. Moreover, to set up network fingerprints at p_i, we measure the response rate as the ratio of detection count and the total number of scans for each beacon as follows:

$$R_{n_c,x} = \frac{\sum_{i=1}^{n_c} b_{x,i}}{n_c}, b_{x,i} = \begin{cases} 1 & \text{if beacon } x \text{ found in } i\text{th scan} \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

where $R_{n_c,x}$ is the response rate of beacon x at p_i and, n_c is the total scan count since the private place was entered. The detection count of each beacon is maintained to identify the frequently occurring beacons. Beacons with higher response rates are used to create the network fingerprint for that p_i. $R_{n_c,x}$ will be maintained in the database of `PL-Protector` to update the response rate of every detected beacon during a specified time interval t spent at private place p_i.

App-Usage Model. We will apply privacy rules to the app sessions taking place at private places. We define "app session" as the duration of the app usage. In `Android`,

Table 4.1 The evaluation metrics for location privacy.

Metric	Description
Pl_{guess}	Number of correctly guessed private locations
Pl_{total}	Number of total collected private locations
Pl_s	Unique value to identify applied privacy settings
Pl_d	Distance between two points with longitude and latitude
P_{high}	Distance is $>$ 111.32km
P_{medium}	Distance is $>$ 11.132km
P_{low}	Distance is $>$ 1.1132km
LoP_{per}	Fraction of achieved privacy level on a per app basis
LoP_{total}	Fraction of overall achieved privacy level in total apps

according to the execution status, an app can run in three different states: foreground, background and perceptible. In general, apps get access to the user's location in the foreground. When the user exits an app, this is cached and moved to background state for faster execution. Persistent status is informed by notifications. Background state allows prolonged location access; therefore, tracking threats are more harmful here. In later Section 4.3 and Chapter 5, we specify how our proposal handles all these three apps running state to mitigate viable location privacy threats.

Privacy Model. Most commonly used metrics to quantify location privacy and evaluate the appropriateness of LPPMs are *k-anonymity* and *entropy* (Shokri et al., 2011). These metrics functions based on pre-defined PoIs and will not be efficient to measure the appropriateness of `PL-Protector` since we aim to achieve location privacy and data minimisation before the private location data is collected by involved service providers. Hence, we identified and applied another evaluation metric, based on well-known `Haversine` formula (Robusto, 1957), to quantify the location privacy and evaluate the appropriateness of `PL-Protector`'s LPPMs. Table 4.1 compiles the metrics to be used for evaluating the location privacy threats. We define the value of Pl_s as the identifier of applied privacy setting and measure the achieved privacy by analysing the collected dataset of the actual location traces at the user's private places. To evaluate location privacy, we use `Haversine` formula to quantify tracking and profiling threats as the distance Pl_d between two positions with longitude and latitude (ϕ, λ) and the radius r of the Earth:

$$Pl_d = 2r\sin^{-1}(\sqrt{\sin^2(\frac{\phi 1 - \phi 2)}{2} + \cos(\phi 1)\cos(\phi 2)\sin^2(\frac{\lambda 2 - \lambda 1}{2}))} \tag{4.4}$$

where the *haversine* function is given by $Hsin(\theta) = sin^2(\frac{\theta}{2})$, $\phi 1$ & $\phi 2$ are the original geo-coordinates, and $\lambda 1$ & $\lambda 2$ are the observed geo-coordinates. Secondly, the privacy rules (see more details in Section 4.3) pre-set by the user will, later, be used to measure achieved privacy using the distance scale $\langle P_{high}, P_{medium}, P_{low}\rangle$. Location privacy threats can be quantified as the probability of occurrence of an event exploiting the vulnerabilities. Therefore, we use the metric given by Freudiger et al. (2011) that allows us to measure the ability of apps to find private locations from the collected location traces and thus pose location privacy threats. Hence, LoP_{per} and LoP_{total} are calculated as:

$$LoP_{per} = \frac{Pl_{guess}}{Pl_{total}}, and \tag{4.5}$$

$$LoP_{total} = \sum LoP_{per} \tag{4.6}$$

4.3 PL-Protector on Android

In this section, we first justify our decision to implement `PL-Protector` as middleware on `Android` platform, we then describe its functionality characteristics, architecture and implementation in detail. Nonetheless, the middleware implementation and results can be extrapolated to other permission-based mobile platforms (e.g., `iOS`).

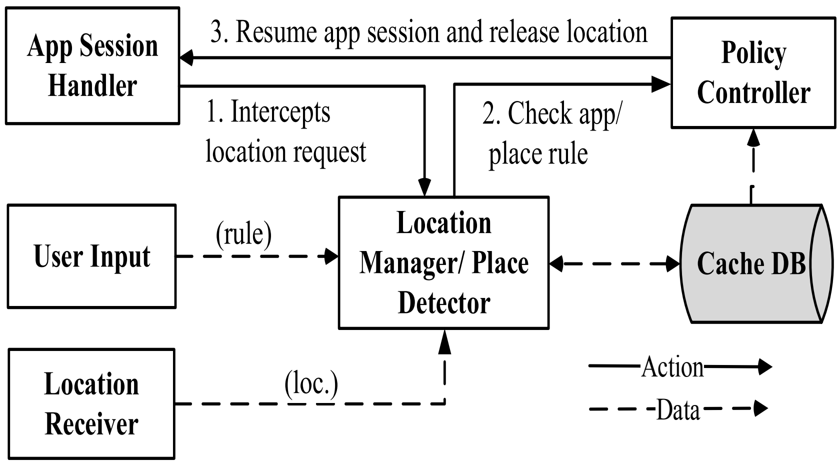


Figure 4.2 Control flow of `PL-Protector`

4.3.1 Architecture of PL-Protector

Based on our user preferences survey results and other existing studies (Almuhimedi et al., 2015; Fawaz and Shin, 2014), it is clear that mobile users are more concerned about sharing their private locations; hence, `PL-Protector` enforces privacy for such user's sensitive locations. `PL-Protector`'s three main design goals are: 1) the third-party app provider will not be able to infer the device's exact location without getting the uses's consent; 2) the user can set distinct privacy preferences for different apps and private places; and 3) `PL-Protector` works independently without the need of modifying the app's code. Figure 4.2 depicts the block diagram for `PL-Protector` architecture; its main components are: user input, app session handler, location manager, place detector, policy controller, cache DB and location receiver.

User Input is the UI (User Interface), which enables users to set and manage their private places and apply improved personalised permissions when running installed location-based apps. Once the user inputs are received, the marked private locations are sent to the *Location Manager* module and the pre-set permissions are sent to the

Policy Controller module.

App Session Handler is responsible for intercepting the event of location access calls and then lead the app's control flow to our middleware. When the flag is positive (i.e., user at private place), it monitors app launch and exit events that need to be intercepted. It first pauses the requesting app's execution, saves its state, and sends the location intent to the *Location Manager* component for rule checking (step 1 in Figure 4.2). Once the privacy rules are applied, the *App-Session Handler* will receive the anonymised/transformed location object from *Policy Controller.* It will then resume the requesting app's control flow to maintain every session (step 3 in Figure 4.2). In `Android`, the *middleware*'s background service frequently checks (every 10 seconds) the currently running foreground app using `getRunningAppProcesses` on older versions and `UsageStasManager` on `Android` 6 (or later). To use the `UsageStasManager` API the user of the device needs to grant permission through `Android`'s settings application. When the app is no longer running in the foreground, it blocks app's background access to location updates that leaves an app limited to foreground sessions. In `Android`, for maintaining a set of session operations, `PL-Protector` executes process isolation and IPC services.

Location Manager and Place Detector are the central components that receive both events (actions) and data from the different components as well as maintaining the *Cache DB* database. The *Location Manager* component receives privacy rules for specific private locations and network fingerprints via *User Input.* Whereas, the *Place Detector* component detects unique identifiers of the surrounding wireless APs and maintains a binary flag to detect private places. When the flag is ON, if the *Location Manager* receives location updates (i.e., Intents) from the *App Session*

Handler, the location data is retrieved from the *Cache DB* and sent to the *Policy Controller*. In case of an unmatched query on the *cache* and the user does not want to input geo-coordinates manually (via maps provided in UI), the location data is received by location providers from the *Location Receiver*. Later, the *Place Detector* monitors user's mobility pattern and updates the mobility profile of the user (see Mobility Model in 4.2.4). Moreover, if the location is one of the user's frequently visited places, the*Place Detector* checks with the user before releasing the location to apps. If the user is not willing to share or reveal, then the location is sent to the *Policy Controller* to be obfuscated or generalised; otherwise, the location is sent to apps as it is.

Policy Controller gathers the location object from the *Location Manager* inorder to apply the corresponding user permissions on the location coordinates, altering it if needed, and transferring the processed location to the *App Session Handler* component. The two privacy policies that the user can set per-app/place basis are the *Standard Policy* and *Per-location Policy* (see Figure 4.5), as follows:

1. The Standard Policy consists of three location settings as follow:

    (a) The *Behaviour Protection* setting implements the geo-coordinate obfuscation equation (defined in 3.4.4) to generate transformed/ obfuscated geo-coordinates (l', l'_g) for every app session. The behaviour protection level is defined by a scale (Low, Medium, and High) that determines randomness of the obfuscation equation's parameters $\langle s, \theta, (l, l_g)\rangle$, where s is the scaling factor, θ is the random rotation angle, and (l, l_g) are the original coordinates.

    (b) The *Location Protection* setting implements the geo-coordinate truncation equation (defined in 3.4.4) and follows a location granularity scale like (Low, Medium, and High) to adjust the location precision level for every app session.

(c) The *Block/Fixed Location* setting picks high behaviour and location protection level by default and determines a constant value of altered geo-coordinates for every app session.

2. The per-location policy allows the User to apply standard policy settings for each pre-marked private places that are displayed on the map.

Once processed geo-coordinates (l' l'_g) that comply with the pre-set privacy rule are generated, we measure the achieved level of a privacy LoP_{per} and LoP_{total} on per-session basis using values of both Pl_d and Pl_s (as defined in Section 4.2.4). Listing 4.2 contains pseudocode for calculating Haversine distance (Pl_d) between two positions with longitude and latitude (ϕ, λ) and the radius r of the Earth.

Listing 4.2 Pseudocode for calculating Haversine Distance (Pl_d) between two positions with longitude and latitude ($\phi = original, \lambda = observed$)

```
initialise static final int EARTH_RADIUS = 6371;
request double haversin(double val)
return  Math.pow(Math.sin(val / 2), 2);
request HavDistance(double originalLat, double originalLong,
        double observedLat, double observedLong)
double dLat  = Math.toRadians((observedLat - originalLat));
double dLong = Math.toRadians((observedLong - originalLong));
originalLat  = Math.toRadians(originalLat);
observedLat  = Math.toRadians(observedLat);
double a     = haversin(dLat) + Math.cos(originalLat)
               * Math.cos(observedLat) * haversin(dLong);
double c     = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
return EARTH_RADIUS * c;
```

Cache DB and Location Receiver *Cache DB* is the established on-device cached database, and it is routinely queried by the *Location Manager* module, which can add, update and delete the cached location data. The locations in *Cache DB* are those

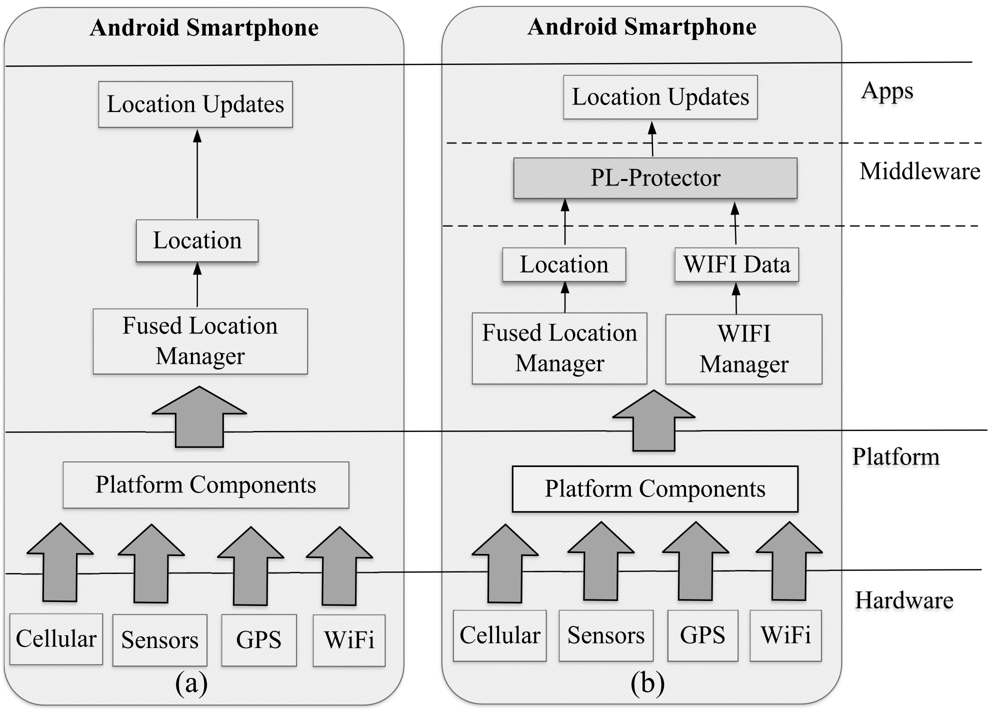


Figure 4.3 Location computation mechanism (a) without and (b) with `PL-Protector`

which are to be protected, and they can also represent regions of space. We used the SQLite (2017) database management system for *Cache DB* generation since it supports embedded databases and `Android` platform. *Cache DB* stores the network fingerprints in an `SQLite` table for each of the observed WiFi APs' signatures that are mapped to their representative geo-coordinates. Each `SQLite` table entry is recorded along with a network fingerprint and geolocation that are acquired either from UI and/or the *Location Receiver*. When the location update request is sent to the *Location Receiver* component, it receives the location object, which includes the user device's geo-coordinates (as seen previous chapter in Listing 3.3), from location providers and sends it over to the *Location Manager* for further processing.

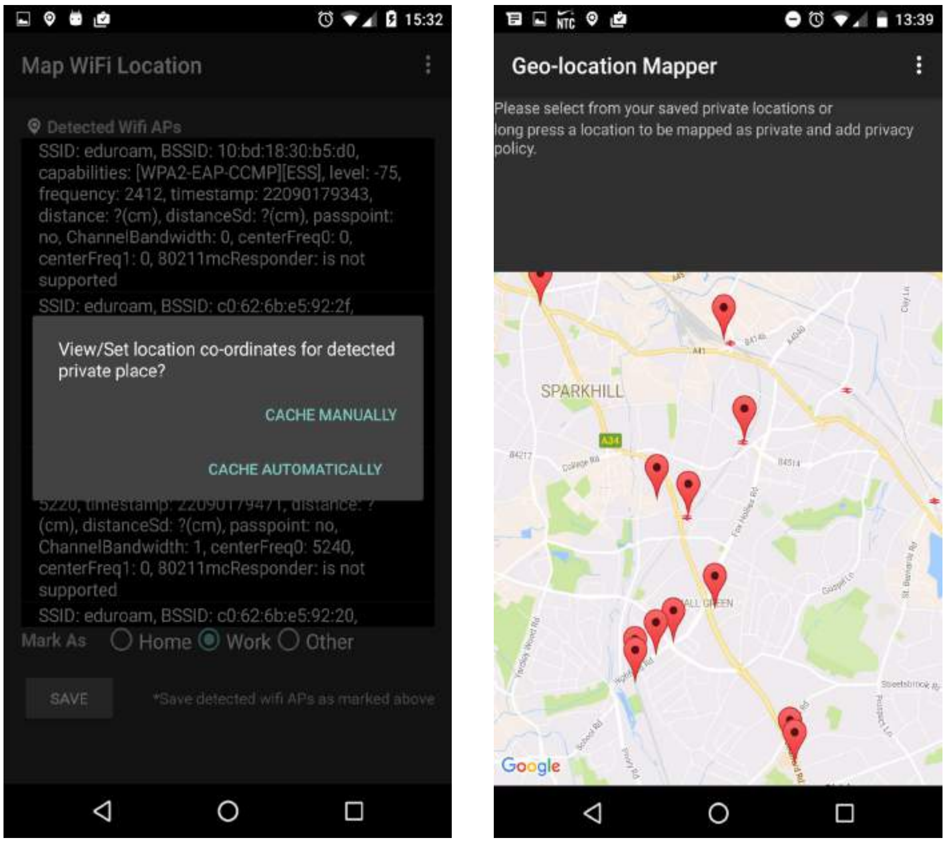
4.3.2 Middleware Implementation

`PL-Protector` orchestrates a mobile platform based privacy protection service on `Android` to modify the location resource handling process. `PL-Protector`'s communication operations only requires process isolation and IPC services; hence, minimising the requirements placed on hardware or OS modifications. As mentioned earlier in Chapter 3, in `Android`, there are two methods to access the user's location: `Location Manager` service, and `Fused Location Manager` service that is part of `Google Play Services`. Both methods require the app to request a callback function to get regular updates by registering a location listener. The app receives a new location object when a new location is available, the callback function is invoked (Figure 4.3 - a). Modifying these two `Google` services is complicated, but we make `PL-Protector` communicate with the location requesting apps by intercepting the location object before it reaches requesting apps (Figure 4.3 - b). One of the main tasks is to add a system service, where the class belongs to the location APIs; thus, the new service is placed in the `android.location package`, which detects private locations via APs and can also be used by other components when calling context. In `Android`, a context allows an app to interact with the OS resources. Similar to Fawaz and Shin (2014) `Android` platform modification method, we add a static context field to the location class, which will be populated when the app is invoked; this enables `PL-Protector` to know which app is currently requesting the location object, and also communicate with the OS. Besides, `Fused Location Manager` combines sensors, GPS, WiFi, and cellular data into a single API for location-based applications (Hellman, 2013), hence separating data from `GPS_PROVIDER` and `NETWORK_PROVIDER` is no longer straightforward. `PL-Protector` addresses this issue by preventing app's location request to reach the `Fused Location Manager` service that collects and sends the network session data to the location provider. Instead, the requested location is retrieved from the on-device cache, and then, it is sent to the requesting app with privacy rules applied. The implementation is further detailed below into additional sub-tasks, which are established to be used for the evaluation of our middleware system (this is given in the next chapter).

Bootstrapping When `PL-Protector` first boots and before turning 'ON' location sharing settings, the user will have to perform the initial setup. This will include WiFi APs scanning, input geo-coordinates and set privacy choices using the UI (Figure 4.4). `PL-Protector`'s UI incorporates a map to get the corresponding geo-coordinates so achieving an effective privacy without affecting the location accuracy. At the same time, this prevents non-authorised sharing of device's exact location and network session data. The UI (Figure 4.5) enables users to set and manage their private locations and apps distinctly.

Runtime awareness of privacy policies. For this, we maintained a user notification service. Every time a new app requests the user's location at run-time, the service sends a notification to the user to select privacy preferences. Figure 4.6 shows a screenshot of a notification dialogue sent to the user when a social networking app (i.e., `Facebook`) requests the user's private location for the first time. The user can then assign the required privacy preferences using these notification dialog boxes/windows.

Generating Network Fingerprint Over-time. The user's private location profile generation is given above in the *bootstrapping* process. However, to improve place detection accuracy over time, we included a method using Equation 4.3 to update beacons rate of frequency, and embedded these counts as a field in the network fingerprinting `SQLite` table. Beacons with higher frequency rate were included to generate signatures of the detected private places at run-time.



Map WiFi Location

Detected Wifi APs

SSID: eduroam, BSSID: 10:bd:18:30:b5:d0, capabilities: [WPA2-EAP-CCMP][ESS], level: -75, frequency: 2412, timestamp: 22090179343, distance: ?(cm), distanceSd: ?(cm), passpoint: no, ChannelBandwidth: 0, centerFreq0: 0, centerFreq1: 0, 80211mcResponder: is not supported

SSID: eduroam, BSSID: c0:62:6b:e5:92:2f,

View/Set location co-ordinates for detected private place?

CACHE MANUALLY

CACHE AUTOMATICALLY

5220, timestamp: 22090179471, distance: ?(cm), distanceSd: ?(cm), passpoint: no, ChannelBandwidth: 1, centerFreq0: 5240, centerFreq1: 0, 80211mcResponder: is not supported

SSID: eduroam, BSSID: c0:62:6b:e5:92:20,

Mark As Home [✓ Work] Other

SAVE *Save detected wifi APs as marked above

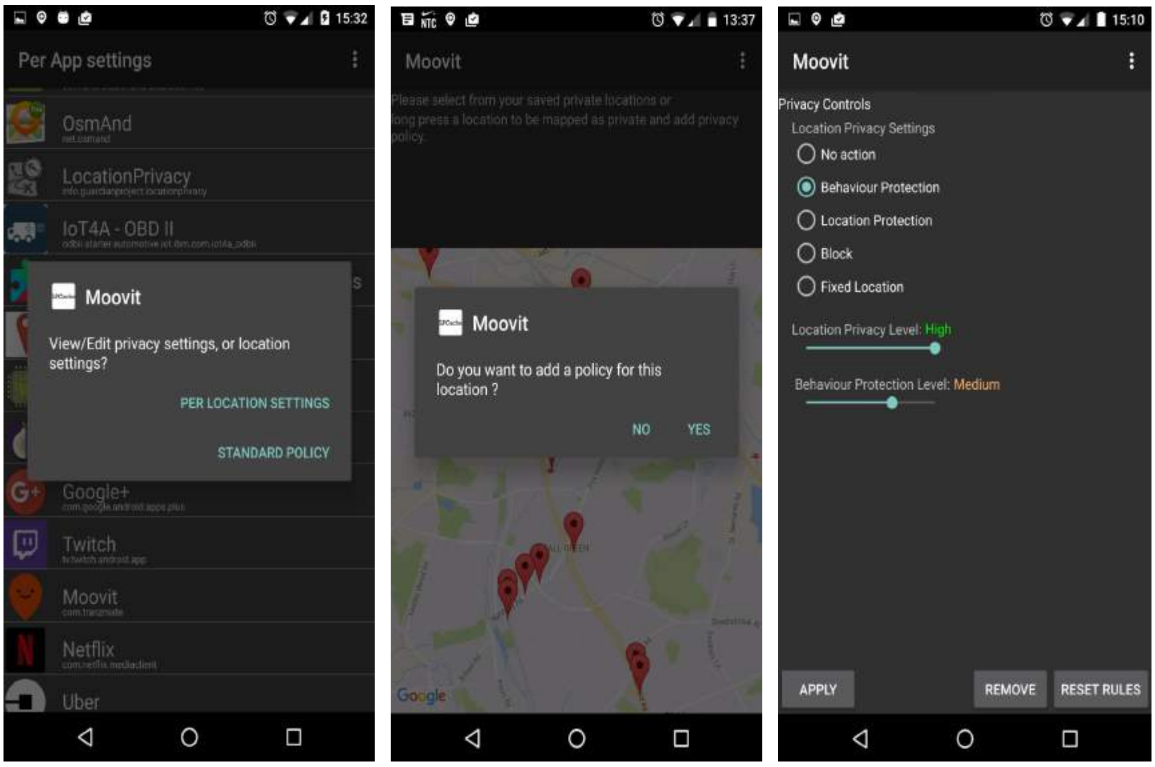
Geo-location Mapper

Please select from your saved private locations or long press a location to be mapped as private and add privacy policy.

Figure 4.4 User Interfaces to manage WiFi/location input settings

Modelling Users' Privacy Preferences in the UI Complex policies, fine-grained configurations and explicit technical details in the UI discourage users from fully exploiting the provided functionalities. To this end, we designed `PL-Protector`'s UI (see Figure 4.4) in an intuitive and straightforward manner that maintains the balance between the usability and expressiveness of users privacy preferences. However, in this research, ease of user access and UI design are excluded. We solely focus on the usability of the two LPPMs.

Buit-In Datasets Collection Mechanism We added regular database collection service running on the `Android` smartphone (`Nexus 6P`). In the initial testing period, this service generated copies of the `PL-Protector`'s database daily and saved it in the device's internal storage; however, we then changed its frequency to weekly. With the



Moovit

View/Edit privacy settings, or location settings?

Please select from your saved private locations or long press a location to be mapped as private and add privacy policy.

Moovit

Do you want to add a policy for this location ?

Privacy Controls

Location Privacy Settings

No action [✓ Behaviour Protection] Location Protection Block Fixed Location

Location Privacy Level: High

Behaviour Protection Level: Medium

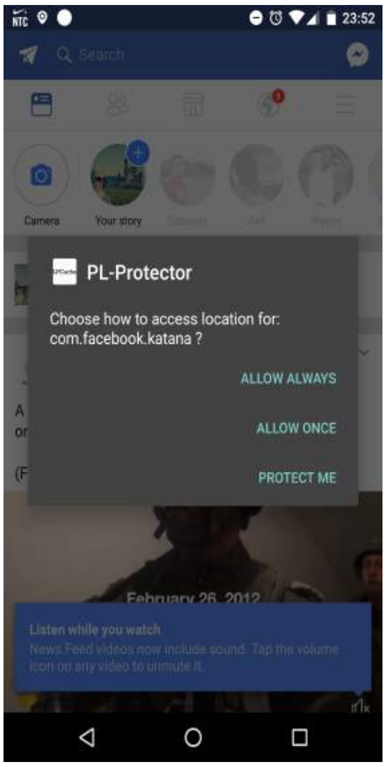
Figure 4.5 User interface of `PL-Protector` to manage per-app/location privacy rule settings

help of the DB Browser (2017) tool, we analysed these collected `SQLite` databases. Furthermore, we added another service to run a script that uses the privacy evaluation metrics (given in Section 4.2.4) to monitor achieved LoP_{per} and LoP_{total} in the collected datasets using the values of both Pl_d and Pl_s. For this purpose, we maintained special tags to record each location access session by every app that will later be used in data analysis and evaluation stages. Additionally, we also used Logcat (2017) the command-line tool provided by the IDE (Integrated Development Environment) of `Android` (Android Studio, 2017) to monitor and record exchanged messages and location traces between apps and `PL-Protector`.

4.3.3 PL-Protector Use-cases

We use the scenario technique (Carroll, 2000) in order to communicate understanding of the proposed system, its adequacy and appropriateness in real-time. We consider a user named Bob, and we have identified three different user scenarios as follows:

Scenario 1: Bootstrapping At home, *Bob* installs `PL-Protector` on his smartphone and activates the monitoring service for the first time. The service will prompt him with initial setup instructions. Using the `PL-Protector`'s UI, he can then mark the current place as private and save it. PL-Protector will mark the current place as Home and, since this is a new request, `PL-Protector` starts scanning the surrounding wireless APs to create a network fingerprint for *Bob*'s home. Once the network fingerprint is created, `PL-Protector` displays a dialog box to *Bob* that contains two options: 1. Cache location manually, and 2. Cache location automatically (see the 1st screen of Figure 4.4). The former option opens another screen containing a map (see the 2nd screen of Figure 4.4) that will allow *Bob* to find his home address on the map. Once found, he will have to long press at the geo-location point on the map; this will allow `PL-Protector` to map and save it as his home's geolocation. The latter, option sends the request to the location provider via the underlying OS and receives the geolocation to be mapped and saved on the device. Note that the request will only be sent to the location provider for the first time. *Bob* can then use the settings UI to apply preferred privacy rule for his home on per app or per location basis (see Figure 4.5). `PL-Protector` now has both *Bob*'s private location and its privacy rule to function properly. Unless *Bob* changes the privacy rule/geolocation via the user interface, the selected options will be saved and applied for all future location update calls from running apps. When *Bob* uses any location-based app at home, `PL-Protector` detects that he is within a private place, retrieves his


PL-Protector

Choose how to access location for:
com.facebook.katana ?

Figure 4.6 A notification sent to the user with location privacy rule settings

home location from the on-device cached database, applies the pre-set privacy rule, and then sends the processed location to the requesting app.

Scenario 2: Regular Since *Bob* pre-set his preferred privacy rule for his home (location), whenever *Bob* uses any app at home, `PL-Protector` follows the same process (i.e., detect, retrieve, push rule and send) for every received location update request. If *Bob* decides to grant location access to the `Moovit` app, he can go to the settings and change the location privacy rule to that particular app (see 1st screen of Figure 4.5). This will enable `Moovit` to access *Bob*'s location while at the same time other apps will be blocked (or restricted) from accessing his home location.

Scenario 3: Run-time notification While `PL-Protector` is running on *Bob*'s smartphone and if he installs the `Facebook` app. When `Facebook` sends a location update request to `PL-Protector` first time a notification (see Figure 4.6) will be sent to him. The notification will prompt *Bob* to select one out of the three permission options. Suppose he picks `Allow Always` option, so `Facebook` will have full access to his home and other locations. If he selects `Allow once` so `PL-Protector` will grant access for that session but block other sessions unless the settings are changed via the UI. If he selects the `Protect Me` option (as seen in Figure 4.6) then the standard settings UI will be displayed on the screen. Once *Bob* selects preferred settings only then the location will be sent to `Facebook`.

4.4 Summary

To summarise, this chapter presented the design and implementation of `PL-Protector`, a location privacy-enhancing middleware, which is a prototype system developed to validate the theoretical model (`LP-Cache`). We mainly focused on supporting our design goals, justifying the design decisions and elaborated on `PL-Protector`'s functionality. We have successfully implemented `PL-Protector` on `Android` platform (`version 6`) to enforce the privacy rules over both the information and control flows occurring between sources and sinks. Through the implementation of the middleware, we proved the deployment feasibility of a new series of privacy controls on a mobile platform to prevent private location disclosure during the formation of LBS queries. This also minimises the interaction and data collection from wireless access points, content distributors, and location providers. Later, we assessed developer efforts and complexity of implementing `PL-Protector` directly into `Android` platform core. In the next chapter, we describe the evaluation methods applied to validate `PL-Protector`'s performance. Followed by the presentation of results and finding in terms of usability and efficiency of `PL-Protector` when interacting with real apps in real-time. We will also present security analysis based on the threat model (defined in Section 4.2.2) to test its compliance with the three privacy settings and achieved privacy guarantees.

Chapter 5

Evaluation and Key Results

Research findings from this chapter have been published in a conference paper[a] *and an article is developed to be published in a journal*[b]

[a] Patel, A.,& Palomar, E., "A middleware enforcing location privacy in mobile platforms". *In: 14th International Conference on Trust and Privacy in Digital Business (TrustBus)*, pp. *32–45*, Springer 2017.

[b]Patel, A., & Palomar, E. "A Practical Cache-based Location Privacy-enhanced Middleware for Mobile Users".

5.1 Introduction

Following a systemic literature review presented in Chapter 2, we proposed a new privacy-enhanced theoretical model (`LP-Cache`), and its prototype implementation (`PL-Protector`). In this chapter, we evaluate `PL-Protector` in terms of performance, security and privacy analysis, and present overall results and findings considering the pre-defined threat model. For `PL-Protector`'s development and evaluation, we largely adopt a comparative study approach since it gives us two main advantages. Firstly, it allows us to compare `PL-Protector`'s functionality with the current process (described in Section 3.2) throughout the development/testing phase. Secondly, this approach makes it easier to evaluate trade-offs for implementing `PL-Protector` on the mobile platform. Furthermore, considering the users' location privacy requirements that we obtained from the field study results, we assess the user acceptance of `PL-Protector`.

The rest of the chapter is organized as follows. Section 5.1 describes the evaluation methods and goals. Section 5.2 presents validation of our system's functionality, and its security and privacy features. Whereas, Section 5.3 presents `PL-Protector`'s performance analysis in terms of cache storage estimation, communication, and computation overheads. Section 5.4 presents the security and privacy analysis, and the field study results. Section 5.5 provides the comparison of `PL-Protector` with related work. Finally, Section 5.6 summarises this chapter.

5.1.1 Evaluation Methods

For our proposed model's evaluation purpose, we have followed three different methods as follows:

1. *Prototype Testing.* We used rapid prototyping method to develop `PL-Protector`, which is utilised to demonstrate and validate the functionality, acceptance and robustness of our proposed theoretical model. A prototyping method helped us present high quality demonstration and validation of our proposal based on the real-life scenarios. We run series of experiments with and without using `PL-Protector` for data collection and evaluation purposes. We designed different experimental setups to test `PL-Protector`'s functionality, usability and efficiency in realtime.

2. *Mobility Datasets Collection and Analysis.* We used smartphones with `Android` 6.0 (API 23) that have 802.11a/b/g/n radio feature for mobility data collection and analysis purposes. We conducted both static and dynamic analysis on the collected mobility datasets. For data consistency, we maintained identities and location categories of 34 different private places throughout the data collection

process. We have collected empirical data from a number of sessions running at different time intervals over a period from 3 to 6 months. We then created two datasets: In a first dataset, we include the session data of ported apps that run over the conventional `Android` environment without interacting with `PL-Protector`. Henceforth, we call the conventional `Android` environment, *Baseline*. The second dataset consists of the same apps but running in the presence of `PL-Protector`. We validated the collected mobility datasets using evaluation metrics pre-defined in Section 4.2.4 to conduct performance and security analysis.

3. *Field Study.* As mentioned in Chapter 1, we conducted a field study using a survey method. However, the aim of the field study is twofold. First, to determine smartphone user's perception of location and mobile apps privacy concerns. Second, to assess user acceptance of `PL-Protector` based on user's location privacy requirements.

5.1.2 Evaluation Goals

We specified in Chapter 4 that, for the purpose of evaluation, we deployed `PL-Protector` as a middleware on a `Nexus 6P` with `Android` 6.0 (API 23) that have 802.11a/b/g/n radio feature so it can operate in both 2.4 GHz and 5 GHz bands at 34 different private places. By successfully designing and implementing `PL-Protector` on `Android` platform, we demonstrate that our theoretical model is a practical solution. However, we still need to evaluate how secure and efficient `PL-Protector` is. Moreover, it is a crucial requirement to measure our middleware's privacy vs usability trade-offs in order to determine its deployment feasibility in the real-world scenarios. We present the evaluation of `PL-Protector` in terms of performance and security by considering three different evaluation criteria:

Criteria 1. What is the performance overhead of `PL-Protector` while interacting with real-world apps over time?

Criteria 2. How well `PL-Protector`'s LPPMs (i.e., the on-device location computation mechanism and the personalised location permissions mechanism) perform in practice. Can the LPPMs find accurate location data corresponding to the cached network fingerprints and apply permission rule in real-time?

Criteria 3. How well `PL-Protector` can perform with respect to user privacy/data leakage using actual mobility traces collected from real-world apps?

5.2 System Validation: Test Cases in Birmingham

`PL-Protector`'s system model, explained in Section 4.2, includes five entities: 1. Middleware, 2. User, 3. Service Providers, 4. Mobile Platform, and 5. Network Infrastructure.

Hence, we validate our developed prototype system based on `PL-Protector`'s interaction with involved entities using a real test case in Birmingham, UK. Once we validate our system on `Android` platform, we can then evaluate its performance and security. We installed `PL-Protector` on an `Android` device (OS Ver. 6) and used it in Birmingham as the system validation test case.

Entities involved in the system validation test case are:

1. *Middleware* → `PL-Protector` *installed on an* `Android`
2. *User → Tester*
3. *Service provider → LBS App Provider*
4. *Mobile Platform* → `Android`
5. *Network Infrastructure → WiFi*

We derived two dedicated test cases to validate: 1. system's functionality, and 2. system's security and privacy features. The former case is predominantly dedicated to identifying the developer/tester encountering a data input failure, control flow breakdown and call execution failure. Whereas, the later validates system's privacy and security features in the presence of adversaries that have passive and active capabilities. These adversarial capabilities can range from curious individual scanning network packets or eavesdropping apps' communication within the user's device; to those with common technical knowledge of sniffers; and to those who are experienced and practicing adversaries regularly collecting and profiling the user's mobility data.

5.2.1 System's Functionality Validation Tests

In this test case, experiments were conducted to assess and validate our system's functionality, which includes implementation of the key methods that handle data input and control flow at runtime. Table 5.1 describes our system methods and calls that are implemented on `Android` platform.

Results for test case 1. Table 5.2 presents our system's functionality validation tests results. During the bootstrapping state, network fingerprints for all of the 34

Table 5.1 Description of data and methods utilised in our system.

System Calls and Methods	Description
$app \Leftrightarrow middleware$	Apps interacting with PL-Protector
$middleware \Leftrightarrow OS$	`PL-Protector` interacting with the OS (`Android`)
$create_{fprints}()$	Create network fingerprints and geolocation for marked private place
$map_{rule}()$	Permissions rule application on location object
$intercept_{locObj}()$	Intercept location update requests
$get_{wifiInfo}()$	Insert, update and verify WiFi data at runtime
$get_{mapsUI}()$	Display maps UI

Table 5.2 System's functionality validation tests results.

System State	**Events: Methods Execution**	**Expected Output**	**Success?**
bootstrap	$create_{fPrints}()$ w.r.t $p_i \in P_l$	Save network fingerprints $\forall$ P_l in *DB*	Yes
$app \Leftrightarrow mid$	$intercept_{locObj}() \rightarrow map_{rule}() \rightarrow send_{locObj}$	Apply rule on every location access call requested by apps when the private location *flag* is ON	Yes
$middleware \Leftrightarrow OS$	$get_{wifiInfo}() \rightarrow get_{mapsUI}()$	Detect beacons and display the map UI at runtime	Yes

marked private places were created, stored and detected at runtime. After this state, our middleware successfully intercepted every event of location access call to apply the location privacy rule. It then resumed the app's control flow by releasing the location object and maintained every session. Our middleware also communicated with the OS regularly to detect the surrounding beacons and to display the map UI to the user when needed. The attained results demonstrate that our system fits the intended use and meets its functionality requirements successfully. Thus, we conclude the implementation of the key system methods that handle data input and control flow have been internally validated. The external validation takes place when it is performed by asking the involved stakeholders (mobile platform providers, app providers, location providers, users) to verify whether our system meets their needs. Due to the research scope we have excluded our system's interaction with the stakeholders; and therefore, conducted tests that only consider internal validations.

5.2.2 System's Security and Privacy Features

Our system guarantees location and wireless network data privacy in location-based mobile environments. Hence, we derived a dedicated test case to validate its security and privacy features. We assume adversaries can have passive and active capabilities to access user's private location and WiFi data.

Location data

Our system protects the user's private locations from privacy threats. Hence, to validate this, we ran a series of tests with the assistance of sniffers such as Wireshark (2016), tPacketcapture (2016) and `Android`'s code monitoring tags (see *the built-in datasets collection mechanism* previously described in Section 4.3.2). This enabled us to monitor data flow and maintain test records of the shared (and leaked) location data from the user's device. Moreover, based on the type of location access that adversaries can obtain while the user is in private places, we categorised them as follows:

- *Passive* - we determine an adversary as passive if he has sporadic access to the user's location objects (i.e., he receives the location object once or twice a day); or he can passively sniff shared location objects within the user's device.
- *Active* - we determine an adversary as active if he has continuous access, which is up to every 10 mins or less, to the user's location objects; or he uses sophisticated sniffing or malware injection tools.

WiFi data

The Article 5 of GDPR (General Data Protection Regulation)[1] states that "data

[1](Url: https://ico.org.uk/for-organisations/data-protection-reform/overview-of-the-gdpr/principles/, accessed Oct. 2017

Table 5.3 The location and WiFi data privacy offered by our system.

Adversary		**Profiling Threat Mitigation**	**Tracking Threat Mitigation**	**Data Minimisation**	**Access Control**
Location	Passive Access	✓	✓	✓	✓
	Active Access	✓	✓	✓	✓
WiFi	Passive Sniffers	✓	✓	✓	✓
	Active Sniffers	-	-	✓	-

minimisation principle requires that the collected data should be adequate, relevant and limited to what is necessary in relation to the purposes for which they are processed". To support the user's data and privacy, our system design implements this principle by minimising the WiFi data sharing and gathering requirement in the location computation process, which is deployed within the existing mobile app ecosystem (defined in Section 3.2). Using aforementioned sniffers and monitoring tags we ran series of tests to: monitor the WiFi data flow, assess our system's capability to mitigate the location privacy threats and quantify achieved data minimisation.

- *Passive* - we define as passive adversaries that have permission to access and collect WiFi data via the user's device; but, they share the collected data with other service providers for commercial and financial gains causing severe privacy leakage. Due the fact that they can passively leak the user's data, we categorise such adversaries as passive sniffers.

- *Active* - we define as active adversaries that do not have the permission, but they can aggressively collect WiFi data with the assistance of sophisticated sniffers. These adversaries can operate in two ways: within the user's device in the form of an application or third-party libraries, and with the wireless network around the user's device actively collecting radio signals.

Results for test case 2. Tests results in Table 5.3 show efficiency of our system's security and privacy features in protecting the user's location and WiFi data from 4 types of adversaries that are mentioned above. These results draw the following conclusions that validate our system's security and privacy features.

1. *Profiling Threat* - the information such as dwell time at a private place or repeated visits in varying frequency and durations can cause profiling threat to the user's privacy. We observed when location privacy setting (i.e., *Behaviour Protection*) was 'ON', our system increased the ambiguity level in the collected mobility data of the user. Geo-coordinates of the user's home (i.e., private) location ($lat = 52.424861$, $long = -1.844832$) were obfuscated for all the location calls sent to our middleware from the installed apps in the obfuscation range (i.e., *lat* = ⟨52.98238466 to 52.5269165386⟩, and *long* = ⟨-3.00256334 to -2.9091795533⟩). Hence, we claim that our system can mitigate profiling threat and prevent the indiscriminate collection of the user's location data.

2. *Tracking Threat* - the information that can cause a tracking threat is the typical route/ circuit path taken by the user as he/she moves from one place to another during a given time interval. We noticed that when the location privacy settings (i.e., *Behaviour Protection* or *Location Protection*) were 'ON' and the tester reached his private places, the requesting apps couldn't record the information needed to track the user at his private places. This was due to the taken routes or circuit paths being broken since the time spent at private place could not be recorded or movement patterns were dispersed. Hence, we claim that our system mitigates tracking threats.

3. *Data Minimisation* - the on-device caching mechanism implemented as part of our middleware minimises the interaction and data collection from wireless access points, content distributors and location providers. We noticed the interaction

with the location provider is reduced (i.e., 1 out of 15 location calls were sent to the location provider) since the location computation is done within the user's device. As a result, the need to periodically send surrounding wireless data to the location provider was significantly reduced by our system. Hence, we claim our system offers data minimisation in the existing ecosystem.

4. *Access Control* - our system's LPPMs provide additional access control to the OS resources that prevent indiscriminate collection of: the user's location data, and the surrounding wireless data that are shared with service providers. We have validated our system's privacy controls in Table 5.3.

Using *WiFi data*, the tracking and profiling threats happen at the data link layer due to the indiscriminate broadcast of unique device identifiers (i.e., MAC addresses), whether the user's device has made use of particular service or not. If an adversary operates outside of the device – such as he actively scans beacons to get the user's link layer identifiers to create his profile or track him – then our system does not provide any data or location privacy protection. However, our system applies LPPMs within the user's device before the data is sent outside in the form of packets. This prevents an active adversary that aims to eavesdrop the data link layer or the network layer communication by capturing these packets sent from the user's device to service providers. This way our system minimises sharing of the user's sensitive location and WiFi data before such data is sent outside of the device via the link or network layer. Hence, we confirm that our system can mitigate such tracking and profiling location privacy concerns.

5.3 Performance Analysis

This section presents comprehensive elaboration on `PL-Protector`'s performance analysis, including experimental results and demonstration of substantial findings.

Experimental setup

LPPMs can incur performance degradation to the location-based apps, and this may create a service obstacle. Considering this and for the purpose of `PL-Protector`'s evaluation, we determined two different experimental settings:

1. *Location-based apps performance.* The first experimental setup studies the efficiency of `PL-Protector` in terms of QoS and usability on operations relevant to the location-based apps and privacy leakage tests. For this purpose, we ported real apps of five different LBS queries categories: 1) Social Networking (e.g., Facebook), 2) Instant Messaging/Chatting (e.g., Whatsapp), 3) Tracking (e.g., Fitness), 4) Utilities (e.g., Weather, Alarm, etc.) and 5) Finder (PoI Finder/Geo-search). Based on these apps' operations, we assume that both 1st and 3rd app categories require continuous access to location data; whereas, 2nd, 4th and 5th app categories involve sporadic access. Henceforth, we set these apps as the performance evaluation benchmarking apps.

2. *WiFi fingerprinting and cache performance.* The statistical analysis, in Section 3.5, on WiFi AP data availability and consistency demonstrates that smartphones regularly detect similar beacons at the frequently visited place; and it confirms that for the place detection at least one beacon should match with the stored WiFi fingerprints. To strengthen this further by conducting dynamic analysis, in the second experimental setup, we investigate the performance of the WiFi fingerprinting method at runtime. This method is instrumented in our middleware as the private place detection source. Using 2 or more different LBS apps at

34 distinct private places, we sent a sequence of location update calls to our middleware to measure the dynamic response rates and cache accuracy over time.

5.3.1 Results for Criteria 1: Privacy and Quality of Service

In this section, we assess and evaluate the system development effort and complexity of implementing `PL-Protector` directly into `Android` platform core, highlighting its efficiency and feasibility. Hence, in the below sections, we will discuss the runtime overhead impact on the underlying OS and apps functionality.

Runtime overhead impact on the underlying OS

1. *Computational effort and memory consumption*

`Android` provides a flexible model of process isolation, ICC (Inter-component communication) and IPC (Inter-process communication) services. Since `Android` development architecture is comprised of multiple components (e.g., *Activities*, *Services* and *Broadcasts*), `PL-Protector` used ICC calls known as *Intents* and IPC services to communicate messages with the components of benchmarked apps. We assess `PL-Protector`'s computational overhead posed on the base OS (i.e., `Android`) while responding to the location calls invoked by apps. We used `Android`'s `MemoryInfo` API and the `Android Device Monitor` tool to log observed computational and memory overheads caused by the presence of `PL-Protector`. Equation 5.1 is used to calculate the total memory consumption that is required for completing every computational lifecycle of `PL-Protector`:

$$C_{comp} = \sum_{i=1}^{l_n} C_i + M_i \tag{5.1}$$

where C_{comp} represents the total memory consumption by the LPPMs, C_i is a location computation operation, M_i is a rule mapping operation, and l_n is the number of received

Table 5.4 Observed difference in monthly dynamic DB storage

Storage Overhead	**1 Month**	**3 Month**	**Observed Change in DB size**
Network Fingerprint	54 KB	108 KB	44 KB Increase in size
Permissions	21 KB	28 KB	7 KB increase in size
Total DB Storage	75 KB	136 KB	51 KB Total increase in size

location update calls from apps. We sent 1 to 10 location calls to the middleware using benchmarked apps and recorded memory consumptions for both *total* and *per* call. Our results show that `PL-Protector`'s core functionality requires 7.3MB of the device memory while its interactions with each app requires 2.8MB of memory on an average. To justify these overhead values, we argue that `Nexus 6P` comes with built-in 3GB memory RAM (Random Access Memory) (< RAM size in later versions) and the official Google Chrome browser app on page load with an empty page consumes around 97MB of the memory. Thus, we argue that `PL-Protector`'s consumption requirement of 21.3MB of memory to communicate with 5 apps (at once) falls within the current acceptable limits of the base OS (i.e., `Android`).

Further, to improve the response time, while the user is stationary the obtained responses (i.e., location objects) are cached in the device memory and shared with other apps that have similar permissions. `PL-Protector` requires 125bytes for caching a single location object in the cache memory. The current built-in caching limit for `Android` is 1MB (Android, 2016). Therefore, the maximum number of location query results saved as cached messages must be < 8388, which is sufficient for `PL-Protector`'s functionality since it only considers the user's private locations.

2. Runtime storage overhead

We created `PL-Protector`'s embedded database structure using the table attributes given in Table 3.2 of Chapter 3. Due to the research scope, we have used index search on `PL-Protector`'s on-device database table entries. However, this can later be improved by including advanced database searching and sorting algorithms. Since `PL-Protector` only stores wireless network fingerprints (i.e., beacons) and geo-coordinates of the user's private locations, we have found that the cache storage requirement is substantially limited compared to other cache-based LPPMs (described in Section 2.7). While `PL-Protector` is installed and running on the `Nexus 6P`, we dynamically collected regular versions of its database for a period of 1 to 3 months using the built-in datasets collection mechanism (see Section 4.3.2).

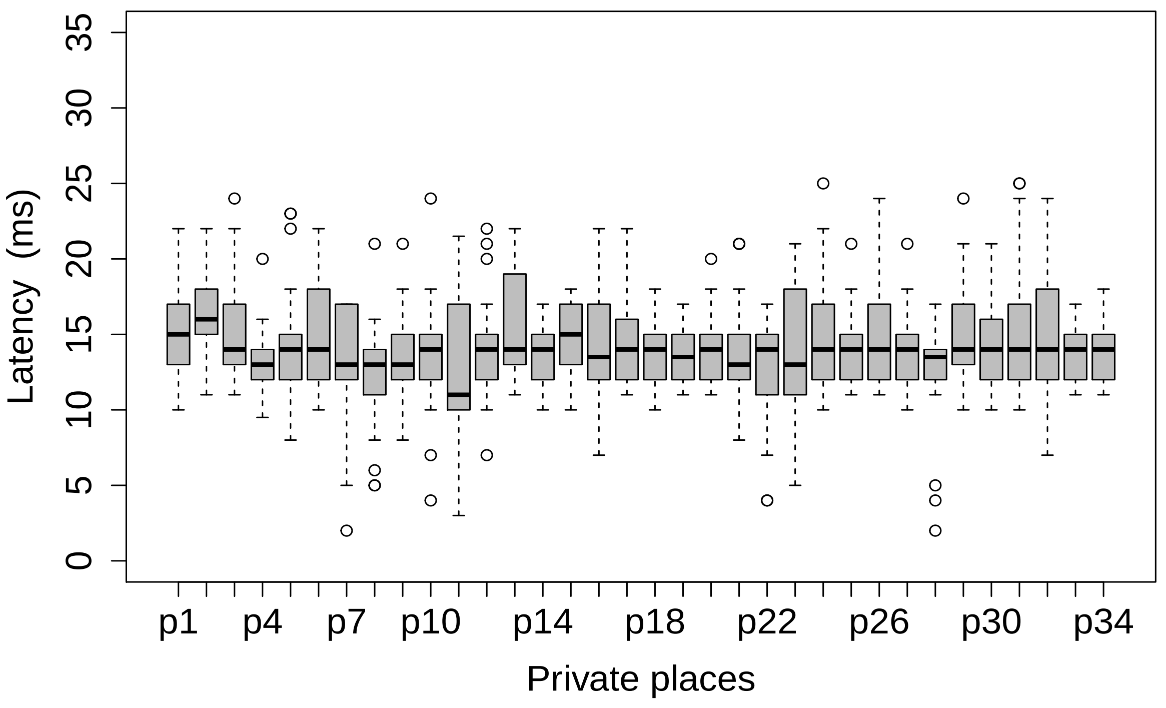
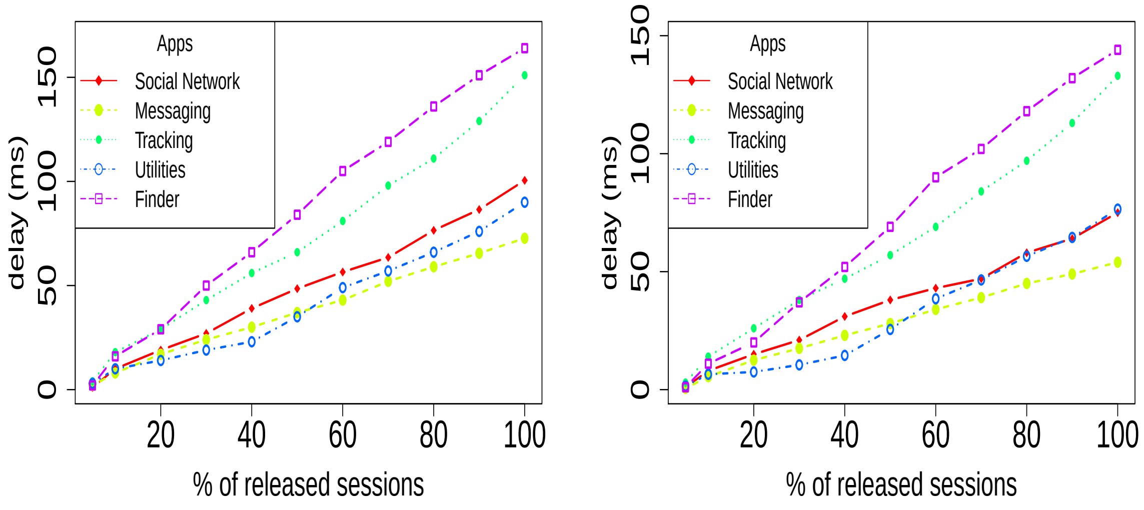
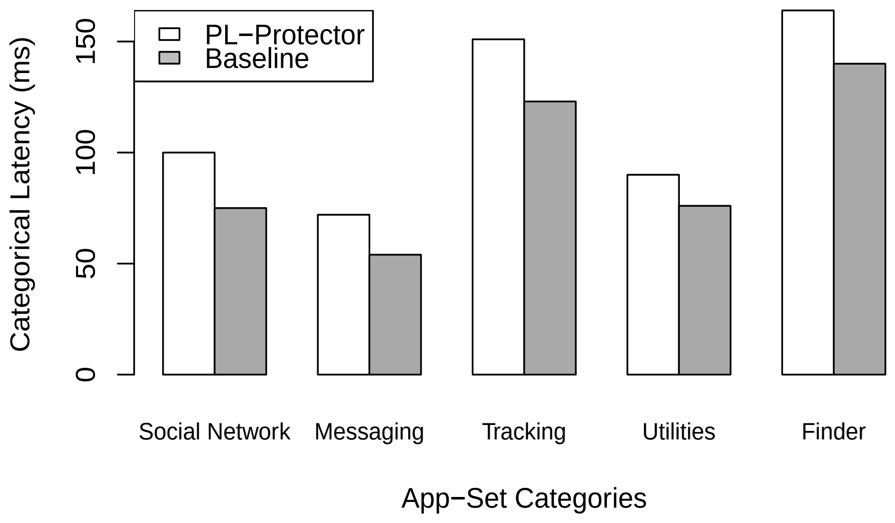


Figure 5.1 `PL-Protector`'s overall computation latency caused at 34 distinct private places.



(a) Apps communication overhead without (left) and with (right) `PL-Protector`.



(b) Categorical latency (ms) on different app-sets categories in both environments.

Figure 5.2 Communication overhead for apps running in both environments baseline (without) vs `PL-Protector`.

Table 5.4 presents the observed monthly increase in the database size. These results evidence that `PL-Protector` does not have a massive on-device cache storage requirements. `PL-Protector`'s database size of 136KB include dynamically collected network fingerprints for 34 private places and privacy rules for 5 location-based apps; and it also includes other table attributes needed by the operations of its LLPMs and the mobility datasets collection purposes. The internal storage limit for `Nexus 6P` range from 32GB, 64GB to 128GB (Nexus 6P, 2017). Hence, we argue that `PL-Protector`'s database size of 136 KB for 3 months is within the acceptable internal storage limit; and the current mobile device internal storage capacity is sufficient for its overall functionality.

Runtime overhead impact on location-based apps functionality

Crucial for its functionality, we measure the delay latency as the time `PL-Protector` takes to interact with the app and perform an entire computational cycle, i.e., to compute the location on-device and to apply the privacy rules. To measure the overall functionality overhead for each app, we varied the range of location calls – over 10 trials of 2 to 5 types of LBS queries – in collected databases for both baseline and `PL-Protector`. For instance, `PL-Protector` took 187 milliseconds to successfully reply to a LBS query requested by the app, compared to 179 milliseconds on the baseline, i.e., 8 milliseconds increase. Figure 5.1 indicates measured delay latencies for all the 34 private places, on an average, `PL-Protector` presents a latency lower than 22 milliseconds to handle all of the location-access calls at runtime. This includes runtime execution of `PL-Protector`'s computational cycle and regulation of pre-set user's privacy policy. We found that the reason for increased latency is due to `PL-Protector`'s load time and cross-process/IPC service transfers of location updates. However, this latency is smaller than 100 milliseconds and, thus, small enough to not cause user-noticeable delays while utilising apps on the device. Furthermore, Figure 5.2 (both 5.2a & 5.2b) shows the communication overhead observed during different sessions, and the overall app functionality overhead for the 5 app categories and compares both baseline and

`PL-Protector` execution environments. In per-location access sessions, we found less than 19 milliseconds and less than 8 milliseconds additional delays in apps that require continuous and sporadic location updates, respectively (see Figure 5.2). Figure 5.3 indicates that `PL-Protector`'s additional communication overhead decreases after a number of repeated sessions, and the observed delays remain well within the acceptable bounds (i.e., less than 100 milliseconds) throughout all the recorded sessions. Thus, we claim that `PL-Protector`'s runtime overhead is acceptable to run existing apps from the aforementioned five LBS categories since their core functionality already accepts delays in this range.

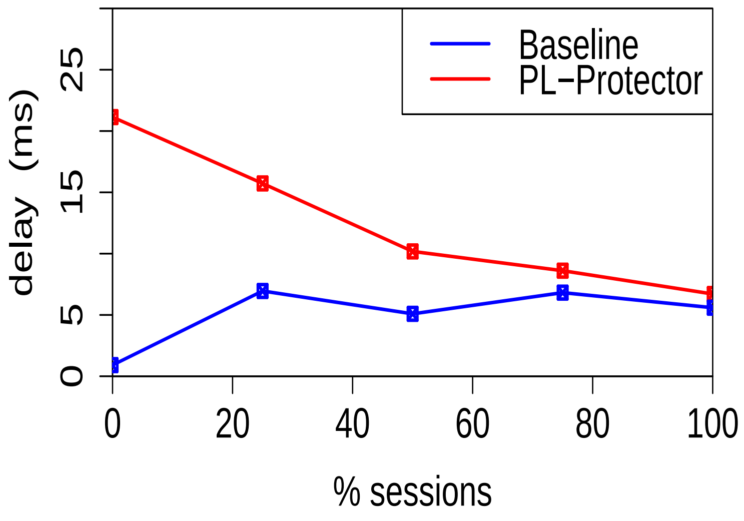


Figure 5.3 Total communication overhead in both environments.

5.3.2 Results for Criteria 2: Performance and Accuracy of LPPMs

Here we evaluate our proposed LPPMs implemented in `PL-Protector`: 1. On-device Cache Database Creation Mechanism, and 2. Personalised Permissions Mechanism.

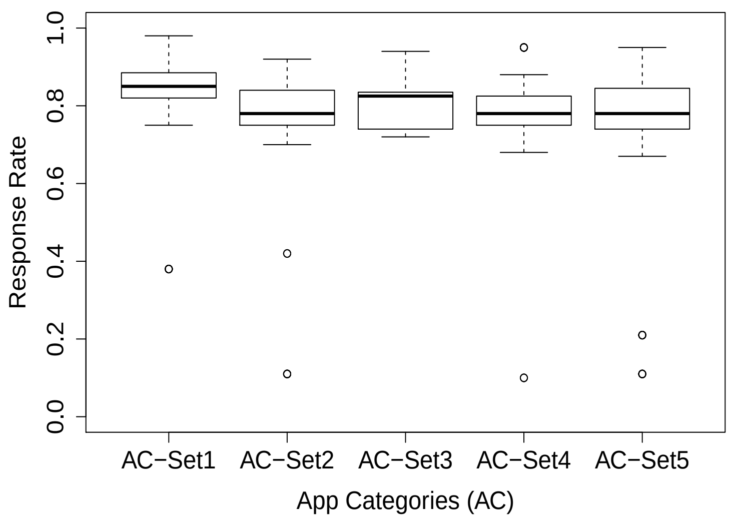


Figure 5.4 Achieved response rate by `PL-Protector` when interacting with apps

LPPM 1. On-device Cache-based Location Computation Mechanism

We focus first on the dynamic analysis of the implemented network fingerprinting method since it is the primary mode of the on-device cache creation in `PL-Protector`'s LPPM-1.

Dynamic analysis of network fingerprinting method - based on the statistical analysis conducted on WiFi feasibility and usability in Section 3.5.1, it is confirmed that smartphones regularly detect similar beacons at the frequently visited place, for place detection at least one beacon should match with the stored fingerprints. However, to analyse the place detection accuracy of the on-device cache method at runtime, we examined occurrence of cache hits and misses that includes three possible outcomes (see Section 3.5.3): (a) *The location is cached and up-to-date*, (b) *The location is cached but is out-of-date*, and (c) *The location is not cached.* As Figure 5.4 shows, the average dynamic response rates range between 70% to 90% of accuracy compared to our statistical analysis results, where the response rates ranged between 75% to 97% of accuracy (see Section 3.5.1). Therefore, it is apparent from the result that the

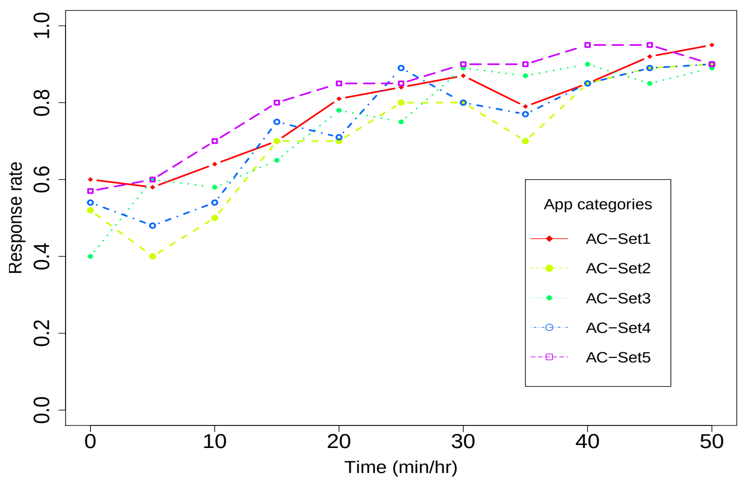


Figure 5.5 On-device cache accuracy of the inter-request interval over time (hr)

WiFi fingerprinting method is effective when used as private place detection source in `PL-Protector`.

Cache accuracy - a further comprehensive description of timely performance of the on-device cache mechanism, Figure 5.5 shows that initially, for up to 8 hrs duration, the result range from 40% to 60% of cache accuracy. However, it gradually increases to 75% to 92% of accuracy indicating that its performance improves over time. The obtained results indicate the suitability of `PL-Protector`'s on-device location computation mechanism to handle apps requiring both sporadic and continuous location-updates. These results also demonstrate that the cache update frequency is within practical limits and provides accurate location data at runtime to all the requesting apps. Due to research scope, we have used index search on on-device cache database table entries; nonetheless, to achieve efficient capability and better accuracy of place recognition via beacons, a *place discovering algorithm* like Kim et al. (2009) or a machine-learning technique like `LearnLoc` (Pasricha et al., 2015), as well as advanced database searching and sorting algorithms can be implemented.

Energy efficiency - currently, for location accuracy `Android` platform, as well as `iOS`, follows fused location sensor practice, where precise location is calculated using all the built-in sensors together to calculate the user's location. This enables the location provider to get user's precise location at all times and, as a result, more effective privacy preservation measures are needed to mitigate location privacy threats. Moreover, GPS and cell-tower based localisation do not work accurately in indoor environment; hence, there is a constant exchange of WiFi and other wireless sensor data with `Android`'s location provider. Furthermore, according to Wang et al. (2016), amongst all the localising clients on the user's devices, GPS is energy-hungry and consumes up to 150 mW at 1 Hz. As a result, if a user constantly uses `Android`'s built-in location services on their device the battery performance is significantly low. Whereas, in `PL-Protector`, the WiFi client is the main power consumption component used in its privacy-enhanced on-device cache-based location calculation mechanism, which improves the accuracy of indoor localisation by creating a low-cost infrastructure-less indoor privacy-enhanced localisation solution. This enables `PL-Protector`'s LPPMs to achieve better energy performance and privacy preservation as compared to existing mechanisms deployed in `Android` platform. Hence, we claim `PL-Protector` successfully overcomes these energy efficient localisation and location privacy issues in the mobile app ecosystem.

LPPM 2. Personalised Location Permissions Mechanism

For this purpose, we used location-based app set-up (see Section 5.3) and selected 2 apps: App 1. with continuous access, App 2. with sporadic access to location data. We examined these apps with following permissions that are required to gain access to user's location and are categorised as dangerous permissions since `Android` version 5.1 (API 22) and later versions:

- ACCESS_FINE_LOCATION

- ACCESS_COARSE_LOCATION

In `Android`, getting a user's location works by means of callback[2]. As aforementioned in Section 4.3.2, the app receives a new location object when a new location is available, the callback function is invoked; `PL-Protector` intercepts these callbacks to communicate with requesting apps. Using `Android`'s location API, the app indicates to receive location updates from the `LocationManager` by calling methods:

- requestLocation();
- getLastKnownLocation();
- requestLocationUpdates().

We made the two selected location-based apps communicate with our `PL-Protector` and instrumented different LBS queries at timed intervals using the three calling methods mentioned above. The observed success rates of instrumented calls listed in Table 5.5 validates the performance of `PL-Protector`'s permission mechanism. The observed performance degrades or missed events during execution of `PL-Protector`'s privacy controls was due to programming and code execution error; however, none of the selected real-time apps crashed or disrupted during the entire testing and data collection period. Misses were observed when `PL-Protector` took long time to detect the stored beacon fingerprints, where it considered user is in a public place and sent the request to the OS that may send user's actual private location to the requesting app. However, such occurrences were minimum as shown in Table 5.5 and can be avoided by adding a waiting time rule in the network fingerprint detection method. This is a suggestion as a future enhancement of the model.
Overall, we find our prototype implementation performs well enough even for real-time location-based apps, and moves the performance bottleneck outside of `PL-Protector`

[2]For more information, please go to this link:
https://developer.android.com/guide/topics/location/strategies.html, accessed on Sep. 2017.

Table 5.5 Location privacy permission controls evaluation results

Privacy controls		**App 1**		**App 2**	
		No. of Instrumented Calls	**Success Rate**	**No. of Instrumented Calls**	**Success Rate**
Standard	Behaviour Protection	50	96% (48)	15	100% (15)
	Location Protection	35	94.28% (33)	15	100% (15)
	Block/ Fixed	15	100% (15)	10	100% (10)
Per-Location	Behaviour Protection	50	94% (47)	15	93.33% (14)
	Location Protection	40	97% (39)	15	100% (15)
	Block/ Fixed	21	100%	10	100% (10)

and into the programming and coding proficiency. By adding a low-cost infrastructure-less indoor localisation, `PL-Protector` provides energy-efficient and privacy-enhanced location calculation solution. The aforementioned results and empirical findings of the performance analysis confirm that `PL-Protector` provides location privacy at the acceptable overheads on the underlying OS and negligible loss in the functionality of location-based apps.

5.4 Security and Privacy Analysis

In this section, we analyse user privacy/data leakage that is likely to affect PL-Protector by an adversary's access to user location in and out the device and can launch an attack.

5.4.1 Security Risks

We discuss data security risks that each of the five apps poses when running on mobile platforms, and find that `PL-Protector` mitigates those risks successfully under privacy leakage tests.

Continuous access Apps (e.g., Social Networking and Tracking) have higher potential to leak location information to attackers, e.g., unauthorized 3rd party service providers and content distributors, since users access such apps for longer duration and in a frequent manner. The social networking app has constant Internet access for ads and for troubleshooting/crash reporting. `PL-Protector` separates private locations from public/less-sensitive locations and enforces privacy rules.

Sporadic access Apps (e.g., Instant Messaging, Utilities or Finder) can leak location data along with other sensitive user information as we note that these apps require Internet access for core functionality. Therefore, under current OS controls it is very easy for this app to leak location data. `PL-Protector` isolates the private location flow to be restricted within the device, eliminating any possibility of cross-flows between apps and 3rd party service providers over the Internet.

Covert access A potential malicious apps leverages vulnerabilities in other already installed benign apps to perform actions that are beyond its individual privileges such as sending device location through text messages. The current mobile OS access control model is per app and it cannot detect such location attacks as it cannot check the security posture of the entire system. `PL-Protector` enforces location privacy policies before releasing the data that mitigates most, if not all, of such covert access vulnerabilities.

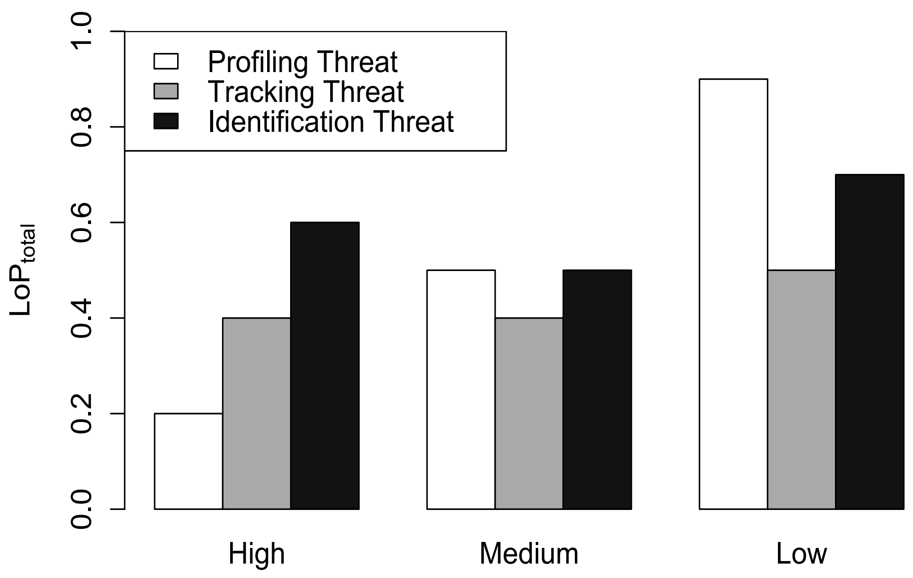


Figure 5.6 *LoP_{total}* achieved by `PL-Protector` to mitigate 3 privacy threats

5.4.2 Results for Criteria 3: Privacy/Data Leakage Test

Here we present a comprehensive security and privacy analyses of `PL-Protector` with respect to satisfaction of the privacy-preserving properties.

Proving Privacy Properties Our model design aims to satisfy the privacy properties that benefit both the end users and service providers within the mobile app ecosystem. Hence, `PL-Protector`'s privacy/data leakage analysis is based on the privacy properties: 1) Unlinkabiliy, 2) Unobservability, 3) Anonymity, and 4) Controlled Information Disclosure (also used in Chapter 2). Pfitzmann and Köhntopp (2001) considers that *anonymity* in a collected data set can be achieved using *unlinkability*, *unobservability* and *controlled information disclosure* properties. In `PL-Protector`, we aim to achieve location privacy by applying *controlled information disclosure* while sharing the user's private location that creates a generalised data set protecting the user's sensitive places. In `PL-Protector`'s context, anonymity translates to *unobservability* and *unlinkability* of private locations to the user within collected datasets. Here we recall `PL-Protector`'s privacy model (see Section 4.2.4), where the privacy evaluation metrics LoP_{per} and LoP_{total} follow as – the "larger" the Haversine distance Pl_d, the "better" the anonymity.

1. Privacy Threat Mitigation

Here we present an analysis concerning the correctness of our proposal in terms of the three fundamental threats: identification, profiling, and tracking. We use the aforementioned privacy metrics (Section 4.2.4) to evaluate achieved privacy. To compute LoP_{per} and (overall) LoP_{total}, We identify applied privacy settings using (labels) value of Pl_s in the collected location traces of apps' sessions and compare them with the privacy rule. We measured LoP_{total} against the observed value of Pl_d for every location-access session, higher the value of LoP more protected is the private location. The following observations are made from the achieved mitigation results considering the fundamental location privacy threats shown in Figure 5.6.

User Tracking Threat Mitigation. This requires protection of private location anonymity and sensitive behaviour pattern from the adversary. We observed that the continuous location updates can pose a high risk to locating the user in real time. `PL-Protector` blocks the app's background access to location updates that restricts location tracking to foreground app sessions, which are considered rare or sporadic (once/twice a day) and initiated from same the private place for about 96%. This enabled `PL-Protector` to block or obfuscate the constant mobility patterns of the user. Thus, by preventing the identification of sensitive behaviour pattern and location prediction of the user, `PL-Protector` mitigated tracking threats in 40%, 40% and 50% of released location-access sessions at P_{low}, P_{medium}, and P_{high} settings, respectively.

Inference and Identification Threat Mitigation. Even sporadic location access can allow an adversary to identify and infer the user's private locations (home and work). These places can then be used as quasi-identities, which can be used to detect the user's identity from anonymised mobility traces. We report that locations released in the sessions at P_l (Equation 4.2) follow privacy rules and prevent location identify in 60% (P_{low}), 50% (P_{medium}) and 78% (P_{high}) of released location traces.

Profiling Threat Mitigation. This requires protection of sensitive behaviour profiling such as health clinics, religious places, shopping habits, etc. Compared to the first dataset, instead of the user's private places, places that the adversary could use to create the user's profile were observed in the mobility traces from the second dataset. Figure 5.6 reports that in the released apps sessions profiling threat increases with more relaxed privacy rule (P_{low}= 20%, P_{medium} = 50% and P_{high} = 90%) since this discloses more information related to the user's private locations $\subseteq$ (P_l).

To summarise, it is suggested from the aforementioned results that `PL-Protector` mitigates the fundamental location privacy threats without affecting the functionality of location-based apps.

2. OS's Middleware Layer Threats

`Android`'s security model considers all apps as potentially malicious and, therefore, runs each app in its own process, known as process isolation, and accesses its own files by default. This security practice protects apps with sensitive information from malware and potential attacks to system resources. Despite this process isolation mechanism, apps can optionally communicate via inter-message passing, or inter-process communication, which can become an attack vector for location privacy violation. Further, location data can be stolen by eavesdroppers and permissions can

Table 5.6 The OS's middleware layer threats and mitigation

Potential Threats	**Attack Vulnerability Type**	**Potential Mitigation by PL-Protector**
Over-privileged app	An *app* forces user to grant a number of permissions, including location access, at runtime to provide its services, which does not require the user's location information.	`PL-Protector` provides truncated or obfuscated locations to such over-privileged apps.
Malicious app	A malicious *app* could send a malicious *Service* that intercepts a location call (an Intent) meant for a legitimate *Service*.	`PL-Protector` blocks continuous background location monitoring of private places; hence, restricting all the installed apps to foreground location access. This mitigates unauthorised access and exposure of user's sensitive locations to such *malicious* apps.
Eavesdropping attack	An eavesdropper can silently read the contents of a location call (a *Broadcast Intent*) without interrupting it.	Since `PL-Protector` restricts apps to foreground location access, passive eavesdropping attacks are unsuccessful while the user is at private places.

be accidentally transferred between apps or third-party services. If a developer sends location data to the unauthorised recipient (intentionally or unintentionally), this can result in sensitive mobility data leakage and scale the location privacy threats mentioned above. However, `PL-Protector`'s LPPMs when implemented as a middleware on `Android` platform can prevent such unintentional exposure to the user's mobility and private locations. Through `PL-Protector`'s privacy preserving on-device location calculation mechanism, we isolate the user's private and public locations and implement enhanced permission controls; to enable robust and efficient source (OS) to sink (apps) flow control; and to protect the user's sensitive mobility data before it is sent to

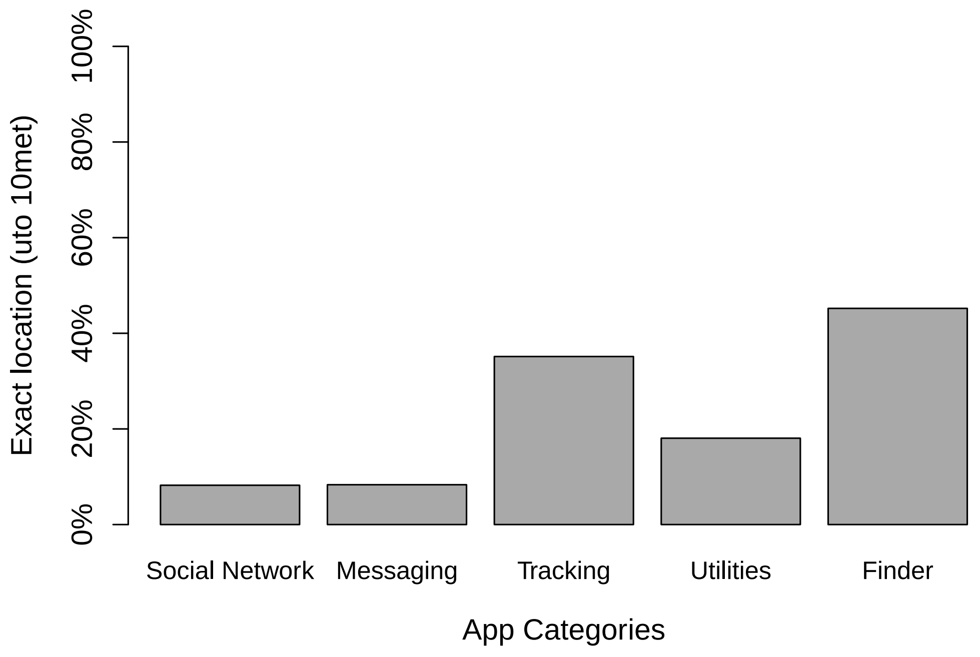


Figure 5.7 Survey results: users' preferred accuracy level to share their private locations with 5 different categories of location-based apps.

app providers. Table 5.6 presents OS's middleware layer potential threats and their mitigation, where it does contain location data and thus be vulnerable to data or privacy leakage. We address different attack vulnerabilities associated with `Android` platform; however, this can also be applied to other permission-based mobile platforms. To sum up, `PL-Protector`'s LPPMs can mitigate such inter-process communication-based attacks on user's private locations from both over-privileged and malicious apps or libraries.

5.4.3 Results of the Field Study: Users' Perspectives

We show the effectiveness and usability of `PL-Protector` through a field study. In total, we surveyed 190 smartphone users within the *University* and via social media platforms.

Users' perspectives on location privacy 89.19% of the respondents expressed that they are concerned about their location privacy, and want to know who can collect or has access to their locations. When using your mobile devices in indoor environments (e.g, shopping malls, retailer shops, etc.), 77.03% of smartphone users think businesses and apps based on indoor-sensing environment (e.g., free hotspots) can pose severe privacy threats, such as unauthorised user tracking, profiling and monitoring of user's movements; whereas, 10.81% were not sure about the potential privacy implications. However, 91.89% of users believe granting permissions to apps on their device to access continuous and precise location can result in a violation of their privacy. And 89.10% (31.08% chose option private and 58.11% chose option both private & public locations) of the respondents are more concerned about their privacy while sharing their private locations such as home and work.

Users' location sharing preferences Furthermore, we asked users to chose their preferred accuracy level while sharing their private locations with 5 different categories of location-based apps: 1. Social networking (e.g., `Facebook`), 2. Instant messaging (e.g., `Whatsapp`), 3. Sports/Fitness Tracking (e.g., Fitness/Workout), 4. Utilities (e.g., Weather, Alarm) and 5. Finder (e.g., PoI-search, Geo-search). Figure 5.7 indicates that, according to smartphone users, not all apps need continuous access to their locations. Out of the 5 given location-based app categories, users where more relaxed to share their private location with (3) Sports and (5) Finder. However, (1) Social, (2) Messaging and (4) Utilities app categories scored the lowest scale that indicate s users are more reluctant to share their private locations with these app categories. 71.62% has already taken actions in the past to protect their location privacy, while 5.41% of users where unaware of any privacy steps or privacy-enhancing tool. Hence, 78.92% of users agreed that there is need to for better privacy controls on their devices and are willing to install a location privacy enhancing tool on their devices. However, when given an option, only 68.92% of users were willing to install a privacy-enhancing tool on their devices.

Users' choices over apps usability vs location privacy Results indicate that 82.18% of the respondents rely on location result accuracy and location-based app functionality when they are anywhere outside or in unknown places. Yet, only 58.11% of the respondents do not mind if the privacy enhancing tool supplies imprecise/fake locations to installed apps. This indicates that users are more likely to select the location generalisation (i.e., truncation) option over location obfuscation (i.e., transformation or fake locations) option while using PL-Protector's enhanced privacy controls.

To summarise, these results suggest that smartphone users are aware of location privacy issues, and they agree with the need for a privacy-enhancing solution in the mobile app ecosystem. This also demonstrates the potential and positive effectiveness and usability of `PL-Protector` amongst smartphone users. However, since smartphone users have not used `PL-Protector` as a privacy enhancing tool on their devices, the precise user study on `PL-Protector`'s usefulness, acceptance and benefits is still an open question. However, our research focuses on identifying location privacy-enhanced mechanisms as a potential solution, which is practical, secure and deployable in the mobile app ecosystem; hence, we do not focus on the individual user stance towards usability of `PL-Protector`. But, this can be done as part of a future-large scale user study by making `PL-Protector` freely available to be used on smartphones. The main challenge to achieving this is that `PL-Protector` has to be fully functional and compatible with all the available smartphones in the market; and this will require further enhancement in the implementation and extra investment of time and money.

5.5 Comparison with Related Work

In this section, we compare `PL-Protector` and its LPPMs with related work. The comparison with related work is summarised in Table 5.7.

Amini et al. (2011) proposed `Caché`, an `Android`-based *service*, that stores location-based data types, such as location contents, maps, geographical regions, LBS queries and the user's PoIs, on the user's device. This approach requires app developers to register with the service to access the user's location in their apps' operations; hence, it relies on app code modification. Additionally, before using such *service*-enabled apps, the user has to pre-fetch the location-content for a geographical area that he/she is interested in (via an internet connection and good bandwidth) from the same *service*. However, this approach can only work if the pre-fetched location-content is useful and correct for the current PoIs/ geographical region of the user in advance; and it can fail if such content needs to changed or updated on a regular basis. Hence, compared to ours, this caching-based LPPM comes with high communication and computation costs: it requires high on-device storage capacity; it can have an adverse effect if the pre-fetched location content changes frequently; it requires high bandwidth to download different location-based data types at runtime; it relies on app developers' willingness to modify their apps' code. Moreover, the feasibility analysis of this cache-based system indicates that the estimated on-device storage can scale up to 65 MB and more; this increase in storage estimation depends on the increase in the number of requesting apps and generated location content over time. On the contrary, `PL-Protector`'s on-device cache storage required only 136 KB of device's internal storage to run 5 different location-based apps and protect the user's 34 private places for a period of 3 months. Moreover, `PL-Protector`'s monthly estimated storage increase is significantly less; and it considers installed apps as black boxes, i.e., it is feasible without modifying installed apps.

Niu et al. (2015) in `MobiCaché` applies k-anonymity as LPPM for caching location based queries; and Zhu et al. (2013) attempts to improve `MobiCach'e` by adding dummy locations along with k-anonymity- based LPPM. Both proposals rely on an unrealistic assumptions that are based on the existence of a trusted infrastructure, i.e., TTP (Trusted Thirst Party), providing privacy preservation and handling every LBS query. Both of these cache-based proposals have been tested on simulated LBS data collection servers but neither are implemented on a mobile platform, nor on the actual app operation. Hence, this lack of deployment feasibility and usability impede the adoption of these proposals. In contrast, the results and findings from this and previous Chapters 3 and 4 signify that our approach is practical, secure and efficient for handling location-demanding apps and can be easily deployed in the current mobile app ecosystem.

Guha et al. (2012) proposed the `Koi` platform, which includes a cloud-based service and a device-based agent, demands numerous changes in the existing smartphone ecosystem. The author proposes *location matching triggers* as an alternative to the existing location API, i.e., latitude-longitude information lookups. It requires developers to use a different API to access device's location, and it implements a new comparison mechanism and a location proximity matching criteria. In short, this platform works at a high level of abstraction that aims to replace most commonly used location API to a new location matching-based API. Adoption of this approach results in developing the current location-based apps in a new way; hence, severely affecting existing mobile platforms functionality, compatibility and interoperability.

Fawaz and Shin (2014) proposed a location-privacy framework, as an `Android`-based service, that neither relies on the adaptation of the app code modification nor on the existence of theoretically trusted infrastructure. It applies *indistinguishability* (i.e., adding noise to user's PoIs, locations and movement patterns) that corresponds to a generalised version of the well-known privacy notion called *differential privacy*, which is mainly deployed and tested as a privacy-preserving data mining technique on the data collection servers since it places heavy computation complexity. However, this framework does not control or minimise the wireless and location data that is shared with the location providers or any third-party. The author has implemented the *indistinguishability*- based LPPM and incorporated the mock location API provided by `Android` platform to generate *dummy locations* and obfuscate the user's PoIs.

Table 5.7 Comparison with related work

Related work	**Proposal Type**	**Privacy Techniques**	**Limitations**
Amini et al. (2011)	On-device Service	Caching	Since it relies on pre-fetched location contents from a remote server at run-time, its QoS and usability both are lost or severely degraded with constant mobility and no internet connection.
Niu et al. (2015) & Zhu et al. (2013)	Trusted Infrastructure	TTP-based Caching	It lacks deployment feasibility since it relies on TTP remote servers to handle every LBS query for privacy preservation.
Guha et al. (2012)	Cloud-based Location API	Cryptographic Protocol-based location triggers.	It suffers from high level of abstractions and demands extreme changes in current mobile app ecosystem. Hence, it lacks user acceptability and deployment feasibility.
Fawaz and Shin (2014)	On-device Service	Indistinguishability & Dummy Locations	It relies on heavy PETs, i.e., indistinguishability & dummy locations, to generate PoIs and maintain mobility histograms that follow the multinomial distribution for location privacy preservation. This increases computation complexity and performance overheads over time and with constant mobility of the user.
`PL-Protector`	On-device Service	Caching, Obfuscation (Truncation and Transformation)	Relies on surrounding network infrastructure availability for on-device location calculation.

Hence, when installed, the user will have to keep the mock location settings enabled in order to use this service on the device. Additionally, this *indistinguishability-*

based LPPM maintains the probability distribution of continuously changing PoI histograms that follow multinomial distribution for privacy preservation. And that can drastically increase the computation complexity on the device hardware/OS, reduce location-based apps functionality and overall QoS over time. Moreover, to generate the user's PoIs as well dummy locations to add noise, this framework relies on `Android`'s built-in `locationManager` class; hence, the user must keep the location services 'ON' to communicate with the location provider. It is a known fact that continuous communication and running of location services in the background can drain the device's battery quickly; whereas, WiFi-based localisation promises better usability and battery life (Wang et al., 2016). In contrast, `PL-Protector` can compute the user's location within the device using beacons in a privacy-preserving manner; hence, it minimises interactions via location APIs with the location provider and potentially improves the user's device battery performance.

5.6 Summary

To summarise, in Chapter 4, through the implementation of our middleware, `PL-Protector`, we proved the deployment feasibility of a new series of privacy controls on a mobile platform to prevent private location disclosure during the formation of LBS queries. To strengthen this further, in this chapter, we comprehensively present and describe our evaluation methods, experiments and results in terms of `PL-Protector`'s performance, privacy and security analysis. We presented the inclusive results and finding in terms of usability and efficiency of `PL-Protector` while interacting with real apps in real-time. We also present security analysis based on the threat model to test its compliance with the three privacy settings and achieved privacy guarantees. We assessed the potential effectiveness, acceptability and usability of `PL-Protector` through a field study on smartphone users' perspectives. Finally, we presented a comparison between `PL-Protector` and related work.

Chapter 6

Conclusions and Future Work

In this thesis, we study the fundamental risks and threats, which are expected to rise with the emergence of the IoT, smart cities or other similar paradigms, concerning both the security and privacy of mobile users. With the focus on users' location data, we analysed the risks and studied threats to users' location privacy in the existing mobile app ecosystem. We considered two main entities: smartphone apps and LBS providers, who have capability of posing location privacy threats. We aimed to determine an effective approach/mechanism to make privacy an integral part of the design of location-based mobile applications and environments. To test our research hypothesis statement, we adopted PbD approach throughout every stage of this research from the systemic literature review to the model development and its evaluation. We also highlighted a number of shortcomings in the state-of-the-art in order to overcome the fundamental location privacy threats such as tracking, identification/inference and profiling. We then proposed a new design principle and privacy policy recommendation in the form of a privacy-preserving model for location-based mobile apps and environments. Our proposed solution includes a platform-independent theoretical model, `LP-Cache`, and its prototype implementation on `Android` platform, `PL-Protector` (i.e., a proof-of-concept) that benefit both end users and service providers.

6.1 Thesis Contributions

In conclusion, this thesis presented three proposals, the `3-Layer Classification`, `LP-Cache`, and `PL-Protector`, that contribute to a practical, theoretically sound, and usable location privacy-preserving solution. Our `3-Layer Classification` intended to guide researchers working towards achieving privacy-enhanced location-based applications: by providing a systemic analysis of the state-of-the-art; by presenting a holistic picture of research gaps, methods, implications; and by assessing them against the satisfaction of the privacy properties. Furthermore, through both proposals `LP-Cache` and `PL-Protector`, we deliver a new privacy-enhancing solution that forces the mobile app ecosystem to make location data usage patterns explicit and maintains the balance between the location privacy and service utility. To our knowledge, our research provided a first working prototype that perused new design principles and policy recommendations to secure the computation and transmission of users' location data within the mobile app ecosystem (Patel and Palomar, 2016a,b, 2017).The main research achievements result in the minimisation of wireless AP data collection and prevention of the user information disclosure via generated LBS queries (e.g., PoIs and nearest neighbors). `PL-Protector`'s implementation and evaluation in terms of performance and security evidence the significance of `LP-Cache`'s three main design goals: 1. Data Minimisation, 2. Practical and Usable Privacy, and 3. Efficiency.

Data Minimisation

Our proposed LPPM-1, *On-device Cache-based Location Calculation*, minimises the interaction and location data collection from wireless access points, content distributors, and location providers by calculating the user's sensitive locations within the device. Whereas, the implementation of LLPM-2, *Personalised Location Permissions*, proved the effectiveness of the new series of privacy controls on `Android` platform. Based on the pre-defined user preferences, `PL-Protector`'s LLPM-2 applied privacy rules on his/her private locations before releasing it to the service providers. Hence, the provision of these settings prevented private location disclosure during the formation of LBS queries and allowed the user to manage each app and private place distinctly. This also mitigated severe location privacy threats and limits the user's sensitive information, which could be leaked in case of a security breach or the service provider's servers being compromised. Thus, this proved that our model successfully limits the collection and usage of both WiFi and location data in the mobile app ecosystem.

Practical and Usable Privacy

`PL-Protector`'s successful implementation as a middleware on the `Android`-based system delivered practical privacy properties guaranteed to mitigate fundamental location privacy threats: user tracking, inference or identification, and profiling. `PL-Protector` ensured that adversaries cannot infer or identify the user's private locations at any given time or session, eliminating the tracking and identification threat. Whereas, `PL-Protector` made the user's mobility or movement pattern dispersed in a given set of location/mobility traces, thus, mitigating profiling threat. Along with this, `PL-Protector`'s LPPMs (1 & 2) also significantly mitigated potential threats to the user's location data in the data link layer and the OS's middleware layer. Moreover, the performance results confirmed that `PL-Protector` does not affect app usability or QoS since it only required modification of `Android` platform, which can be done by rooting the device without any app code modification. The inclusive evaluation results confirmed that the instrumentation of `PL-Protector` enabled robust and efficient source (OS) to sink (3rd party app provider) data and flow control on `Android` platform. Since our middleware (PL-Protector) is fully compatible with all the involved entities, if the stakeholder, `Android` platform provider (or other mobile platform providers), agrees to willingly include it in the OS, then the rooting is no longer needed and it can become a standard process.

In addition, findings from the field study, which was conducted to analyse users' perspectives towards location privacy, demonstrated the potential and positive acceptability and usability of `PL-Protector` amongst mobile users. The outcome of this study indicated that smartphone users are aware of location privacy issues; they agree with the need for a privacy-enhancing solution in the system; and they are willing to use a location privacy-enhancing tool on their devices. This study further supported the practical, usable, and effective aspects of `PL-Protector`'s acceptability as a location privacy-enhancing tool amongst end-users.

Efficiency

By adding a low-cost infrastructure-less indoor localisation, `PL-Protector` provided energy-efficient and privacy-enhanced LPPMs as compared to the existing mechanisms deployed in mobile platforms. The outcome of the performance and security analysis signified that `PL-Protector` handled every location call and the cached database efficiently. This confirmed that `PL-Protector` enabled the user to set distinct privacy preferences for every app and private place. These results also made it apparent that the computation intensity and overhead does not affect location-based apps' performance or the underlying OS. Moreover, the outcome of the privacy and data leakage test indicated that service providers are unable to infer the device's exact location without getting prior user's consent. Further, `PL-Protector`'s LPPMs also mitigated unauthorised access to private locations and inter-process communication-based attacks on private locations from both over-privileged and malicious apps or in-app libraries. Hence, these results have further strengthened our confidence in the efficiency of `PL-Protector`; and that our solution provided users with location privacy at acceptable overheads.

To sum up, by following PbD approach, we proposed a model, which can evidently be incorporated into existing mobile platforms without needing any changes in the ecosystem, to preserve users' location privacy. This model determined a client-side service (i.e., on-device middleware), which neither relies on the theoretical assumptions (e.g., third-party infrastructure) nor on the trustworthiness of service providers. The proposed model scaled better than other cache-based or mobile platform-based approaches and is resistant to service providers' distrust and curious behaviors. By achieving data minimisation in the current location calculation process and reducing LBS related information exchanged and stored, our model protected both the end-user privacy and service providers' liability in case of a security breach or their servers being compromised.

6.2 Future Research Directions

There are a few related open research questions that deserve exploration to enhance the development, deployment, and evaluation of our proposals, the model (`LP-Cache`) and prototype (`PL-Protector`). In this section, we present two of the research questions that have not been addressed: 1) scalability and 2) user perceived QoS.

Scalability

Our proposed model relies on the availability of the surrounding network infrastructure for the on-device location calculation mechanism. The achieved performance results are determined with regards to the availability of network infrastructure within the selected geographical region, i.e., Birmingham, UK. The study on WiFi availability and feasibility was conducted in 34 selected places within the city; but obtained results can be assumed for other similar places (i.e., cities and countries). We argue that the privacy issues will be high in the regions that have high internet connectivity (wireless networks), hence, our proposal's performance will be better in such regions. However, further investigation and experimentation are needed to account for verifying this argument. Since our results can be extrapolated, `PL-Protector`'s implementation and evaluation decision are limited to `Android` platform and WiFi data. But, the model, `LP-Cache`, was developed as platform independent and can be similarly tested on other permission-based mobile platforms. To verify this, our prototype could be extended to other mobile platforms (e.g., `iOS`) and wireless data (e.g., cellular network) can be incorporated. We have included five categories of location-based apps; however, it would be interesting to assess the effects of our model when these apps communicate with IoT devices or sensors and other connected appliances at the user's private places.

User Perceived Quality of Service

The term "user-perceived" QoS involve system attributes such as ease of use, quantifying rate and tolerance of user errors, minimising events of user errors, and so on (Dzida et al., 1978). Our evaluation methodology suffers an inherent limitation, we made an intentional decision to exclude the requirement of user feedback regarding `PL-Protector`'s usability performance. Our research attempts to improve privacy mechanism objectively to guaranty location privacy for mobile users. It does not focus on individual user stance towards usability of `PL-Protector`. Moreover, user satisfaction survey would require additional research work on other usability fixes in the developed proof-of-concept. This would also need additional funding to compensate the involved participants. However, to achieve a practical user experience, we have considered existing surveys that studies individuals' privacy preferences and behaviour. In the future, we intend to conduct a thorough user study to determine comfortability of the users to accommodate `PL-Protector` in their devices. To achieve user-perceived

QoS, we will examine the user feedback to improve the privacy model, system usability, and user satisfaction. `PL-Protector` is designed in a way to minimise the frequency of notifying the user for privacy related decisions and inputs. However, this will need a well-defined user requirement criteria to assess up to what extent `PL-Protector`'s notifications and prompts will become obtrusive to users.

Summary of publications

The findings of this thesis have resulted in 5 publications, we present a summary of each them below:

Patel and Palomar (2014) This paper presented our classification model that consists of three layers to classify all the protocols, mechanisms and interfaces covering from the application layer to the network layer. It provided a comparative analysis of the state-of-the-art approaching privacy-preserving mobile LBS applications;

Patel and Palomar (2016a) This paper introduced our platform-independent theoretical model, `LP-Cache`, for mobile apps that overcomes the shortcomings related to user privacy in the current mobile app ecosystem;

Patel and Palomar (2016b) This paper is an updated and extended version of Patel and Palomar (2016a) that includes additional results and model improvements;

Patel and Palomar (2017) This paper presented the design, deployment and evaluation of the middleware, `PL-Protector`, which implements the `LP-Cache` model. It demonstrated the implementation of `PL-Protector` on `Android` ver. 6, conducted experiments and the performance evaluation to assess delay overheads;

Patel and Palomar [2018][1] This developed manuscript presents an inclusive contribution to design, deployment and evaluation of a fully-practical middleware(`PL-Protector`), including the following: comprehensive demonstration of experiments, mitigation of location privacy threats, and results related to the performance, security and privacy achievements.

[1]Patel, A., & Palomar, E. "A Practical Cache-based Location Privacy-enhanced Middleware for Mobile Users" is developed for publication.

References

3GPP, April 2017. The 3rd Generation Partnership Project. http://www.3gpp.org/about-3gpp.

Aad, I., Niemi, V., 2010. Nrc data collection and the privacy by design principles. Proc. of PhoneSense, 41–45.

Agarwal, Y., Hall, M., 2013. ProtectMyPrivacy: detecting and mitigating privacy leaks on iOS devices using crowdsourcing. In: Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services. ACM, pp. 97–110.

Almuhimedi, H., Schaub, F., Sadeh, N., Adjerid, I., Acquisti, A., Gluck, J., Cranor, L., Agarwal, Y., 2015. Your Location has been Shared 5,398 Times! A Field Study on Mobile App Privacy Nudging. In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI). ACM, pp. 787–796.

Amini, S., Lindqvist, J., Hong, J., Lin, J., Toch, E., Sadeh, N., 2011. Caché: caching location-enhanced content to improve user privacy. In: Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services. ACM, pp. 197–210.

Andrés, M. E., Bordenabe, N. E., Chatzikokolakis, K., Palamidessi, C., 2013. Geo-indistinguishability: Differential privacy for location-based systems. In: Proceedings of the 13 ACM SIGSAC Conference on Computer & Communications Security (CCS). ACM, pp. 901–914.

Andrienko, G., Gkoulalas-Divanis, A., Gruteser, M., Kopp, C., Liebig, T., Rechert, K., 2013. Report from dagstuhl: the liberation of mobile location data and its implications for privacy research. ACM SIGMOBILE Mobile Computing and Communications Review 17 (2), 7–18.

Android, March 2016. http://developer.android.com/guide /topics/data/ data- storage.html.

Android, March 2016. Android Developer Reference. http://developer.android.com/reference/.

Android, July 2017. https://developer.android.com/training/articles/security-tips.html.

Android, July 2017. Android Security Overview. http://source.android.com/devices/tech/security/index.html.

Android Studio, August 2017. Android studio: The official ide for android. https://developer.android.com/studio/index.html.

Android Things, July 2017. https://developer.android.com/things/hardware/index.html.

Anonymizer, March 2016. http://www.anonymizer.com/.

Apple, August 2016. https://developer.apple.com/app-store/review/guidelines/.

Apple, August 2016. iOS Security. http://images.apple.com/ipad/business /docs/iOS_Security_Feb14.pdf.

Backes, M., Bugiel, S., Derr, E., McDaniel, P. D., Octeau, D., Weisgerber, S., 2016. On demystifying the android application framework: Re-visiting android permission specification analysis. In: USENIX Security Symposium. pp. 1101–1118.

Bell, S., Jung, W. R., Krishnakumar, V., 2010. WiFi-based enhanced positioning systems: accuracy through mapping, calibration, and classification. In: Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness. ACM, pp. 3–9.

Beresford, A. R., Rice, A., Skehin, N., Sohan, R., 2011. Mockdroid: trading privacy for application functionality on smartphones. In: Proceedings of the 12th Workshop on Mobile Computing Systems and Applications. ACM, pp. 49–54.

Beresford, A. R., Stajano, F., 2004. Mix zones: User privacy in location-aware services. In: Proceedings of the 2nd IEEE Annual Conference on Pervasive Computing and Communications Workshops. IEEE, pp. 127–131.

Bettini, C., Riboni, D., 2015. Privacy protection in pervasive systems: State of the art and technical challenges. Pervasive and Mobile Computing 17, 159–174.

Bettini, C., Wang, X. S., Jajodia, S., 2005. Protecting privacy against location-based personal identification. In: Workshop on Secure Data Management. Springer, pp. 185–199.

Bugiel, S., Heuser, S., Sadeghi, A.-R., 2013. Flexible and fine-grained mandatory access control on android for diverse security and privacy policies. In: Usenix security. pp. 131–146.

Burguera, I., Zurutuza, U., Nadjm-Tehrani, S., 2011. Crowdroid: behavior-based malware detection system for android. In: Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices. ACM, pp. 15–26.

Carroll, J. M., 2000. Making use: scenario-based design of human-computer interactions. MIT press.

Chaum, D., 1988. The dining cryptographers problem: Unconditional sender and recipient untraceability. Journal of cryptology 1 (1), 65–75.

Chen, X., Pang, J., 2012. Measuring query privacy in location-based services. In: Proceedings of the 2nd ACM Conference on Data and Application Security and Privacy. ACM, pp. 49–60.

Chow, C.-Y., Mokbel, M. F., 2009. Privacy in location-based services: a system architecture perspective. Sigspatial Special 1 (2), 23–27.

Cranor, L. F., Sadeh, N., 2013. A shortage of privacy engineers. IEEE Security & Privacy (2), 77–79.

Damiani, M. L., 2011. Third party geolocation services in LBS: Privacy requirements and research issues. Trans. on Data Privacy 4 (2), 55–72.

DB Browser, August 2017. Db browser for sqlite. http://sqlitebrowser.org/.

Doty, N., Wilde, E., 2010. Geolocation privacy and application platforms. In: Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS. ACM, pp. 65–69.

Duckham, M., Kulik, L., 2006. Location privacy and location-aware computing. Dynamic & mobile GIS: investigating change in space and time 3, 35–51.

Dzida, W., Herda, S., Itzfeldt, W. D., 1978. User-perceived quality of interactive systems. IEEE Transactions on Software Engineering (4), 270–276.

Egele, M., Kruegel, C., Kirda, E., Vigna, G., 2011. PiOS: Detecting Privacy Leaks in iOS Applications. In: NDSS.

Enck, W., 2011. Defending users against smartphone apps: Techniques and future directions. In: Proceedings of the International Conference on Information Systems Security. Springer, pp. 49–70.

Enck, W., Gilbert, P., Han, S., Tendulkar, V., Chun, B.-G., Cox, L. P., Jung, J., McDaniel, P., Sheth, A. N., 2014. TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. ACM Transactions on Computer Systems (TOCS) 32 (2), 5.

Enck, W., Ongtang, M., McDaniel, P., 2009. On lightweight mobile phone application certification. In: Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS). ACM, pp. 235–245.

Erdin, E., Zachor, C., Gunes, M. H., 2015. How to find hidden users: A survey of attacks on anonymity networks. IEEE Communications Surveys & Tutorials 17 (4), 2296–2316.

Ericsson, July 2017. https://www.ericsson.com/res/docs/2016/ericsson-mobility-report-2016.pdf.

European Commission, 2016. Protection of personal data. Http://ec.europa.eu/justice/data-protection/.

European Commission, July 2017. https://ec.europa.eu/digital-single-market/en/policies/improving-connectivity-and-access.

Fawaz, K., Feng, H., Shin, K. G., 2015. Anatomization and protection of mobile apps' location privacy threats. In: 24th USENIX Security Symposium (USENIX Security 15). USENIX Association, pp. 753–768.

Fawaz, K., Shin, K. G., 2014. Location Privacy Protection for Smartphone Users. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, pp. 239–250.

Felt, A. P., Egelman, S., Wagner, D., 2012. I've got 99 problems, but vibration ain't one: a survey of smartphone users' concerns. In: Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices. ACM, pp. 33–44.

Fernandes, E., Jung, J., Prakash, A., 2016a. Security analysis of emerging smart home applications. In: Security and Privacy (SP), 2016 IEEE Symposium on. IEEE, pp. 636–654.

Fernandes, E., Paupore, J., Rahmati, A., Simionato, D., Conti, M., Prakash, A., 2016b. Flowfence: Practical data protection for emerging iot application frameworks. In: USENIX Security Symposium. pp. 531–548.

Fernandes, E., Rahmati, A., Jung, J., Prakash, A., 2017. Security implications of permission models in smart-home application frameworks. IEEE Security & Privacy 15 (2), 24–30.

Freudiger, J., Shokri, R., Hubaux, J.-P., 2011. Evaluating the privacy risk of location-based services. In: Financial Cryptography. Vol. 7035. Springer, pp. 31–46.

Gartner, July 2015. https://www.gartner.com/newsroom/id/3115517.

Gartner, July 2017a. https://www.gartner.com/newsroom/id/3598917.

Gartner, July 2017b. http://www.gartner.com/newsroom/id/3725117.

Ghinita, G., 2013. Privacy for location-based services. Synthesis Lectures on Information Security, Privacy, & Trust 4 (1), 1–85.

Gibler, C., Crussell, J., Erickson, J., Chen, H., 2012. AndroidLeaks: Automatically detecting potential privacy leaks in Android applications on a large scale. In: Trust and Trustworthy Computing. Springer, pp. 291–307.

Google Location Service, March 2016. https://support .google.com/gmm/answer/1646140?hl=en-GB.

Google Weave, Feb 2017. Weave. https://developers.google.com/weave/.

Gov. UK, July 2017. UK Digital Strategy. https://www.gov.uk/government/publications/uk-digital-strategy.

Grace, M. C., Zhou, W., Jiang, X., Sadeghi, A.-R., 2012. Unsafe exposure analysis of mobile in-app advertisements. In: Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks. ACM, pp. 101–112.

Greenstein, B., McCoy, D., Pang, J., Kohno, T., Seshan, S., Wetherall, D., 2008. Improving wireless privacy with an identifier-free link layer protocol. In: Proceedings of the 6th international conference on Mobile systems, applications, and services. ACM, pp. 40–53.

Gruteser, M., Grunwald, D., 2003. Anonymous usage of location-based services through spatial and temporal cloaking. In: Proceedings of the 1st international conference on Mobile systems, applications and services. ACM, pp. 31–42.

Guha, S., Cheng, B., Francis, P., 2011. Privad: Practical privacy in online advertising. In: USENIX conference on Networked systems design and implementation. pp. 169–182.

Guha, S., Jain, M., Padmanabhan, V. N., 2012. Koi: A location-privacy platform for smartphone apps. In: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. USENIX Association, pp. 14–14.

Gupta, R., Rao, U. P., 2017. An exploration to location based service and its privacy preserving techniques: A survey. Wireless Personal Communications 96 (2), 1973–2007.

Gürses, S., del Alamo, J. M., 2016. Privacy engineering: Shaping an emerging field of research and practice. IEEE Security & Privacy 14 (2), 40–46.

Haddadi, H., Hui, P., Brown, I., 2010. MobiAd: private and scalable mobile advertising. In: Proceedings of the fifth ACM international workshop on Mobility in the evolving internet architecture. ACM, pp. 33–38.

Hellman, E., 2013. Android programming: Pushing the limits. John Wiley & Sons.

Hornyack, P., Han, S., Jung, J., Schechter, S., Wetherall, D., 2011. These aren't the droids you're looking for: Retrofitting android to protect data from imperious applications. In: Proceedings of the 18th ACM conference on Computer and communications security. ACM, pp. 639–652.

Hu, H., Xu, J., Ren, C., Choi, B., 2011. Processing private queries over untrusted data cloud through privacy homomorphism. In: Data Engineering (ICDE), 2011 IEEE 27th International Conference on. IEEE, pp. 601–612.

ICO, April 2018. Information commisioner's office: Privacy by design. https://ico.org.uk/for-organisations/guide-to-data-protection/privacy-by-design/.

IETF, March 2017. Geographic Location Privacy. http://datatracker.ietf.org/wg/geopriv/charter/.

Jain, A. K., Shanbhag, D., 2012. Addressing security and privacy risks in mobile applications. IT Professional 14 (5), 0028–33.

Jan, O., Horowitz, A. J., Peng, Z.-R., 2000. Using global positioning system data to understand variations in path choice. Transportation Research Record: Journal of the Transportation Research Board 1725 (1), 37–44.

Jeon, J., Micinski, K. K., Vaughan, J. A., Fogel, A., Reddy, N., Foster, J. S., Millstein, T., 2012. Dr. Android and Mr. Hide: fine-grained permissions in android applications. In: Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices. ACM, pp. 3–14.

Jones, T. S., Richey, R. C., 2000. Rapid prototyping methodology in action: A developmental study. Educational Technology Research and Development 48 (2), 63–80.

Kaemarungsi, K., Krishnamurthy, P., 2004. Modeling of indoor positioning systems based on location fingerprinting. In: INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies. Vol. 2. IEEE, pp. 1012–1022.

Kelly, D., Raines, R., Baldwin, R., Grimaila, M., Mullins, B., 2012. Exploring extant and emerging issues in anonymous networks: A taxonomy and survey of protocols and metrics. Communications Surveys & Tutorials, IEEE 14 (2), 579–606.

Ketelaar, P. E., van Balen, M., 2018. The smartphone as your follower: The role of smartphone literacy in the relation between privacy concerns, attitude and behaviour towards phone-embedded tracking. Computers in Human Behavior 78, 174–182.

Khoshgozaran, A., Shahabi, C., 2010. A taxonomy of approaches to preserve location privacy in location-based services. International Journal of Computational Science and Engineering 5 (2), 86–96.

Khoshgozaran, A., Shahabi, C., Shirani-Mehr, H., 2011. Location privacy: going beyond K-anonymity, cloaking and anonymizers. Knowledge and Information Systems 26 (3), 435–465.

Kido, H., Yanagisawa, Y., Satoh, T., 2005. An anonymous communication technique using dummies for location-based services. In: Pervasive Services, 2005. ICPS'05. Proceedings. International Conference on. IEEE, pp. 88–97.

Kim, D. H., Hightower, J., Govindan, R., Estrin, D., 2009. Discovering semantically meaningful places from pervasive RF-beacons. In: Proceedings of the 11th international conference on Ubiquitous computing. ACM, pp. 21–30.

Kim, Y. S., Tian, Y., Nguyen, L. T., Tague, P., 2014. LAPWiN: Location-aided probing for protecting user privacy in Wi-Fi networks. In: Proceedings of the 14th International Conference on Communications and Network Security (CNS). IEEE, pp. 427–435.

Krasnova, A., Neikes, M., Schwabe, P., 2016. Footprint scheduling for dining-cryptographer networks. In: International Conference on Financial Cryptography and Data Security. Springer, pp. 385–402.

Krumm, J., 2009. A survey of computational location privacy. Personal and Ubiquitous Computing 13 (6), 391–399.

Lee, B., Oh, J., Yu, H., Kim, J., 2011. Protecting location privacy using location semantics. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp. 1289–1297.

Lee, Y.-K., Chang, C.-T., Lin, Y., Cheng, Z.-H., 2014. The dark side of smartphone usage: Psychological traits, compulsive behavior and technostress. Computers in Human Behavior 31, 373–383.

Lin, D., Bertino, E., Cheng, R., Prabhakar, S., 2008. Position transformation: a location privacy protection method for moving objects. In: Proceedings of the SIGSPATIAL ACM GIS 2008 International Workshop on Security and Privacy in GIS and LBS. ACM, pp. 62–71.

Liu, B., Andersen, M. S., Schaub, F., Almuhimedi, H., Zhang, S. A., Sadeh, N., Agarwal, Y., Acquisti, A., 2016. Follow my recommendations: A personalized privacy assistant for mobile app permissions. In: Symposium on Usable Privacy and Security.

Liu, D., Gao, X., Wang, H., 2017. Location privacy breach: Apps are watching you in background. In: IEEE 37th International Conference on Distributed Computing Systems (ICDCS). IEEE, pp. 2423–2429.

Liu, X., Li, X., 2012. Privacy Preserving Techniques for Location Based Services in Mobile Networks. In: Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International. IEEE, pp. 2474–2477.

Logcat, August 2017. Android studio: Logcat command-line tool. https://developer.android.com/studio/command-line/logcat.html.

Michael, K., Clarke, R., 2013. Location and tracking of mobile devices: Überveillance stalks the streets. Computer Law & Security Review 29 (3), 216–228.

Microsoft Phone, July 2017a. https://msdn.microsoft.com/en-GB/library/windows/apps/dn764944.aspx.

Microsoft Phone, July 2017b. https://msdn.microsoft.com/en-us/library/dn756283.aspx.

Mokbel, M. F., 2007. Privacy in location-based services: State-of-the-art and research directions. In: Proceedings of the 8th International Conference on Mobile Data Management (MDM). IEEE, pp. 228–228.

Montoliu, R., Blom, J., Gatica-Perez, D., 2013. Discovering places of interest in everyday life from smartphone data. Multimedia tools and applications 62 (1), 179–207.

Muslukhov, I., Boshmaf, Y., Kuo, C., Lester, J., Beznosov, K., 2012. Understanding users' requirements for data protection in smartphones. In: Proceedings of the 28th International Conference on Data Engineering Workshops (ICDEW). IEEE, pp. 228–235.

Mylonas, A., Dritsas, S., Tsoumas, B., Gritzalis, D., 2011. Smartphone security evaluation-the malware attack case. SECRYPT 11, 25–36.

Mylonas, A., Kastania, A., Gritzalis, D., 2013. Delegate the smartphone user? security awareness in smartphone platforms. Computers & Security 34, 47–66.

Navizon, March 2016. http://www.navizon.com.

NetworkInfoIi, March 2016. http://play.google.com/store/apps/.

Nexus 6P, August 2017. https://www.google.com/nexus/6p/.

Niu, B., Li, Q., Zhu, X., Cao, G., Li, H., 2015. Enhancing Privacy through Caching in Location-Based Services. In: Proc. of IEEE INFOCOM.

Ozer, N., Conley, C., O'Connell, D. H., Gubins, T. R., Ginsburg, E., 2010. Location-based services: time for a privacy check-in. ACLU of Northern California.

Pan, X., Xu, J., Meng, X., 2012. Protecting location privacy against location-dependent attacks in mobile services. Knowledge and Data Engineering, IEEE Transactions on 24 (8), 1506–1519.

Papadopoulos, S., Bakiras, S., Papadias, D., 2010. Nearest neighbor search with strong location privacy. Proceedings of the VLDB Endowment 3 (1-2), 619–629.

Pasricha, S., Ugave, V., Anderson, C. W., Han, Q., 2015. Learnloc: A framework for smart indoor localization with embedded mobile devices. In: Proceedings of the 10th International Conference on Hardware/Software Codesign and System Synthesis. IEEE Press, pp. 37–44.

Patel, A., Palomar, E., 2014. Privacy Preservation in Location-Based Mobile Applications: Research Directions. In: Proceedings of the 9th International Conference on Availability, Reliability and Security (ARES). IEEE, pp. 227–233.

Patel, A., Palomar, E., 2016a. LP-Caché: Privacy-aware cache model for location-based apps. In: Proceedings of the 13th International Conference on Security and Cryptography (SECRYPT). pp. 183–194.

Patel, A., Palomar, E., 2016b. Protecting smartphone users' private locations through caching. In: Proceedings of the 13th International Conference on E-Business and Telecommunications. Springer, pp. 316–337.

Patel, A., Palomar, E., 2017. A middleware enforcing location privacy in mobile platform. In: Proceedings of the 14th International Conference on Trust and Privacy in Digital Business (TrustBus). Springer, pp. 32–45.

Pennekamp, J., Henze, M., Wehrle, K., 2017. A survey on the evolution of privacy enforcement on smartphones and the road ahead. Pervasive and Mobile Computing.

Pfitzmann, A., Hansen, M., 2008. Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management-a consolidated proposal for terminology. Version v0 31, 15, http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.28.pdf.

Pfitzmann, A., Köhntopp, M., 2001. Anonymity, unobservability, and pseudonymity—a proposal for terminology. In: Designing Privacy Enhancing Technologies. Springer, pp. 1–9.

Pingley, A., Zhang, N., Fu, X., Choi, H.-A., Subramaniam, S., Zhao, W., 2011. Protection of query privacy for continuous location based services. In: INFOCOM, 2011 Proceedings IEEE. IEEE, pp. 1710–1718.

Pontes, T., Vasconcelos, M., Almeida, J., Kumaraguru, P., Almeida, V., 2012. We know where you live: Privacy characterization of foursquare behavior. In: Proceedings of the 2012 ACM Conference on Ubiquitous Computing. ACM, pp. 898–905.

Pu, C.-C., Pu, C.-H., Lee, H.-J., 2011. Indoor location tracking using received signal strength indicator. INTECH Open Access Publisher.

Puttaswamy, K., Wang, S., Steinbauer, T., Agrawal, D., El Abbadi, A., Kruegel, C., Zhao, B., 2012. Preserving location privacy in geo-social applications.

Quercia, D., Leontiadis, I., McNamara, L., Mascolo, C., Crowcroft, J., 2011. Spotme if you can: Randomized responses for location obfuscation on mobile phones. In: Distributed Computing Systems (ICDCS), 2011 31st International Conference on. IEEE, pp. 363–372.

Reiter, M. K., Rubin, A. D., 1998. Crowds: Anonymity for web transactions. ACM Transactions on Information and System Security (TISSEC) 1 (1), 66–92.

Ren, J., Wu, J., 2010. Survey on anonymous communications in computer networks. Computer Communications 33 (4), 420–431.

Robusto, C. C., 1957. The cosine-haversine formula. The American Mathematical Monthly 64 (1), 38–40.

Roman, R., Zhou, J., Lopez, J., 2013. On the features and challenges of security and privacy in distributed internet of things. Computer Networks 57 (10), 2266–2279.

Rozgonjuk, D., Elhai, J., 2018. Problematic smartphone usage, emotion regulation, and social and non-social smartphone use. In: Proceedings of the Technology, Mind, and Society. ACM, p. 35.

Samsung Smartthings, Feb 2017. smartthings. http://www.samsung.com/uk/smartthings/.

Sapiezynski, P., Gatej, R., Mislove, A., Lehmann, S., 2015. Opportunities and challenges in crowdsourced wardriving. In: Proceedings of the 2015 ACM Conference on Internet Measurement Conference. ACM, pp. 267–273.

Shabtai, A., Kanonov, U., Elovici, Y., Glezer, C., Weiss, Y., 2012. “andromaly”: a behavioral malware detection framework for android devices. Journal of Intelligent Information Systems 38 (1), 161–190.

Shields, C., Levine, B. N., 2000. A protocol for anonymous communication over the internet. In: Proceedings of the 7th ACM conference on Computer and communications security. ACM, pp. 33–42.

Shih, F., Liccardi, I., Weitzner, D., 2015. Privacy tipping points in smartphones privacy preferences. In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems. ACM, pp. 807–816.

Shin, K. G., Ju, X., Chen, Z., Hu, X., 2012. Privacy protection for users of location-based services. Wireless Communications, IEEE 19 (1), 30–39.

Shklovski, I., Mainwaring, S. D., Skúladóttir, H. H., Borgthorsson, H., 2014. Leakiness and creepiness in app space: Perceptions of privacy and mobile app use. In: Proceedings of the 32nd annual ACM conference on Human factors in computing systems. ACM, pp. 2347–2356.

Shokri, R., Theodorakopoulos, G., Le Boudec, J.-Y., Hubaux, J.-P., 2011. Quantifying location privacy. In: IEEE Symposium on Security and privacy. IEEE, pp. 247–262.

Skyhook, March 2016. http://www.skyhookwireless.com/.

Skyhook, 2016. Submit a Wi-Fi Access Point. \url{http://www.skyhookwireless.com/submit-access-point}.

SQLite, August 2017. https://www.sqlite.org/.

Suikkola, V., 2010. Open exposure of telco capabilities-identification of critical success factors for location-based services in open telco. In: Wireless and Mobile Communications (ICWMC), 2010 6th International Conference on. IEEE, pp. 202–208.

Sweeney, L., 2002. Achieving k-anonymity privacy protection using generalization and suppression. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10 (05), 571–588.

Symantec, July 2017. A window into mobile device security. http://www.symantec.com/content/en/us/enterprise/white_papers/b-mobile-device-security_WP.en-us.pdf.

Toch, E., Cranshaw, J., Hankes-Drielsma, P., Springfield, J., Kelley, P. G., Cranor, L., Hong, J., Sadeh, N., 2010. Locaccino: a privacy-centric location sharing application. In: Proceedings of the 12th ACM International Conference adjunct papers on Ubiquitous Computing-Adjunct. ACM, pp. 381–382.

TOR, March 2016. http://www.torproject.org/.

tPacketcapture, March 2016. http://play.google.com/.

UWP, July 2017. https://docs.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide.

Van Kleek, M., Liccardi, I., Binns, R., Zhao, J., Weitzner, D. J., Shadbolt, N., 2017. Better the devil you know: Exposing the data sharing practices of smartphone apps. In: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems. ACM, pp. 5208–5220.

Vratonjic, N., Huguenin, K., Bindschaedler, V., Hubaux, J.-P., 2013. How others compromise your location privacy: The case of shared public IPs at hotspots. In: Privacy Enhancing Technologies. Springer, pp. 123–142.

Wang, T., Lu, K., Lu, L., Chung, S. P., Lee, W., 2013. Jekyll on ios: When benign apps become evil. In: USENIX Security Symposium. Vol. 13. pp. 559–572.

Wang, Y., Bu, Y., Jin, Q., Vasilakos, A. V., 2016. Energy-efficient localization and tracking on smartphones: Design principle and solutions. In: Proceedings of the 11th International Conference on Future Internet Technologies. ACM, pp. 29–35.

Wernke, M., Skvortsov, P., Dürr, F., Rothermel, K., 2014. A classification of location privacy attacks and approaches. Personal and Ubiquitous Computing 18 (1), 163–175.

WiEye, March 2016. Http://play.google.com/store/apps/.

Wijesekera, P., Baokar, A., Tsai, L., Reardon, J., Egelman, S., Wagner, D., Beznosov, K., 2017. The feasibility of dynamically granted permissions: Aligning mobile privacy with user preferences. In: IEEE Symposium on Security and Privacy (SP). IEEE, pp. 1077–1093.

Wireshark, March 2016. https://www.wireshark.org.

Wong, W. K., Cheung, D. W.-l., Kao, B., Mamoulis, N., 2009. Secure knn computation on encrypted databases. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of data. ACM, pp. 139–152.

Xie, J., Knijnenburg, B. P., Jin, H., 2014. Location sharing privacy preference: analysis and personalized recommendation. In: Proceedings of the 19th International Conference on Intelligent User Interfaces. ACM, pp. 189–198.

You, T.-H., Peng, W.-C., Lee, W.-C., 2007. Protecting moving trajectories with dummies. In: Mobile Data Management, 2007 International Conference on. IEEE, pp. 278–282.

Zhou, W., Zhou, Y., Jiang, X., Ning, P., 2012a. Detecting repackaged smartphone applications in third-party android marketplaces. In: Proceedings of the second ACM conference on Data and Application Security and Privacy. ACM, pp. 317–326.

Zhou, Y., Wang, Z., Zhou, W., Jiang, X., 2012b. Hey, you, get off of my market: Detecting malicious apps in official and alternative android markets. In: Proceedings of the 19th Annual Network and Distributed System Security Symposium. pp. 5–8.

Zhou, Y., Zhang, X., Jiang, X., Freeh, V. W., 2011. Taming information-stealing smartphone applications (on android). In: International conference on Trust and trustworthy computing. Springer, pp. 93–107.

Zhu, X., Chi, H., Niu, B., Zhang, W., Li, Z., Li, H., 2013. Mobicache: When k-anonymity meets cache. In: Global Communications Conference (GLOBECOM), 2013 IEEE. IEEE, pp. 820–825.

Zhuang, Y., Syed, Z., Georgy, J., El-Sheimy, N., 2015. Autonomous smartphone-based WiFi positioning system by using access points localization and crowdsourcing. Pervasive and Mobile Computing.

Appendix A

User Survey

Research Survey: Privacy-preservation in Location-based Mobile Applications

I am a final year PhD student at Birmingham City University. This study aims to investigate and contribute to enhancing location privacy preservation in the current mobile app ecosystem. Any smartphone user who downloads and use apps from app repositories (e.g., App store or Google Play) can participate in this online survey to help us understand and gather required information. There are only 10 multiple choice questions. Please spare 5 to 10 minutes to complete this survey.

Your participation is voluntary, and it is your right to unconditionally withdraw at any time and for any reason. There are no particular risks associated with your participation, and no personally identifiable information will be collected. In case of questions/comments/suggestions about the survey, please contact asma.patel@bcu.ac.uk If you agree to the above, please proceed with the survey.

1. Do you care about potential privacy threats that your mobile device can pose?
   ☐ Yes
   ☐ No

2. Do you care about who access your location information?
   ☐ Yes
   ☐ No

3. Are you concerned about your privacy when granting location permission to apps that can continuously access your precise location, even while it is running in the background? Do you think such continuous access to your precise locations can allow apps to violate your privacy?
   ☐ Yes
   ☐ No

4. Do you care about your privacy when using your mobile devices in indoor environments (e.g, shopping malls, retailer shops, etc.)? Do you think businesses based on indoor sensing environment (e.g., free hotspots) can pose severe privacy threats, such as unauthorised user tracking, profiling and monitoring of your movements?
   ☐ Yes
   ☐ No
   ☐ I don't know

5. Have you taken any step to protect your location privacy?
   ☐ Regularly change location sharing settings to OFF/ON on your devices based on the usage, e.g., ON when using a location-based app and then turn it OFF when not in use.
   ☐ Never changed location settings on your device.
   ☐ I don't know

6. Would you install an app or a tool that manages all the installed apps on your mobile device and protects your location privacy?
   ☐ Yes
   ☐ No

7. Would you mind if such location privacy enhancing tool supplies imprecise locations to installed apps?
   ☐ Yes
   ☐ No

8. When are you more concerned about your privacy? while sharing your
☐ Private locations (e.g., home, work or any other frequently visited private place)
☐ Public locations (e.g., malls, shop, or any rarely visited places)
☐ Both
☐ None

9. You rely more on location based app functionality and result accuracy, when you are
☐ At home or work
☐ Anywhere outside

10. Do you think all apps need continuous access to your precise location? (If your answer to first question is yes, please select 4th Scale (Exact Location) for all the following 5 app categories) If given an option to choose, what will be your preferred location accuracy level when sharing private locations (e.g., home, work or any other private location) with the following app categories?

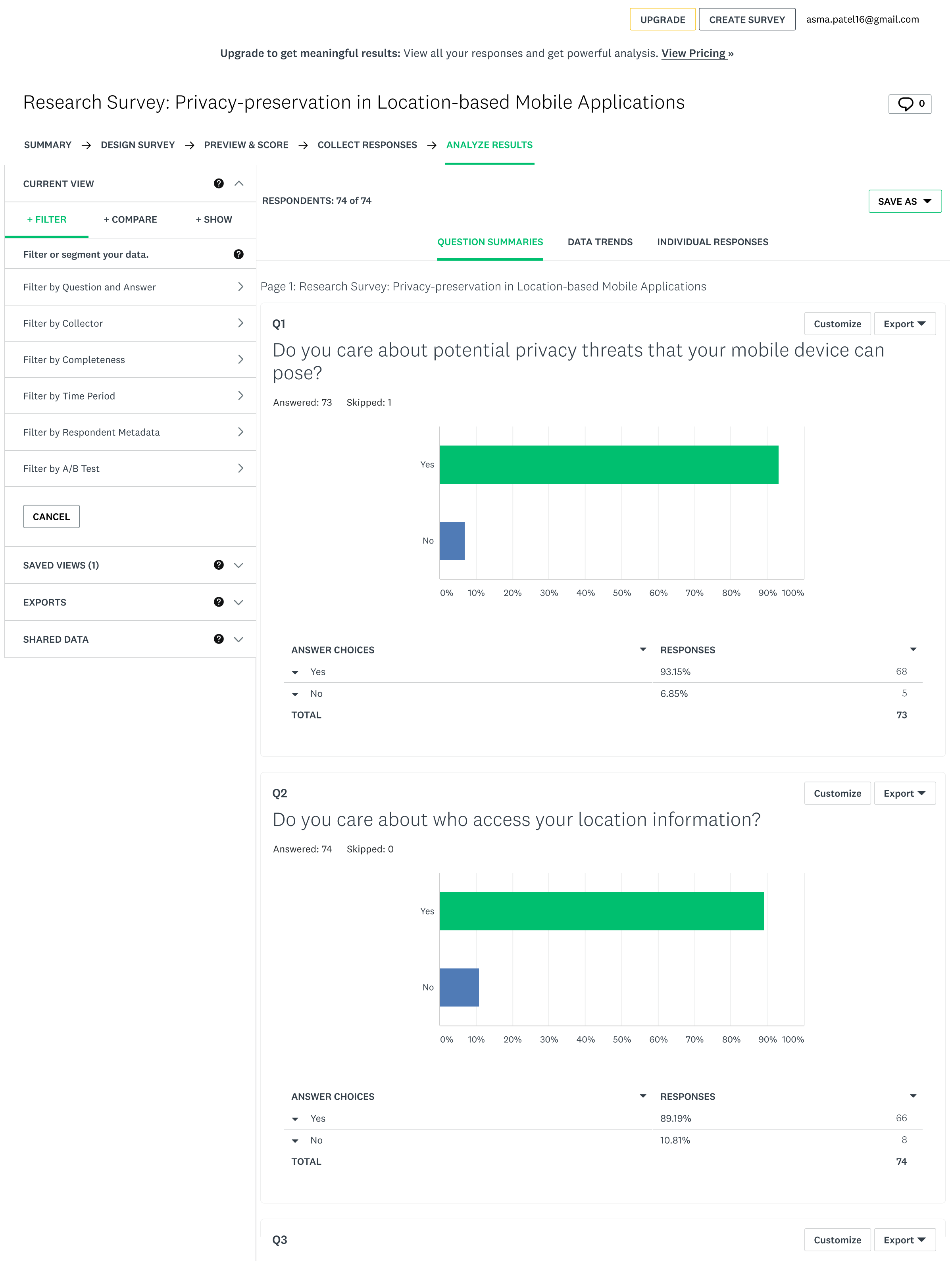
App Category	Random/Fake	City Level	Street level	Exact Location (upto 10 meters accuracy)
Social Networking (e.g., Facebook)	○	○	○	○
Instant Messaging/Chatting (e.g., Whatsapp)	○	○	○	○
Sports/Fitness tracking (e.g., Fitness)	○	○	○	○
Utilities (e.g., Weather, Alarm, etc.)	○	○	○	○
Finder (PoI Finder/Geo-search)	○	○	○	○

Appendix B

User Survey Results

SurveyMonkey Analyze - Research Survey: Privacy-preservation in Loca... https://www.surveymonkey.com/analyze/tgO4Sqo8wP4AOiA3pBuGe...

Online Survey Collector A: Results



Upgrade to get meaningful results: View all your responses and get powerful analysis. View Pricing »

Research Survey: Privacy-preservation in Location-based Mobile Applications

0
SUMMARY → DESIGN SURVEY → PREVIEW & SCORE → COLLECT RESPONSES → ANALYZE RESULTS
CURRENT VIEW
+ FILTER
+ COMPARE
+ SHOW

Filter or segment your data.

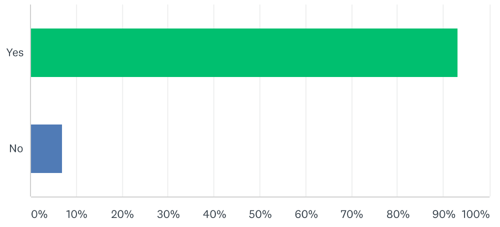
RESPONDENTS: 74 of 74

Page 1: Research Survey: Privacy-preservation in Location-based Mobile Applications

Q1

Do you care about potential privacy threats that your mobile device can pose?

Answered: 73 Skipped: 1

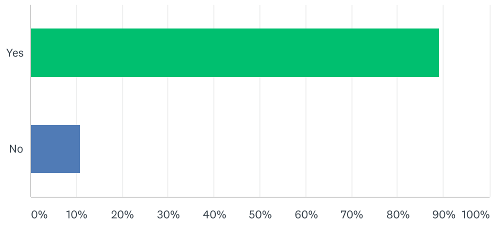


ANSWER CHOICES	RESPONSES	
Yes	93.15%	68
No	6.85%	5
TOTAL		73

Q2

Do you care about who access your location information?

Answered: 74 Skipped: 0



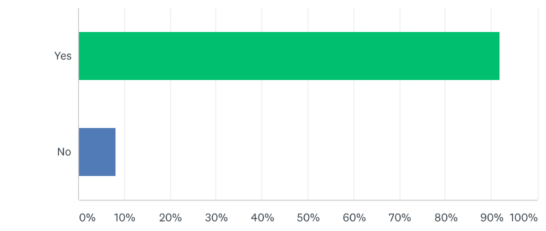
ANSWER CHOICES	RESPONSES	
Yes	89.19%	66
No	10.81%	8
TOTAL		74

Q3

Online Survey Collector A: Results

Are you concerned about your privacy when granting location permission to apps that can continuously access your precise location, even while it is running in the background? Do you think such continuous access to your precise locations can allow apps to violate your privacy?

Answered: 74 Skipped: 0



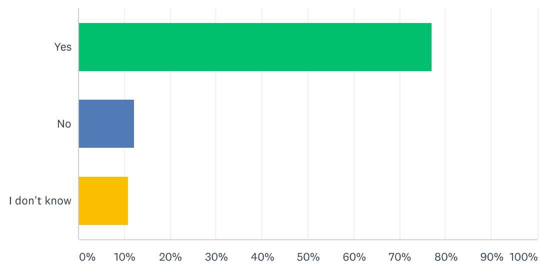
ANSWER CHOICES	RESPONSES	
Yes	91.89%	68
No	8.11%	6
TOTAL		74

Q4

Customize Export

Do you care about your privacy when using your mobile devices in indoor environments (e.g, shopping malls, retailer shops, etc.)? Do you think businesses based on indoor sensing environment (e.g., free hotspots) can pose severe privacy threats, such as unauthorised user tracking, profiling and monitoring of your movements?

Answered: 74 Skipped: 0



ANSWER CHOICES	RESPONSES	
Yes	77.03%	57
No	12.16%	9
I don't know	10.81%	8
TOTAL		74

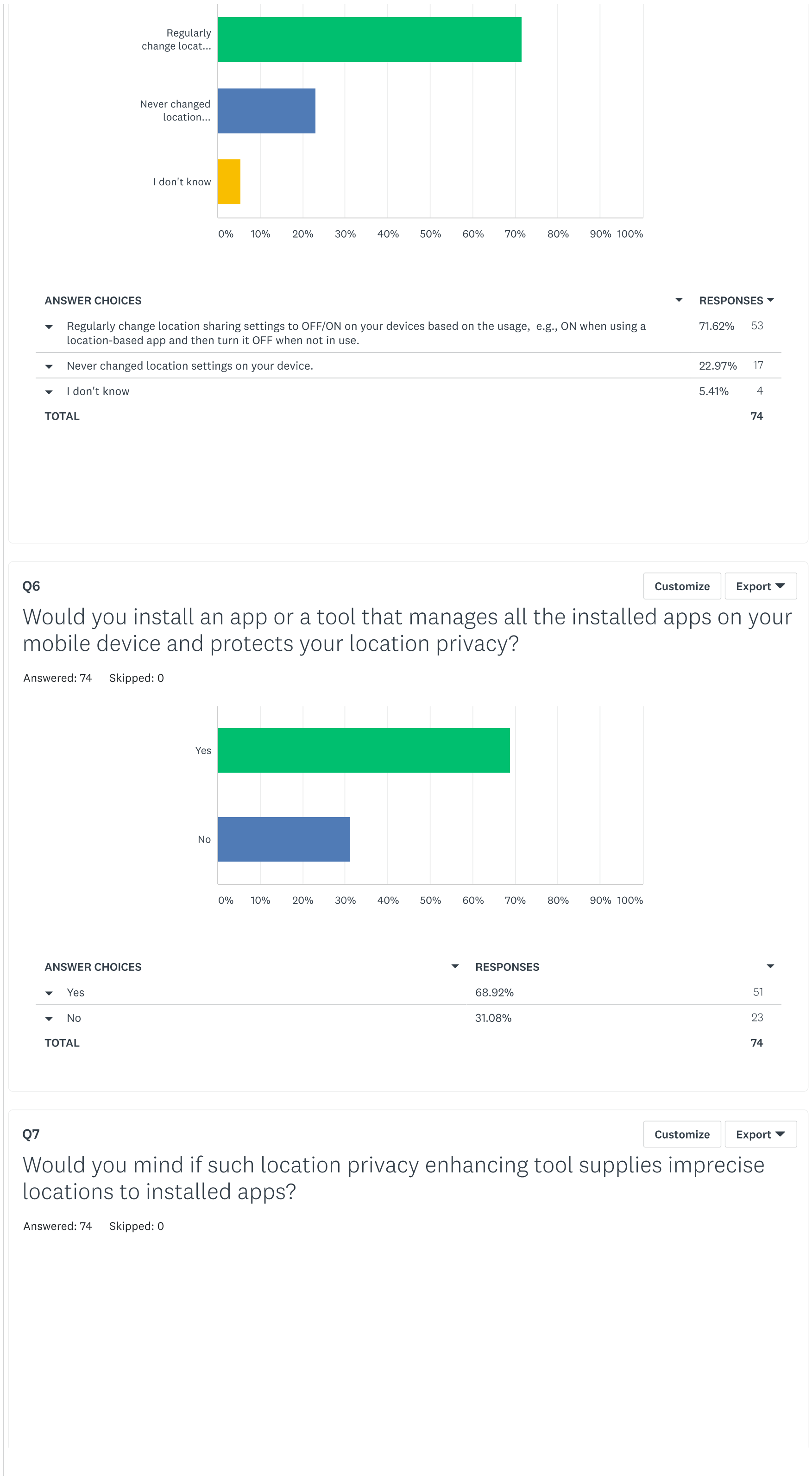
Q5

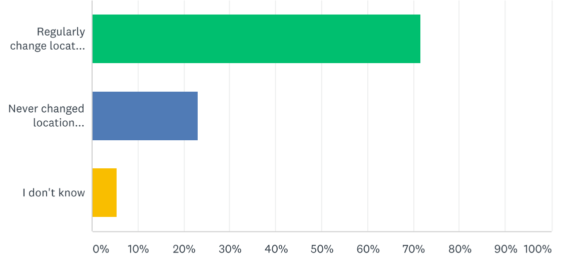
Customize Export

Have you taken any step to protect your location privacy?

Answered: 74 Skipped: 0

Online Survey Collector A: Results



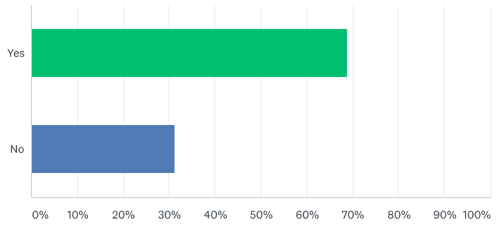


ANSWER CHOICES	RESPONSES	
Regularly change location sharing settings to OFF/ON on your devices based on the usage, e.g., ON when using a location-based app and then turn it OFF when not in use.	71.62%	53
Never changed location settings on your device.	22.97%	17
I don't know	5.41%	4
TOTAL		74

Q6 Customize Export

Would you install an app or a tool that manages all the installed apps on your mobile device and protects your location privacy?

Answered: 74 Skipped: 0



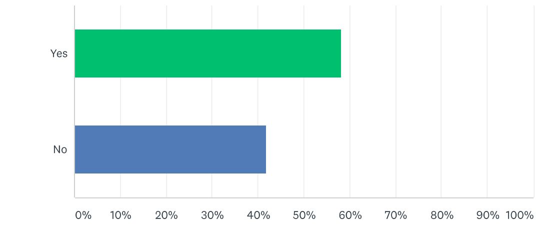
ANSWER CHOICES	RESPONSES	
Yes	68.92%	51
No	31.08%	23
TOTAL		74

Q7 Customize Export

Would you mind if such location privacy enhancing tool supplies imprecise locations to installed apps?

Answered: 74 Skipped: 0

Online Survey Collector A: Results



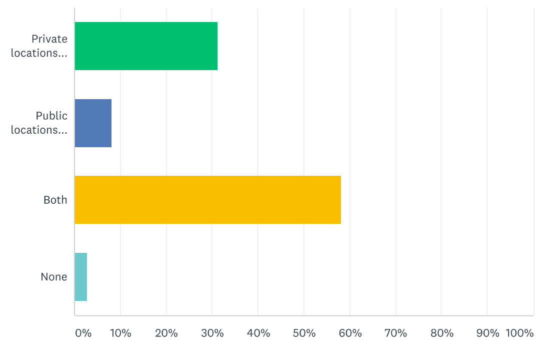
ANSWER CHOICES	RESPONSES	
Yes	58.11%	43
No	41.89%	31
TOTAL		74

Q8

Customize Export

When are you more concerned about your privacy? while sharing your

Answered: 74 Skipped: 0



ANSWER CHOICES	RESPONSES	
Private locations (e.g., home, work or any other frequently visited private place)	31.08%	23
Public locations (e.g., malls, shop, or any rarely visited places)	8.11%	6
Both	58.11%	43
None	2.70%	2
TOTAL		74

Q9

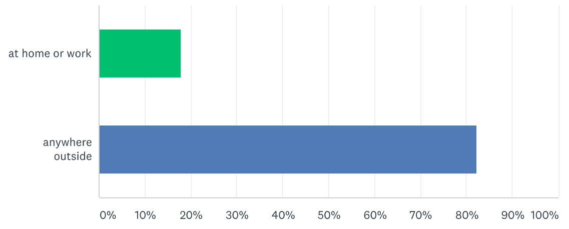
Customize Export

You rely more on location based app functionality and result accuracy, when you are

Answered: 73 Skipped: 1

SurveyMonkey Analyze - Research Survey: Privacy-preservation in Loca... https://www.surveymonkey.com/analyze/tgO4Sqo8wP4AOiA3pBuGe...

Online Survey Collector A: Results



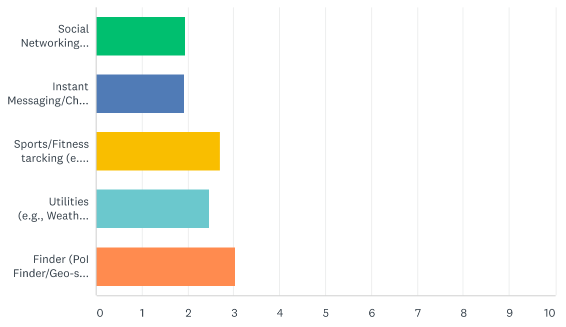
ANSWER CHOICES	RESPONSES	
at home or work	17.81%	13
anywhere outside	82.19%	60
TOTAL		73

Q10

Customize Export

Do you think all apps need continuous access to your precise location? (If your answer to first question is yes, please select 4th Scale (Exact Location) for all the following 5 app categories) If given an option to choose, what will be your preferred location accuracy level when sharing private locations (e.g., home, work or any other private location) with the following app categories?

Answered: 74 Skipped: 0



	RANDOM/FAKE	CITY LEVEL	STREET LEVEL/LOCAL AREA	EXACT LOCATION (UPTO 10 METERS ACCURACY)	TOTAL	WEIGHTED AVERAGE
Social Networking (e.g., Facebook)	34.25% 25	45.21% 33	12.33% 9	8.22% 6	73	1.95
Instant Messaging/Chatting (e.g., Whatsapp)	37.50% 27	41.67% 30	12.50% 9	8.33% 6	72	1.92
Sports/Fitness tarcking (e.g., Fitness)	24.32% 18	16.22% 12	24.32% 18	35.14% 26	74	2.70
Utilities (e.g., Weather, Alarm, etc.)	9.72% 7	51.39% 37	20.83% 15	18.06% 13	72	2.47
Finder (PoI Finder/Geo-search)	9.59% 7	21.92% 16	23.29% 17	45.21% 33	73	3.04

5 of 6 06/04/2018, 16:42

Online Survey Collector B: Results

UPGRADE CREATE SURVEY asma.patel16@gmail.com

Upgrade to get meaningful results: View all your responses and get powerful analysis. **View Pricing »**

Research Survey: Privacy-preservation in Location-based Mobile Applications 0

SUMMARY → DESIGN SURVEY → PREVIEW & SCORE → COLLECT RESPONSES → ANALYZE RESULTS

CURRENT VIEW

+ FILTER + COMPARE + SHOW

Filter or segment your data.

Filter by Question and Answer

Filter by Collector

Filter by Completeness

Filter by Time Period

Filter by Respondent Metadata

Filter by A/B Test

CANCEL

SAVED VIEWS (1)

EXPORTS

SHARED DATA

RESPONDENTS: 71 of 71

SAVE AS

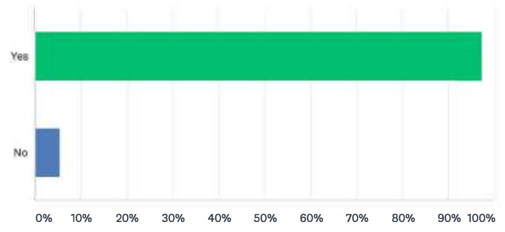
QUESTION SUMMARIES DATA TRENDS INDIVIDUAL RESPONSES

Page 1: Research Survey: Privacy-preservation in Location-based Mobile Applications

Q1 Customize Export

Do you care about potential privacy threats that your mobile device can pose?

Answered: 71 Skipped: 0

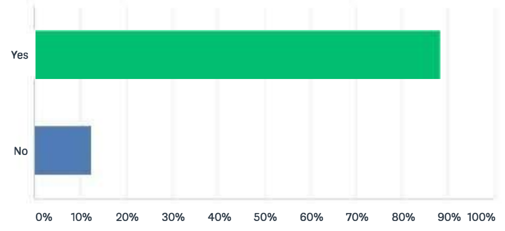


ANSWER CHOICES	RESPONSES	
Yes	95.77%	68
No	4.22%	3
TOTAL		71

Q2 Customize Export

Do you care about who access your location information?

Answered: 71 Skipped: 0



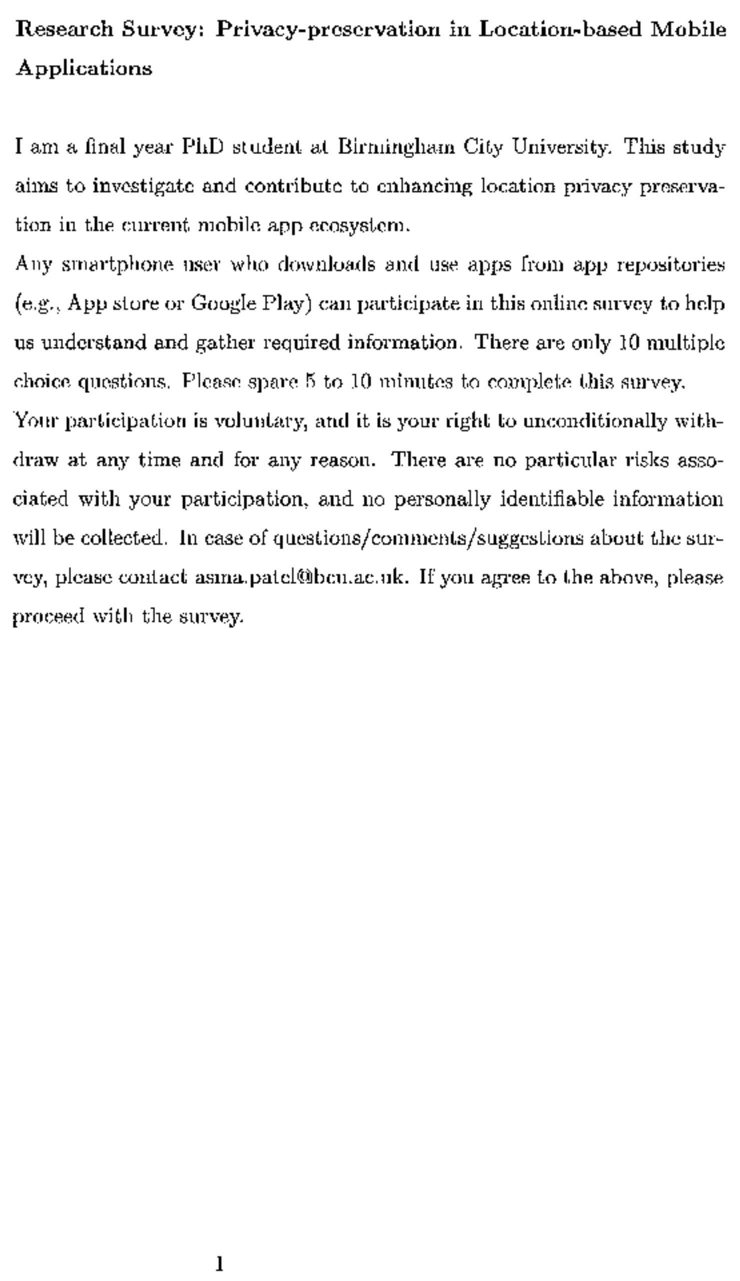
ANSWER CHOICES	RESPONSES	
Yes	88.73%	63
No	11.26%	8
TOTAL		71

Q3 Customize Export

A sample of the paper survey response

(Collected 45 responses in total)

Survey Respondent - No. 5.



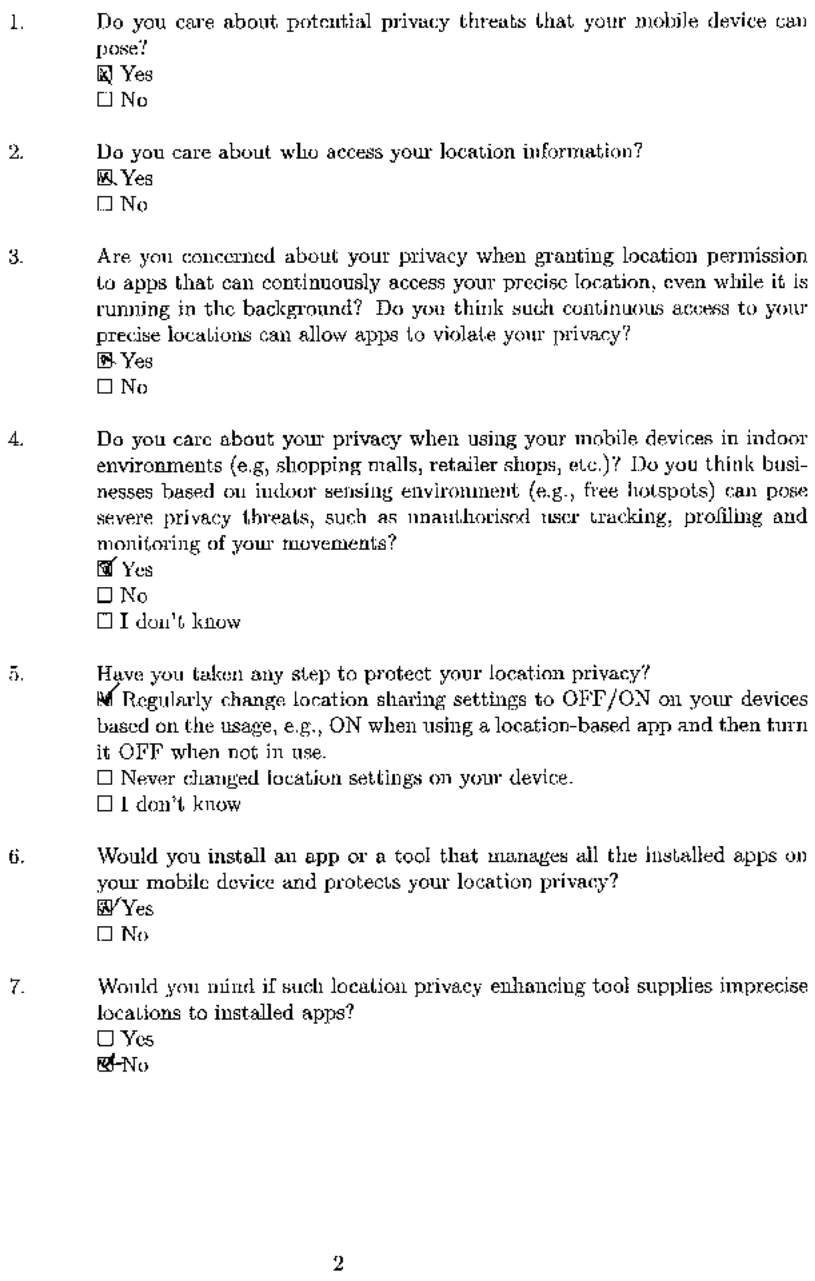
Research Survey: Privacy-preservation in Location-based Mobile Applications

I am a final year PhD student at Birmingham City University. This study aims to investigate and contribute to enhancing location privacy preservation in the current mobile app ecosystem.
Any smartphone user who downloads and use apps from app repositories (e.g., App store or Google Play) can participate in this online survey to help us understand and gather required information. There are only 10 multiple choice questions. Please spare 5 to 10 minutes to complete this survey.
Your participation is voluntary, and it is your right to unconditionally withdraw at any time and for any reason. There are no particular risks associated with your participation, and no personally identifiable information will be collected. In case of questions/comments/suggestions about the survey, please contact asma.patel@bcu.ac.uk. If you agree to the above, please proceed with the survey.

1

A sample of the paper survey response

(Collected 45 responses in total)



1. Do you care about potential privacy threats that your mobile device can pose?
   [✓ Yes]
   ☐ No

2. Do you care about who access your location information?
   [✓ Yes]
   ☐ No

3. Are you concerned about your privacy when granting location permission to apps that can continuously access your precise location, even while it is running in the background? Do you think such continuous access to your precise locations can allow apps to violate your privacy?
   [✓ Yes]
   ☐ No

4. Do you care about your privacy when using your mobile devices in indoor environments (e.g, shopping malls, retailer shops, etc.)? Do you think businesses based on indoor sensing environment (e.g., free hotspots) can pose severe privacy threats, such as unauthorised user tracking, profiling and monitoring of your movements?
   [✓ Yes]
   ☐ No
   ☐ I don't know

5. Have you taken any step to protect your location privacy?
   [✓ Regularly change location sharing settings to OFF/ON on your devices based on the usage, e.g., ON when using a location-based app and then turn it OFF when not in use.]
   ☐ Never changed location settings on your device.
   ☐ I don't know

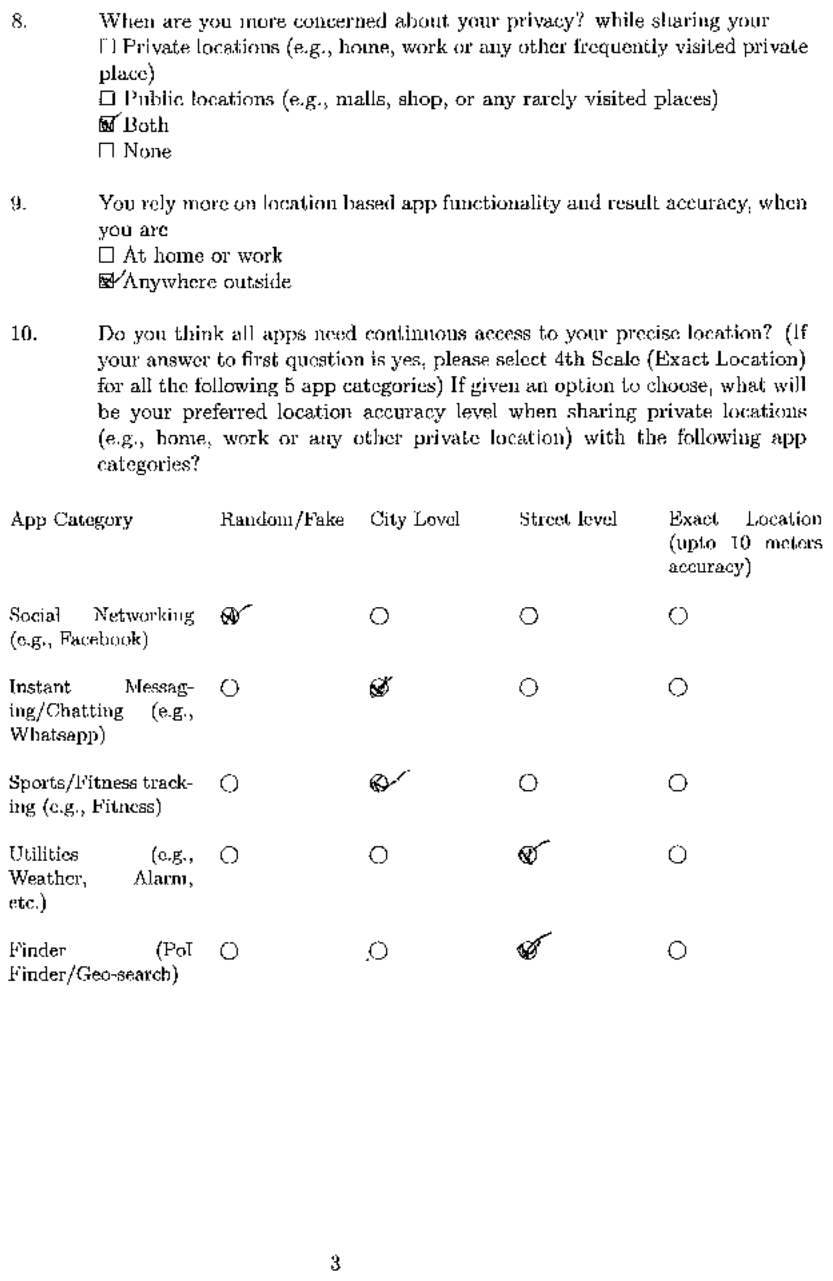
6. Would you install an app or a tool that manages all the installed apps on your mobile device and protects your location privacy?
   [✓ Yes]
   ☐ No

7. Would you mind if such location privacy enhancing tool supplies imprecise locations to installed apps?
   ☐ Yes
   [✓ No]

2

A sample of the paper survey response

(Collected 45 responses in total)



8. When are you more concerned about your privacy? while sharing your
   - ☐ Private locations (e.g., home, work or any other frequently visited private place)
   - ☐ Public locations (e.g., malls, shop, or any rarely visited places)
   - [✓ Both]
   - ☐ None

9. You rely more on location based app functionality and result accuracy, when you are
   - ☐ At home or work
   - [✓ Anywhere outside]

10. Do you think all apps need continuous access to your precise location? (If your answer to first question is yes, please select 4th Scale (Exact Location) for all the following 5 app categories) If given an option to choose, what will be your preferred location accuracy level when sharing private locations (e.g., home, work or any other private location) with the following app categories?

App Category	Random/Fake	City Level	Street level	Exact Location (upto 10 meters accuracy)
Social Networking (e.g., Facebook)	✓	○	○	○
Instant Messaging/Chatting (e.g., Whatsapp)	○	✓	○	○
Sports/Fitness tracking (e.g., Fitness)	○	✓	○	○
Utilities (e.g., Weather, Alarm, etc.)	○	○	✓	○
Finder (PoI Finder/Geo-search)	○	○	✓	○

3

Appendix C

The Middleware Implementation Source Code

Listing C.1 contains block of code to monitor running app on Android 6.0 device. The *middleware*'s background service frequently checks the running foreground app using `getRunningAppProcesses` on older versions and `UsageStasManager` on `Android` 6 (or later). However,to use the `UsageStasManager` API the user of the device needs to grant permission through `Android`'s settings application.

Listing C.1 Block of code to monitor running app on Android 6.0 device

```
 protected Void doInBackground(Void... arg0) {
String appNew = "NULL";
String appNew = am.getRunningTasks(1).get(0).topActivity.
getPackageName();
   if(Build.VERSION.SDK_INT > 20){
   if(android.os.Build.VERSION.SDK_INT > =
   android.os.Build.VERSION_CODES.LOLLIPOP) {
        UsageStatsManager usm =
       (UsageStatsManager)context.getSystemService("usagestats");
```

```
        long time = System.currentTimeMillis();
       List<UsageStats> appList = usm.queryUsageStats
    (UsageStatsManager.INTERVAL_DAILY,  time - 1000*1000, time);
if (appList != null && appList.size() > 0) {
   SortedMap<Long, UsageStats> mySortedMap =
    new TreeMap<Long, UsageStats>();
    for (UsageStats usageStats : appList) {
   mySortedMap.put(usageStats.getLastTimeUsed(), usageStats);
}
if (mySortedMap != null && !mySortedMap.isEmpty()) {
appNew = mySortedMap.get(mySortedMap.lastKey()).getPackageName();
    }
   }
}
else {
ActivityManager am =
(ActivityManager)context.getSystemService(Activity.ACTIVITY_SERVICE);
    List<ActivityManager.RunningAppProcessInfo> tasks =
    am.getRunningAppProcesses();
    appNew  = tasks.get(0).processName;
    }
    Log.e("adapter", "Current_App_in_foreground_is:" + appNew);
String appNew = am.getRunningAppProcesses().get(0).processName;
    boolean lockedNew = isDeviceLocked(context);
    appSessionRecord retVal = isNewRecord(appOld,
    lockedOld, appNew, lockedNew);
if (retVal != null) {
   handleAppChange(appOld, appNew, retVal.end - retVal.start,
    retVal.placeStart, retVal.placeEnd);
    }
    appOld = appNew;
    lockedOld = lockedNew;
```

```
        return null;
    }
}
```

Listing C.2 contains Java/Android code to generate copies of the dataset in the Android device's internal storage. This dataset was then opened in the `DB Browser` tool (see Figure D.1) to maintain logs for the development and testing of the middleware.

Listing C.2 Block of code to to generate copies of the dataset in the Android device's internal storage

```
public void exportDatabse(String databaseName) {
   File sd = Environment.getExternalStorageDirectory();
   File data = Environment.getDataDirectory();
if (sd.canWrite()) {
   String currentDBPath =
   "//data//"+getPackageName()+ "//databases//"+databaseName+"";
   String backupDBPath = "backupname.db";
               File currentDB = new File(data, currentDBPath);
               File backupDB = new File(sd, backupDBPath);
if (currentDB.exists()) {
FileChannel src = new FileInputStream(currentDB).getChannel();
FileChannel dst = new FileOutputStream(backupDB).getChannel();
dst.transferFrom(src, 0, src.size());
     src.close();
     dst.close();
     }
           }
}
```

Listing C.3 contains Java/Android code to add session and location access tags to every app's location access requests. These tags were used to maintain control flow of every intercepted app and monitoring privacy rules.

Listing C.3 Block of code that adds session and location access tags to every app's location access requests

```java
public static void addAppSession
   (Context context, String app, boolean isLocationAccessed)
{
SharedPreferences sharedPref = context.getSharedPreferences
(PREF_FILE, Context.MODE_PRIVATE);
       SharedPreferences.Editor editor = sharedPref.edit();
       String keySessionTotal = app + SESSION_TAG;
       String keySessionLocationAccess = app + LOCATION_TAG;
int sessionTotal = getSessionTotal(context, app);
int sessionsWithLocationAccess =
   getSessionsWithLocationAccess(context, app);
       editor.putInt(keySessionTotal, sessionTotal + 1);
if (isLocationAccessed) {
 editor.putInt(keySessionLocationAccess,
 sessionsWithLocationAccess +1);
   }
       editor.commit();
   }
```

Listing C.4 contains block of sqlite database script to fetch applied privacy rule for every app's location access requests in the database to calculate *LoP*. This was used to calculate achieved privacy.

Listing C.4 Block of code to fetch applied privacy rule for every app's location access requests in SQLite database to calculate *LoP*

```
public LOPRule fetchUpToDateRule(String app, int place) {
RuleData appRule = new RuleData(app);
String strWhereFirstClause = "(" +
 RuleEntry.COLUMN_NAME_PLACE_ID + "=" + place + "OR" +
 RuleEntry.COLUMN_NAME_PLACE_ID + "=-1" + "OR" +
 RuleEntry.COLUMN_NAME_PLACE_ID + "=" + RuleInterface.PLACE_FLAG +
  ")";
  String strWhereSecondClause =
  "(" + RuleEntry.COLUMN_NAME_PACKNAME + "='"
  + app + "'OR" + RuleEntry.COLUMN_NAME_PACKNAME +
   "='" + RuleInterface.APP_FLAG + "')";
  String strWhere =
  strWhereFirstClause + "AND" + strWhereSecondClause;
       Util.Log(Util.DB_TAG, strWhere);
       SQLiteDatabase db = App.getReadableDB();
       String[] projection = {
               RuleEntry.COLUMN_NAME_PLACE_ID,
               RuleEntry.COLUMN_NAME_FORERULE,
               RuleEntry.COLUMN_NAME_BACKRULE,
               RuleEntry.COLUMN_NAME_PRULE,
               RuleEntry.COLUMN_NAME_LOC_PRIVACY_LEVEL,
               RuleEntry.COLUMN_NAME_LEVEL_OF_PRIVACY,
               RuleEntry.COLUMN_NAME_LAT,
               RuleEntry.COLUMN_NAME_LON
       };
   String sortOrder = RuleEntry.COLUMN_NAME_PLACE_ID + "DESC," +
    RuleEntry.COLUMN_NAME_PACKNAME + "DESC";
    String limit = "1";
    Cursor c = db.query(
     RuleEntry.TABLE_NAME, // The table to query
               projection,
               strWhere,
               null,
               null,
               null,
               sortOrder,
               limit
       );
       c.moveToFirst();
       // no matching record
       if (c.getCount() == 0) {
           return null;
       }
       return appRule.fetchSubRule(c);
   }
```

Listing C.4 contains the code used for monitoring locations calls received from running apps. In this code, the requested private location is checked and annonymised/altered as per the privacy rules pre-set by the user, and then it is released to the requesting app.

Listing C.5 Block of code to anonymise locations requested by running apps

```
public void anonymiseLocation(String app, String result,
String source, int reportedPlaceID, int privacyLevel) {
currentAnonymisationLevel = privacyLevel;//routinely updated
 //anonymise location by applying privacy rules to the location
 //check if there is a cached location for place and app to use
 //if not, create a new one and use. Access the current place
 LatLng alteredLocation =
 getAlteredLocationForCurrentPlace(app, privacyLevel);
 mainService.startLocationAnon(alteredLocation);
 //behavior varies for runtime notification or app launch event
  if (!source.equals(Util.FAKE_INTENT_SOURCE)) {
  finishDecisionMaking(app, result, source, reportedPlaceID);
  }
}
private LatLng getAlteredLocationForCurrentPlace(String app,
int privacyLevel) {
double lat = 0;
double lon = 0;
 //add an interface using shared preferences (app+place as key)
 //doesn't exist, create one. And then add to it
 //doesn't interfere with the existing structure of privacy rules
 //everything passes through the rule interface
 Util.Log("rule", "" + privacyLevel);
 int currentPlace = mainService.getCurrentPlace().getPlaceID();
 LatLng currentLoc =
 mainService.getCurrentPlace().getCurrentLoc();
 long t1 = System.nanoTime();
 //privacy level via settings UI
 LatLng alteredLocation = ruleBridge.getFixedLocation(app,
 currentPlace, privacyLevel, context);
 Util.Log("lp_time", "cached␣location:␣" +
 (System.nanoTime() - t1));
  if (alteredLocation != null) {
 Util.Log("rule", "found␣existing␣rule" +
 alteredLocation.toString());
   return alteredLocation;
   }
 //now if null, get it from math tools
alteredLocation =
MathTools.getAlteredLocation(currentLoc, privacyLevel);
ruleBridge.setFixedLocation(app, currentPlace, alteredLocation,
privacyLevel, context);
return alteredLocation;
}
//rule decision making method and resume the app
public void finishDecisionMaking(String appLPPM, String result,
String source, int reportedPlaceID) {
//send the intent back to the app launcher to allow the app to start
mainService.instructApptoLaunch(appLPPM,
result, source, reportedPlaceID);
appHelper.setStartedApp(appLPPM, reportedPlaceID);
}
```

Listing C.6 contains block for the three types of location privacy rules: adjust granularity (i.e., truncate), obfuscate (i.e., transform) and no change are applied on the user's geo-coordinates. To apply privacy rules, we need to convert the geo-coordinates (received via location object) from dms (degree, minutes, seconds) to decimal format. For this, we used geo-coordinate parser util library. The geo-coordinate truncation and transformation levels and scaling factor both are decided by the user inputs. Hence, for truncation, geo-coordinates are rounded to 6/7/8 decimals (1m to 1km precision).

Listing C.6 Block of code for the location privacy rules given in the personalised permissions algorithm

```
// convert geo-coordinates from dms to decimal format.
public class MathTools{
 public static LatLng getAlteredLocation(LatLng currentLoc,
 int accuLevel) {
        double altLat;
        double altLong;
        double radius = getRadius(accuLevel);
        double privacyLevel = Math.log(4);
        Random rand = new Random();//generate randomness
        double theta = rand.nextDouble() * 360;
  double phi = privacyLevel / radius; // (~1m to 1km precision)

        altLat = currentLoc.lat
        altLong = currentLoc.long
 altLat = Math.asin(Math.cos(theta) Math.sin(currentLoc.lat) -
Math.cos(currentLoc.long) Math.sin(theta) Math.cos(currentLoc.lat))
altLong = Math.atan2(sin(currentLoc.long), Math.tan(currentLoc.lat)
Math.sin(theta) + Math.cos(currentLoc.long) Math.cos(theta)) - phi
 return Util.calculateDerivedPosition(currentLoc,
 altLat, altLong, theta);
 }
 private static double getRadius(int accuLevel) {
        switch (accuLevel) {
            case RuleData.ANON_LOW:
                return 10;
            case RuleData.ANON_MEDIUM:
                return 100;
            case RuleData.ANON_HIGH:
                return 1000;
            default:
                return 1000;
        }
 }
 // truncate geo-cordinates (x) using value of p set by the user.
private static Double TrunctateToDecimals(LatLng currentLoc, int p)
 {
    public  parseLatLongtoDecimal(LatLng currentLoc){
```

```
    public CoordinateParseUtils cordParse =
    new CoordinateParseUtils()
    cordParse.OccurrenceParseResult(String.valueOf(currentLoc));
    return this;
    }
    return x == null ? null :
    Math.round(currentLoc * Math.pow(10, p))/ Math.pow(10, p);
    }
}
// no change
private static Double noChange(LatLng currentLoc){
return currentLoc;
    }
}
```

Appendix D

Results for the Middleware Evaluation

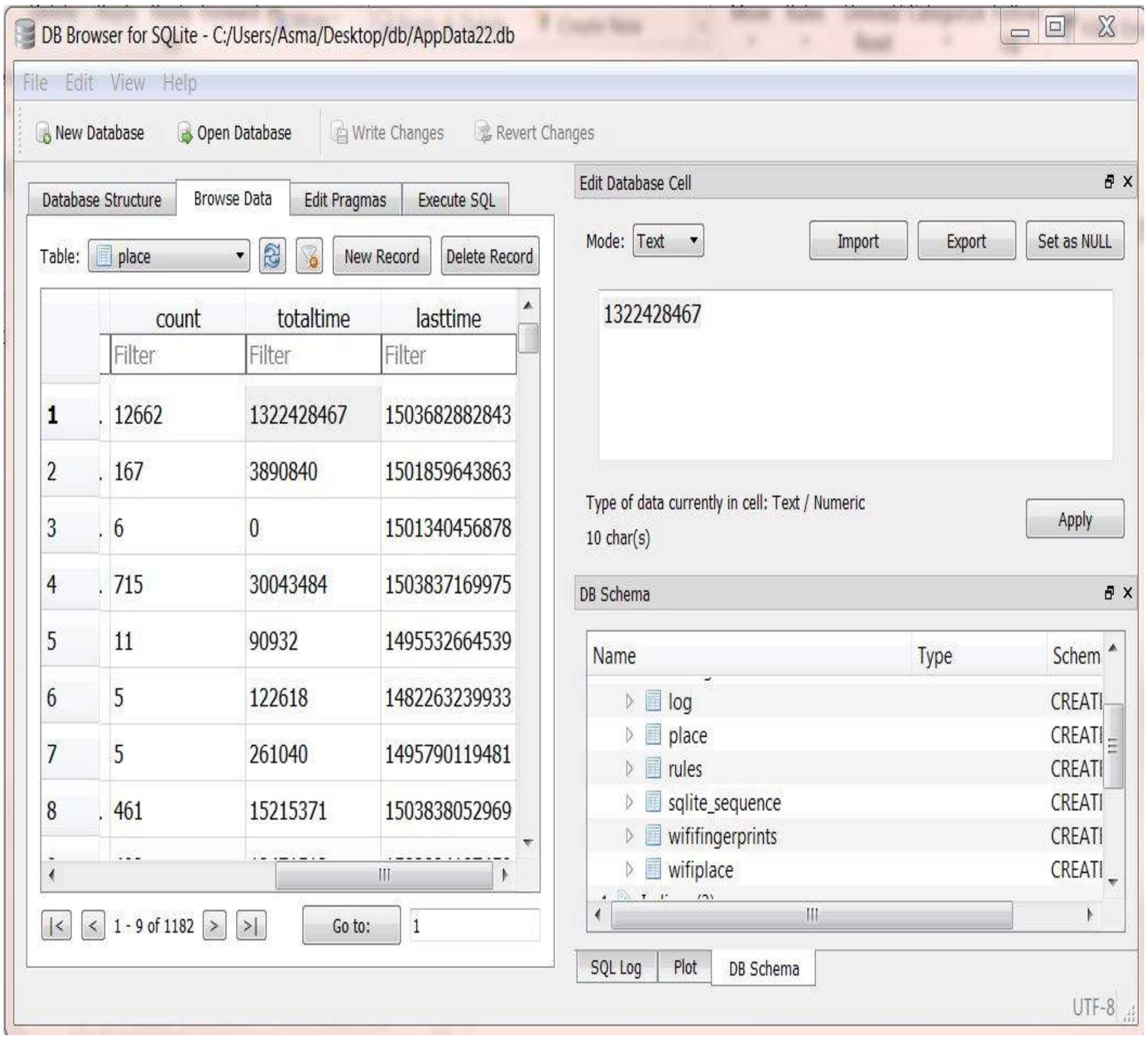


Table: place

	count	totaltime	lasttime
1	12662	1322428467	1503682882843
2	167	3890840	1501859643863
3	6	0	1501340456878
4	715	30043484	1503837169975
5	11	90932	1495532664539
6	5	122618	1482263239933
7	5	261040	1495790119481
8	461	15215371	1503838052969

1 - 9 of 1182 Go to: 1

Mode: Text

1322428467

Type of data currently in cell: Text / Numeric
10 char(s)

Name	Type	Schem
log		CREATI
place		CREATI
rules		CREATI
sqlite_sequence		CREATI
wififingerprints		CREATI
wifiplace		CREATI

Figure D.1 Screenshot of a collected SQLite database opened on DB Browser tool
The collected SQLite database, i.e., AppData22.db, is opened on DB Browser tool. We used this tool to analyse the collected versions of SQLite databases and monitor the increase of the on-device cache storage.

Table D.1 Part A: Data collected for calculating PL-Protector's computation latency at private places (P1 to P17).

Part A

Sessions	P1-Set1	P2-Set2	P3-Set3	P4-Set4	P5-Set5	P6-Set6	P7-Set7	P8-Set8	P9-Set9	P10-Set10	P11-Set11	P12-Set12	P13 -Set13	P14 -Set14	P15 -Set15	P16 -Set16	P17-Set17
1	10	19	18	10	9.5	18	17	15	14	10	8	10	19	17	18	19	16
2	18	17	15	14	9.5	18	17	15	14	7	8	17	19	10	18	7	16
3	14	18	13	16	23	12	15	12	13	4	10	10	14	13	12	11	13
4	17	15	14	9.5	18	17	15	16	12	13	8	20	14	15	17	15	21
5	12	21	17	12	13	10	12	13	12	14	3	14	11	10	13	12	21
6	14	15	11	12	15	14	17	8	15	17	11	13	14	15	16	17	12
7	15	17	18	14	14	13	12	14	13	11	10	12	13	14	15	13	12
8	17	15	14	12	13	14	12	13	21	18	5	7	22	14	13	14	14
9	17	15	24	16	17	12	13	11	8	14	13	15	14	12	10	17	14
10	15	14	12	14	12	11	13	14	15	24	10	12	13	14	13	11	11
11	13	16	13	13	15	12	12	11	13	12	17	14	19	16	17	18	11
12	10	19	18	10	9.5	18	17	15	14	13	18	10	19	17	18	14	22
13	18	22	15	14	9.5	18	17	15	14	18	18	17	19	10	18	22	16
14	14	18	13	16	22	12	15	12	13	15	10	10	14	13	12	12	13
15	17	15	14	9.5	18	17	15	16	12	13	18	21	14	15	17	13	21
16	12	21	17	12	13	21	12	13	12	14	13	14	11	11	13	18	21
17	14	15	11	12	15	14	17	8	15	17	11	13	14	15	16	15	12
18	15	17	18	14	14	13	12	14	13	11	10	12	13	14	15	13	12
19	17	15	14	12	13	14	2	13	11	18	15	17	12	14	13	14	14
20	17	15	14	16	17	22	13	21	18	14	13	15	14	12	11	17	14
21	15	14	12	14	12	11	13	14	15	14	21.5	12	13	14	13	11	11
22	13	16	13	13	15	12	12	11	13	12	17	14	19	16	17	18	11
23	10	19	18	10	9.5	18	17	5	14	13	18	10	19	17	18	14	16
24	18	17	15	14	9.5	18	17	5	14	17	18	17	19	10	18	14	16
25	14	18	13	16	23	12	5	12	13	14	10	10	14	13	12	12	13
26	17	15	14	9.5	8	17	15	6	12	14	8	22	14	15	17	13	21
27	12	21	17	12	13	21	12	13	12	11	3	14	11	11	13	12	11
28	14	15	11	12	15	14	17	8	15	14	11	13	14	15	16	17	12
29	15	17	18	14	14	13	12	14	13	15	11	12	13	14	15	14	12
30	17	15	14	12	13	14	12	13	11	12	15	17	12	14	13	12	14
31	17	15	14	16	17	12	13	11	18	12	13	15	14	12	11	13	14
32	15	14	22	14	12	11	13	14	15	17	11	12	13	14	13	12	11
33	13	16	13	13	15	12	12	11	13	15	17	14	19	16	17	12	11
34	22	11	17	20	14	21	12	13	12	15	13	14	21	10	10	12	21

Table D.2 Part B: Data collected for calculating PL-Protector's computation latency at private places (P18 to P34).

Part B

Sessions	P18 -Set18	P19 -Set19	P20 -Set20	P21 -Set21	P22 -Set22	P23 -Set23	P24 -Se 24	P25 -Set25	P26 -Set26	P27 -Set27	P28 -Set28	P29 -Set29	P30 -Set30	P31 -Set31	P32 -Set32	P33 -Set33	P34 -Set34
1	18	17	15	14	10	18	10	18	17	15	14	10	18	10	18	17	15
2	18	17	15	14	7	18	17	18	17	15	4	17	18	17	18	17	15
3	12	15	12	13	4	10	10	12	15	12	13	14	10	10	12	15	12
4	17	15	16	12	4	18	22	17	15	16	12	14	18	25	7	15	16
5	11	12	13	12	10	13	14	11	12	13	2	21	13	14	11	12	13
6	14	17	18	15	15	11	13	14	17	18	15	14	15	13	14	17	18
7	13	12	14	13	14	11	12	13	13	14	14	15	14	13	13	12	14
8	10	12	13	21	14	15	17	14	14	13	14	13	14	14	14	12	13
9	15	13	20	8	12	13	15	12	17	21	12	14	12	17	17	13	11
10	14	13	14	15	14	10	12	21	11	14	14	17	14	11	11	13	14
11	14	12	11	13	16	17	14	12	18	11	16	11	16	18	18	12	11
12	12	17	15	14	17	18	10	18	14	15	17	18	17	14	14	17	11
13	14	17	15	14	10	8	17	18	24	15	11	14	11	24	24	17	15
14	16	15	12	13	13	10	10	12	12	12	14	24	14	12	12	15	14
15	17	15	16	12	15	18	21.5	11	13	16	13	12	13	13	13	15	14
16	10	11	13	12	11	13	14	15	18	17	14	13	14	18	18	12	12
17	13	15	18	21	15	11	13	14	15	10	12	18	12	15	15	11	14
18	15	14	14	13	14	11	12	14	13	13	11	15	11	13	13	15	16
19	11	14	13	11	14	15	17	12	14	15	15	13	14	14	14	14	17
20	15	12	11	18	12	13	15	14	17	11	14	14	16	17	17	14	11
21	14	14	14	15	14	11	12	16	13	15	14	17	17	12	20	12	14
22	14	16	11	13	16	17	14	17	12	14	12	15	11	14	12	14	13
23	12	17	15	14	17	18	10	11	17	14	14	10	14	10	18	16	14
24	14	11	15	14	17	18	17	14	17	12	16	17	13	17	18	17	12
25	16	14	12	13	14	10	10	13	15	14	17	14	14	10	12	11	12
26	17	13	16	12	14	18	25	14	15	16	11	14	12	25	17	14	16
27	11	14	13	12	11	13	14	12	12	17	14	11	11	14	21	13	13
28	14	12	18	15	14	21	13	14	17	11	13	14	11	13	14	14	18
29	13	12	14	13	15	21	12	13	12	14	14	15	21	12	13	12	14
30	14	12	13	21	12	5	17	14	12	13	12	12	15	17	14	12	13
31	12	13	11	18	12	13	15	12	13	14	11	12	13	15	12	13	11
32	11	13	14	15	17	21	12	11	13	12	5	17	11	12	11	13	14
33	12	12	11	13	15	17	14	12	12	11	13	15	17	14	12	12	11
34	10	12	11	12	11	13	14	11	11	12	12	11	13	14	11	16	13

Table D.3 Data collected to calculate communication overhead for apps.
Please note: we collected datasets of 5 apps running in both environments baseline ($WDelay$ and $Wfreq$) vs `PL-Protector` ($Delay$ and $Freq$).

AppSets	Sessions	Delay	Freq	WDelay	WFreq
AppSet 1	5	1	1	1	1
AppSet 1	10	9	10	7	8
AppSet 1	20	9	19	7	15
AppSet 1	30	8	27	6	21
AppSet 1	40	12	39	10	31
AppSet 1	50	9.5	48.5	7	38
AppSet 1	60	8	56.5	5	43
AppSet 1	70	7	63.5	4	47
AppSet 1	80	13	76.5	11	58
AppSet 1	90	10	86.5	6	64
AppSet 1	100	14	100.5	11	75
AppSet 2	5	2	2	0.5	0.5
AppSet 2	10	6	8	5	5.5
AppSet 2	20	9	17	7	12.5
AppSet 2	30	7	24	5	17.5
AppSet 2	40	6	30	5.5	23
AppSet 2	50	7	37	5	28
AppSet 2	60	6	43	6	34
AppSet 2	70	9	52	5	39
AppSet 2	80	7	59	6	45
AppSet 2	90	6.5	65.5	4	49
AppSet 2	100	7.2	72.7	5	54
AppSet 3	5	4	4	3	3
AppSet 3	10	14	18	11	14

AppSets	Sessions	Delay	Freq	WDelay	WFreq
AppSet 3	20	11	29	12	26
AppSet 3	30	14	43	12	38
AppSet 3	40	13	56	9	90
AppSet 3	50	10	66	10	97
AppSet 3	60	15	81	12	99
AppSet 3	70	17	98	15	101
AppSet 3	80	13	111	13	107
AppSet 3	90	18	129	16	113
AppSet 3	100	22	151	20	123
AppSet 4	5	3	3	1.5	1.5
AppSet 4	10	7	10	5	6.5
AppSet 4	20	4	14	1	7.5
AppSet 4	30	5	19	3	10.5
AppSet 4	40	4	23	4	14.5
AppSet 4	50	12	35	11	25.5
AppSet 4	60	14	49	13	38.5
AppSet 4	70	8	57	8	46.5
AppSet 4	80	9	66	10	56.5
AppSet 4	90	10	76	8	64.5
AppSet 4	100	14	90	12	76.5
AppSet 5	5	2	2	1	1
AppSet 5	10	14	16	10	11
AppSet 5	20	13	29	9	20
AppSet 5	30	21	50	17	37
AppSet 5	40	16	66	15	52
AppSet 5	50	18	84	17	79
AppSet 5	60	21	105	21	80

AppSets	Sessions	Delay	Freq	WDelay	WFreq
AppSet 5	70	14	119	12	98
AppSet 5	80	17	136	16	118
AppSet 5	90	15	151	14	122
AppSet 5	100	13	164	12	140

Table D.4 Data collected to calculate response rate by PL-Protector when interacting with apps

No.	AC-Set1	AC-Set2	AC-Set3	AC-Set4	AC-Set5
1	0.85	0.7	0.83	0.82	0.78
2	0.87	0.92	0.92	0.7	0.86
3	0.85	0.77	0.83	0.68	0.79
4	0.84	0.76	0.82	0.78	0.85
5	0.91	0.84	0.83	0.8	0.73
6	0.82	0.42	0.72	0.75	0.84
7	0.87	0.72	0.73	0.77	0.73
8	0.75	0.75	0.88	0.88	0.75
9	0.85	0.77	0.82	0.95	0.84
10	0.81	0.89	0.81	0.86	0.76
11	0.38	0.78	0.72	0.81	0.78
12	0.77	0.78	0.82	0.78	0.77
13	0.84	0.84	0.83	0.81	0.88
14	0.97	0.85	0.94	0.69	0.95
15	0.84	0.77	0.74	0.75	0.76
16	0.98	0.88	0.72	0.84	0.21
17	0.88	0.81	0.83	0.75	0.11
18	0.82	0.11	0.84	0.81	0.87
19	0.89	0.74	0.82	0.74	0.78

No.	AC-Set1	AC-Set2	AC-Set3	AC-Set4	AC-Set5
20	0.89	0.81	0.91	0.83	0.67
21	0.85	0.7	0.83	0.82	0.78
22	0.87	0.92	0.92	0.7	0.86
23	0.85	0.77	0.83	0.68	0.79
24	0.84	0.74	0.82	0.78	0.85
25	0.91	0.84	0.83	0.8	0.73
26	0.82	0.77	0.72	0.75	0.84
27	0.87	0.72	0.73	0.77	0.73
28	0.75	0.85	0.88	0.88	0.75
29	0.85	0.77	0.82	0.95	0.84
30	0.81	0.89	0.81	0.86	0.76
31	0.38	0.78	0.72	0.1	0.78
32	0.77	0.78	0.72	0.78	0.77
33	0.84	0.84	0.83	0.81	0.88
34	0.97	0.75	0.94	0.69	0.95
35	0.84	0.87	0.74	0.75	0.76
36	0.98	0.88	0.72	0.84	0.21
37	0.88	0.81	0.83	0.75	0.11
38	0.82	0.11	0.84	0.81	0.87
39	0.89	0.78	0.82	0.74	0.78
40	0.89	0.81	0.91	0.83	0.67
41	0.87	0.7	0.83	0.82	0.78
42	0.84	0.92	0.92	0.7	0.86
43	0.85	0.77	0.83	0.68	0.79
44	0.87	0.76	0.82	0.78	0.85
45	0.91	0.84	0.83	0.8	0.73
46	0.82	0.42	0.72	0.75	0.84

No.	AC-Set1	AC-Set2	AC-Set3	AC-Set4	AC-Set5
47	0.87	0.72	0.73	0.77	0.73
48	0.75	0.75	0.88	0.88	0.75
49	0.85	0.77	0.82	0.95	0.84
50	0.81	0.89	0.81	0.86	0.76
51	0.38	0.78	0.72	0.81	0.78
52	0.77	0.78	0.82	0.78	0.77
53	0.84	0.84	0.83	0.81	0.88
54	0.97	0.85	0.94	0.69	0.95
55	0.84	0.77	0.74	0.75	0.76
56	0.98	0.88	0.72	0.84	0.21
57	0.88	0.81	0.83	0.75	0.11
58	0.82	0.11	0.84	0.81	0.87
59	0.89	0.74	0.82	0.74	0.78
60	0.89	0.81	0.91	0.83	0.67
61	0.85	0.7	0.83	0.82	0.78
62	0.87	0.92	0.92	0.7	0.86
63	0.85	0.77	0.83	0.68	0.79
64	0.89	0.74	0.82	0.78	0.85
65	0.85	0.84	0.83	0.8	0.73
66	0.87	0.77	0.72	0.75	0.84
67	0.85	0.72	0.73	0.77	0.73
68	0.75	0.85	0.88	0.88	0.75
69	0.85	0.77	0.82	0.95	0.84
70	0.81	0.89	0.81	0.86	0.76
71	0.38	0.78	0.72	0.1	0.78
72	0.77	0.78	0.72	0.78	0.77
73	0.84	0.84	0.83	0.81	0.88

No.	AC-Set1	AC-Set2	AC-Set3	AC-Set4	AC-Set5
74	0.97	0.75	0.94	0.69	0.95
75	0.84	0.87	0.74	0.75	0.76
76	0.98	0.88	0.72	0.84	0.21
77	0.88	0.81	0.83	0.75	0.11
78	0.82	0.11	0.84	0.81	0.87
79	0.75	0.78	0.82	0.74	0.78
80	0.91	0.81	0.91	0.83	0.67

Table D.5 Data collected to calculate on-device cache mechanism accuracy of the inter-request interval over time for 5 apps (Time vs Response rate)

Time	Appset 1	AppSet 2	AppSet 3	AppSet 4	AppSet 5
0	0.6	0.52	0.4	0.54	0.57
5	0.58	0.4	0.6	0.48	0.6
10	0.64	0.5	0.58	0.54	0.7
15	0.7	0.7	0.65	0.75	0.8
20	0.81	0.7	0.78	0.71	0.85
25	0.84	0.8	0.75	0.89	0.85
30	0.87	0.8	0.89	0.8	0.9
35	0.79	0.7	0.87	0.77	0.9
40	0.85	0.85	0.9	0.85	0.95
45	0.92	0.89	0.85	0.89	0.95
50	0.95	0.9	0.89	0.9	0.9

Appendix E

Publications

List of Publications:

1. Patel, A., & Palomar, E., 2017. A middleware enforcing location privacy in mobile platforms. In: 14th Int. Conf. on Trust and Privacy in Digital Business (TrustBus). Springer, pp. 32–45.

2. Patel, A., & Palomar, E., 2016b. Protecting Mobile Users' Private Locations through Caching. In: 13th Int. Conf. on E-Business and Telecommunications, Springer, pp. 316-337.

3. Patel, A., & Palomar, E., 2016a. LP-Cache: Privacy-aware Cache Model for Location-based Apps. In: 13th Int. Conf. on Security and Cryptography (SECRYPT), pp. *183-194*.

4. Patel, A., & Palomar, E., 2014. Privacy Preservation in Location-Based Mobile Applications: Research Directions". In: 9th Int. Conf. on Availability, Reliability and Security (ARES), IEEE, pp. *227-233*.

5. Patel, A., & Palomar, E., 2018. A Practical Cache-based Location Privacy-enhanced Middleware for Mobile Users (manuscript developed for publication).

A Middleware Enforcing Location Privacy in Mobile Platforms

Asma Patel(✉) and Esther Palomar

School of Computing and Digital Technology,
Birmingham City University, Birmingham, UK
{asma.patel,esther.palomar}@bcu.ac.uk
http://www.bcu.ac.uk/

Abstract. Emerging indoor positioning and WiFi infrastructure enable building apps with numerous Location-based Services (LBS) that represent critical threats to smartphone users' location privacy provoking continuous tracking, profiling and unauthorized identification. Currently, the app eco-system relies on permission-based access control, which is proven ineffective at controlling how third party apps and/or library developers use and share users' data. In this paper we present the design, deployment and evaluation of PL-Protector, a location privacy-enhancing middleware, which through a caching technique minimises the interaction and data collection from wireless access points, content distributors and location providers. PL-Protector also provides a new series of control settings and privacy rules over both, the information and control flows between sources and sinks, to prevent user information disclosure during LBS queries. We implement PL-Protector on Android 6, and conduct experiments with real apps from five different categories of location-based services such as instant messaging and navigation. Experiments demonstrate acceptable delay overheads (lower than 22 ms) within practical limits; hence, our middleware is practical, secure and efficient for location-demanding apps.

Keywords: Location privacy · Location-based services · Smartphones · Caching · Location-based applications · Android · Mobile platforms

1 Introduction

The explosive growth of Internet of Things, Smart Cities and Smart-Home application frameworks, e.g., Samsung SmartThing [23] and Google Weave/Android of Things [12], leverage third party developers to build apps that compute on user's sensitive data. For instance, emergent context-aware mobile apps bring about tremendous opportunities for a whole new class of Location-Based Services (LBS) [21]. Geo-marketing and geo-social networking, location-based games, and assisted eHealth represent a small subset of these opportunities that can certainly pose a serious threat to the users' privacy [17,24].

© Springer International Publishing AG 2017
J. Lopez et al. (Eds.): TrustBus 2017, LNCS 10442, pp. 32–45, 2017.
DOI: 10.1007/978-3-319-64483-7_3

Currently, privacy settings of user location on smartphones[1] are modeled after existing permission controls that are based on a binary process[2]. In general, apps use permission-based access control for data sources and sinks, but they do not control flows between the authorised sources and sinks. Besides, users are forced to rely on third party service providers/sinks that in many cases continuously collect, use and share their location data, and in some cases even prompt the user to give away their position on page load [1,24]. Moreover, both academia and industry agree on the urgent need of adopting a Privacy-by-Design (PbD) approach for the development of more user-friendly and socially-accepted solutions to privacy preservation on their mobile products and services [6].

To encounter this challenge, our approach campaigns new design principle and privacy policy recommendation that forces the smartphone app ecosystem to make location data use patterns explicit, while preventing all other sensitive data flows. In this paper, we present the design, deployment and evaluation of the middleware called *Private Location Protector* (PL-Protector), which implements the LP-Caché model introduced in [20]. PL-Protector envisions beyond the simple grant/deny access method and provides the user with advanced mechanisms to decide the extent of disclosing location data with service providers. It also incorporates caching technique to determine users' geographical location in a privacy preserving manner by means of wireless access points, and with minimum cache storage requirements. The contributions of this work are as follows:

– To identify the required functionality goals that protect users from location-demanding apps through the analysis of location privacy specific challenges and security design issues within the existing mobile platforms.
– Design of the on-device location computation model that enables robust and efficient source to sink (unauthorised apps and third party app providers) flow control. We present implementation details of the PL-Protector middleware for mobile platforms. Our prototype runs on a Nexus 6p with Android that acts as platform's location privacy knob. PL-Protector only requires process isolation and IPC services from the underlying operative system (OS); thus, minimizing the requirements placed on the hardware/OS.
– Evaluation of PL-Protector in terms of Quality of Service (QoS), communication and computational overheads. We ported five real location-based apps to PL-Protector. Macro-benchmarks of these apps (latency) and our empirical settings indicate that PL-Protector performance overheads are acceptable, and it is practical, secure, and efficient middleware for location-demanding apps.

The rest of the paper is organized as follows. Section 2 outlines the background and related work. Section 3 overviews PL-Protector's privacy model.

[1] Throughout this paper, we use the terms Smartphone and Mobile interchangeably.

[2] Data protection directives and acts [8,15] across the globe state that personal data should not be disclosed or shared with third parties without consent from subject(s). Such a consent is typically obtained by mandatory acceptance of the conditions mentioned in the End User License Agreement (EULA), or through opt-out possibilities and other regulations [16].

Section 4 fully elaborates on the design decisions, architecture and implementation of the middleware. We evaluate PL-Protector's performance overheads in Sect. 5. Finally, Sect. 6 concludes and sets future plans.

2 Background

In this section we analyze the location privacy threats within the smartphone app ecosystem. This includes studying how the location calculation process works in smartphones and how LBS apps collect the user location data. We also justify our decision on implementing PL-Protector as middleware for Android platforms. Nonetheless, our results can be extrapolated to other permission-based mobile platforms such as iOS.

2.1 Location Sources

To understand location privacy specific challenges and security design issues, we start analysing the process of location calculation in smartphones when using LBS. Based on our prior study [20], we understand that the three major existing mobile platforms, namely *Android*, *Windows* and *iOS* that span the domains of smartphones, follow common patterns of location data retrieval. Basically, the standard architecture of using LBS on a mobile platform comprises of four main entities: (1) User Device i.e. installed apps, (2) App Provider, (3) Network Infrastructure, and (4) Location Provider. The user device collects the unique identifiers from the surrounding network access points along with GPS data, and sends these over to the location provider to get the exact device location. Calculation[3] of the user device's actual position is then performed at the location provider who sends it back to the user device in the shape of a location object containing geo-coordinates. At the device, this location object is shared amongst apps, and then, it is transmitted to app providers along with LBS query via the standard programming interface/API [3].

Simple eavesdropping on this location object is a major threat to this architecture even if users put in place the corresponding location sharing preferences[4], which generally are highly context sensitive and use dependent [26]. Moreover, existing OS's location access controls by system services respond inadequately to major privacy threats [1,10].

2.2 Operating System Controls and Apps' Location Access

In Android, apps can only access sensitive resources through the official APIs once the corresponding permissions declared at the manifest files are granted and

[3] i.e., WiFi Triangulation and Cell-tower Triangulation, GPS Mapping, etc.

[4] Types and levels of controls for user location privacy settings depend on the OS and apps. In some cases, apps do not allow users to control others' access to their location data.

authorised by the user. Since Android 6.0 (API level 23), users grant permissions to apps while the app is running, not when they install the app. However, in both cases, a positive user authorisation might result in other remote third parties and external sources benefiting from this information made available in ad-libraries for commercial purposes and/or untrusted code execution [7,9]. These existing studies and reports of data-stealing malware on smartphones clearly show the need of a better run-time permission method regulating the way apps and ad libraries are integrated into Android and other permission based platforms.

2.3 Middleware

Privacy Enhancing Techniques (PETs) and other cryptographic schemes [19,25] have been proposed to the location query formation and privacy preservation between the app/LBS providers in the different architectures and settings. Besides, several proposals apply caching scheme, along with PETs, to address location privacy challenges. Most of these techniques, however, rely on a series of theoretical assumptions such as the existence of a trusted infrastructure providing privacy protection, a group of similar app users being at the same time and same place, or a data collection servers complying with location privacy regulations, e.g., *MobiCaché* [4,18,27].

Caché [2] maintains an on-device cache to store entire location based queries contents and data-types to be re-used in future LBS queries that increases the cache storage requirements. Besides the storage overhead, Caché also requires the abilities of app developer to modify the way app access location data. PL-Protector only caches the network fingerprints and mapped geo-coordinates, which reduces the memory requirements significantly. Our middleware, PL-Protector, considers installed apps as black boxes; this way, it does not require app developer to modify the app code.

The service called *Koi* in [13] is cloud-based and demands numerous changes in the existing smartphone ecosystem. It requires developers to use a different API for the local access to the device location and implements a comparison mechanism and location criteria. Similar to ours, solution proposed in [11] does not rely neither on the adaptation of the apps code nor on the existence of theoretically trusted infrastructure. However, it does not allow the user to control wireless and location data that is shared with the location provider and/or the app provider. This issue is mainly due to considering the location provider as the only source of the user location when developing location-based apps. Moreover, it applies *indistinguishability* on the location data, which increases the computational overhead over time.

LP-Caché introduced in [20] and its implementation as the middleware PL-Protector are, to our knowledge, first working prototype leading the new design principles and policy recommendations to secure the computation and transmission of user location data within the existing mobile ecosystem. Main achievements lead to a minimisation of the wireless AP data collection by both, the wireless content distributors and location providers, and the provision of the control settings preventing user information disclosure from the generated LBS queries (e.g., points of interests (PoIs) and nearest neighbors).

3 Privacy Model

We characterize PL-Protector's privacy model considering the threat model and evaluation metrics for both app usage and privacy protection.

3.1 Threat Model

We consider two different threat scenarios to PL-Protector mainly caused by the level of access to user location in and out the device, as follows:

Android's Middleware Layer Threats. A series of attacks operates at Android's middleware layer [5]. PL-Protector mitigates the location privacy attacks coming from over-privileged and malicious 3rd party apps and libraries.

Privacy Threats. User tracking, identification and profiling (i.e., personal habits, movement patterns, etc.) are fundamental threats to location privacy [25]. Without PL-Protector, the continuous flow of LBS queries between user devices and service providers, that include device's exact geo-coordinates information, leverages malicious misuse of the user location data, especially in the presence of a malicious location provider, app provider and via advanced network sniffing practices.

PL-Protector computes the exact location within the user device, without the location provider's involvement, whilst trusting the device on the storage of sensitive data. However, the user has still the option of giving consent for app providers and/or location providers to access location data. Mobile network providers might, however, collect user location data via cellular clients. It is also excluded from our work the option of manually inserting the location data (e.g., street name and post code) within the LBS query.

3.2 Preliminaries

We now model the user mobility and app usage (specifically at private places) as a series of privacy evaluation metrics that will be used to validate PL-Protector's working assumptions.

Mobility Model. We formulate users' points of interests (PoIs), (e.g., Home or Work) as private places that users frequently visit. Hence, p_i represent ith private place identification, which is derived from a series of scanned beacons n_x and the representative location l_r for that private place, and P_l is a set of user's total private places, as shown in Eqs. 1 and 2.

$$p_i = [n_1], [n_2], \ldots, [n_x] \rightarrow [l_r] \tag{1}$$

$$P_l = [p_i], [p_j], \ldots, [p_n] \tag{2}$$

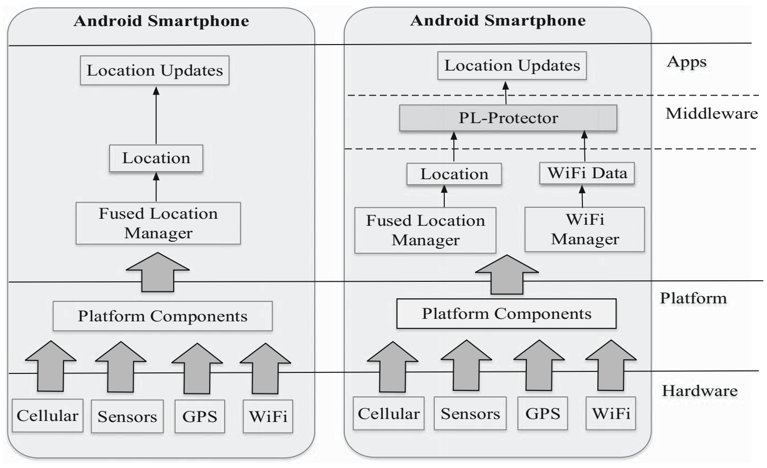


Fig. 1. Mechanism to access user's location in Android (left), and PL-Protector's deployment in Android (right).

At location p_i, the user can then visit a subset of private places $U_{p_i} \subseteq p_1, p_2, \cdots, p_j$, running the different LBS apps on his device. PL-Protector relies on the user input to define the set of private places that are distinct for every user mobility profile. Moreover, to set up network fingerprints at p_i, we measure the response rate as the ratio of detection count and the total number of scans for each beacon as follows:

$$R_{n_c,x} = \frac{\sum_{i=1}^{n_c} b_{x,i}}{n_c}, b_{x,i} = \begin{cases} 1 & \text{if beacon}\, x \,\text{found in}\, i\text{th scan} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $R_{n_c,x}$ is the response rate of beacon x at p_i and n_c is the total scan count since the private place was entered. The detection count of each beacon is maintained to identify the frequently occurring beacons. Beacons with higher response rate are used to create the network fingerprint for that p_i. $R_{n_c,x}$ will be maintained in the PL-Protector database to update the response rate of every detected beacon during a specified time interval t spent at private place p_i.

App-Usage Model. We will apply privacy rules to the app sessions taking place at private places. We define "app session" as the duration of the app usage. In Android, according to the execution status, an app can run in three different states: foreground, background and perceptible. In general, apps get access to the user's location in foreground. When the user exits an app, this is cached and moved to background state for faster execution. Persistent status is informed by notifications. Background state allows prolonged location access; therefore, tracking threats are more harmful here.

3.3 Privacy Metrics

Table 1 compiles the hereinafter metrics to be used for evaluating the location privacy threats. We define value of Pl_s as the identifier of applied privacy setting and measure the achieved privacy by analysing the collected dataset of the actual location traces at user's private places. To evaluate location privacy, we use Haversine formula in [22] to quantify tracking and profiling threats as the distance Pl_d (Eq. 4) between two positions with longitude and latitude (ϕ, λ) and the radius r of the Earth:

$$Pl_d = 2r \sin^{-1}(\sqrt{\sin^2(\frac{\phi1 - \phi2)}{2} + \cos(\phi1)\cos(\phi2)\sin^2(\frac{\lambda2 - \lambda1}{2}}) \qquad (4)$$

where the haversine function is given by $Hsin(\theta) = sin^2(\frac{\theta}{2})$, $\phi1$ & $\phi2$ are the original geo-coordinates, and $\lambda1$ & $\lambda2$ are the observed geo-coordinates. Secondly, the privacy rules (see more details in Sect. 4.1) pre-set by user will, later, be used to measure achieved privacy using the distance scale $\langle P_{high}, P_{medium}, P_{low} \rangle$.

Table 1. The evaluation metrics for the location privacy threats.

Metric	Description
Pl_s	Unique value to identify applied privacy settings
Pl_d	Distance between two points with longitude and latitude
P_{high}	Distance is >111.32 km
P_{medium}	Distance is >11.132 km
P_{low}	Distance is >1.1132 km

4 PL-Protector on Android

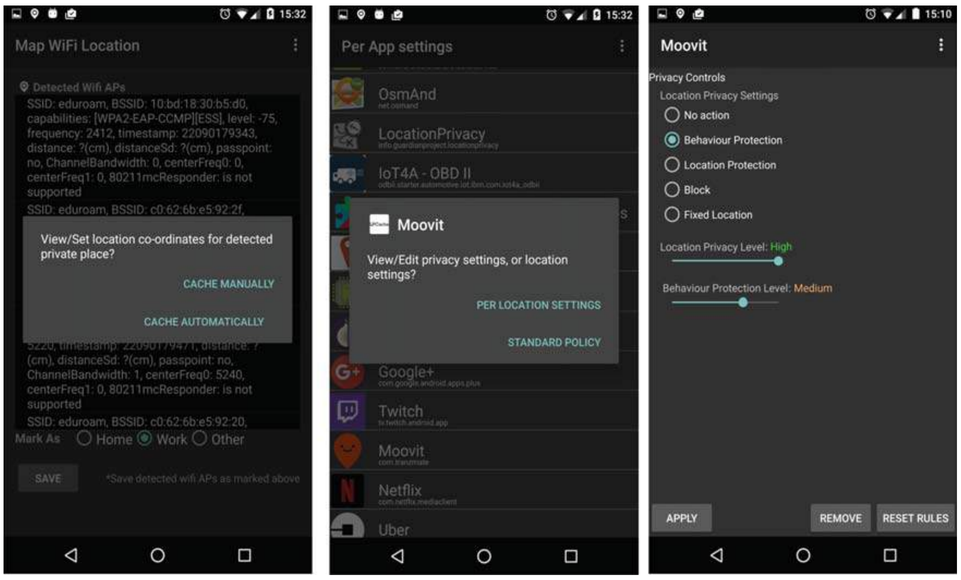
In this section we describe the architecture and implementation decisions for PL-Protector middleware on Android.

4.1 Architecture

PL-Protector enables users to control per-location access sessions.

App-Session Handler component is responsible of intercepting location access events so to lead the LBS apps' control flow to our middleware. It first pauses the requesting app and, retrieves the newly acquired location object to be sent to the *Private Location Manager* component for rule checking. Once the privacy policy rules are applied, the App-Session Handler will receive the anonymised/transformed location and resume the requesting app's control flow.

The **Private Location Manager** is the central component that receives both events (actions) and data from the different components as well as it maintains



```
SSID: eduroam, BSSID: 10:bd:18:30:b5:d0,
capabilities: [WPA2-EAP-CCMP][ESS], level: -75,
frequency: 2412, timestamp: 22090179343,
distance: ?(cm), distanceSd: ?(cm), passpoint:
no, ChannelBandwidth: 0, centerFreq0: 0,
centerFreq1: 0, 80211mcResponder: is not
supported
SSID: eduroam, BSSID: c0:62:6b:e5:92:2f,
5220, timestamp: 22090179471, distance: ?
(cm), distanceSd: ?(cm), passpoint: no,
ChannelBandwidth: 1, centerFreq0: 5240,
centerFreq1: 0, 80211mcResponder: is not
supported
SSID: eduroam, BSSID: c0:62:6b:e5:92:20,
```

View/Set location co-ordinates for detected private place?

Mark As ○ Home [✓ Work] ○ Other

View/Edit privacy settings, or location settings?

○ No action
[✓ Behaviour Protection]
○ Location Protection
○ Block
○ Fixed Location

Location Privacy Level: High

Behaviour Protection Level: Medium

Fig. 2. User interface: (left) WiFi settings screen, and (middle and right) screens to manage per-app/location rule settings

the *Cache DB* database. User inputs via user interface (UI) are used to create privacy rules for specific private locations and network fingerprints. Moreover, the Private Location Manager detects unique identifiers of the surrounding wireless APs and maintains a binary flag to detect private places. When the flag is ON, the location data is retrieved from the *Cache DB* and sent to the *Policy Controller*. In case of an unmatched query on the cached locations, and the user do not want to input geo-coordinates manually (via maps provided in UI) the location data is received by location providers from the *Location Receiver*.

The **Policy Controller** gathers the location objects from the *Private Location Manager* as to apply the corresponding user permissions on the location coordinates, altering them if needed, and transferring the processed location to the *App Session Handler* module. The two privacy policies that the User can set per-app/place basis are the *Standard Policy* and *Per-location Policy* (see Fig. 2), as follows:

1. The Standard Policy consists of three location settings as follow:
   (a) The *Behaviour Protection* setting implements the geo-coordinate obfuscation equation defined in [20] to generate transformed/obfuscated geo-coordinates (l', l'_g) for every app session. The behaviour protection level is defined by a scale (Low, Medium, and High) that determines randomness of the obfuscation equation's parameters $\langle s, \theta, (l, l_g)\rangle$, where s is the scaling factor, θ is the random rotation angle, and (l, l_g) are the original coordinates.
   (b) The *Location Protection* setting implements the geo-coordinate truncation equation defined in [20] and follows a location granularity scale like

(Low, Medium, and High) to adjust the location precision level for every app session.

(c) The *Block/Fixed Location* setting picks high behaviour and location protection level by default and determines a constant value of altered geo-coordinates for every app session.

2. The Per-location policy allows the User to apply standard policy settings for each pre-marked private places that are displayed on the map.

Once processed geo-coordinates (l' l'_g) that comply with pre-set privacy rule are generated, we measure achieved level of privacy on per-session basis using values of both Pl_d and Pl_s (as defined in Sect. 3.3).

The **Location Receiver** component receives a location object, which includes the user device's geo-coordinates from location providers, and sends it over to the *Private Location Manager* for further processing.

4.2 Middleware Implementation

PL-Protector orchestrates a mobile platform based location protection service on Android to modify the location resource handling process. The middleware communication requires process isolation and IPC services; hence, minimising the requirements placed on hardware or OS modifications.

PL-Protector Life Cycle. In Android, there are two methods to access user's location: (1) Location Manager Service (Old), and (2) Fused Location Manager Service (New) that are part of Google Play Services. However, both methods require the app to request a callback function to get regular updates by registering a location listener. The app receives a new location object when a new location is available, the callback function is invoked (Fig. 1 (left)). Modifying these two Google services is complicated, but we make PL-Protector communicate with the location requesting apps by intercepting the location object before it reaches requesting apps (Fig. 1-(right)). One of the main task is to add a system

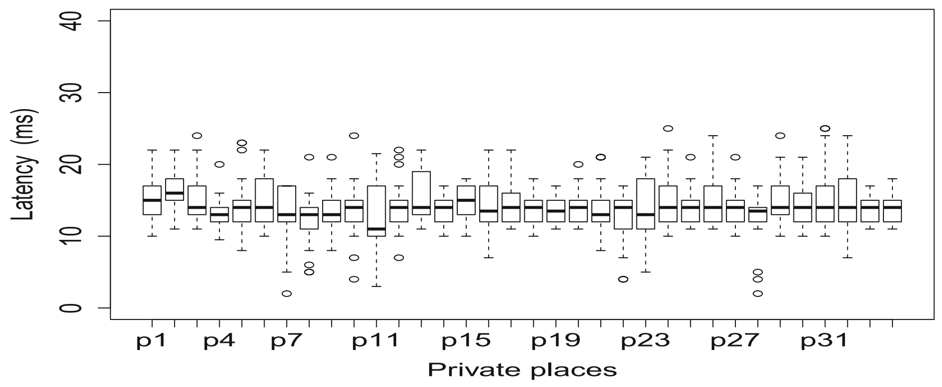


Fig. 3. PL-Protector's overall computation latency caused at 34 distinct private places.

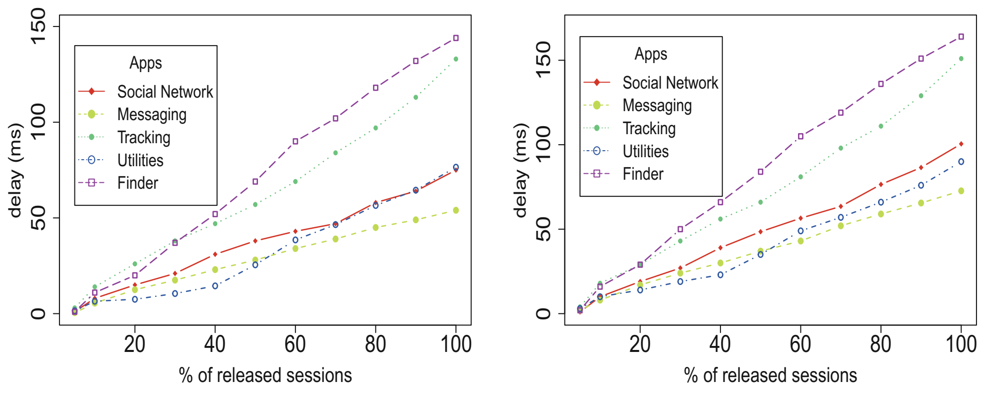


Fig. 4. Apps communication overhead without (left) and with (right) PL-Protector.

service, where the class belongs to the location APIs. Thus, the PL-Protector's service is placed in the `android.location package`, which detects private locations via WiFi APs and can also be used by other components when calling context. In Android, a context allows an app to interact with the OS resources. Similar to [11], we add a static context field to the location class, which will be populated when the app is invoked; this enables PL-Protector to know which app is currently requesting the location object, and also communicate with the OS. Besides, `Fused Location Manager` combines sensors, GPS, Wi-Fi, and cellular data into a single API for location-based applications [14], hence separating data from `GPS_PROVIDER` and `NETWORK_PROVIDER` is no longer straight forward. PL-Protector addresses this issue by preventing app's location request to reach the `Fused Location Manager` that collects and sends the network session data to the location provider. Instead, the requested location is retrieved from the on-device cache, and then, it is sent to the requesting app (with privacy rules applied).

Bootstrapping. When PL-Protector first boots and before turning 'ON' the location sharing setting, the user will have to perform an initial setup. This will allow WiFi AP scanning, input geo-coordinates and set privacy choices using *User Interfaces* (UI) (Fig. 2 - left). PL-Protector's UI incorporates a map to get the corresponding geo-coordinates so achieving an effective privacy without affecting the location accuracy. At the same time, this prevents non-authorised sharing of device's exact location and network session data. The UI (Fig. 2 - middle and right) enables users to set and manage their private locations and apps distinctly.

5 Evaluation

We evaluate PL-Protector in terms of QoS, communication and computational overheads.

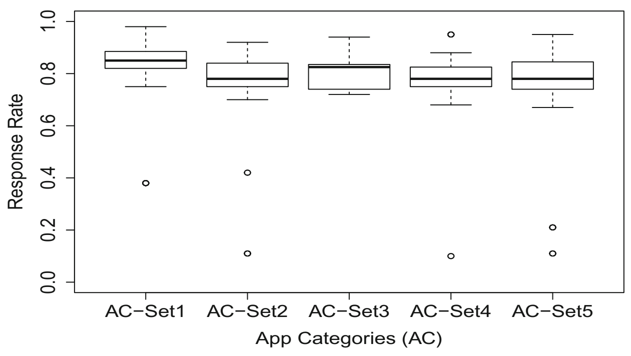


Fig. 5. Total response rate by PL-Protector when interacting with apps

5.1 Experimental Setup

We deployed PL-Protector middleware on a Nexus 6 with Android 6.0 (API 23) and ported apps of five different LBS queries categories namely (1) Social Networking (e.g., Facebook), (2) Instant Messaging/chatting (e.g., Whatsapp), (3) Tracking (e.g., Fitness), (4) Utilities (e.g., Weather, Alarm, etc.) and (5) Finder (PoI Finder/Geo-search). Based on app operations, we assume that both types (1) and (3) require continuous access to location data; whereas, types (2), (4) and (5) involve sporadic access. We have collected empirical data from a number of sessions running at different time intervals over a period from 1 to 6 months. We then created two datasets at 34 selected private places. In the first dataset, we include the ported apps' session data running over the conventional Android environment without interacting with PL-Protector. The second dataset consists of the same apps running in the presence of PL-Protector.

5.2 Impact on the Quality of Service

Crucial for its functionality, we measure latency as the time PL-Protector takes to interact with the app and perform an entire computational cycle, i.e., to compute the location on-device and to apply the privacy rules. On average, PL-Protector presents a latency lower than 22 ms upon all the location-access calls executing PL-Protector's privacy controls at runtime (as shown in Fig. 3). The reason for increased latency is due to PL-Protector's load time, and cross-process/IPC service transfers of location updates. However, this latency is smaller than 100 ms and, thus, small enough to not cause user-noticeable delays while utilising apps on the device. Furthermore, Fig. 4 show the communication overhead for the 5 app categories and compares both execution environments. In per-location access sessions, we found <19 ms delays when continuous location updates, and <8 ms delay for sporadic location updates. Thus, PL-Protector is suitable to run all the existing apps of aforementioned five LBS categories since their core functionality already accepts delays in this range.

5.3 Cache Accuracy

To analyse the accuracy of the on-device cache method at runtime, we measure cache hits and misses that includes three possible outcomes: 1. *The location is cached and up-to-date*, 2. *The location is cached but is out-of-date*, and 3. *The location is not cached*. The total observed response rates range between 70% to 90% accuracy (Fig. 5) that demonstrates the suitability of PL-Protector's on-device location computation process with types of apps requiring both sporadic and continuous location-updates. This indicates PL-Protectors's on-device cache update frequency is within practical limits, and it provides accurate location data at runtime.

6 Conclusion

In this paper, we have presented the design, deployment and evaluation of PL-Protector, a location privacy-enhancing middleware, which minimises the interaction and data collection from wireless access points, content distributors and location providers. The middleware also provides a new series of control settings to prevent user information disclosure during formation of LBS queries. PL-Protector enforces these privacy rules over both the information and control flows occurred between sources and sinks. We have fully implemented PL-Protector on Android 6 and validated with real apps from five different LBS queries categories. Experiments demonstrated acceptable delay overheads and within efficient and practical limits. Immediate future work pursues analysis on the threat model to read its compliance with the three privacy settings and measure achieved level of privacy. Additionally, we plan deployment improvements related to the on-device computation and cache storage, i.e., by incorporating PL-Protector as a part of Android custom Read-Only-Memory. We are also planning to conduct a usability study that will allow us to enhance PL-Protector's privacy and usability rates.

References

1. Almuhimedi, H., Schaub, F., Sadeh, N., Adjerid, I., Acquisti, A., Gluck, J., Cranor, L.F., Agarwal, Y.: Your Location has been Shared 5,398 Times! A Field Study on Mobile App Privacy Nudging. In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, pp. 787–796. ACM (2015)
2. Amini, S., Lindqvist, J., Hong, J., Lin, J., Toch, E., Sadeh, N.: Caché: caching location-enhanced content to improve user privacy. In: Proceedings of ACM International Conference on Mobile Systems, Applications, and Services, pp. 197–210. ACM (2011)
3. Android Developer Reference: March 2016. http://developer.android.com/reference/
4. Ardagna, C.A., Livraga, G., Samarati, P.: Protecting privacy of user information in continuous location-based services. In: 2012 IEEE 15th International Conference on Computational Science and Engineering (CSE), pp. 162–169. IEEE (2012)

5. Bugiel, S., Heuser, S., Sadeghi, A.R.: Flexible and fine-grained mandatory access control on android for diverse security and privacy policies. In: Usenix Security, pp. 131–146 (2013)
6. Cranor, L.F., Sadeh, N.: A shortage of privacy engineers. IEEE Secur. Priv. **2**, 77–79 (2013)
7. Enck, W., Gilbert, P., Han, S., Tendulkar, V., Chun, B.G.: Others: TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. TOCS **32**(2), 5 (2014)
8. European Commission: Protection of personal data (2016). http://ec.europa.eu/justice/data-protection/
9. Faruki, P., Bharmal, A., Laxmi, V., Ganmoor, V., Gaur, M.S., Conti, M., Rajarajan, M.: Android security: a survey of issues, malware penetration, and defenses. IEEE Commun. Surv. Tutor. **17**(2), 998–1022 (2015)
10. Fawaz, K., Feng, H., Shin, K.G.: Anatomization and protection of mobile apps' location privacy threats. In: 24th USENIX Security Symposium (USENIX Security 15), pp. 753–768. USENIX Association (2015)
11. Fawaz, K., Shin, K.G.: Location privacy protection for smartphone users. In: Proceedings of ACM Conference on Computer and Communications Security, pp. 239–250. ACM (2014)
12. Google Weave: Weave. https://developers.google.com/weave/
13. Guha, S., Jain, M., Padmanabhan, V.N.: Koi: a location-privacy platform for smartphone apps. In: Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, pp. 14–14. USENIX Association (2012)
14. Hellman, E.: Android Programming: Pushing the Limits. Wiley, Hoboken (2013)
15. IETF: Geographic Location Privacy, March 2016. http://datatracker.ietf.org/wg/geopriv/charter/
16. Michael, K., Clarke, R.: Location and tracking of mobile devices: Überveillance stalks the streets. Comput. Law Secur. Rev. **29**(3), 216–228 (2013)
17. Muslukhov, I., Boshmaf, Y., Kuo, C., Lester, J., Beznosov, K.: Understanding users' requirements for data protection in smartphones. In: IEEE International Conference on Secure Data Management on Smartphones and Mobiles, pp. 228–235. IEEE (2012)
18. Niu, B., Li, Q., Zhu, X., Cao, G., Li, H.: Enhancing privacy through caching in location-based services. In: Proceedings of IEEE INFOCOM (2015)
19. Patel, A., Palomar, E.: Privacy preservation in location-based mobile applications: research directions. In: Proceeings of IEEE International Conference on Availability, Reliability and Security (ARES), pp. 227–233. IEEE (2014)
20. Patel, A., Palomar, E.: LP-Caché: Privacy-aware cache model for location-based apps. In: Proceedings of the 13th International Conference on Security and Cryptography (SECRYPT), pp. 183–194 (2016)
21. Pontes, T., Vasconcelos, M., Almeida, J., Kumaraguru, P., Almeida, V.: We know where you live: privacy characterization of foursquare behavior. In: Proceedings of ACM Conference on Ubiquitous Computing, pp. 898–905. ACM (2012)
22. Robusto, C.C.: The cosine-haversine formula. Am. Math. Mon. **64**(1), 38–40 (1957)
23. Samsung: smartthings. http://www.samsung.com/uk/smartthings/
24. Shklovski, I., Mainwaring, S.D., Skúladóttir, H.H., Borgthorsson, H.: Leakiness and creepiness in app space: perceptions of privacy and mobile app use. In: Proceedings of ACM Conference on Human factors in computing systems, pp. 2347–2356. ACM (2014)

25. Wernke, M., Skvortsov, P., Dürr, F., Rothermel, K.: A classification of location privacy attacks and approaches. Pers. Ubiquit. Comput. **18**(1), 163–175 (2014)
26. Xie, J., Knijnenburg, B.P., Jin, H.: Location sharing privacy preference: analysis and personalized recommendation. In: Proceedings of the 19th International Conference on Intelligent User Interfaces, pp. 189–198. ACM (2014)
27. Zhu, X., Chi, H., Niu, B., Zhang, W., Li, Z., Li, H.: Mobicache: when k-anonymity meets cache. In: GLOBECOM, pp. 820–825. IEEE (2013)

Protecting Smartphone Users' Private Locations through Caching[⋆]

Asma Patel and Esther Palomar

School of Computing and Digital Technology
Birmingham City University, UK
{asma.patel,esther.palomar}@bcu.ac.uk
http://www.bcu.ac.uk/

Abstract. Smartphones equipped with advanced positioning technology continuously collect users' location information and make that information easily accessible to third party app and/or library developers. Users are becoming increasingly aware of the resultant privacy threats, and demanding effective privacy preserving solutions that will allow them to securely use location-based services. In addition, academic and industrial communities are paying special attention to the development of more friendly and socially-accepted approaches to location privacy. In this work, we model, design and evaluate LP-Caché, a mobile platform based service that protects locations by modifying the location resource handling process. It applies caching technique to protect users' private locations and establishes personalised location permission controls. We define the design decisions and implementation requirements towards the viability and feasibility of the model deployment. We also evaluate resources and storage requirements in order to minimise the computational and communication overheads. Empirical results of 2 months comparative study show a 2.26% change in the network fingerprints at 34 distinct places that required only 2.07% change in the overall cache storage. Both these results demonstrate feasibility of the model.

Keywords: Location Privacy, Location-based Services, Smartphones, Caching, Location-based Applications

1 Introduction

The explosive growth of context-aware mobile apps has leveraged tremendous opportunities for a whole new class of Location-Based Services (LBS) [32]. Geo-marketing and geo-social networking, location-based games, monitoring, assisted eHealth, and energy consumption 3D maps represent a small subset of the third-party apps nowadays available as LBS and can certainly pose a serious threat to the users' privacy [26, 33].

[⋆] Updated and extended version of SECRYPT 2016 conference paper with title "LP-Caché: Privacy-aware Cache Model for Location-based Apps."

Currently, approaches to privacy settings of user location on smartphones[1] are based on a binary process[2]. Users are forced to rely on third party service providers that in many cases continuously collect, use and share their location data, and in some cases even prompt the user to give away their position on page load [2, 26, 16, 33]. Moreover, both academia and industry agree on the urgent need of adopting a Privacy-by-Design (PbD) approach for the development of more user-friendly and socially-accepted solutions to location privacy preservation on their mobile products and services [9].

To encounter these challenges, in [31] the authors introduced the model called *Location Privacy Caché* (LP-Caché). LP-Caché envisions beyond the simple grant/deny access method and provides the user with advanced mechanisms to decide the extent of disclosing location data with service providers. Several caching based solutions [40, 3, 29] have been proposed to minimise the risk of major location privacy threats, but lacking of deployment feasibility. They rely on unrealistic assumptions such as vast cache data storage requirements, or on the app developers modifying the code to incorporate their cached databases. LP-Caché incorporates caching technique to determine users' geographical location in a privacy preserving manner, and with minimum cache storage requirements.

In this paper we overview the main contributions presented in [31] and, further prove LP-Caché's features in an extended experimental setting. In particular, we describe

– A detailed analysis of the current location computation process deployed in smartphones when running location-based apps.
– A detailed definition of the LP-Caché model and architecture as well as its main implementation requirements.
– A complete performance evaluation of LP-Caché, analising the wireless access point data availability and consistency, and the estimated user resource and storage requirements. We will also show that LP-Caché is feasible without modifying installed apps. Estimated storage requirements and monthly datasets of wireless acess points have been analysed. Results from the extended experimental setting help us to determine the scalability of LP-Caché.

The rest of the paper is organized as follows. Section 2 outlines the current location computation process and its evaluation. Section 3 reviews the related work. Section 4 presents the design and architecture of LP-Caché, and Section 5 fully elaborates on design decisions and implementation requirements. We evaluate the feasibility of WiFi APs availability, resources and storage requirements in Section 6. Finally, Section 7 concludes and describes current work as well as sets future research plans.

[1] Throughout this paper, we use the terms smartphones and mobile interchangeably
[2] Data protection directives and acts [14, 20] across the globe state that personal data should not be disclosed or shared with third parties without consent from subject(s). Such a consent is typically obtained by mandatory acceptance of the conditions mentioned in the End User License Agreement (EULA), or through opt-out possibilities and other regulations[25].

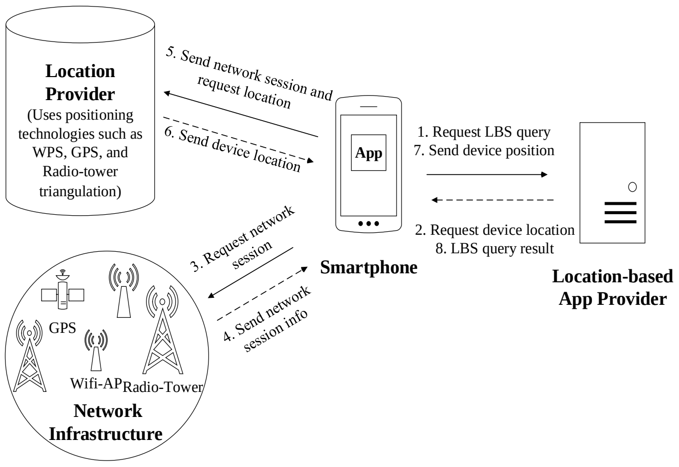

Location Provider
(Uses positioning technologies such as WPS, GPS, and Radio-tower triangulation)
5. Send network session and request location
6. Send device location
App
Smartphone
1. Request LBS query
7. Send device position
2. Request device location
8. LBS query result
Location-based App Provider
3. Request network session
4. Send network session info
GPS
Wifi-AP
Radio-Tower
Network Infrastructure

Fig. 1. Current location computation architecture

2 Overview of the Current Location Computation Process

In this section, we describe roles and processes involved in the current architecture for computing user's device location.

2.1 Current Architecture

The current location computation architecture to use location-based apps on smartphones comprises four main entities: 1. Smartphones with installed apps, 2. App Provider, 3. Network Infrastructure, and 4. Location Provider. This architecture (Figure 1) mainly relies on third party location providers, e.g., Google Location Service [18], Skyhook [34], and Navizon [27]. The location provider represents the central database, which maps the received signatures of nearby wireless access points to the geo-coordinates, i.e., latitude and longitude, so handling every geo-location request. Therefore, the location provider has constant access to the user's location as well as to the trajectory data. To respond to any location request, the location provider maintains a database of surrounding network infrastructure, including WiFi Access Points (APs), cellular-towers, and IP addresses, which must be mapped to their exact geographical co-ordinates. Compared to GPS and cell-tower based positioning, WiFi Positioning Systems (WPS) is nowadays considered as a very accurate method for location calculation [34]. Location providers rather use enhanced WPS than GPS, primarily due to current smart-mobile devices benefit from built-in WiFi clients that perform faster than most expensive GPS receivers. This enables the service provider to get user's precise location at all times and, as a result, more effective privacy preservation measures are needed in the current process to mitigate privacy threats.

WiFi APs continuously announce their existence in the way of network frames/beacons and transmit their Service Set Identifier (SSID) and Basic Service Set Identifier (BSSID)/MAC addresses. Location providers use these WiFi APs identifers to create network signatures and map them with geo-coordinates, also called geolocation. IEEE 802.11 states two standardised ways to collect beacons from WiFi APs: 1. Active scanning, and 2. Passive scanning. Location providers are capable of deploying systems with either active scanning, passive scanning, or both together. Location providers use three different ways to collect geo-location of WiFi APs:

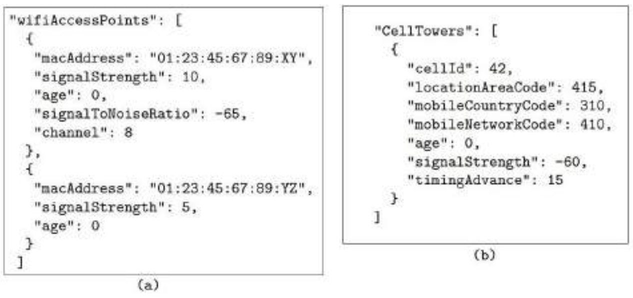
1. *Statically-* They collect WiFi beacons by the so called *wardiving* process. Basically, they map the equipped vehicle's exact geo-coordinates along with the signal strength of the captured beacons from surrounded APs.
2. *Dynamically-* They can collect data from WiFi APs automatically once the user device uses location services, e.g. Maps and Navigation applications. The user device as configured to be geolocated acquires unique identifiers from the surrounding WiFi APs, even if the network is encrypted, and then sends it over to the location provider in order to perform geolocation calculation. The collected information is utilised to build and update the database autonomously, for example, by applying crowdsourcing [41].
3. *User input-* They encourage users to manually input the WiFi APs' information, i.e., BSSID and the geo-coordinates, into their databases, e.g., Skyhook[3] to register WiFi APs.

2.2 Evaluation of Current Location Computation Process

We conducted a series of experiments on different mobile devices installed with *Android*, *Windows Phone*, and *iOS* operating systems to categorise the data flow in the current location computation process. With the assistance of sniffers, such as *Wireshark* [39] and *tPacketCapture* [36], we captured and analysed sequence and location data transmission when using location-based apps, e.g., Navigation and Friend Finder.

Observation. These experiments were designed to understand whether there is any difference on the location calculation process on each of these three mobile operating systems. Based on the results, all of them display common patterns of location data retrieval. The user device collects the unique identifiers from the surrounding network along with GPS data, and sends it to the location

[3] Submit a Wi-Fi Access Point. See http://www.skyhookwireless.com/submit-access-point (last access in March 2016).



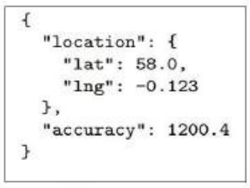
```
"wifiAccessPoints": [
  {
   "macAddress": "01:23:45:67:89:XY",
   "signalStrength": 10,
   "age": 0,
   "signalToNoiseRatio": -65,
   "channel": 8
  },
  {
   "macAddress": "01:23:45:67:89:YZ",
   "signalStrength": 5,
   "age": 0
  }
]
```

(a)

```
"CellTowers": [
  {
    "cellId": 42,
    "locationAreaCode": 415,
    "mobileCountryCode": 310,
    "mobileNetworkCode": 410,
    "age": 0,
    "signalStrength": -60,
    "timingAdvance": 15
  }
]
```

(b)

Fig. 2. Structure of (a) WiFi AP object and (b) cell-tower object sent to the location provider [31].



```
{
  "location": {
    "lat": 58.0,
    "lng": -0.123
  },
  "accuracy": 1200.4
}
```

Fig. 3. Structure of the location object received from the location provider [31].

provider to get the exact device location. Figure 2 shows the structure of the WiFi and Cell-tower objects sent to the location provider. Once calculation is performed, the location provider sends to the device the precise location in the way of a geo-location object containing geo-coordinates. Figure 3 represents the structure of the location object received from the location provider. In short, the app developer over any mobile platform can utilize this location object to get the user's geo-location with no need of focusing on the details of the underlying location technology. In the following section, we give the detailed description of the process sequence.

Process Sequence. Figure 4 illustrates the sequence of processes and messages involved in the current location computation architecture. Note that, on a smartphone, location sharing service settings must be 'ON' while using any location-based app. If the location sharing is 'OFF', then the device prompts for changing the setting from 'OFF' to 'ON'; otherwise, user cannot use the service. Once the app obtains the location object from OS, it is then used by the app provider to send the corresponding reply to LBS query via the standard programming interface/API [5].

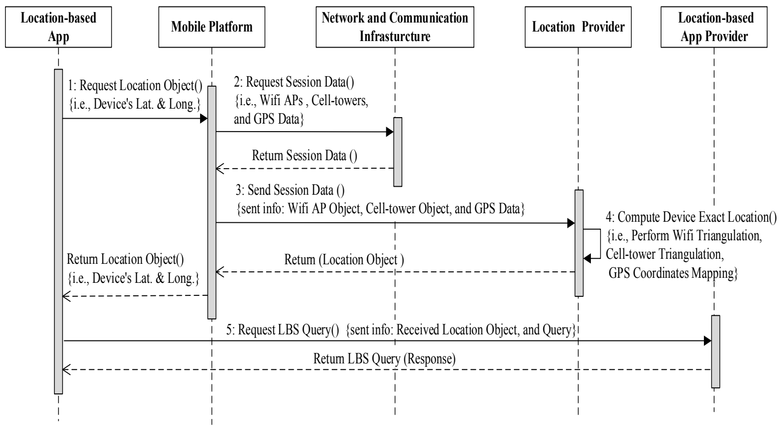

Location-based App
Mobile Platform
Network and Communication Infrasturcture
Location Provider
Location-based App Provider
1: Request Location Object() {i.e., Device's Lat. & Long.}
2: Request Session Data() {i.e., Wifi APs , Cell-towers, and GPS Data}
Return Session Data ()
3: Send Session Data () {sent info: Wifi AP Object, Cell-tower Object, and GPS Data}
4: Compute Device Exact Location() {i.e., Perform Wifi Triangulation, Cell-tower Triangulation, GPS Coordinates Mapping}
Return Location Object() {i.e., Device's Lat. & Long.}
Return (Location Object)
5: Request LBS Query() {sent info: Received Location Object, and Query}
Return LBS Query (Response)

Fig. 4. Sequence diagram of current location computation process [31].

3 Related Work

Existing approaches to preservation of the location privacy can be classified into three categories: 1. Mobile Platform[4], 2. Location Query, and 3. Privacy-aware Network Communication.

3.1 Mobile Platform

A few studies have proposed static and dynamic methods to detect privacy leaks in mobile platforms. The former method statistically analyses apps by creating permission mapping, generating call graphs and data flow analysis to report privacy leaks for further auditing, e.g, AndroidLeaks [17] and PiOS [12] for Android and Apple iOS, respectively. The application of dynamic methods involves modification of the existing mobile platform. For example, TaintDroid [13] adds taint tracking information to sensitive sources calls from apps, and it tracks location data flow as it generated through applications during execution. MockDroid [8] relies on instrumenting `Android's` manifest permission mechanism to mock sensitive data from OS resource, including location data, which can affect apps' usability and functionality. LP-Caché not only monitors the location sources but also modifies, if required, the generated location data based on defined user permissions. In another attempt[15], *indistinguishability* technique is applied as location privacy preservation mechanism into the advertising and analytics libraries as well as on installed apps; however, it does not give control

[4] Throughout this paper, we use the terms mobile platform and operating system interchangeably

on the amount of WiFi and location data that is being shared with the location provider. Moreover, *indistinguishability* technique increases computational overhead on smartphones.

3.2 Location Query

Apps share location information with the provider in the form of LBS queries. The transmission of such queries to the location server may allow attackers to gain access to user location data. Privacy Enhancing Techniques (PET) like k-anonymity, dummy locations, region cloaking, location perturbation and obfuscation, and mix-zone pseudonyms have been applied to different architectures for location query formation and privacy preservation from LBS providers [30, 37, 22]. Most of these techniques rely on theoretical assumptions - like trusted infrastructure to provide the privacy protection, requiring a group of similar app users to be at the same time and same place. The main issue with PETs and cryptographic schemes is that it relies entirely on the data collection servers to comply with location privacy.

Caching. Several authors have used caching scheme along with PETs to build to a database consisting of different contents/datatypes used within location based queries to be re-used in future LBS queries. MobiCaché [40] applies k-anonymity for caching location based queries. Similarly, Niu et al. [29] attempt to improve k-anonymity based caching by adding dummy locations. Both proposals require a trusted infrastructure to maintain privacy. Caché [3] maintains a local cache within the device to reuse the data types available from applications in future location based queries; however, storing entire LBS query data increases the cache storage requirements. Besides, Caché also requires app developer to modify the way app access location data. By contrast, LP-Caché caches the network fingerprints and geo-coordinates, which reduces the storage overhead drastically; it considers installed apps as black box, and therefore, does not require app developer to modify the code, it works as a middleware between the app and the mobile platform. All these cache-based systems either intent to generalise or obfuscate the LBS query or minimise the number of queries sent to the app providers, but they do not provide privacy from WiFi content distributors. Besides, mobile devices not only send vast amounts of location data to app providers but also to location providers creating different location privacy shortcomings [2, 33]. In this regard, limited work has been published on privacy preservation from the location provider's perspective [10, 11]. Damiani (2011) proposes a theoretical approach for privacy-aware geolocation-based web services to encourage further research to minimise the amount of location data being shared with the location provider. This is mainly due to that the location provider is considered as the only source to get the user location when developing any location-based app. In LP-Caché, we minimise the process of wireless AP data collection by the WiFi content distributors or location providers, and we control information disclosure within the generated LBS query (e.g., points of interests (POIs) and nearest neighbor) since it will be sent to the third-party app provider.

3.3 Privacy-aware Network Communication

Besides location queries, device's IP address can also reveal user's private locations. To this regard, anonymous communication protocols, e.g., Anonymizer [6] and TOR [35], deal with anonymous service usage at the network layer while communicating over Internet (i.e., the server cannot infer user's location via received device's IP address along with the location query), and they are most prominent and commonly used network layer solutions.

4 LP-Caché Model

In this section, we describe LP-Caché's threat model, design goals, architecture and main processes' sequence diagram.

4.1 Threat Model

Apps deliberately collect user's sensitive data, including location and other sensitive information as part of their operations. User tracking, identification and profiling (i.e. personal habits, movement patterns, etc.) are fundamental threats to location privacy [16, 37]. Furthermore, the current direct link of smartphones to the location provider and the continuous flow of LBS queries that include device's exact geo-coordinates over network create a serious risk to the protection of users' sensitive information, even more challenging, in the presence of a malicious location provider and via advanced network sniffing practices.

LP-Caché computes the exact location within user device, without service provider's involvement, and trusts the device on the storage of sensitive data. However, the user has still the option of giving consent for app providers or location provider to access their location. Mobile network providers might, however, collect user location data via cellular clients. It is also excluded from our model the option of manually inserting the location data (e.g., street name, zip code, post code) within LBS query.

4.2 LP-Caché Control Flow Architecture

LP-Caché's three main design goal are: 1) the third-party app provider will not be able to infer the device's exact location without getting uses's consent; 2) the user can set distinct privacy preferences for different apps and private places; and 3) the model works independently without the need of modifying the app's code. Figure 5 depicts the block diagram for LP-Caché architecture; its main components are:

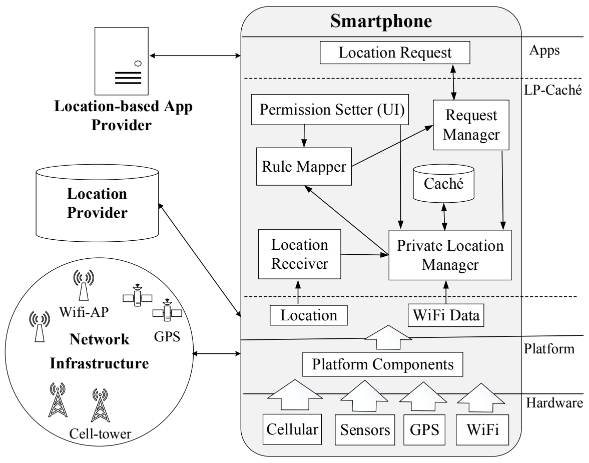


Fig. 5. LP-Caché architecture [31].

Permission Setter is the user interface (UI), which enables users to set and manage their private places and apply improved personalised permissions when running installed location-based apps. Once received the user inputs, pre-set private locations are sent to the *Private Location Manager* module, and permissions are sent to the *Rule Mapper* Module.

Request Manager is responsible to intercept the event of location access calls, and then lead the app's control flow to the *Private Location Manager* module. Besides, it will also be in control of receiving the processed user location (i.e., could be either anonymised or altered) from *Rule Mapper*, and then delivering it to the app in order to maintain every session's control flow.

Private Location Manager module's main task is to detect unique identifers of the surrounding WiFi APs and compare them with the stored network fingerprints to determine whether the user is within the set of private places. User inputs from the *Permission Setter* will create network fingerprints for known private locations, which are then added or updated in the *Cach* database. Moreover, it maintains a binary flag to detect private places. In the case of a hit the location data is sent to the *Rule Mapper*. Otherwise, the location is received from the *Location Receiver*. Whenever the *Private Location Manager* receives a new private place request, the received location is mapped to the detected network fingerprint and stored in the *Caché* database.

Rule Mapper dynamically collects and checks set permissions from *Permission Setter*. Once the representative location object is received from the *Private Location Manager*, it applies the user permissions on the location co-ordinates, alters them (if required), and outputs the processed location

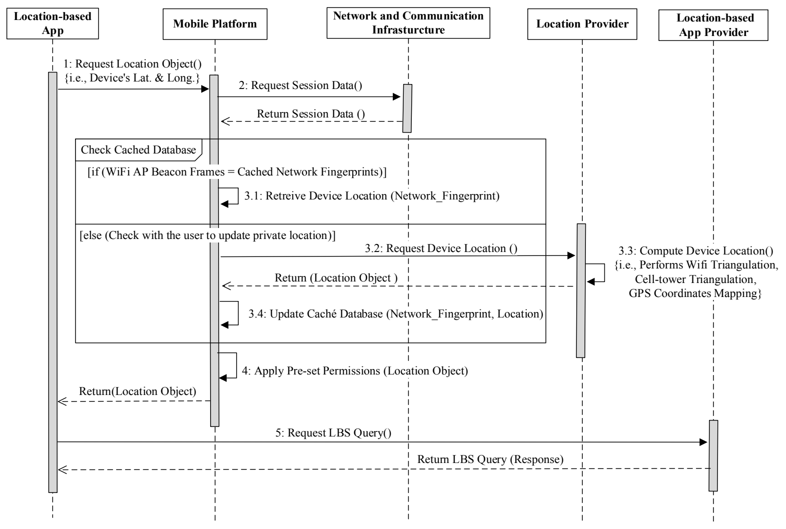

Location-based App
Mobile Platform
Network and Communication Infrasturcture
Location Provider
Location-based App Provider
1: Request Location Object() {i.e., Device's Lat. & Long.}
2: Request Session Data()
Return Session Data ()
Check Cached Database
[if (WiFi AP Beacon Frames = Cached Network Fingerprints)]
3.1: Retreive Device Location (Network_Fingerprint)
[else (Check with the user to update private location)]
3.2: Request Device Location ()
3.3: Compute Device Location() {i.e., Performs Wifi Triangulation, Cell-tower Triangulation, GPS Coordinates Mapping}
Return (Location Object)
3.4: Update Caché Database (Network_Fingerprint, Location)
4: Apply Pre-set Permissions (Location Object)
Return(Location Object)
5: Request LBS Query()
Return LBS Query (Response)

Fig. 6. Sequence diagram of location computation process using proposed model [31].

to the *Request Manager* module. If the flag is negative, then it forwards the exact location.

Cache is the established on-device cached database, and it is routinely queried by the *Private Location Manager* module, which can add, update and delete the cached location data. The locations in cach are those which are to be protected, and they can also represent regions of space. Each entry is recorded along with a network fingerprint and geo-location that is acquired from the location provider.

Location Receiver module receives a location object, which includes the user device's geo-coordinates (as in Figure 3), from location providers and sends it over to the *Private Location Manager* for further processing.

4.3 Process Sequence

LP-Caché modifies the current location resource handling process; however, the involved entities (as in Section 2) remain the same. Figure 6 illustrates the sequence of processes and messages involved in LP-Caché:

1. At the event of app requesting the device location, our service will intercept the request to get the location from the cache database instead of sending the request to the location provider.
2. Upon receiving the location request, our service will scan the surrounding network infrastructure.

3. Using observed network frames our service will execute as follows:
   (a) Our service compares the collected beacons with the stored network fingerprints to retrieve corresponding stored representative location coordinates.
   (b) In the case of an unmatched entry on the database, the LP-Caché prompts the user two options either input the location using UI, or allow the query to be sent to location provider that will calculate and send the current location coordinates. Note that this will only occur if the user has set the current location as private but the geo-coordinates are not cached.
   (c) The received location data for the encountered APs will be tracked within the local cache database for future use.
4. User location coordinates can be altered based on the privacy settings. LP-Caché provides three options for controlled information disclosure: (1) Adjust Location Granularity, (2) Obfuscate Location, and (3) No Change. Computed location is populated in the location object and sent to the app.
5. Once the app obtains the location object, it is then used by the app provider to send the corresponding reply to LBS query via the standard programming interface/API [5].

5 LP-Caché Implementation Requirements

In the following sections, we describe LP-Caché's implementation requirements. LP-Caché orchestrates a mobile platform based location protection service to modify the location resource handling process. For instrumenting the LP-Caché implementation, `Android` will be the best choice since it is open source; however, it can also be implemented on other permission-based mobile platforms.

5.1 Bootstrapping

LP-Caché aims to protect user's private places. Initially, LP-Caché does not have enough information to function, the two main required information are private places's network fingerprints and geo-coordinates. LP-Caché cannot collect network fingerprints and geo-cordinates for private places at runtime, as by the time we have this information, other installed apps will have access to it. Therefore, when LP-Caché first boots and before turning 'ON' location sharing settings, user will have to do the initial setup, which includes allow WiFi AP scanning, input geo-cordinates and set privacy choices (see Section 5.4). In 2013, Google presented a new service API (also works on older `Android` versions) for location-based apps that allows developers to use the new and advanced *Location and Activity API*, i.e., they changed `Location Manager` to `Fused Location Manager`, hence combining sensors, GPS, Wi-Fi, and cellular data into a single API for location-based applications [19]. As a result, separating data from `GPS_PROVIDER` and `NETWORK_PROVIDER` is no longer straight forward. LP-Caché addresses this issue by preventing app's location request to reach the `Fused Location Manager` that collects and sends the network session data to the location provider. Instead, the requested location is retrieved from the on-device cache, and then, it is sent to the requesting app (with privacy rules applied). Besides, geographic tool[5] can be incorporated in the LP-Cache's UI to get the corresponding geo-cordinates. This will allow LP-Caché to achieve effective privacy without affecting location accuracy, at the same time, prevent from the non-authorised sharing of device's exact location and network session data.

5.2 Mobile Platform

For performance evaluation, there are two possible ways of implementing LP-Caché location protection service. The first requires modifying the app's location accessing interfaces and intercepting location updates before they reach the app provider. Whereas, the second option requires modifying the platform and changing the location data before reaching the app.

App Code Modification. This comprises unpacking the app, rewriting the code to work according to the new rules, and then repackaging it again, e.g.,[21]. However, app repackaging changes the signature and stops future updates, and therefore, affects its functionality. Another way to modify app's location accessing interfaces is through the creation of an `Android` service and allowing apps to register with it. Then, Apps can use the location data provided by this service. This approach is easy to implement but relies heavily on app developers to modify their app's code, which is highly infeasible and unrealistic. Nonetheless, this approach can be used as simulated testing environment for any developed service.

Platform Modification. For the sake of experimentation, we develop LP-Caché via platform modification. One of the main task is to add a system service, where the class belongs to the location APIs; thus, it is placed in the `android.location package`, which detects private locations via APs and can also be used by other components when calling context. In Android, a context allows an app to interact with the OS resources. Another task is to make LP-Caché communicate with location requesting apps. On `Android` there are two methods to access user's location: 1) Location Manager Service (Old), and 2) Fused Location Manager Service (New) that is a part of Google Play Services. Both methods require the app to request a callback function to get regular updates by registering a location listener. The app receives a new location object when a new location is available, the callback function is invoked. Modifying these two Google services is complicated, but there is a possibility to intercept the location object before it reaches requesting apps. We will add a static context field to the location class, which will be populated when the app is invoked; this will enable us to know which app is currently requesting the location object, and also communicate with the OS [15]. The created location object will have reference to the requesting app's context, and therefore, it can interact with our external service.

[5] LatLong is a geographic tool. See http://www.latlong.net/ (last access in March. 2016)

Algorithm 1 Location Calculation Algorithm [31]

```
Require: $n_x$: Network Frames
Ensure: $l_r$: Representative Location
 1: $n_x = 0$
 2: read $n_x$
 3: while $n_x \neq$ null do
 4:     if $n_x = n_i$ , $\forall\ i \in p$  then
 5:         (step 1) retrieve the corresponding $l_r$
 6:         add flag $f$ = (if private 1, else 0)
 7:         send $l_r$
 8:     else
 9:         (step 2) request $l_r$ from user or location provider
10:         set received $l_r$ to corresponding $p_i$
11:         update $c$
12:         send $l_r$
13:     end if
14: end while
```

5.3 On-device Cache Database Creation

LP-Caché requires fixed wireless APs data (i.e., 802.11) to create cached database of private locations. Initially, we decided to focus on WPS since it infers accurate user location. However, we can later include other fixed radio sources (e.g., cell-tower unique identifiers).

Network Fingerprinting. We can distinguish two main types of WPS techniques to determine the position of client devices with respect to APs [7]: 1) Signal trilateration and 2) Fingerprinting. The former undertakes trilateration of Received Signal Strength (RSS), Angle of Arrival (AoA), and Time of Flight (ToF) from observed APs, and the later involves mapping observed APs signatures with a stored database. LP-Caché uses fingerprinting to create cached location database; however, fingerprinting performance is highly related to the number of APs. Therefore, in Section 6 we have evaluated WiFi AP availability and consistency. The detected network management frames/beacons are mapped with the device's representative geo-location to create a network fingerprint, which is then stored in the local cached database, an example private location fingerprint is shown in Equation 1. Moreover, to reduce storage and computation overhead, our model only caches network fingerprints of private places (e.g., home, work, frequently visited places or particular stores), and it relies on user input for initial pre-set up. The user will have to select the option (via LP-Caché UI) to set current location as private place p_i, and then fingerprint will be recorded. Later, the private place will be detected automatically with respect to observed beacons $[n_x]$, such that

$$p_i = [n_1], [n_2], ..., [n_x] \rightarrow [l_r] \tag{1}$$

where p_i represent ith private place IDs, n is the scanned beacon, and l_r is a representative location for that private place. WiFi AP beacons $[n_x]$ consists of

Algorithm 2 Enhanced Permissions Algorithm [31]

```
Require: $l_r$: Representative Location
Ensure: $l'_r$: Processed Location
 1: $u_p$ = User Input
 2: read $l$, $l_g$, $f$, $u_p$
 3: if $u_p$ = Adjust Granularity then
 4:     check granularity level $g_l$
 5:     truncate($l$, $l_g$)
 6:     replace $l$ to $l'$ and $l_g$ to $l'_g$
 7:     return $l'_r$
 8: else if $u_p$ = Obfuscate then
 9:     randomly generate angle $\theta$
10:     obfuscate($l$, $l_g$, $\theta$)
11:     replace $l$ to $l'$ and $l_g$ to $l'_g$
12:     return $l'_r$
13: else
14:     unchanged
15:     return $l'_r$
16: end if
```

four attributes ⟨SSID, BSSID/MAC address, Signal-strength, and Timestamp⟩. The private representative location $[l_r]$ consists of a tuple ⟨*Lattitude*, *Longitude*, and *Accuracy*⟩.

In LP-Caché, to set up network fingerprints at every private place, we measure the response rate as the ratio of detection count and the total number of scans for each beacon:

$$R_{l_c,x} = \frac{\sum_{i=1}^{n_{l_c}} b_{x,i}}{n_c}, b_{x,i} = \begin{cases} 1 & \text{if beacon } x \text{ found in } i\text{th scan} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $R_{l_c,x}$ is the response rate of beacon x at the current private location l_c and, n_{l_c} is the total scan count since the private place was entered. The detection count of each beacon is maintained to identify the frequently occurring beacons; and therefore, beacons with higher response rate are used to create the network fingerprint for the current private place $[l_c]$. $R_{l_c,x}$ will be maintained in the LP-Caché database to update the response rate of every detected beacon during a specified time interval spent at private place $[l_c]$.

On-device Cache-based Location Calculation Algorithm. The detailed steps of privacy-aware geo-location calculation process are summarised in *Algorithm* 1. The surrounded beacons $[n_x]$ are scanned and compared to the list of private WiFi fingerprints $[n_i]$ to detect private place $[p_i]$ stored in cached database $[c]$. Further, the representative $[l_r]$ is altered based on set permissions (see Section 5.4).

5.4 Personalised Permissions for Location Sharing

A general LBS query consists of different attributes, e.g., LBS query {*POI, Latitude and Longitude, User-Info*}, where included geo-coordinates estimate the device's geo-location. To satisfy one of the privacy property called controlled information disclosure, we designed enhanced permission mechanism to control these geo-coordinates before it is sent to app providers. When using LP-Caché, for every installed app and set private place, the UI provides three distinct privacy settings: (1) Adjust Location Granularity, (2) Obfuscate Location and, (3) No Change. In the first option, geo-coordinate truncation method adjusts location precision level; in the second option, geo-coordinate transformation obfuscate user's location; whereas, in the third option, the exact unchanged geo-coordinates are sent to the requesting app.

Enhanced Permissions Algorithm. Once LP-Caché receives an invoked location object $[l_r]$, it alters the location data according to the enhanced permission settings and returns processed location $[l'_r]$. The steps involved in enhanced permission mechanism are summarised in *Algorithm* 2, where u_p is the set permission, g_l is the adjusted location precision level, l is the latitude, and l_g is the longitude.

Geo-coordinates Truncation. The geographical coordinates are represented by a tuple consisting of {*latitude* : 52°28'59.200" *N*, *and longitude* : 1°53'37.0001" *W*}, where the last digits specify more accurate geo-location. Geo-coordinate truncation method will enable us to adjust the location precision level, i.e., by removing last digits and rounding the location accuracy from street to city level or even more general. Generally, for any third party reuse, service providers or data collectors assure in the EULA that this method will be applied on the collected data since the truncated coordinates increase the ambiguity level [1]. On the contrary, LP-Caché applies this method on the user device with user's permission in order to minimise the user's sensitive data collection and privacy concerns.

Geo-coordinates Transformation. For privacy preservation, position transformation functions such as scaling, rotation and translation have been used by location data distributors or anonymisers [24, 37]. In LP-Caché, we use geo-coordinate transformation on the device to obfuscate user's private locations. Our service represents the geo-coordinate transformation using scaling and rotation, and denotes its parameters as a tuple $\langle s, \theta, (l, l_g)\rangle$, where s is the scaling factor, θ is the rotation angle, and (l, l_g) are the original coordinates. It applies Equation 3 to generate transformed or obfuscated geo-cordinates (l', l'_g), where angle θ is randomly generated.

$$\begin{aligned} l' &= \theta(s.l) \\ l'_g &= \theta(s.l_g) \end{aligned} \tag{3}$$

Table 1. WiFi measurement dataset comparative summary.

	1 Month	2 Months	Total scans.
Total number of scans	25480	21140	46620
Distinct private locations selected	34		% Change
Total APs detected	486	497	2.26%
Average APs detected	396	393	- 0.76 %

6 Feasibility and Usability Analysis

LP-Caché's actual performance evaluation depends on the location-based apps performance. In this section, we analise the WiFi AP data availability and consistency to measure feasibility of WiFi fingerprinting method to be included in LP-Caché's implementation. In [31], we presented a WiFi APs dataset summary for a month; we have extended the sample size for a couple of months and conducted a comparative study of the observations from both, 1st and 2nd month datasets to evaluate the scalability of the WiFi fingerprinting method. For the sake of illustration, we have maintained unique ID and sequence for all the selected 34 private places.

6.1 WiFi APs Availability and Consistency

Experimental Set-up. The experimental set-up to measure WiFi AP data availability and consistency consists of the following three steps:

1. *Data collection.* We collected beacons from fixed WiFi APs using `WiEye` [38] and `Network Info II` [28] apps on `Android` smartphones that have 802.11a/b/g/n radio feature so they can operate in both 2.4GHz and 5GHz bands at 34 different private places for a period of two months.
2. *Location categorisation.* App users are more concern about sharing their private locations[2]; therefore, in our analysis, we selected three distinct categorise of private places: 1. Home (i.e., residential place), 2. Work (i.e., commercial place), and 3. Arbitrary (i.e., any frequently visited place) to determine categorical distribution pattern of WiFi APs.
3. *Data analysis.* We collected and statistically analysed the scanned WiFi AP data. Table 1 compiles the included sample size and the measured percentage changes; whereas, Figure 7 shows the relative difference between WiFi APs density, and Figure 8 depicts the relative average accuracy distribution pattern of detected WiFi APs for each category over the 2 months period.

Observation 1. For each category of private places, experiments revealed the following:

Home The results demonstrate that Wifi APs are fixed and frequent and the difference between number of constant beacons and minimum number of similar beacons is comparatively less, and therefore, it achieved highest accuracy

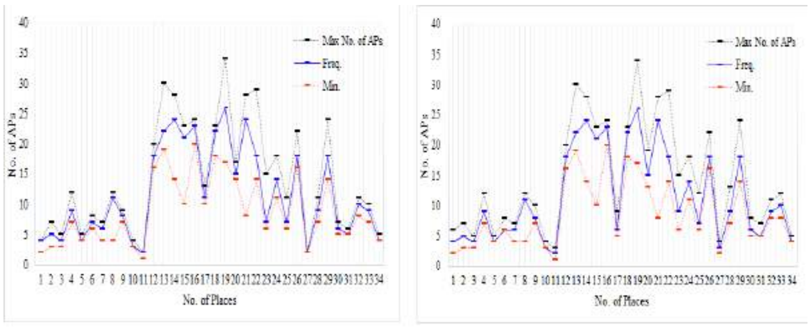


Fig. 7. Measured density of 1 month (left) and 2 months (right) detected WiFi APs at private places.

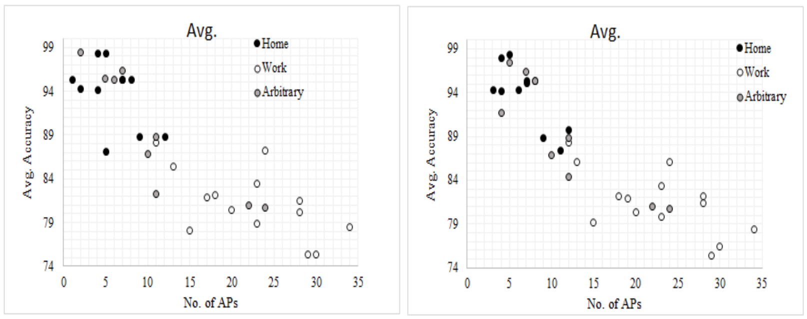


Fig. 8. Relative average accuracy distribution pattern of detected WiFi APs at private places of 1 month (left) and 2 months (right).

rate. Moreover, the ratio of SSID to BSSID is 1:1, i.e., 1 SSID (abc) has 1 BSSID (a0:12:b3:c4:56:78), this makes fingerprints distinct so improving the location detection performance.

Work This category has many fixed WiFi APs but with fluctuating signal strengths, and therefore, the sequence of available APs changes. However, the observed ratio of SSID to BSSID is many to one, i.e., 1 SSID has many BSSIDs; therefore, in this case, SSIDs along with BSSIDs can be used as unique identifiers to create a fingerprint to detect a private place dynamically.

Arbitrary In this category, the data collector could select any frequently visited locations, e.g., gym, shop, or friend's home. Figure 8 demonstrates that the outcome of this category is related to the other two categories, it either shows results similar to home or work.

The range of average accuracy for all the three categories of private places falls between 75% to 97%. Hence, it is evident that smartphones regularly detect similar beacons at frequently visited place, for place detection at least one beacon should match with the stored fingerprints. Thus, the result demonstrates that WiFi fingerprinting can be effectively used as private place detection source in LP-Caché. Nonetheless, to achieve efficient capability for place recognition via beacons, a *place discovering algorithm* like [23] can be implemented (in future work).

Observation 2. Table 1 shows comparative analysis of WiFi APs data that has been scanned and collected during both 1st and 2nd month. Considering percentage changes, the number of total detected APs has increased with 2.26% and the number of frequently detected APs remained without change, i.e, with a negligible difference of -0.76%. Equation 2 has been used statistically to identify frequently detected beacons whilst at a particular private place. Pre-set unique IDs and a sequence for all the selected 34 private places allowed us to measure the relative density and distribution pattern of the WiFi APs during both 1st and 2nd month. Figure 7 shows the relative difference between WiFi APs density, and Figure 8 depicts the relative average accuracy distribution pattern of detected WiFi APs for each category over the period on 2 months.

6.2 Estimated storage requirements

Location-based queries that are generated/received from running applications and service providers include several attributes, and their data types require vast amount of storage space. This is the case of some location privacy solutions (e.g., [3, 40]) that apply caching techniques on location-based queries as a resukt of their storage requirement. LP-Caché does not cache location-based queries, instead it stores the WiFi AP data and geo-coordinates of users' private locations. Moreover, the user's pre-set privacy rules are applied to the mapped geo-coordinates at runtime. As a result, comparatively, LP-Caché's on-device cache database does not demand massive storage requirement. By considering the 802.1 Standard and datatypes sizes, Table 2 presents the storage requirements in bytes and database components, where network fingerprint table is a tuple of $\langle no.ofbeacons, beaconfield, counter\rangle$, and permission table is a tuple of $\langle location, placeid, accuracy\ counter, no.ofprivateplaces\rangle$. Moreover, Table 3 presents the measured changes in the 1st and the 2nd month of WiFi data collected at 34 distinct private places. The results conclude that the average change has increased by 2.07%. cache storage of total 25844 bytes that includes the sum of permissions and network fingerprints for 34 distinct private places. Thus, we can anticipate that the current mobile device internal storage capacity is sufficient for the required storage [4].

Table 2. Estimated storage

Field storage		**Size**	**Database Component**	**Size**
Beacon field	BSSID/MAC	6 bytes	Network fingerprint table	$= (beaconfield + counter) \times no.ofbeacons$
	SSID	32 bytes		
	Place ID	3 bytes		
	Timestamp/Age	8 bytes		
Location field	Geo-coordinates	32 bytes	Permission table	$= (location +$ $placeid + accu.\ counter) \times no.ofprivateplaces$
	Region	32 bytes		

Table 3. Relative difference of monthly storage

Storage	**1 Month**	**2 Months**	**% Change**
Network fingerprint	25272 bytes	25844 bytes	2.26% increase
Permissions	2380 bytes	2380 bytes	No change
Total storage	27652 bytes	28224 bytes	2.07% total increase

6.3 Cache hits and cache misses

In LP-Caché, up-to-date cache database and network fingerprint search result accuracy are main challenges. The three possible outcomes when looking for device's location based on the scanned beacons are:

1. *The location is cached and up-to-date* This case comes with positive result, and therefore, data can be used effectively.
2. *The location is cached but is out-of-date* This can occur if the network infrastructure changes, e.g., if router is changed then the cache data needs to be updated. The overall results of Section 6.1 and Section 6.2 prove that this case does not occur frequently. Nonetheless, for data accuracy a method that uses Equation will be incorporated to detect and measure occurrence of such situations of cache misses at runtime. Moreover, the developed method can likewise be deployed to maintain *data freshness* and *data consistency.*
3. *The location is not cached* This occurs when the observed WiFi AP is not cached and/or mapped to the private locations, then our service will have to interact with user to update the location cache. Besides, the response rate $R_{l_c,x}$ can be further extended to measure runtime occurrences of these outcomes.

6.4 Ongoing Evaluation of Caching Method

Following WiFi data availability and consistency analysis, LP-Caché's feasibility evaluation will be extended to analyse how frequently the cache needs to be updated and what are the trade-offs between the cache update frequency and location privacy and accuracy in order to measure computational and communication overheads. We also intent to conduct a thorough user study to determine usability for the users to accommodate LP-Caché's functionality. Moreover, we plan to extend the fundamental caching-related technical challenges such as cache hits and cache misses, data freshness, data consistency, and estimated bandwidth requirements in an advanced development and implementation of LP-Caché hence paying special attention to storage-efficient caching.

7 Conclusion

Secure gathering and transmission of the location data by mobile apps while preserving users' privacy is a major concern that needs reconsideration. Evaluation of current location handling process confirms that it is vulnerable to location privacy attacks; therefore, we presented a privacy-aware model that provides users with advanced location controls to mitigate major privacy threats. Within a dataset generated in 2 months of experimental setting, we observed a 2.26% change in detected WiFi APs at 34 distinct places and 2.07% change in estimated storage. These results are promising and benefit deployment of LP-Caché's. In this paper, we mainly focused on establishing the design decisions, implementation requirements, and on measuring the feasibility of LP-Caché. Work in progress is on (1) deploying our model on a mobile platform to measure its functionality and efficiency while interacting with different location-based apps; (2) carrying out run-time measurements of the cache storage over an extended period of time, and (3) performing critical analysis of the network fingerprinting and permission mapping methods with dynamic movement patterns. We plan to further assess LP-Caché's scalability in future large scale scenarios and, address end user as well as service providers concerns.

References

1. Aad, I., Niemi, V.: NRC data collection and the privacy by design principles. In: PhoneSense (2010)
2. Almuhimedi, H., Schaub, F., Sadeh, N., Adjerid, I., Acquisti, A., Others: Your Location has been Shared 5,398 Times! A Field Study on Mobile App Privacy Nudging. In: Procs. of ACM Conf. on Human Factors in Computing Systems. pp. 787–796. ACM (2015)
3. Amini, S., Lindqvist, J., Hong, J., Lin, J., Toch, E., Sadeh, N.: Caché: caching location-enhanced content to improve user privacy. In: Procs. of ACM Int. Conf. on Mobile Systems, Applications, and Services. pp. 197–210. ACM (2011)
4. Android: (March 2016), http://developer.android.com/guide/ topics/data/data-storage.html
5. Android Developer Reference: (March 2016), http://developer.android.com/reference/
6. Anonymizer: (March 2016), http://www.anonymizer.com/
7. Bell, S., Jung, W.R., Krishnakumar, V.: WiFi-based enhanced positioning systems: accuracy through mapping, calibration, and classification. In: Procs. of ACM SIGSPATIAL Int. Workshop on Indoor Spatial Awareness. pp. 3–9. ACM (2010)

8. Beresford, A.R., Rice, A., Skehin, N., Sohan, R.: Mockdroid: trading privacy for application functionality on smartphones. In: Procs. of ACM Workshop on Mobile Computing Systems and Applications. pp. 49–54. ACM (2011)
9. Cranor, L.F., Sadeh, N.: A shortage of privacy engineers. IEEE Security & Privacy (2), 77–79 (2013)
10. Damiani, M.L.: Third party geolocation services in LBS: Privacy requirements and research issues. Trans. on Data Privacy 4(2), 55–72 (2011)
11. Doty, N., Wilde, E.: Geolocation privacy and application platforms. In: Procs. of ACM SIGSPATIAL Int. Workshop on Security and Privacy in GIS and LBS. pp. 65–69. ACM (2010)
12. Egele, M., Kruegel, C., Kirda, E., Vigna, G.: PiOS: Detecting Privacy Leaks in iOS Applications. In: NDSS (2011)
13. Enck, W., Gilbert, P., Han, S., Tendulkar, V., Chun, B.G., Others: TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. TOCS 32(2), 5 (2014)
14. European Commission: Protection of personal data (2016), http://ec.europa.eu/justice/data-protection/
15. Fawaz, K., Shin, K.G.: Location Privacy Protection for Smartphone Users. In: Procs. of ACM SIGSAC Conf. on Computer and Communications Security. pp. 239–250. ACM (2014)
16. Felt, A.P., Egelman, S., Wagner, D.: I've got 99 problems, but vibration ain't one: a survey of smartphone users' concerns. In: Procs. of ACM Workshop on Security and Privacy in Smartphones and Mobile Devices. pp. 33–44. ACM (2012)
17. Gibler, C., Crussell, J., Erickson, J., Chen, H.: AndroidLeaks: Automatically detecting potential privacy leaks in Android applications on a large scale. In: TRUST 2012, pp. 291–307. Springer (2012)
18. Google Location Service: (March 2016), https://support .google.com/gmm/answer/1646140?hl=en-GB
19. Hellman, E.: Android programming: Pushing the limits. John Wiley & Sons (2013)
20. IETF: Geographic Location Privacy (March 2016), http://datatracker.ietf.org/wg/geopriv/charter/
21. Jeon, J., Micinski, K.K., Vaughan, J.A., Fogel, A., Reddy: Dr. Android and Mr. Hide: fine-grained permissions in android applications. In: In Procs. of ACM Workshop on Security and Privacy in Smartphones and Mobile Devices. pp. 3–14. ACM (2012)
22. Khoshgozaran, A., Shahabi, C., Shirani-Mehr, H.: Location privacy: going beyond K-anonymity, cloaking and anonymizers. Knowledge and Information Systems 26(3), 435–465 (2011)
23. Kim, D.H., Hightower, J., Govindan, R., Estrin, D.: Discovering semantically meaningful places from pervasive RF-beacons. In: Procs. of ACM Int. Conf. on Ubiquitous Computing. pp. 21–30. ACM (2009)
24. Lin, D., Bertino, E., Cheng, R., Prabhakar, S.: Position transformation: a location privacy protection method for moving objects. In: Procs. of the SIGSPATIAL ACM GIS 2008 Int. Workshop on Security and Privacy in GIS and LBS. pp. 62–71. ACM (2008)
25. Michael, K., Clarke, R.: Location and tracking of mobile devices: Überveillance stalks the streets. Computer Law & Security Review 29(3), 216–228 (2013)
26. Muslukhov, I., Boshmaf, Y., Kuo, C., Lester, J., Beznosov, K.: Understanding users' requirements for data protection in smartphones. In: ICDE Workshop, IEEE Int. Conf. on Secure Data Management on Smartphones and Mobiles. pp. 228–235. IEEE (2012)

27. Navizon: (March 2016), http://www.navizon.com
28. NetworkInfoIi: (March 2016), http://play.google.com/store/apps/
29. Niu, B., Li, Q., Zhu, X., Cao, G., Li, H.: Enhancing Privacy through Caching in Location-Based Services. In: Proc. of IEEE INFOCOM (2015)
30. Patel, A., Palomar, E.: Privacy Preservation in Location-Based Mobile Applications: Research Directions. In: Procs. of IEEE Int. Conf. on Availability, Reliability and Security (ARES). pp. 227–233. IEEE (2014)
31. Patel, A., Palomar, E.: LP-Caché: Privacy-aware cache model for location-based apps. In: Proceedings of the 13th International Joint Conference on e-Business and Telecommunications (ICETE 2016) - Volume 4: SECRYPT, Lisbon, Portugal, July 26-28, 2016. pp. 183–194 (2016)
32. Pontes, T., Vasconcelos, M., Almeida, J., Kumaraguru, P., Almeida, V.: We know where you live: Privacy characterization of foursquare behavior. In: Procs. of ACM Conf. on Ubiquitous Computing. pp. 898–905. ACM (2012)
33. Shklovski, I., Mainwaring, S.D., Skúladóttir, H.H., Others: Leakiness and Creepiness in App Space: Perceptions of Privacy and Mobile App Use. In: Procs. of ACM Conf. on Human factors in computing systems. pp. 2347–2356. ACM (2014)
34. Skyhook: (March 2016), http://www.skyhookwireless.com/
35. TOR: (March 2016), http://www.torproject.org/
36. tPacketcapture: (March 2016), http://play.google.com/
37. Wernke, M., Skvortsov, P., Dürr, F., Rothermel, K.: A classification of location privacy attacks and approaches. Personal and Ubiquitous Computing 18(1), 163–175 (2014)
38. WiEye: (March 2016), http://play.google.com/store/apps/
39. Wireshark: (March 2016), https://www.wireshark.org
40. Zhu, X., Chi, H., Niu, B., Zhang, W., Li, Z., Li, H.: Mobicache: When k-anonymity meets cache. In: GLOBECOM. pp. 820–825. IEEE (2013)
41. Zhuang, Y., Syed, Z., Georgy, J., El-Sheimy, N.: Autonomous smartphone-based WiFi positioning system by using access points localization and crowdsourcing. Pervasive and Mobile Computing (2015)

LP-Caché: Privacy-aware Cache Model for Location-based Apps

Asma Patel and Esther Palomar
School of Computing and Digital Technology, Birmingham City University, Birmingham, U.K.

Keywords: Location Privacy, Location-based Services, Smartphones, Caching, Location-based Applications.

Abstract: The daily use of smartphones along with third-party apps, which involve location data to be continuously collected, shared and used, have become a significant privacy concern. Besides, taking advantage of the rapid growth of wireless access points, the capability of these location-based services to track users' lives, even sometimes with their consent, creates an urgent need for the development of more user-friendly and socially-accepted approaches to location privacy preservation. In this paper, we introduce a novel privacy-aware model for location-based apps to overcome the shortcomings related to user privacy during the location calculation process. By making the user device play a bigger role in the process, our model prevents users from relying on service providers' trustworthiness. The model applies a cache-based technique to determine the position of client devices by means of wireless access points and achieve data minimisation in the current process. The model also establishes new personalised permission settings for the users while sharing their location information. We outline possible implementation of the proposal, and preliminary findings of the work-in-progress evaluation on the wireless data feasibility and usability that demonstrate deployment viability.

1 INTRODUCTION

The explosive growth of context-aware mobile apps has leveraged tremendous opportunities for a whole new class of Location-Based Services (LBS) (Pontes et al., 2012). Geo-marketing and geo-social networking, location-based games, monitoring, assisted eHealth, and energy consumption 3D maps represent a small subset of the third-party apps nowadays available as LBS and can certainly pose a serious threat to the users' privacy. (Muslukhov et al., 2012; Shklovski et al., 2014).

Currently, approaches to privacy settings of user location on smartphones are based on a binary process[1]. Users are forced to rely on third party service providers that in many cases continuously collect, use and share their location data, and in some cases even prompt the user to give away their position on page load (Almuhimedi et al., 2015; Muslukhov et al., 2012; Felt et al., 2012; Shklovski et al., 2014). Moreover, both academia and industry agree on the urgent need of adopting a Privacy-by-Design (PbD) approach for the development of more user-friendly and socially-accepted solutions to location privacy preservation on their mobile products and services (Cranor and Sadeh, 2013).

To encounter these challenges, we introduce the model called *Location Privacy Caché* (LP-Caché). LP-Caché envisions beyond the simple grant/deny access method and provides the user with advanced mechanisms to decide the extent of disclosing location data with service providers. Several caching based solutions (Zhu et al., 2013; Amini et al., 2011; Niu et al., 2015) have been proposed to minimise the risk of major location privacy threats, but lacking of deployment feasibility. They rely on unrealistic assumptions such as vast cache data storage requirements, or on the app developers modifying the code to incorporate their cached databases. LP-Caché incorporates caching technique to determine users' geographical location in a privacy preserving manner, and with minimum cache storage requirements. The main contributions in this paper are:

- Detailed analysis of the current location computation process deployed in smartphones when running location-based apps.
- Definition of LP-Caché and its main implementa-

[1]Data protection directives and acts (European Commission, 2016; IETF, 2016) across the globe state that personal data should not be disclosed or shared with third parties without consent from subject(s). Such a consent is typically obtained by mandatory acceptance of the conditions mentioned in the End User License Agreement (EULA), or through opt-out possibilities and other regulations(Michael and Clarke, 2013).

Patel, A. and Palomar, E.
LP-Cache: Privacy-aware Cache Model for Location-based Apps.
DOI: 10.5220/0005970101830194
In *Proceedings of the 13th International Joint Conference on e-Business and Telecommunications (ICETE 2016) - Volume 4: SECRYPT*, pages 183-194
ISBN: 978-989-758-196-0

tion requirements. The main identified benefits of LP-Caché are twofold:

– Provides enhanced personalised location privacy settings that control every installed app distinctly and protect sharing of user's private location with third-party app providers.
– Minimises the amount of wireless access point data that is being shared within the current architecture for computing the device's location by means of minimum on-device caching.

- Evaluation of wireless access point data availability and consistency, and of the implementation requirements to demonstrate that LP-Caché is feasible without modifying installed apps.

The rest of the paper is organized as follows. Section 2 outlines the current location computation process and its evaluation. Section 3 reviews the related work. Section 4 presents the design and architecture of LP-Caché, and Section 5 fully elaborates on design decisions and possible implementation. We evaluate the feasibility and usability requirements in Section 6. Finally, Section 7 concludes and sets future research plans.

2 CURRENT LOCATION COMPUTATION PROCESS

In this section, we describe roles and processes involved in the current architecture for computing user's device location.

2.1 Current Architecture

The current location computation architecture to use location-based apps on smartphones comprises four main entities: 1. Smartphones with installed apps, 2. App Provider, 3. Network Infrastructure, and 4. Location Provider. This architecture mainly relies on third party location providers, e.g., Google Location Service (Google Location Service, 2016), Skyhook (Skyhook, 2016), and Navizon (Navizon, 2016). The location provider represents the central database, which maps the received signatures of nearby wireless access points to the geo-coordinates, i.e., latitude and longitude, so handling every geo-location request. Therefore, the location provider has constant access to the user's location as well as to the trajectory data. To respond to any location request, the location provider maintains a database of surrounding network infrastructure, including WiFi Access Points (APs), cellular-towers, and IP addresses, which must be mapped to their exact geographical coordinates. Compared to GPS and cell-tower based positioning, WiFi Positioning Systems (WPS) is nowadays considered as a very accurate method for location calculation (Skyhook, 2016). Location providers rather use enhanced WPS than GPS, primarily due to current smart-mobile devices benefit from built-in WiFi clients that perform faster than most expensive GPS receivers. This enables the service provider to get user's precise location at all times and, as a result, more effective privacy preservation measures are needed in the current process to mitigate privacy threats.

WiFi APs continuously announce their existence in the way of network frames/beacons and transmit their Service Set Identifier (SSID) and Basic Service Set Identifier (BSSID)/MAC addresses. Location providers use these WiFi APs identifers to create network signatures and map them with geo-coordinates, also called geolocation. IEEE 802.11 states two standardised ways to collect beacons from WiFi APs: 1. Active scanning, and 2. Passive scanning. Location providers are capable of deploying systems with either active scanning, passive scanning, or both together. Location providers use three different ways to collect geo-location of WiFi APs:

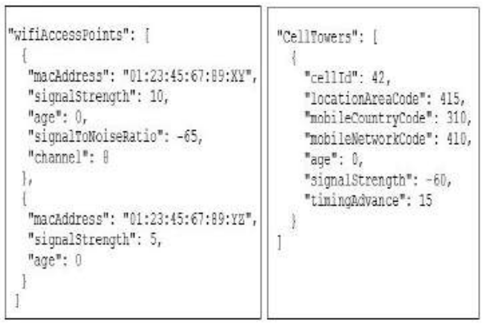
1. *Statically*- They collect WiFi beacons by the so called *wardiving* process. Basically, they map the equipped vehicle's exact geo-coordinates along with the signal strength of the captured beacons from surrounded APs.
2. *Dynamically*- They can collect data from WiFi APs automatically once the user device uses location services, e.g. Maps and Navigation applications. The user device as configured to be geolocated acquires unique identifiers from the surrounding WiFi APs, even if the network is encrypted, and then sends it over to the location provider in order to perform geolocation calculation. The collected information is utilised to build and update the database autonomously, for example, by applying crowdsourcing (Zhuang et al., 2015).
3. *User input*- They encourage users to manually input the WiFi APs' information, i.e., BSSID and the geo-coordinates, into their databases, e.g., Skyhook[2] to register WiFi APs.

[2]Submit a Wi-Fi Access Point. See http://www.skyhookwireless.com/submit-access-point (last access in March. 2016).

2.2 Evaluation of Current Location Computation Process

We conducted a series of experiments on different mobile devices installed with *Android*, *Windows Phone*, and *iOS* operating systems to categorise the data flow in the current location computation process. With the assistance of sniffers, such as *Wireshark* (Wireshark, 2016) and *tPacketCapture* (tPacketcapture, 2016), we captured and analysed sequence and location data transmission when using location-based apps, e.g., Navigation and Friend Finder.

Observation. These experiments were designed to understand whether there is any difference on the location calculation process on each of these three mobile operating systems/ platforms. Based on the results, all of them display common patterns of location data retrieval. The user device collects the unique identifiers from the surrounding network along with GPS data, and sends it to the location provider to get the exact device location. Figure 1 shows the structure of the WiFi and Cell-tower objects sent to the location provider. Once calculation is performed, the location provider sends to the device the precise location in the way of a geo-location object containing geo-coordinates. Figure 2 represents the structure of the location object received from the location provider. In short, the app developer over any mobile platform can utilize this location object to get the user's geo-location with no need of focusing on the details of the underlying location technology. In the following section, we give the detailed description of the process sequence.

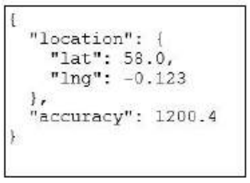


```
"wifiAccessPoints": [
  {
   "macAddress": "01:23:45:67:89:XY",
   "signalStrength": 10,
   "age": 0,
   "signalToNoiseRatio": -65,
   "channel": 8
  },
  {
   "macAddress": "01:23:45:67:89:YZ",
   "signalStrength": 5,
   "age": 0
  }
]
```

```
"CellTowers": [
  {
    "cellId": 42,
    "locationAreaCode": 415,
    "mobileCountryCode": 310,
    "mobileNetworkCode": 410,
    "age": 0,
    "signalStrength": -60,
    "timingAdvance": 15
  }
]
```

Figure 1: Structure of WiFi AP object (left) and cell-tower object(right) sent to the location provider.

Process Sequence. Figure 3 illustrates the sequence of processes and messages involved in the current location computation architecture. Note that, on a smartphone, location sharing service settings must be 'ON' while using any location-based app. If the location sharing is 'OFF', then the device prompts for changing the setting from 'OFF' to 'ON'; otherwise, user cannot use the service.



```
{
  "location": {
    "lat": 58.0,
    "lng": -0.123
  },
  "accuracy": 1200.4
}
```

Figure 2: Structure of the location object received from the location provider.

3 RELATED WORK

Existing approaches to location privacy preservation can be classified into three categories: 1. Mobile Platform, 2. Location Query, and 3. Privacy-aware Network Communication.

3.1 Mobile Platform

A few studies have proposed static and dynamic methods to detect privacy leaks in mobile platforms. The former method statistically analyse apps by creating permission mapping, generating call graphs, and data flow analysis to report privacy leaks for further auditing, e.g, AndroidLeaks (Gibler et al., 2012) and PiOS (Egele et al., 2011) for Android and Apple iOS, respectively. The latter involves modification of existing mobile platforms, such as TaintDroid (Enck et al., 2014) and MockDroid (Beresford et al., 2011). TaintDroid adds taint tracking information to sensitive sources calls from apps, and it tracks location data flow as it generated through applications during execution. Whereas, MockDroid relies on instrumenting `Android′s` manifest permission mechanism to mock sensitive data from OS resource, including location data, which can affect apps' usability and functionality. LP-Caché not only monitors the location sources but also modify, if required, the generated location data based on defined user permissions. Fawaz and Shin (Fawaz and Shin, 2014) apply *indistinguishability* into the advertising and analytics libraries as well as on installed apps as location privacy preservation mechanism; however, it does not give control on the amount of WiFi and location data that is being shared with the location provider, *indistinguishability* technique increases computational overhead on smartphones.

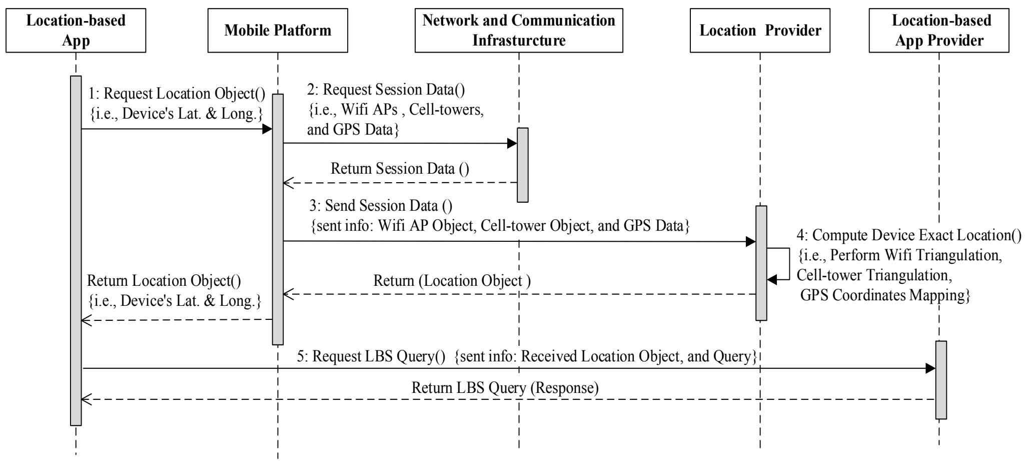

Location-based App
Mobile Platform
Network and Communication Infrasturcture
Location Provider
Location-based App Provider
1: Request Location Object() {i.e., Device's Lat. & Long.}
2: Request Session Data() {i.e., Wifi APs , Cell-towers, and GPS Data}
Return Session Data ()
3: Send Session Data () {sent info: Wifi AP Object, Cell-tower Object, and GPS Data}
4: Compute Device Exact Location() {i.e., Perform Wifi Triangulation, Cell-tower Triangulation, GPS Coordinates Mapping}
Return Location Object() {i.e., Device's Lat. & Long.}
Return (Location Object)
5: Request LBS Query() {sent info: Received Location Object, and Query}
Return LBS Query (Response)

Figure 3: Sequence diagram of current location computation process.

3.2 Location Query

Apps share location information with the provider in the form of LBS queries. The transmission of such queries to the location server may allow attackers to gain access to user location data. Privacy Enhancing Techniques (PET) like k-anonymity, dummy locations, region cloaking, location perturbation and obfuscation, and mix-zone pseudonyms have been applied to different architectures for location query formation and privacy preservation from LBS providers (Patel and Palomar, 2014; Wernke et al., 2014; Khoshgozaran et al., 2011). Most of these techniques rely on theoretical assumptions - like trusted infrastructure to provide the privacy protection, requiring a group of similar app users to be at the same time and same place. The main issue with PETs and cryptographic schemes is that it relies entirely on the data collection servers to comply with location privacy.

Caching. Several authors have used caching scheme along with PETs to build privacy preserving location based queries. MobiCaché (Zhu et al., 2013) applies k-anonymity for caching location based queries. Similarly, Niu et al. (Niu et al., 2015) attempt to improve k-anonymity based caching by adding dummy locations. Both proposals require trusted infrastructure to maintain privacy. Caché (Amini et al., 2011) maintains a local caché within the device to reuse the data types available from applications in future location based queries; however, storing entire LBS query data increases the cache storage requirements. Besides, Caché also requires app developer to modify the way app access location data. Whereas, LP-Caché caches the network fingerprints and geo-coordinates, which reduces the storage overhead drastically; it considers installed apps as black box, and therefore, does not require app developer to modify the code, it works as a middleware between the app and the mobile platform. All these cache-based systems either intent to generalise or obfuscate the LBS query or minimise the number of queries sent to the app providers, but they do not provide privacy from WiFi content distributors. Besides, mobile devices not only send vast amounts of location data to app providers but also to location providers creating different location privacy shortcomings (Almuhimedi et al., 2015; Shklovski et al., 2014). In this regard, limited work has been published on privacy preservation from the location provider perspective (Damiani, 2011; Doty and Wilde, 2010). Damiani (2011) proposes a theoretical approach for privacy-aware geolocation-based web services to encourage further research to minimise the amount of location data being shared with the location provider. This is mainly due to that the location provider is considered as the only source to get the user location when developing any location-based app. In LP-Caché, we minimise the process of wireless AP data collection by the WiFi content distributors or location providers, and we control information disclosure within the generated LBS query (e.g., points of interests (POIs) and nearest neighbor) since it will be sent to the third-party app provider.

3.3 Privacy-aware Network Communication

Besides location queries, device's IP address can also reveal user's private locations. To this regard, anonymous communication protocols, e.g., Anonymizer (Anonymizer, 2016) and TOR (TOR, 2016), deal with anonymous service usage at the network layer while communicating over Internet (i.e., the server cannot infer user's location via received device's IP address along with location query), and they are most prominent and commonly used network layer solutions.

4 LP-CACHÉ MODEL

In this section, we describe LP-Caché's threat model, design goals, architecture and main processes' sequence diagram.

4.1 Threat Model

Apps deliberately collect user's sensitive data, including location and other sensitive information as part of their operations. User tracking, identification and profiling (i.e. personal habits, movement patterns, etc.) are fundamental threats to location privacy (Felt et al., 2012; Wernke et al., 2014). Furthermore, the current direct link of smartphones to the location provider and the continuous flow of LBS queries that include device's exact geo-coordinates over network create a serious risk to the protection of users' sensitive information, even more challenging, in the presence of a malicious location provider and via advanced network sniffing practices.

LP-Caché computes the exact location within user device, without service provider's involvement, and trusts the device on the storage of sensitive data. However, the user has still the option of giving consent for app providers or location provider to access their location. Mobile network providers might, however, collect user location data via cellular clients. It is also excluded from our model the option of manually inserting the location data (e.g. street name, zip code, post code) within LBS query.

4.2 LP-Caché Control Flow Architecture

LP-Caché's three main design goal are: 1) the third-party app provider will not be able to infer the device's exact location without getting uses's consent; 2) the user can set distinct privacy preferences for different apps and private places; and 3) the model works independently without the need of modifying the app's code. Figure 4 depicts the block diagram for LP-Caché architecture; its main components are:

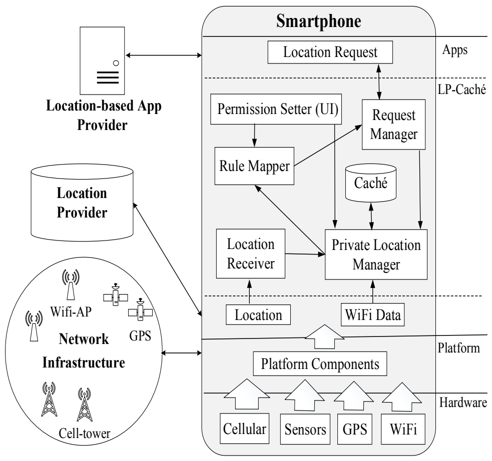


Figure 4: LP-Caché architecture.

Permission Setter is the user interface (UI), which enables users to set and manage their private places and apply improved personalised permissions when running installed location-based apps. Once received the user inputs, pre-set private locations are sent to the *Private Location Manager* module, and permissions are sent to the *Rule Mapper* Module.

Request Manager is responsible to intercept the event of location access calls, and then lead the app's control flow to the *Private Location Manager* module. Besides, it will also be in control of receiving the processed user location (i.e., could be either anonymised or altered) from *Rule Mapper*, and then delivering it to the app in order to maintain every session's control flow.

Private Location Manager module's main task is to detect unique identifers of the surrounding WiFi APs and compare them with the stored network fingerprints to determine whether the user is within the set of private places. User inputs from the *Permission Setter* will create network fingerprints for known private locations, which are then added or updated in the *Caché* database. Moreover, it maintains a binary flag to detect private places. In the case of a hit the location data is sent to the *Rule Mapper*. Otherwise, the location is received from the *Location Receiver*. Whenever the *Private Location Manager* receives a new private place request, the received location is mapped to the detected network fingerprint and stored in the *Caché* database.

Rule Mapper dynamically collects and checks set permissions from *Permission Setter*. Once the representative location object is received from the *Private Location Manager*, it applies the user permissions on the location co-ordinates, alters them (if required), and outputs the processed location to the *Request Manager* module. If the flag is negative, then it forwards the exact location.

Caché is the established on-device cached database, and it is routinely queried by the *Private Location Manager* module, which can add, update and delete the cached location data. The locations in caché are those which are to be protected, and they can also represent regions of space. Each entry is recorded along with a network fingerprint and geo-location that is acquired from the location provider.

Location Receiver module receives a location object, which includes the user device's geo-coordinates (as in Figure 2), from location providers and sends it over to the *Private Location Manager* for further processing.

4.3 Process Sequence

LP-Caché modifies the current location resource handling process, however, the involved entities (as in Section 2) remain the same. Figure 5 illustrates the sequence of processes and messages involved in LP-Caché:

1. At the event of app requesting the device location, our service will intercept the request to get the location from the cache database instead of sending the request to the location provider.
2. Upon receiving the location request, our service will scan the surrounding network infrastructure.
3. Using observed network frames our service will execute as follows:
   (a) Our service compares the collected beacons with the stored network fingerprints to retrieve corresponding stored representative location coordinates.
   (b) In the case of an unmatched entry on the database, the LP-Caché prompts the user two options either input the location using UI, or allow the query to be sent to location provider that will calculate and send the current location coordinates. Note that this will only occur if the user has set the current location as private but the geo-coordinates are not cached.
   (c) The received location data for the encountered APs will be tracked within the local cache database for future use.
4. User location coordinates can be altered based on the privacy settings. LP-Caché provides three options for controlled information disclosure: (1) Adjust Location Granularity, (2) Obfuscate Location, and (3) No Change. Computed location is populated in the location object and sent to the app.
5. Once the app obtains the location object, it is then used by the app provider to send the corresponding reply to LBS query via the standard programming interface/API (Android Developer Reference, 2016).

5 IMPLEMENTATION REQUIREMENT

In the following sections, we describe LP-Caché's implementation requirements. LP-Caché orchestrates a mobile platform based location protection service to modify the location resource handling process. For instrumenting the LP-Caché implementation, `Android` will be the best choice since it is open source; however, it can also be implemented on other permission-based mobile platforms.

5.1 Bootstrapping

LP-Caché aims to protect user's private places. Initially, LP-Caché does not have enough information to function, the two main required information are private places's network fingerprints and geo-coordinates. LP-Caché cannot collect network fingerprints and geo-cordinates for private places at runtime, as by the time we have this information, other installed apps will have access to it. Therefore, when LP-Caché first boots and before turning 'ON' location sharing settings, user will have to do the initial setup, which includes allow WiFi AP scanning, input geo-cordinates and set privacy choices (see Section 5.4). In 2013, Google presented a new service API (also works on older `Android` versions) for location-based apps that allows developers to use the new and advanced *Location and Activity API*, i.e., they changed `Location Manager` to `Fused Location Manager`, hence combining sensors, GPS, Wi-Fi, and cellular data into a single API for location-based applications (Hellman, 2013). As a result, separating data from `GPS_PROVIDER` and `NETWORK_PROVIDER` is no longer straight forward. LP-Caché addresses this issue by

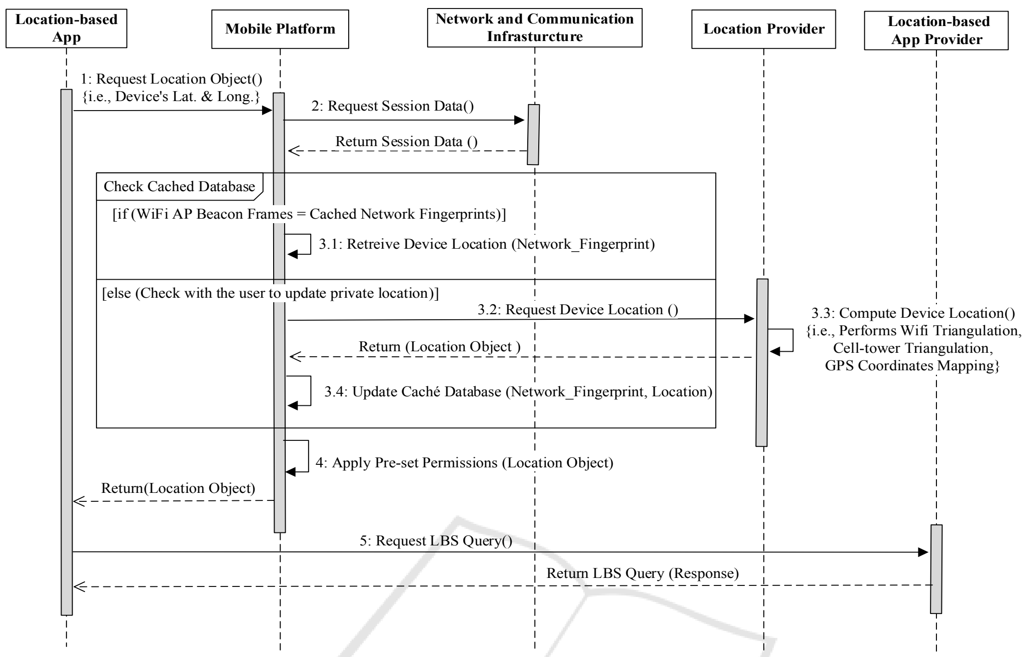

Location-based App
Mobile Platform
Network and Communication Infrasturcture
Location Provider
Location-based App Provider
1: Request Location Object() {i.e., Device's Lat. & Long.}
2: Request Session Data()
Return Session Data ()
Check Cached Database
[if (WiFi AP Beacon Frames = Cached Network Fingerprints)]
3.1: Retreive Device Location (Network_Fingerprint)
[else (Check with the user to update private location)]
3.2: Request Device Location ()
Return (Location Object)
3.3: Compute Device Location() {i.e., Performs Wifi Triangulation, Cell-tower Triangulation, GPS Coordinates Mapping}
3.4: Update Caché Database (Network_Fingerprint, Location)
4: Apply Pre-set Permissions (Location Object)
Return(Location Object)
5: Request LBS Query()
Return LBS Query (Response)

Figure 5: Sequence diagram of location computation process using proposed model

preventing app's location request to reach the `Fused Location Manager` that collects and sends the network session data to the location provider. Instead, the requested location is retrieved from the on-device cache, and then, it is sent to the requesting app (with privacy rules applied). Besides, geographic tool[3] can be incorporated in the LP-Cache's UI to get the corresponding geo-cordinates. This will allow LP-Caché to achieve effective privacy without affecting location accuracy, at the same time, prevent from the non-authorised sharing of device's exact location and network session data.

5.2 Mobile Platform

For performance evaluation, there are two possible ways of implementing LP-Caché location protection service. The first requires modifying the app's location accessing interfaces and intercepting location updates before they reach the app provider. Whereas, the second option requires modifying the platform and changing the location data before reaching the app.

App Code Modification. This comprises unpacking the app, rewriting the code to work according to the new rules, and then repackaging it again, e.g.,(Jeon et al., 2012). However, app repackaging changes the signature and stops future updates, and therefore, affects its functionality. Another way to modify app's location accessing interfaces is through the creation of an `Android` service and allowing apps to register with it. Then, Apps can use the location data provided by this service. This approach is easy to implement but relies heavily on app developers to modify their app's code, which is highly infeasible and unrealistic. Nonetheless, this approach can be used as simulated testing environment for any developed service.

Platform Modification. For the sake of experimentation, we develop LP-Caché via platform modification. One of the main task is to add a system service, where the class belongs to the location APIs; thus, it is placed in the `android.location package`, which detects private locations via APs and can also be used by other components when calling context. In Android, a context allows an app to interact with the OS resources. Another task is to make LP-Caché communicate with location requesting apps. On `Android` there are two methods to access user's location: 1) Location Manager Service (Old), and 2) Fused Location Manager Service (New) that is a part of Google Play

[3]LatLong is a geographic tool. See http://www.latlong.net/ (last access in March. 2016)

Services. Both methods require the app to request a callback function to get regular updates by registering a location listener. The app receives a new location object when a new location is available, the callback function is invoked. Modifying these two Google services is complicated, but there is a possibility to intercept the location object before it reaches requesting apps. We will add a static context field to the location class, which will be populated when the app is invoked; this will enable us to know which app is currently requesting the location object, and also communicate with the OS (Fawaz and Shin, 2014). The created location object will have reference to the requesting app's context, and therefore, it can interact with our external service.

5.3 On-device Cache Database Creation

LP-Caché requires fixed wireless APs data (i.e., 802.11) to create cached database of private locations. Initially, we decided to focus on WPS since it infers accurate user location. However, we can later include other fixed radio sources (e.g., cell-tower unique identifiers).

Network Fingerprinting. We can distinguish two main types of WPS techniques to determine the position of client devices in respect to APs (Bell et al., 2010): 1) Signal trilateration and 2) Fingerprinting. The former undertakes trilateration of Received Signal Strength (RSS), Angle of Arrival (AoA), and Time of Flight (ToF) from observed APs, and the later involves mapping observed APs signatures with a stored database. LP-Caché uses fingerprinting to create cached location database; however, fingerprinting performance is highly related to the number of APs. Therefore, in Section 6 we have evaluated WiFi AP availability and consistency. The detected network management frames/beacons are mapped with the device's representative geo-location to create a network fingerprint, which is then stored in the local cached database, an example private location fingerprint is shown in Equation 1. Moreover, to reduce storage and computation overhead, our model only caches network fingerprints of private places (e.g., home, work, frequently visited places or particular stores), and it relies on user input for initial pre-set up. The user will have to select the option (via LP-Caché UI) to set current location as private place p_i, and then fingerprint will be recorded. Later, the private place will be detected automatically with respect to observed beacons.

$$p_i = [n1], [n2], ..., [nx] \rightarrow [l_r] \qquad (1)$$

Algorithm 1: Location Calculation Algorithm.

```
Input: $n_x$: Network Frames
Output: $l_r$: Representative Location
 1: $n_x = 0$
 2: read $n_x$
 3: while $n_x \neq$ null do
 4:     if $n_x = n_i$ , $\forall\, i \in p$ then
 5:         (step 1) retrieve the corresponding $l_r$
 6:         add flag $f$ = (if private 1, else 0)
 7:         send $l_r$
 8:     else
 9:         (step 2) request $l_r$ from user or location
            provider
10:         set received $l_r$ to corresponding $p_i$
11:         update $c$
12:         send $l_r$
13:     end if
14: end while
```

where p_i represent ith private place IDs, n is the scanned beacon, and l_r is a representative location for that private place. WiFi AP beacon frame n consists of four attributes ⟨SSID, BSSID/MAC address, Signal-strength, and Timestamp⟩. The private representative location l_r consists of a tuple ⟨*Lattitude*, *Longitude*, and *Accuracy*⟩.

On-device Cache-based Location Calculation Algorithm. The detailed steps of privacy-aware geo-location calculation process are summarised in *Algorithm* 1. The surrounded beacons n_x are scanned and compared to the list of private WiFi fingerprints n_i to detect private place p stored in cached database c. Further, the representative l_r is altered based on set permissions (see Section 5.4).

5.4 Personalised Permissions for Location Sharing

A general LBS query consists of different attributes, e.g., LBS query {*POI, Latitude and Longitude, User-Info*}, where included geo-coordinates estimate the device's geo-location. To satisfy one of the privacy property called controlled information disclosure, we designed enhanced permission mechanism to control these geo-coordinates before it is sent to app providers. When using LP-Caché, for every installed app and set private place, the UI provides three distinct privacy settings: (1) Adjust Location Granularity, (2) Obfuscate Location and, (3) No Change. In the first option, geo-coordinate truncation method adjusts location precision level; in the second option, Algorithm 2: Enhanced Permissions Algorithm.

```
Input: $l_r$: Representative Location
Output: $l'_r$: Processed Location
 1: $u_p$ = User Input
 2: read $l$, $l_g$, $f$, $u_p$
 3: if $u_p$ = Adjust Granularity then
 4:     check granularity level $g_l$
 5:     truncate($l$, $l_g$)
 6:     replace $l$ to $l'$ and $l_g$ to $l'_g$
 7:     return $l'_r$
 8: else if $u_p$ = Obfuscate then
 9:     randomly generate angle $\theta$
10:     obfuscate($l$, $l_g$, $\theta$)
11:     replace $l$ to $l'$ and $l_g$ to $l'_g$
12:     return $l'_r$
13: else
14:     unchanged
15:     return $l'_r$
16: end if
```

geo-coordinate transformation obfuscate user's location; whereas, in the third option, the exact unchanged geo-coordinates are sent to the requesting app.

Enhanced Permissions Algorithm. Once LP-Caché receives an invoked location object l_r, it alters the location data according to the enhanced permission settings and returns processed location l'_r. The steps involved in enhanced permission mechanism are summarised in *Algorithm* 2, where u_p is the set permission, g_l is the adjusted location precision level, l is the latitude, and l_g is the longitude.

Geo-coordinates Truncation. The geographical coordinates looks like 52°28'59.200" N 1°53'37.0001" W, where the last digits specify more accurate geo-location. Geo-coordinate truncation method will enable us to adjust the location precision level, i.e., by removing last digits and rounding the location accuracy from street to city level or even more general. Generally, for any third party reuse, service providers or data collectors assure in the EULA that this method will be applied on the collected data since the truncated coordinates increase the ambiguity level(Aad and Niemi, 2010). On contrary, LP-Caché applies this method on the user device with user's permission in order to minimise the user's sensitive data collection and privacy concerns.

Geo-coordinates Transformation. For privacy preservation, position transformation functions such as scaling, rotation and translation have been used by location data distributors or anonymisers (Lin et al., 2008; Wernke et al., 2014). In LP-Caché, we use geo-coordinate transformation on the device to obfuscate user's private locations. Our service represents the geo-coordinate transformation using scaling and rotation, and denotes its parameters as a tuple $\langle s, \theta, (l, l_g)\rangle$, where s is the scaling factor, θ is the rotation angle, and (l, l_g) are the original coordinates. It applies Equation 2 to generate transformed or obfuscated geo-cordinates (l', l'_g), where angle θ is randomly generated.

Table 1: WiFi measurement dataset summary.

Total number of scans	25480
Distinct private locations selected	34
Total APs detected	486
Average APs detected	396

$$\begin{aligned} l' &= \theta(s.l) \\ l'_g &= \theta(s.l_g) \end{aligned} \tag{2}$$

6 FEASIBILITY AND USABILITY ANALYSIS

LP-Caché's actual performance evaluation depends on the location-based apps performance. In this section, we analise the WiFi AP data availability and consistency to measure feasibility of WiFi fingerprinting method to be included in LP-Caché's implementation.

6.1 WiFi APs Availability and Consistency

Experimental Set-up. The experimental set-up to measure WiFi AP data availability and consistency consists of three steps:

1. *Data collection.* We collected beacons from fixed WiFi APs using `WiEye` (WiEye, 2016) and `Network Info II` (NetworkInfoIi, 2016) apps on `Android` smartphones that have 802.11a/b/g/n radio feature so they can operate in both 2.4GHz and 5GHz bands at 34 different private places for a period of one month.
2. *Location categorisation.* App users are more concern about sharing their private locations(Almuhimedi et al., 2015); therefore, in our analysis, we selected three distinct categorise of private places: 1. Home (i.e., residential place), 2. Work (i.e., commercial place), and 3. Arbitrary (i.e., any frequently visited place) to determine categorical distribution pattern of WiFi APs.

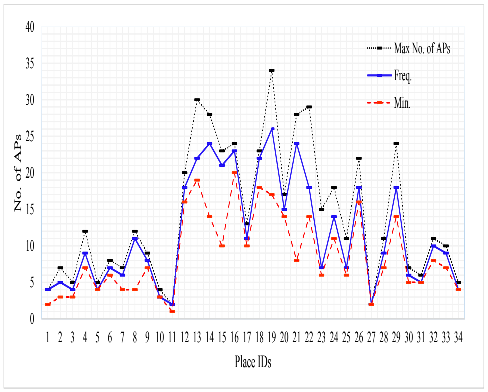


Figure 6: Measured density of detected WiFi APs at private places.

3. *Data analysis.* We collected and statistically analysed the scanned WiFi AP data. Table 1 compiles the included sample size; whereas, Figure 6 shows the relative difference between WiFi APs density, and Figure 7 depicts the relative average accuracy distribution pattern of detected WiFi APs for each category.

Observation. For each category of private places, experiments revealed the following:

Home The results demonstrate that Wifi APs are fixed and frequent and the difference between number of constant beacons and minimum number of similar beacons is comparatively less, and therefore, it achieved highest accuracy rate. Moreover, the ratio of SSID to BSSID is 1:1, i.e., 1 SSID (abc) has 1 BSSID (a0:12:b3:c4:56:78), this makes fingerprints distinct so improving the location detection performance.

Work This category has many fixed WiFi APs but with fluctuating signal strengths, and therefore, the sequence of available APs changes. However, the observed ratio of SSID to BSSID is many to one, i.e., 1 SSID has many BSSIDs; therefore, in this case, SSIDs along with BSSIDs can be used as unique identifiers to create a fingerprint to detect a private place dynamically.

Arbitrary In this category, the data collector could select any frequently visited locations, e.g., gym, shop, or friend's home. Figure 7 demonstrates that the outcome of this category is related to the other two categories, it either shows results similar to home or work.

The range of average accuracy for all the three categories of private places falls between 74% to 96%.

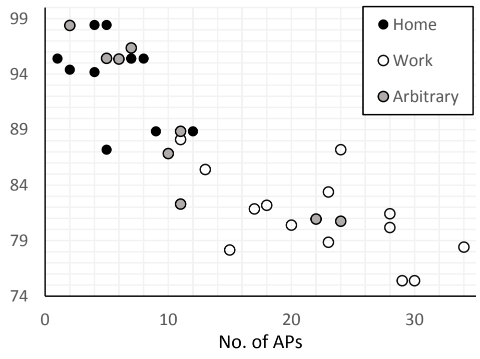


Figure 7: Relative average accuracy distribution pattern of detected WiFi APs at private places.

Hence, it is evident that smartphones regularly detect similar beacons at frequently visited place, for place detection at least one beacon should match with the stored fingerprints. Thus, the results demonstrates that WiFi fingerprinting can be effectively used as private place detection source in LP-Caché. Nonetheless, to achieve efficient capability for place recognition via beacons *place discovering algorithm* like (Kim et al., 2009) can be implemented (in future work).

6.2 Ongoing Evaluation of Caching Method

Following WiFi data availability and consistency analysis, LP-Caché's feasibility evaluation will be extended to analyse how frequently cache needs to be updated and what are the trade-offs of cache update frequency vs location privacy and accuracy in order to measure further computational and communication overheads. We also intent to conduct a thorough user study to determine comfortability of the users to accommodate LP-Caché's functionality. Moreover, we plan to consider the fundamental caching based technical challenges such as cache hits and cache misses, data freshness, data consistency, and estimated bandwidth requirements in the further development and implementation of LP-Caché paying special attention to storage-efficient caching.

7 CONCLUSIONS

Secure gathering and transfer of location data via smart mobile devices while at the same time preserving users' privacy are concerning needs. Research and industry communities are making joint efforts to identify the requirements for LBS applications in future large scale scenarios and addressing end user concerns. Studies based on cryptographic schemes and PETs have been tested on service provider's data collection servers but neither are implemented on the mobile platform, nor on the actual app operation. In addition, the lack of usability is one of the factors that hinders the adoption of existing privacy-aware solutions. We presented detailed analysis of the current location computation process and proposed a novel privacy-aware model. Both the end users and service providers benefits from LP-Caché since the on-device caching technique works on the minimisation of the user's private location collection process. With a personalised permission mechanism, users can manage each app and private place distinctly. Immediate future work focuses on the implementation of the model to measure the feasibility, usability and efficiency of our approach while interacting with different location-based apps.

REFERENCES

Aad, I. and Niemi, V. (2010). NRC data collection and the privacy by design principles. In *PhoneSense*.

Almuhimedi, H., Schaub, F., Sadeh, N., Adjerid, I., Acquisti, A., and Others (2015). Your Location has been Shared 5,398 Times! A Field Study on Mobile App Privacy Nudging. In *Procs. of ACM Conf. on Human Factors in Computing Systems*, pages 787–796. ACM.

Amini, S., Lindqvist, J., Hong, J., Lin, J., Toch, E., and Sadeh, N. (2011). Caché: caching location-enhanced content to improve user privacy. In *Procs. of ACM Int. Conf. on Mobile Systems, Applications, and Services*, pages 197–210. ACM.

Android Developer Reference (2016). http://developer.android.com/reference/.

Anonymizer (2016). http://www.anonymizer.com/.

Bell, S., Jung, W. R., and Krishnakumar, V. (2010). WiFi-based enhanced positioning systems: accuracy through mapping, calibration, and classification. In *Procs. of ACM SIGSPATIAL Int. Workshop on Indoor Spatial Awareness*, pages 3–9. ACM.

Beresford, A. R., Rice, A., Skehin, N., and Sohan, R. (2011). Mockdroid: trading privacy for application functionality on smartphones. In *Procs. of ACM Workshop on Mobile Computing Systems and Applications*, pages 49–54. ACM.

Cranor, L. F. and Sadeh, N. (2013). A shortage of privacy engineers. *IEEE Security & Privacy*, (2):77–79.

Damiani, M. L. (2011). Third party geolocation services in LBS: Privacy requirements and research issues. *Trans. on Data Privacy*, 4(2):55–72.

Doty, N. and Wilde, E. (2010). Geolocation privacy and application platforms. In *Procs. of ACM SIGSPATIAL Int. Workshop on Security and Privacy in GIS and LBS*, pages 65–69. ACM.

Egele, M., Kruegel, C., Kirda, E., and Vigna, G. (2011). PiOS: Detecting Privacy Leaks in iOS Applications. In *NDSS*.

Enck, W., Gilbert, P., Han, S., Tendulkar, V., Chun, B.-G., and Others (2014). TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *TOCS*, 32(2):5.

European Commission (2016). Protection of personal data. http://ec.europa.eu/justice/data-protection/.

Fawaz, K. and Shin, K. G. (2014). Location Privacy Protection for Smartphone Users. In *Procs. of ACM SIGSAC Conf. on Computer and Communications Security*, pages 239–250. ACM.

Felt, A. P., Egelman, S., and Wagner, D. (2012). I've got 99 problems, but vibration ain't one: a survey of smartphone users' concerns. In *Procs. of ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, pages 33–44. ACM.

Gibler, C., Crussell, J., Erickson, J., and Chen, H. (2012). AndroidLeaks: Automatically detecting potential privacy leaks in Android applications on a large scale. In *TRUST 2012*, pages 291–307. Springer.

Google Location Service (2016). https://support.google.com/gmm/answer/1646140?hl=en-GB.

Hellman, E. (2013). *Android programming: Pushing the limits*. John Wiley & Sons.

IETF (2016). Geographic Location Privacy. http://datatracker.ietf.org/wg/geopriv/charter/.

Jeon, J., Micinski, K. K., Vaughan, J. A., Fogel, A., and Reddy (2012). Dr. Android and Mr. Hide: fine-grained permissions in android applications. In *In Procs. of ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, pages 3–14. ACM.

Khoshgozaran, A., Shahabi, C., and Shirani-Mehr, H. (2011). Location privacy: going beyond K-anonymity, cloaking and anonymizers. *Knowledge and Information Systems*, 26(3):435–465.

Kim, D. H., Hightower, J., Govindan, R., and Estrin, D. (2009). Discovering semantically meaningful places from pervasive RF-beacons. In *Procs. of ACM Int. Conf. on Ubiquitous Computing*, pages 21–30. ACM.

Lin, D., Bertino, E., Cheng, R., and Prabhakar, S. (2008). Position transformation: a location privacy protection method for moving objects. In *Procs. of the SIGSPATIAL ACM GIS 2008 Int. Workshop on Security and Privacy in GIS and LBS*, pages 62–71. ACM.

Michael, K. and Clarke, R. (2013). Location and tracking of mobile devices: Überveillance stalks the streets. *Computer Law & Security Review*, 29(3):216–228.

Muslukhov, I., Boshmaf, Y., Kuo, C., Lester, J., and Beznosov, K. (2012). Understanding users' requirements for data protection in smartphones. In *ICDE Workshop, IEEE Int. Conf. on Secure Data Management on Smartphones and Mobiles*, pages 228–235. IEEE.

Navizon (2016). http://www.navizon.com.

NetworkInfoIi (2016). http://play.google.com/store/apps/.

Niu, B., Li, Q., Zhu, X., Cao, G., and Li, H. (2015). Enhancing Privacy through Caching in Location-Based Services. In *Proc. of IEEE INFOCOM*.

Patel, A. and Palomar, E. (2014). Privacy Preservation in Location-Based Mobile Applications: Research Directions. In *Procs. of IEEE Int. Conf. on Availability, Reliability and Security (ARES)*, pages 227–233. IEEE.

Pontes, T., Vasconcelos, M., Almeida, J., Kumaraguru, P., and Almeida, V. (2012). We know where you live: Privacy characterization of foursquare behavior. In *Procs. of ACM Conf. on Ubiquitous Computing*, pages 898–905. ACM.

Shklovski, I., Mainwaring, S. D., Skúladóttir, H. H., and Others (2014). Leakiness and Creepiness in App Space: Perceptions of Privacy and Mobile App Use. In *Procs. of ACM Conf. on Human factors in computing systems*, pages 2347–2356. ACM.

Skyhook (2016). http://www.skyhookwireless.com/.

TOR (2016). http://www.torproject.org/.

tPacketcapture (2016). http://play.google.com/.

Wernke, M., Skvortsov, P., Dürr, F., and Rothermel, K. (2014). A classification of location privacy attacks and approaches. *Personal and Ubiquitous Computing*, 18(1):163–175.

WiEye (2016). http://play.google.com/store/apps/.

Wireshark (2016). https://www.wireshark.org.

Zhu, X., Chi, H., Niu, B., Zhang, W., Li, Z., and Li, H. (2013). Mobicache: When k-anonymity meets cache. In *GLOBECOM*, pages 820–825. IEEE.

Zhuang, Y., Syed, Z., Georgy, J., and El-Sheimy, N. (2015). Autonomous smartphone-based WiFi positioning system by using access points localization and crowdsourcing. *Pervasive and Mobile Computing*.

2014 9th International Conference on Availability, Reliability and Security

Privacy Preservation in Location-based Mobile Applications: Research Directions

Asma Patel and Esther Palomar
School of Computing, Telecommunications and Networks
Faculty of Technology, Engineering and the Environment
Birmingham City University
Email: {asma.patel, esther.palomar}@bcu.ac.uk

***Abstract*—Proliferation of mobile devices equipped with position sensors has made Location-based Service (LBS) increasingly popular. These mobile devices send user's actual location information to the third party location servers, which compile and, in some cases, share with other service providers. As a result, users aware of the privacy implications feel continuously tracked. Effective and, even more important, socially-accepted privacy enhancing technologies for these services have recently received a lot of attention in academia and industry. This paper presents an overview of the privacy preserving techniques currently applied by LBS applications. It classifies these techniques into a classification model consisting of three layers. Thus, a brief description of all the protocols, mechanisms and interfaces covering from the application layer to the network layer are presented, also providing a comparative analysis of current privacy-aware location solutions. To guide future research, a new perspective of the literature findings is proposed and research questions, methods and implications are discussed. Novel to related work, our classification embraces a holistic picture of approaching privacy-aware mobile LBS.**

I. Introduction

Mobile devices equipped with improved positioning technology (e.g. GPS, wifi triangulation, IP address approximation, cell tower based identification, user provided information) have drastically increased the use of Location-Based Services (LBSs). In addition, cheap data storage allows information collectors to record many activities of such advanced mobile device users[1]. Nowadays, LBS applications are being leveraged by many companies because of the business growing around them, e.g. location-based games, location-based media, geo-marketing and geo social networking, to name a few. However, such applications can also be a serious threat to users' privacy [2]. The traditional approach to guarantee users' privacy has been based on the End User License Agreement (EULA) and other regulations or privacy policies, which do more to protect company's interests than to safeguard users' rights [3]. As a result, users aware of the privacy implications are reluctant to use such services on their mobile devices [4].

Example 1: Consider Foursquare, one of the most popular LBS which is completely location-based. To use the service, the user must turn on the location sharing settings on his/her device, and provide personal details for registration. At this stage, the user agrees on the terms and conditions via EULA stating "By submitting any personal information, I consent to having my personal information transferred to and processed in the US, which I understand may have different data protection rules than my country". This is also followed by another two EULAs: (i) giving consent of access to the user's contacts, and (ii) to push notifications. Some studies have already proven that users sensitive information collected by Foursquare's third party location servers is shared with other services causing sever privacy leakage [5]. However, benefits of LBS for the society are reasonably apparent, especially when they are associated with Assistive Healthcare Systems and emergent wearable technologies.

Certainly, a natural conflict arises when attempting to protect user privacy while building a system that allows for flexible use of location information [6]. Approaches to this challenge have applied cryptography and Privacy Enhancing Technologies (PETs) [7], [8], [9] which are still far from being socially-accepted. To avoid major privacy implications well known companies (e.g. `Google` and `Microsoft`) have identified the need of adopting a Privacy-by-Design (PbD) approach to consider privacy as an integral part in the design process of LBS applications and enhance PETs [10].

Thus, to guide future research a brief description of all the protocols, mechanisms and interfaces covering from the application layer to the network layer are presented in this paper. Compared to other related work, our classification embraces a holistic picture of approaching privacy-aware mobile LBS. This new perspective allows us to characterize and classify the protocols, to analyze the tradeoffs produced by different design decisions for mobile location-based applications, and to link the various aspects and data flows at every stage of the service, ranging from the users using the mobile device to the transmission of the user sensitive information within the query over the network.

The rest of the paper is organized as follows. Section II outlines our novel classification consisting of three layers for privacy protection of mobile LBS applications. We fully elaborate on each layer: mobile platform, query formation, and network communication in Sections III, IV, and V, respectively. In these sections, we highlight existing techniques, protocols, mechanisms, potential research areas, and related challenges also providing a comparative analysis of current privacy-aware location solutions. Section VI will conclude discussing promising directions for future research.

978-1-4799-4223-7/14 $31.00 © 2014 IEEE
DOI 10.1109/ARES.2014.37

II. Overview of our Classification

According to Duckham and Kulik [11], "location privacy is the right of individuals to decide how, when, and for which purposes their location information can be released to other parties". Furthermore, Suikkola [12] specified that the success of any PETs for LBS applications depends on critical factors such as the need of standardized interfaces, usability, security, privacy and society-acceptance.

Therefore, to identifying most effective solution, we must know who can officially collect such information? and how this information can violate users privacy? Observing the common architecture for LBS - which consist of four main entities: mobile device, positioning technologies, network communication, and service providers- there are three main types of service providers: 1.) Mobile Network Operator (MNO), 2.) Mobile Platform Providers (MPP), and 3.) Mobile Application Providers (MAP).

Example 2. Consider a mobile device with LBS application installed. MNO collects location information for providing signals. MPP provides services to develop mobile applications and distribute on there official repositories. MAP has to register with MPP (e.g. `Android` or `iOS`) in order to publish their application on the official repositories. Alternatively, such application can also be distributed on third party repositories (e.g. `Cydia` and `Amazon`).

Therefore, MNO/MPP/MAP are the main service providers, who can collect users location data and, in some cases, share for third party use. But, existing work presented so far surveying on privacy preservation in LBS [13], [14], [15] have considered the above challenge mainly by following an application-specific approach or by only focusing on the application of cryptographic primitives. Coming to network layer solutions, these solutions mainly depend on anonymous network communication techniques, in which even if the data packet is encrypted, both the source and destination addresses located in the packet's IP header are still visible to an eavesdropper. In our classification, we examine all the roles of these service providers at each layer based on the common LBS architecture to identify the most potential and promising directions to achieve socially-accepted PETs. Such, socially-accepted PETs will also benefit the sharing of location-based information amongst all the three service providers. Figure 1 depicts our classification which consists of three layers: Layer 1.) Mobile OS and APPs - to provide privacy-aware mobile device, Layer 2.) Query Formation - to provide privacy-aware location query processing, and Layer 3.) Network Communication - to provide privacy-aware networks communication.

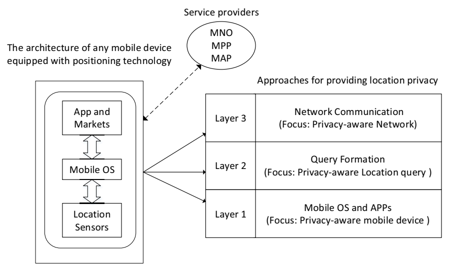

Service providers
MNO
MPP
MAP
The architecture of any mobile device equipped with positioning technology
Approaches for providing location privacy
App and Markets
Mobile OS
Location Sensors
Layer 3
Network Communication (Focus: Privacy-aware Network)
Layer 2
Query Formation (Focus: Privacy-aware Location query)
Layer 1
Mobile OS and APPs (Focus: Privacy-aware mobile device)

Fig. 1. Overview of Classification

III. Layer 1. Mobile Platform

In this layer, we analyse existing work on improving potential security mechanism and minimizing privacy risks in major MPP. Later, we propose some guidelines for further research directions in this domain. In particular, Layer 1 classifies existing and other proposed solutions used to provide users' privacy and security for using any third party applications (i.e. by thirty party MAP). Thus, this layer is divided into two sub topics: A) Current mobile OS protection mechanisms and B) Securing Mobile Applications and Repositories.

A. Current mobile OS protection mechanisms

Existing literature confirms that attackers mainly target users of popular MPP like `Android`, `iOS`, and `Windows Mobile` to get maximum amount of sensitive information. Therefore, lets understand the current prevention mechanism in such popular mobile platforms security models.

a) Android Security models: Android OS and Windows Mobile have similar security models for approving any third party applications to be distributed over their official repositories. Android OS security model consist of three main techniques:

- *Sandboxes*: Sandbox approach (i.e. creating an environment in which the actions of a process are restricted according to a security policy) for dynamic monitoring of third-party apps so they are restricted from accessing files stored by other apps, cannot make changes to the device and have limited access to the OS [16].
- *Digital Signature or Code Signing*: Developer must digitally sign the installation package file of every mobile application. The security model then maps the signature of the developer with a unique ID of the application package and enforces signature level permission authorization [17].
- *Permission based controls*: Android OS uses permission control using manifest file to show all required permissions to the user at installation time. If the user decides to grant permission to the application, then the protected resources are available to the application, otherwise the access to the resources is blocked [18].

b) iOS Security Model: On the contrary, `Apple` prefers keeping the actual vetting/reviewing process for a third-party application to be published on their official repositories unknown [19]. Yet, similar to techniques used by `Android OS` and `Windows Mobile`, `iOS` security model uses sandboxing and digital signatures but do not give permission based controls to their device users. Instead it uses heuristic-based method, i.e. defines and distributes application configuration settings by an installed configuration profile in the form of XML file and user cannot make changes in this file. According to apple's security policy, the identity verification process of developers to issue certificate is always unknown. Like, other platforms any third-party applications must conform to the predefined privacy rules and must be digitally signed using certificates issued by apple [20]. Moreover, if the user installs application from third party repositories or the device is jailbroken, `Apple` do not take responsibility of any private data leakage. This makes iOS platform very restricted, it prevents the developers to understand the flows in iOS security models and creates an assumption amongst apple users that the restricted and pre-defined terms and conditions are built to safeguard their privacy and security.

However, Mylonas have experimented location privacy attack on all mobile platforms, including `Android` OS, `BlackBerry` OS, `Apple` iOS, `Windows Mobile`, and `Symbian` OS, the result demonstrates that all the security models of existing mobile platforms are still vulnerable to location tracking attack and malicious code attack [21]. This proves that security models based on signature-based methods and heuristic-based methods (i.e. pre-defined rules) cannot secure user's privacy rights and require several major fixes. For this reason, many researchers[22], [23] are trying to incorporate another techniques (e.g. host-based intrusion detection application) to improve current security models. Table I compiles more details of the literature findings at this layer.

B. Securing Mobile Applications and Repositories

We now analyse the incorporation of security and privacy solutions into LBS applications running on users mobile devices. We mentioned earlier once the LBS application is installed and user grants access to their location, all the existing mobile OS security models are vulnerable towards location data tracking/leakage attacks. In essence, all the existing security models focus more towards companies policies rather than securing user's privacy rights. As a consequence, malicious attacks on location-based mobile applications (or Apps) is becoming very common amongst the the official marketplaces (e.g. `Apple store` or `Google market`) or the third-party App marketplaces (e.g. `Cydia`, `Amazon AppStore`) [36].

Many studies proposed application-specific layer solutions, mainly for `Android` platform. These studies are based on `Android`'s permission mechanisms for addressing two main goals. First, there is a focus on detecting and providing alerts when a leakage is recognized by applying a re-delegation of permission mechanism, e.g. `AndroidLeaks` [37], `TaintDroid` [38], and `Kirin` [39]. Secondly, to empower users with privacy controls aims at adjusting permission settings at run-time, e.g. `TISSA` [25] and `Locaccino` [26]. In addition, `DroidMOSS` [36] uses fuzzy hashing technique to match digital signatures and to detect repackaged Apps in third-party `Android` repositories. `DroidRanger` [40] uses permission-based behavioral footprinting and heuristics-based filtering to detects repackaged Apps in both, official and third-party, `Android` repositories. Both approaches identify that the third-party repositories facilitate users' information leakage.

In contrast, `Apple` does not provide user with permission-based control, this makes `iOS` platform restricted, limiting the monitoring and identification of potential sensitive information leakages. Nevertheless, `PiOS` [19] demonstrated several privacy threats within the `iOS` platform. Also, some applications available on `iOS` official repositories prompt for location sharing settings to be switched on, although it is not needed at all. Thus, users/MPP are unaware of such data gathering, manipulation or other usage of sensitive information by malicious MAP. In this layer, most promising research direction is to minimise/avoid unnecessary collection of users' personal information by the third-party applications and improve existing MPP security model against location tracking attack.

IV. Layer 2: Location Query Formation

LBS applications share location information with service providers in the form of query. The sharing of such queries may allow attackers to gain access to user location data. In location tracking applications, users' privacy can be at a higher risk because of continuous location queries are sent to the location server. Hence, in this layer, we analyse different privacy concepts and approaches to form privacy preserving location-based queries.

M. F. Mokbel [14] classified location-based queries into three categories namely private queries over public data, public queries over private data, and private queries over private data. Additionally, Shin et al. [13] categorised LBS privacy queries into two types: query privacy and location privacy. Wernke et al. [15] stated that privacy protection relies on three different attributes of the location query: identity, position, and time. In short, all of them emphasize that the privacy of LBS query depends on two main factors: the protection of users' sensitive data within the query, and the unlinkability between the user's location and the query.

Most proposals so far have classified the existing techniques for location privacy protection focusing on two popular location privacy metrics: entropy and k-anonymity [15], [41], [42]. Additionally, Chow and Mokbel [43] described how these techniques can be used in different architectures including Client-Server, Trusted Third Party (TTP), Mobile Peer-to-Peer P2P and fully distributed. We refer reader to the cited bibliography for details. Most commonly used techniques are briefly described as follows:

1) Privacy metrics: Most existing techniques in the privacy literature largely adopt privacy metrics such as k-anonymity (i.e. a total of k locations are sent in a request and the service provider is then unable to identify the user's real location with a probability higher than $\frac{1}{k}$) and location entropy, but are mainly centralized (i.e. rely on trusted location-based server/Anonymisers).

a) Dummy locations: Dummy locations are generated and sent to location servers along with the actual position to hide users' location [27]. This technique uses k-anonymity

TABLE I
ANALYSIS OF TECHNIQUES AND APPROACH STUDIED FOR EACH LAYER.

Layers		Techniques	Drawbacks	Related Work	Privacy properties (● = Fully, △ = Partially, ○ = Not at all)			
					Unlinkability	**Unobservability**	**Anonymity**	**Controlled info Disclosure**
L1	Mobile OS	- Permission based controls	Over declaration	[16]	○	△	○	●
		- Sandbox and digital signature	Weak to detect colluding applications attacks	[20][16]	○	○	○	●
		- Host based IDS	Needs to be installed as a separate application	[22]	○	○	○	●
	Secure Mobile Apps and repositories	- Re-delegation of permissions and policies	Weak towards malicious codes	[24]	○	○	○	●
		- Runtime Privacy Controls	Requires user interference	[25] [26]	○	△	○	●
L2	Location query using Privacy metrics	-Dummy locations	Usability issue	[27][28]	△	●	●	●
		-Cloaked region	Vulnerable to continuous querying		●	●	●	●
		-Mix- zone	Specifically for location tracking applications	[29]	●	●	●	○
		-Location Perturbation and Obfuscation	Obfuscation might reduce usability of data	[30]	●	●	●	○
	Location query using cryptography	-PIR	High computational and communication overhead	[9]	●	●	●	●
		-Encryption	Requires strong cryptography	[31]	●	●	●	●
L3	Anonymous communication	Anonymous routing protocols	More Expensive and vulnerable to internal attacks	[32][33] [34]	●	●	●	○
		Cryptographic primitive	Computational overhead	[35]	●	●	●	○

metric to measure users' location privacy. This approach can decrease the usability of actual data over dummy data, but essential advantage of this approach is that the user is able to generate dummies without the need of TTPs [28].

b) Cloaked region: Based on k-anonymity, several approaches [44] [7] used spatial and temporal cloaking to preserve location privacy. This technique can also be used with other privacy metrics such as l-diversity in `CliqueCloak` [45]. It is more suited for LBS requiring user's location information at a specific time, but with continuous querying the cloaked region becomes extremely large, and therefore, it fails to provide accuracy/usability, e.g. `ICliqueCloak` [46].

c) Mix-zone: A mix-zone technique refers to a service restricted area where mobile users can change their pseudonyms so that the mappings between their old pseudonyms (i.e. entering restricted area) and new pseudonyms (i.e. exiting restricted area) are not disclosed [29]. This techniques is suitable for services that continuously tracking users' movement in LBS applications. It depends on geometric location transformation and uses privacy metrics to measure the level of location privacy.

d) Location Perturbation and Obfuscation: Obfuscation-based techniques perturb (i.e. alter or move from its original position) the actual location information while maintaining a binding with the users identities. The idea of this approach is to utilize a log of historical user locations rather than the real-time location information to generate cloaked or obfuscated regions[30][47]. Again, it uses k-anonymity as a metric to measure user's location privacy.

2) Cryptographic approach: Cryptographic based techniques for query privacy are mainly used when location servers are not trusted. Following are few cryptographic approaches used in LBS applications for query executions:

a) PIR based techniques: Private Information Retrieval (PIR) based approach uses standard public key cryptography to provide location privacy without the need of privacy metrics or TTP. PIR protocols allow clients to retrieve the information privately from the server, without the server learning what information was requested by client. This techniques provides maximum query privacy but it causes high computational and communication overhead on the server, which makes it hard to be practically implemented in LBS [48] [49].

b) Other encryption base techniques for query privacy: Encryption based approach to query un-trusted servers are heavy-weight homomorphic [50], symmetric or asymmetric [51]. Compared to symmetric key encryption, asymmetric key encryption nodes are relatively much more expensive and it can be used for location query in location based social applications, e.g. `LocX` [31]. Other cryptographic primitives like group signature, blind signature and ring signatures are used into privacy preserving communication protocols. These cryptographic primitives have been used for anonymous user authentication or other access control operations. In LBS applications cryptographic primitives are mainly used in continuous spacial-temporal tracking applications [35].

In short, query formation using cryptographic approach is more suited for privacy preserving but cryptographic operations can incur high query processing, computational and communication overheads on the servers. As a result, this opens another research direction to minimise these overheads on the location servers.

V. Layer 3: Privacy-aware Network Communication

This layer reviews major technologies and communication protocols including mixed networks, onion routing to identify possibility of improvements and variations for mobile/LBS scenario. In LBS, mobile network is used for calculating the user location by having network nodes communicating between each others. Existing network layer solutions mainly depend on anonymous network communication techniques, in which even if the data packet is encrypted, both the source and destination addresses located in the packet's IP header are still visible to an eavesdropper. Therefore, academia and industrial research are focus on designing and building privacy and security schemes as an infrastructure that run on the top of existing Internet protocols allowing users to communicate with each others without disclosing their network identifiers. Some of the successful anonymous routing protocols are:

A. Anonymous communication techniques

Communicating over anonymous network can provide LBS users with strong unlinkability, unobservability, and anonymity. Privacy preserving solutions like `Anonymizer` [33], and `TOR` [32] deal with anonymous service usage at the network layer while communicating over Internet (i.e. the server can see the location data without knowing the identity of users). They create a circuit with randomly selected `Tor` relays, which encrypts the original data from source to destination including the destination IP address. Another solution called `Crowds` [34] is based on the idea of blending into a crowd (i.e. users request can either be submitted to the end location server or forward it to another random chosen router). As a result, no third-party knows the origin of the requested query. `Hordes` [52], uses multicast routing to achieve anonymous communication over the Internet. However, all of these protocols have to be prevented from several security attacks such as the inference of the user location using the GPS trace without affecting the performance of the LBSs. Amongst all the above protocols only few anonymizing networks have been tested for the mobile Internet scenario and it is an area that attracted research interest.

B. Measuring Anonymity

Anonymity set size, k-anonymity, individual anonymity degree, and entropy anonymity are few anonymity metrics to measure anonymity in an anonymous network [53].

In this layer, existing literature proves that both anonymous routing and anonymous communication protocols, requires further research to complement/improve current privacy preserving network layer solutions. Also, there are other cryptographic primitives that provide a high degree of anonymity, but cannot be applied into network communication due to high cost in terms of network traffic and processing, e.g. the Dining Cryptographers.

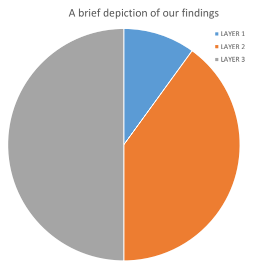


Fig. 2. A summary using pie diagram.

VI. Conclusion and Future Direction

Standards for LBS applications exist in both the 3GPP [54] and IETF [55] arenas. The need to securely gather and transfer location information for LBS, while at the same time protecting the privacy of the users involved have been identified. However, industry and academia are making joint efforts to identify the requirements for LBSs in future large scale community and better addressing end user concerns. Current research demands PETs to protect user location privacy while encouraging the use of LBS applications, this will benefit both the end users and service providers. In this paper, we have proposed a novel classification which embraces a holistic picture of approaching privacy-aware mobile LBS. Hence, our survey provides better focused classification of the possible solutions towards improving location privacy preserving mechanisms. We have identified three different layers: Layer 1) Mobile OS and APPs, Layer 2) Query formation, and Layer 3) Network Communication.

In summary, to come up with an efficient socially-acceptable solution, all the above layers must be considered. We have described each layer's shortcomings. Figure. 2 provides a sketch of our findings, and Table I illustrates each layer's most common approaches satisfying the privacy properties. Based on entire conducted analysis, compared to approaches in all the layers in our classification Layer 2 approaches fully satisfy the required privacy properties. Besides, following are the three main directions we have identified for future research in each layer:

- Layer 1: Insights from current mobile OS security models regarding location attacks attract the attention for a more rigorous vetting process for regulation of third-party location application and repositories. The future research that aims at an application level solution must incorporate other techniques against intrusion.
- Layer 2: We distinguish two direction for future work to this regard. First, new metrics can be applied such as l-diversity, t-closeness and differential privacy to measure query privacy. Secondly, applying better cryptographic

approaches but sametime reducing the computational and communication overheads on the location servers' side.

- Layer 3: Anonymous and unlinkable sharing of location information between the LBS applications and the location servers over the network can be further elaborated to provide a privacy-aware LBS framework form the network layer's viewpoint.

References

[1] R. Roman, J. Zhou, and J. Lopez, "On the features and challenges of security and privacy in distributed internet of things," *Computer Networks*, vol. 57, no. 10, pp. 2266–2279, 2013.

[2] I. Shklovski, S. D. Mainwaring, H. H. Skúladóttir, H. Borgthorsson, and R. L. Vej, "Leakiness and creepiness in app space: Perceptions of privacy and mobile app use," 2014.

[3] N. Ozer, C. Conley, D. H. O'Connell, T. R. Gubins, and E. Ginsburg, "Location-based services: time for a privacy check-in," *ACLU of Northern California*, 2010.

[4] I. Muslukhov, Y. Boshmaf, C. Kuo, J. Lester, and K. Beznosov, "Understanding users' requirements for data protection in smartphones," in *Data Engineering Workshops (ICDEW), 2012 IEEE 28th International Conference on*. IEEE, 2012, pp. 228–235.

[5] T. Pontes, M. Vasconcelos, J. Almeida, P. Kumaraguru, and V. Almeida, "We know where you live: Privacy characterization of foursquare behavior," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM, 2012, pp. 898–905.

[6] G. Andrienko, A. Gkoulalas-Divanis, M. Gruteser, C. Kopp, T. Liebig, and K. Rechert, "Report from dagstuhl: the liberation of mobile location data and its implications for privacy research," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 17, no. 2, pp. 7–18, 2013.

[7] M. Gruteser and D. Grunwald, "Anonymous usage of location-based services through spatial and temporal cloaking," in *Proceedings of the 1st international conference on Mobile systems, applications and services*. ACM, 2003, pp. 31–42.

[8] Y. Sun, T. F. La Porta, and P. Kermani, "A flexible privacy-enhanced location-based services system framework and practice," *Mobile Computing, IEEE Transactions on*, vol. 8, no. 3, pp. 304–321, 2009.

[9] A. Khoshgozaran, C. Shahabi, and H. Shirani-Mehr, "Location privacy: going beyond k-anonymity, cloaking and anonymizers," *Knowledge and Information Systems*, vol. 26, no. 3, pp. 435–465, 2011.

[10] L. F. Cranor and N. Sadeh, "A shortage of privacy engineers," *Security & Privacy, IEEE*, vol. 11, no. 2, pp. 77–79, 2013.

[11] M. Duckham and L. Kulik, "Location privacy and location-aware computing," *Dynamic & mobile GIS: investigating change in space and time*, vol. 3, pp. 35–51, 2006.

[12] V. Suikkola, "Open exposure of telco capabilities-identification of critical success factors for location-based services in open telco," in *Wireless and Mobile Communications (ICWMC), 2010 6th International Conference on*. IEEE, 2010, pp. 202–208.

[13] K. G. Shin, X. Ju, Z. Chen, and X. Hu, "Privacy protection for users of location-based services," *Wireless Communications, IEEE*, vol. 19, no. 1, pp. 30–39, 2012.

[14] M. F. Mokbel, "Privacy in location-based services: State-of-the-art and research directions," in *Mobile Data Management, 2007 International Conference on*. IEEE, 2007, pp. 228–228.

[15] M. Wernke, P. Skvortsov, F. Dürr, and K. Rothermel, "A classification of location privacy attacks and approaches," *Personal and Ubiquitous Computing*, vol. 18, no. 1, pp. 163–175, 2014.

[16] Android, "Android security overview," http://source.android.com/devices/tech/security/index.html.

[17] A. K. Jain and D. Shanbhag, "Addressing security and privacy risks in mobile applications," *IT Professional*, vol. 14, no. 5, pp. 0028–33, 2012.

[18] A. Mylonas, S. Dritsas, B. Tsoumas, and D. Gritzalis, "Smartphone security evaluation-the malware attack case." *SECRYPT*, vol. 11, pp. 25–36, 2011.

[19] M. Egele, C. Kruegel, E. Kirda, and G. Vigna, "Pios: Detecting privacy leaks in ios applications." in *NDSS*, 2011.

[20] Apple, "ios security," http://images.apple.com/ipad/business /docs/iOS_Security_Feb14.pdf.

[21] A. Mylonas, A. Kastania, and D. Gritzalis, "Delegate the smartphone user? security awareness in smartphone platforms," *Computers & Security*, vol. 34, pp. 47–66, 2013.

[22] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss, "andromaly: a behavioral malware detection framework for android devices," *Journal of Intelligent Information Systems*, vol. 38, no. 1, pp. 161–190, 2012.

[23] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid: behavior-based malware detection system for android," in *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*. ACM, 2011, pp. 15–26.

[24] W. Enck, M. Ongtang, P. D. McDaniel *et al.*, "Understanding android security." *IEEE Security & Privacy*, vol. 7, no. 1, pp. 50–57, 2009.

[25] Y. Zhou, X. Zhang, X. Jiang, and V. W. Freeh, "Taming information-stealing smartphone applications (on android)," in *Trust and Trustworthy Computing*. Springer, 2011, pp. 93–107.

[26] E. Toch, J. Cranshaw, P. Hankes-Drielsma, J. Springfield, P. G. Kelley, L. Cranor, J. Hong, and N. Sadeh, "Locaccino: a privacy-centric location sharing application," in *Proceedings of the 12th ACM international conference adjunct papers on Ubiquitous computing-Adjunct*. ACM, 2010, pp. 381–382.

[27] H. Kido, Y. Yanagisawa, and T. Satoh, "An anonymous communication technique using dummies for location-based services," in *Pervasive Services, 2005. ICPS'05. Proceedings. International Conference on*. IEEE, 2005, pp. 88–97.

[28] T.-H. You, W.-C. Peng, and W.-C. Lee, "Protecting moving trajectories with dummies," in *Mobile Data Management, 2007 International Conference on*. IEEE, 2007, pp. 278–282.

[29] A. R. Beresford and F. Stajano, "Mix zones: User privacy in location-aware services." in *PerCom Workshops*, 2004, pp. 127–131.

[30] A. Pingley, N. Zhang, X. Fu, H.-A. Choi, S. Subramaniam, and W. Zhao, "Protection of query privacy for continuous location based services," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 1710–1718.

[31] K. Puttaswamy, S. Wang, T. Steinbauer, D. Agrawal, A. El Abbadi, C. Kruegel, and B. Zhao, "Preserving location privacy in geo-social applications," 2012.

[32] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," DTIC Document, Tech. Rep., 2004.

[33] Anonymizer, "Anonymizer," https://anonymizer.com.

[34] M. K. Reiter and A. D. Rubin, "Crowds: Anonymity for web transactions," *ACM Transactions on Information and System Security (TISSEC)*, vol. 1, no. 1, pp. 66–92, 1998.

[35] J. Ren and J. Wu, "Survey on anonymous communications in computer networks," *Computer Communications*, vol. 33, no. 4, pp. 420–431, 2010.

[36] W. Zhou, Y. Zhou, X. Jiang, and P. Ning, "Detecting repackaged smartphone applications in third-party android marketplaces," in *Proceedings of the second ACM conference on Data and Application Security and Privacy*. ACM, 2012, pp. 317–326.

[37] C. Gibler, J. Crussell, J. Erickson, and H. Chen, "Androidleaks: Automatically detecting potential privacy leaks in android applications on a large scale," in *Trust and Trustworthy Computing*. Springer, 2012, pp. 291–307.

[38] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. Sheth, "Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones." in *OSDI*, vol. 10, 2010, pp. 1–6.

[39] W. Enck, M. Ongtang, and P. McDaniel, "On lightweight mobile phone application certification," in *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, 2009, pp. 235–245.

[40] Y. Zhou, Z. Wang, W. Zhou, and X. Jiang, "Hey, you, get off of my market: Detecting malicious apps in official and alternative android markets," in *Proceedings of the 19th Annual Network and Distributed System Security Symposium*, 2012, pp. 5–8.

[41] J. Krumm, "A survey of computational location privacy," *Personal and Ubiquitous Computing*, vol. 13, no. 6, pp. 391–399, 2009.

[42] A. Khoshgozaran and C. Shahabi, "A taxonomy of approaches to preserve location privacy in location-based services," *International Journal of Computational Science and Engineering*, vol. 5, no. 2, pp. 86–96, 2010.

[43] C.-Y. Chow and M. F. Mokbel, "Privacy in location-based services: a system architecture perspective," *Sigspatial Special*, vol. 1, no. 2, pp. 23–27, 2009.

[44] L. Sweeney, "Achieving k-anonymity privacy protection using generalization and suppression," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 571–588, 2002.
[45] B. Lee, J. Oh, H. Yu, and J. Kim, "Protecting location privacy using location semantics," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 1289–1297.
[46] X. Pan, J. Xu, and X. Meng, "Protecting location privacy against location-dependent attacks in mobile services," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 24, no. 8, pp. 1506–1519, 2012.
[47] D. Quercia, I. Leontiadis, L. McNamara, C. Mascolo, and J. Crowcroft, "Spotme if you can: Randomized responses for location obfuscation on mobile phones," in *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*. IEEE, 2011, pp. 363–372.
[48] G. Ghinita, "Privacy for location-based services," *Synthesis Lectures on Information Security, Privacy, & Trust*, vol. 4, no. 1, pp. 1–85, 2013.
[49] S. Papadopoulos, S. Bakiras, and D. Papadias, "Nearest neighbor search with strong location privacy," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 619–629, 2010.
[50] H. Hu, J. Xu, C. Ren, and B. Choi, "Processing private queries over untrusted data cloud through privacy homomorphism," in *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*. IEEE, 2011, pp. 601–612.
[51] W. K. Wong, D. W.-l. Cheung, B. Kao, and N. Mamoulis, "Secure knn computation on encrypted databases," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. ACM, 2009, pp. 139–152.
[52] C. Shields and B. N. Levine, "A protocol for anonymous communication over the internet," in *Proceedings of the 7th ACM conference on Computer and communications security*. ACM, 2000, pp. 33–42.
[53] D. Kelly, R. Raines, R. Baldwin, M. Grimaila, and B. Mullins, "Exploring extant and emerging issues in anonymous networks: A taxonomy and survey of protocols and metrics," *Communications Surveys & Tutorials, IEEE*, vol. 14, no. 2, pp. 579–606, 2012.
[54] 3GPP, "Functional stage 2 description of location services (lcs)," 23.271 Version 6.8.0 (2006-06).
[55] IETF, "Geopriv working group," http://www.ietf.org/html.charters/geopriv-charter.html.

Manuscript developed for publication

A Practical Cache-based Location Privacy-enhanced Middleware for Mobile Users

Asma Patel[a,b,], Esther Palomar[a]

[a]*School of Computing and Digital Technology*
Birmingham City University, UK
[b]*School of Computing and Digital Technology*
Staffordshire University, UK

Abstract

In the evolving mobile age, use of smartphones equipped with advanced positioning technology can pose critical threats to users' location privacy provoking continuous tracking, profiling and unauthorized identification. At present, the mobile app ecosystem relies on permission-based access control and policy practices, which are proven ineffective at controlling how third party apps and/or library developers use and share users' data. In this paper, we present and evaluate a privacy-enhancing middleware that enforces location privacy over both the information and control flows occurred between sources and sinks. Through a caching-based mechanism, our middleware minimises the interaction and data collection from wireless access points, content distributors and location providers, and it provides enhanced privacy controls to prevent disclosure of user's private locations. We implement our middleware on Android 6, and present its comprehensive evaluation in terms of performance and security. Using our middleware, we conducted series of experiments with real apps from five different categories of location-based services, e.g., instant messaging and navigation. Experiments demonstrate acceptable computational efforts, memory consumption (upto 21.3 MB of RAM usage), delay overheads within practical limits (below 22 milliseconds), limited storage overhead (136 KB on device storage for a period of 3 months). To assess privacy/data leakage within the collected datasets, we conduct security analysis that demonstrate our middleware mitigates critical location privacy threats at a tolerable loss of in Quality-of-Service (QoS) of location-based apps and the underlying OS. Hence, our middleware is practical, secure and efficient location privacy preserving solution.

Keywords: Location Privacy, Location-based Services, Smartphones, Caching, Location-based Applications, Android, Mobile Platforms

1. Introduction

Recent advancement in wireless indoor positioning techniques and systems have leveraged tremendous opportunities for a whole new class of Location-Based Services(LBS) that mobile[1] users can benefit from while at home, work, shopping mall, or university [1]. Moreover, today's demand for indoor localisation services have become a key prerequisite in some commercial markets that lead to futuristic geo-marketing and geo-social networking, monitoring, and assisted eHealth techniques, to name a few, as well as some new platforms, e.g., for location-based educational augmented reality games or energy consumption 3D maps in smart communities. Such growing businesses and easy access to location data via mobile devices both result in high number of location-demanding apps, which compute on user's sensitive data and pose a serious threat to the users' privacy [2, 3].

Common approaches to privacy of user location on smartphones are based on two methods namely i) permission controls as a binary process[2] and ii) privacy policies[3]. In former method, mobile Operating Systems (OS[4]) implement permission-based access control for data sources and sinks (third-party apps and libraries); however, they do not control *flows* between the authorised sources and sinks. In latter method, privacy policies are encoded in natural language and are directly enforced by users. Hence, users are forced to rely on third party service providers that in many cases continuously collect, use and share their location data and, in some other cases, prompt the user to give away geo-position upon page loading [7, 2, 8, 3].

Motivation. To analyse users' location privacy preferences, we conducted a field study. We designed a survey questionnaire and distributed it within the University and on social media platforms. In total, we surveyed 190 smartphone users. 89.19% of the respondents expressed that they are concerned about their location privacy, and that they care about who has access to their location information. Whereas, 91.89% users think granting permissions to apps on their device to access continuous and precise location can result in violation of their privacy. 89.10% the respondents are more concerned about their privacy while sharing their private locations such as home and work. 82.18 % of the respondents rely on location result accuracy and location-based app functionality when they are anywhere outside or unknown places. Further, 78.92% users agreed that their is need to for better privacy controls on their devices and are willing to install a location privacy enhancing tool on their devices.

Hence, we adopt a Privacy-by-Design (PbD) approach for the development of a more user-friendly and socially-accepted solution to location privacy on LBS, in compliance with the existing privacy regulations for mobile products and services [9]. In particular,

[1]Throughout this paper, we use the terms smartphones and mobile interchangeably

[2]Data protection directives and acts [4, 5] across the globe state that personal data should not be disclosed or shared with third parties without consent from subject(s). Such a consent is typically obtained by mandatory acceptance of the conditions mentioned in the End User License Agreement (EULA), or through opt-out possibilities and other regulations[6].

[3]A privacy policy specifies the privacy practices of an organisation, basically what kind of personal information is collected, the purpose and how the information will be used/shared.

[4]Throughout this paper, we use the terms OS and mobile platform interchangeably

our approach campaigns for a new design principle and privacy policy recommendation that forces the smartphone app ecosystem to make location data use patterns explicitly, while preventing all other sensitive data flows from unauthorised leakage.

In this article, we present the design, deployment and evaluation of the middleware called *Private Location Protector* (PL-Protector), which implements the LP-Caché model introduced in [10] and enhances [11]. PL-Protector envisions beyond the simple grant/deny access method and provides the user with advanced mechanisms to decide the extent of disclosing location data with service providers. It also incorporates caching technique to determine users' geographical location in a privacy preserving manner by means of wireless access points (AP), and with minimum cache storage requirements. Several caching based solutions [12, 13, 14] have been proposed to minimise the risk of major location privacy threats, but lacking of deployment feasibility. They rely on unrealistic assumptions such as vast cache data storage requirements, or on the app developers modifying the code to incorporate their cached databases. By contrast, PL-Protector incorporates caching technique with minimum cache storage requirements. The main contributions of our proposal are as follows:

- Based on the analysis of existing mobile platforms and users' perspectives, we present enhanced design and implementation of a location privacy-enhancing middleware, which is a prototype developed to validate the theoretical model, LP-Caché. We have successfully implemented our middleware on Android Platform (`Android version 6`) to enforce the privacy rules over both the information and control flow occurred between source (OS) and sink (apps).
- Through the implementation of our middleware, we proved the deployment feasibility of a new series of privacy controls on a mobile platform to prevent private location disclosure during the formation of LBS queries. It only requires process isolation and IPC services from the underlying OS; thus, minimizing the requirements placed on the hardware/OS.
- We perform a thorough evaluation of our middleware in terms of performance and security analysis. Our results show that our middleware provides users with location privacy at a tolerable loss of app functionality and acceptable overheads on the underlying OS. Hence, we find our proposal to be a practical, privacy-enhanced, secure, and efficient for location-demanding apps.

The rest of the paper is organized as follows. Section 2 outlines the current location computation process and its evaluation. Section 3 overviews PL-Protector's system model system roles, threat, mobility, app usage and privacy model. In Section 4, we fully elaborate on the design decisions, architecture and implementation of the middleware. We evaluate PL-Protector's performance in terms of developer efforts, cache storage overheads, communication and computation overheads, and analyse its security all in Section 5. Section 7 reviews the related work. Finally, Section 8 concludes and describes current work as well as future research plans.

2. Background

In this section, we analyse the location privacy threats within the smartphone app ecosystem by studying how the location calculation process works in smartphones and how LBS apps collect the user location data. We also justify our decision on implementing PL-Protector as middleware for Android platforms. Nonetheless, our results can be extrapolated to other permission-based mobile platforms such as iOS.

2.1. Location sources

To understand location privacy specific challenges and security design issues, we start analysing the process of location calculation in smartphones when using LBS. Based on our prior study [10], we understand that the three major existing mobile platforms, namely *Android*, *Windows* and *iOS* that span the domains of smartphones, follow common patterns of location data retrieval. Basically, the standard architecture of using LBS on a mobile platform comprises four main entities: 1) User Device i.e. installed apps, 2) App Provider, 3) Network Infrastructure, and 4) Location Provider. The user device collects the unique identifiers from the surrounding network access points along with GPS data, and sends these over to the location provider to get the exact device location. Listing 1 and 2show the structure of the WiFi and Cell-tower objects sent to the location provider.

Listing 1 Structure of WiFi AP object sent to the location provider

```
{"wifiAccessPoints": [{
  "macAddress/BSSID": "11:22:33:44:55:YZ",
  "signalStrength": 50,
  "age": 0,
  "signalToNoiseRatio":-60,
  "channel": 8
  {
  "macAddress/BSSID": "11:22:33:44:55:YZ",
  "signalStrength": 50,
  "age": 0,
  }
}]
}
```

Listing 2 Structure of cell-tower object sent to the location provider

```
"CellTowers": [
{
    "cellId": 01,
    "locationAreaCode": 415 ,
    "mobileCountryCode": 310,
    "mobileNetworkCode": 410'
    "age": 0,
    "signalStrength": -60,
    "timingAdvance":15
} ]
```

Calculation[5] of the user device's actual position is then performed by the location provider who sends back a location object (see Listing 3) containing exact geo-coordinates.

Listing 3 Structure of the location object received from the location provider

```
{"location": {
  "lat": 54.0,
  "lng": -0.012,
  },
  "accuracy": 190.2,
}
```

At the user device, this location object is shared amongst installed apps as well as with the app provider who will transmit it as a LBS query via the standard programming interface/API [15]. Moreover, this location object is also used to estimate places of interest (PoI) of users' daily lives. Simple eavesdropping on this location object is a major threat to this architecture even if users put in place the corresponding location sharing preferences[6], which generally are highly context sensitive and use dependent [16].

2.2. *Estimation of PoIs*

A general LBS query consists of different attributes, e.g., LBS query { PoI, Latitude and Longitude, User-Info}, where included geo-coordinates estimate the device's geo-location and then generate user's PoIs. Service providers use two categories of techniques to estimate the user's PoI from collected smartphone data [17]:

Geometry-based techniques. Geometry-based algorithms use LBS queries (i.e., location data) to trace geo-coordinates, circles or polygons of regions to define the significant PoIs or private places where the user goes in real life.

Fingerprint-based algorithms. Fingerprint (or signature) based algorithms obtain a list of places where the user goes, but provide no direct information about where the place is geographically located unless the signature of observed fixed network infrastructure is mapped to the geo-coordinates. In general, fingerprint-based technique detects fixed radio/wireless environments that indicate a stay or frequent visits of user device for PoI estimation. WiFi-based fingerprinting involves mapping of observed wireless APs with signatures, pre-stored on a remote location servers, to generate list of PoIs.

[5]Location calculation is commonly based on the positioning technologies such as WiFi Triangulation and Cell-tower Triangulation, GPS Mapping, etc.

[6]Types and levels of controls for user location privacy settings depend on the OS and apps. In some cases, apps do not allow users to control others' access to their location data.

PL-Protector uses fingerprinting to create private location database within the device instead of storing it on a remote location servers. This minimises the process of wireless AP data collection by the WiFi content distributors or location providers. In addition, PL-Protector controls information disclosure within the generated LBS query (e.g., PoIs and nearest neighbor) since it will be sent to third-party app providers.

2.3. Operating System Controls and Apps' Location Access

In any permission-based mobile OS, apps can only access sensitive resources through the official APIs once the corresponding permissions declared at the manifest files are granted and authorised by the user. For instance, in iOS and Android (6.0 or API level 23 onwards) users grant permissions to apps while the app is running, not when they install the app. However, in both cases, a positive user authorisation might result in other remote third parties and external sources benefiting from this information made available in ad-libraries for commercial purposes and/or untrusted code execution [18, 19]. These existing studies and reports of data-stealing malware on smartphones clearly show the need of a better run-time permission method regulating the way apps and ad libraries are integrated into Android and other permission based platforms. Since existing OS's location access controls by system services respond inadequately to major privacy threats [7, 20], we deployed enhanced permission mechanism to control user's sensitive location data before it is sent to app providers.

3. System Model

We characterize PL-Protector's system model considering system roles, the threat model and evaluation metrics for both app usage and privacy protection.

3.1. System roles

PL-Protector modifies the current location resource handling process in mobile systems; however, the involved entities and their roles remain the same. PL-Protector follows the sequence of processes and messages as described in [10]. Here we clarify the role of each entity by describing the overall communication procedure in presence of PL-Protector as follows (Figure 1).

1. *Middleware* determines two main components privacy settings and on-device cache to overcome the shortcomings related to user privacy within the existing mobile systems. It then functions according to the pre-set privacy rules that secure the calculation and transmission of sensitive location data.

2. *User* needs to set privacy choices using settings, i.e., User Interface (UI), by marking sensitive locations as private, selecting preferred level of privacy for every app and location distinctly. User also has an option to manually map the location (i.e, geo-coordinate) with the observed network signature that will result into no involvement of the *location provider*.

Manuscript developed for publication

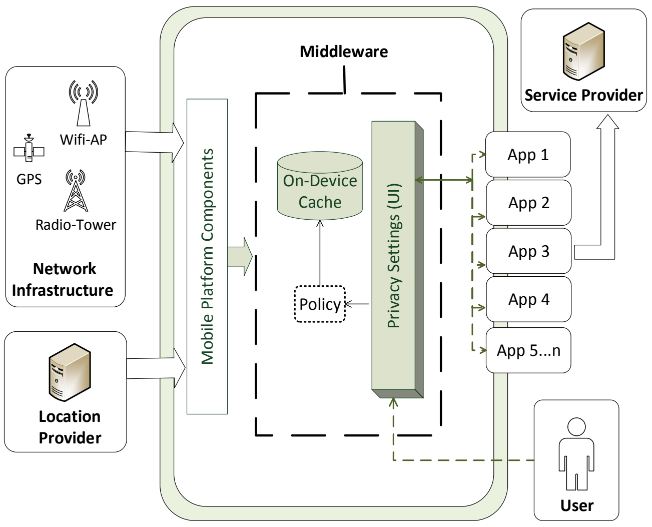


Figure 1: System roles

3. *Service providers'* operations are unaffected, i.e., they can follow their standard process. However, *service providers* benefit from secure gathering and transfer of location data using mobile devices while at the same time preserving users' privacy since the *middleware* minimises sensitive location data flows and privacy concerns without affecting their app's operations or QoS.

4. *Mobile platform*'s location access controls respond inadequately to major privacy threats [7, 20]; however, the *middleware* complements existing controls by better regulating the way apps and ad libraries access private location data at run-time.

5. *Network infrastructure* availability and consistency is extremely important to the performance of included fingerprinting method, which requires fixed wireless APs[7] data to create on-device cache of private locations.

6. In case of an unmatched entry on the cached locations, only if the user has set the current location as private and do not want to input geo-coordinates manually (via UI), the *location provider* is required to calculate and send the exact geo-coordinates (via location object, see Listing 3) first time.

Pseudocode in Listing 4 illustrates a social networking app interacting with PL-Protector. Our middleware ensures that when the app requests a location update at private place, the privacy policies are applied before sending LBS query to the service provider (i.e., the social networking app provider).

[7]Initially, we decided to focus on WiFi APs since they infer accurate user location. However, we can later include other fixed radio sources (e.g., Cell tower unique identifiers).

```
application  SocialApp
request  {LBS_Query.NearestNeighbour -> loc}
return flag(PrivLoc_recog(appContext, nBeacons));
if (flag = ON)
Location loc = extractCacheLoc(appContext, nBeacons);
void applyPolicy(loc);
Handle appSession = sessionHandler(appContext);
receive loc from PL-Protector;
send request to service provider;
```

Listing 4 Pseudocode for a social networking app interacting with PL-Protector - get location data and send LBS query response.

3.2. Threat Model

We consider two different attack scenarios to PL-Protector mainly caused by the level of access to user location in and out the device, as follows:

OS's Middleware Layer Threats: A series of attacks operates at Android's middleware layer[21]. PL-Protector mitigates the location privacy attacks coming from over-privileged and malicious 3rd party apps and libraries. The former can threaten user privacy by gaining unauthorized access to location, and other user sensitive information, that are not required for their operation. Underlying purpose lies in general in feeding advertisement libraries and, ultimately, exploiting the permissions of the host app [22]. The later can leverage unauthorized location permissions for financial gains and leak users' mobility and behaviour information (e.g., unauthorized profiling).

Privacy Threats: User tracking, identification and profiling (i.e. personal habits, movement patterns, etc.) are fundamental threats to location privacy [23]. Without PL-Protector, there is a continuous flow of LBS queries between user devices and location providers that include device's exact geo-coordinates and other sensitive information. This can leverage malicious misuse of location data, especially in the presence of a malicious location provider and via advanced network sniffing practices.

PL-Protector computes the exact location within the user device, without the service provider's involvement, whilst trusting the device on the storage of sensitive data. However, the user has still the option of giving consent for app providers and/or location providers to access location data. Mobile network providers might, however, collect user location data via cellular clients. It is also excluded from our work the option of manually inserting the location data (e.g., street name, post code, post code) within the LBS query.

3.3. *Preliminaries*

We now model the user mobility and app usage (specifically at private places) as a series of privacy evaluation metrics that will be used to validate PL-Protector's working assumptions.

3.3.1. *Mobility Model*

We formulate user's PoIs (e.g., Home and Work) as private places that the user frequently visit; hence, p_i represent ith private place identification, which is derived from a series of scanned beacons n_x and the representative location l_r for that private place, as shown in Eq. 1 and 2).

$$p_i = [n_1], [n_2], ..., [n_x] \rightarrow [l_r] \tag{1}$$

$$P_l = [p_i], [p_j], ..., [p_n] \tag{2}$$

At location p_i, the user can then visit a subset of private places $U_{p_i} \subseteq p_1, p_2, \cdots, p_x$ while running different LBS apps on his device. Hence, P_l is the total number of user's private locations (as given in Equation 2). PL-Protector relies on the user input to define the set of private places that are distinct for every user mobility profile. Moreover, to set up network fingerprints at p_i, we measure the response rate as the ratio of detection count and the total number of scans for each beacon as follows:

$$R_{n_c,x} = \frac{\sum_{i=1}^{n_c} b_{x,i}}{n_c}, b_{x,i} = \begin{cases} 1 & \text{if beacon } x \text{ found in } i\text{th scan} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $R_{n_c,x}$ is the response rate of beacon x at p_i and, n_c is the total scan count since the private place was entered. The detection count of each beacon is maintained to identify the frequently occurring beacons. Beacons with higher response rate are used to create the network fingerprint for that p_i. $R_{n_c,x}$ will be maintained in the PL-Protector database to update the response rate of every detected beacon during a specified time interval t spent at private place p_i.

3.3.2. *App-Usage Model*

We will apply privacy rules to the app sessions taking place at private places. We define "app session" as the duration of the app usage. In Android, according to the execution status, an app can run in three different states: foreground, background and perceptible. In general, apps get access to the user's location in foreground. When the user exits an app, this is cached and moved to background state for faster execution. Persistent status is informed by notifications. Background state allows prolonged location access; therefore, tracking threats are more harmful here.

In further sections, we will specify how our work handles all these three app running state to mitigate viable location privacy threats.

Table 1: The evaluation metrics for the location privacy threats.

Metric	Description
Pl_{guess}	Number of correctly guessed private locations
Pl_{total}	Number of total collected private locations
Pl_s	Unique value to identify applied privacy settings
Pl_d	Distance between two points with longitude and latitude
P_{high}	Distance is > 111.32km
P_{medium}	Distance is > 11.132km
P_{low}	Distance is > 1.1132km
LoP_{per}	Fraction of achieved privacy level at a private place
LoP_{total}	Fraction of achieved privacy level at all private place

3.4. Privacy Model

Table 1 compiles the hereinafter metrics to be used for evaluating the location privacy threats. We define value of Pl_s as the identifier of applied privacy setting and measure the achieved privacy by analysing the collected dataset of the actual location traces at user's private places. To evaluate location privacy, we use Haversine formula in [24] to quantify tracking and profiling threats as the distance Pl_d (Eq. 4) between two positions with longitude and latitude (ϕ, λ) and the radius r of the Earth:

$$Pl_d = 2r\sin^{-1}(\sqrt{\sin^2(\frac{\phi 1 - \phi 2}{2}) + \cos(\phi 1)\cos(\phi 2)\sin^2(\frac{\lambda 2 - \lambda 1}{2})}) \tag{4}$$

where the haversine function is given by $Hsin(\theta) = sin^2(\frac{\theta}{2})$, $\phi 1$ & $\phi 2$ are the original geo-coordinates, and $\lambda 1$ & $\lambda 2$ are the observed geo-coordinates. Secondly, the privacy rules (see more details in Section 4.1) pre-set by user will, later, be used to measure achieved privacy using the distance scale $\langle P_{high}, P_{medium}, P_{low} \rangle$. Location privacy threats can be quantified as the probability of occurrence of an event exploiting the vulnerabilities. Therefore, we use the metric given in [25] that allows us to measure the ability of apps to find private locations from the collected location traces and thus pose location privacy threats. Hence, LoP_{per} and LoP_{total} are calculated as:

$$LoP_{per} = \frac{Pl_{guess}}{Pl_{total}}, and \tag{5}$$

$$LoP_{total} = \sum LoP_{per} \tag{6}$$

4. PL-Protector on Android

In this section we describe the architecture and implementation for PL-Protector middleware on Android.

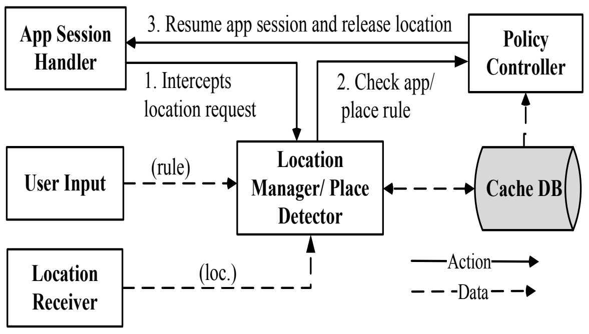


Figure 2: Control Flow

4.1. Architecture

Since app users are more concern about sharing their private locations[7], PL-Protector enforces privacy for such user's sensitive locations. PL-Protector's three main design goal are: 1) the third-party app provider will not be able to infer the device's exact location without getting uses's consent; 2) the user can set distinct privacy preferences for different apps and private places; and 3) the model works independently without the need of modifying the app's code. Figure 2 depicts the block diagram for PL-Protector architecture; its main components are:

User Input is the User Interface (UI), which enables users to set and manage their private places and apply improved personalised permissions when running installed location-based apps. Once received the user inputs, pre-set private locations are sent to the *Location Manager* module, and permissions are sent to the *Policy Controller* module.

App Session Handler is responsible to intercept the event of location access calls and then lead the app's control flow to our middleware. When the flag is positive (i.e., user at private place), it monitors app launch and exit events that need to be intercepted. It first pauses the requesting app's execution, saves its sate, and sends the location intent to the *Location Manager* component for rule checking (step 1 in Figure 2). Once the privacy rules are applied, the *App-Session Handler* will receive the anonymised/transformed location object from *Policy Controller*. It will then resume the requesting app's control flow to maintain every session (step 3 in Figure 2). In Android, the *middleware*'s background service frequently checks (every 10s) the currently running foreground app using `getRunningAppProcesses` on older versions and `UsageStasManager` on Android 6 (or later). To use the `UsageStasManager` API the user of the device needs to grant permission through the Android's Settings application. When an app is no longer running in the foreground, it blocks app's background access to location updates that leaves an app limited to foreground sessions. In Android, for maintaining a set of session operations, PL-Protector executes process isolation and IPC services .

Location Manager & Place Detector are the central components that receive both events (actions) and data from the different components as well as it maintains the *Cache DB* database. The *Location Manager* component receives privacy rules for specific private locations and network fingerprints via *User Input*. Whereas, the *Place Detector* component detects unique identifiers of the surrounding wireless APs and maintains a binary flag to detect private places. When the flag is ON, if the *Location Manager* receives location updates (i.e., Intents) from *App Session Handler*, the location data is retrieved from the *Cache DB* and sent to the *Policy Controller*. In case of an unmatched query on the *cache* and the user do not want to input geo-coordinates manually (via maps provided in UI), the location data is received by location providers from the *Location Receiver*. Later, the *Place Detector* monitors user's mobility pattern and updates the mobility profile of the user (see Mobility Model in Section 3.3.1). Moreover, if the location is user's one of the frequently visited places, the*Place Detector* consults with the user to check if he is comfortable with release of the location. If the user isn't comfortable, the location is sent to the *Policy Controller* to hide the visited place, else the location is released as it is.

Policy Controller it gathers the location object from the *Location Manager* as to apply the corresponding user permissions on the location coordinates, altering it if needed, and transferring the processed location to the *App Session Handler* component. The two privacy policies that the user can set per-app/place basis are the *Standard Policy* and *Per-location Policy* (see Figure 4), as follows:

1. The Standard Policy consists of three location settings as follow:
   (a) The *Behaviour Protection* setting implements the geo-coordinate obfuscation equation defined in [10] to generate transformed/ obfuscated geo-coordinates (l', l'_g) for every app session. The behaviour protection level is defined by a scale (Low, Medium, and High) that determines randomness of the obfuscation equation's parameters $\langle s, \theta, (l, l_g)\rangle$, where s is the scaling factor, θ is the random rotation angle, and (l, l_g) are the original coordinates.
   (b) The *Location Protection* setting implements the geo-coordinate truncation equation defined in [10] and follows a location granularity scale like (Low, Medium, and High) to adjust the location precision level for every app session.
   (c) The *Block/Fixed Location* setting picks high behaviour and location protection level by default and determines a constant value of altered geo-coordinates for every app session.
2. The Per-location policy allows the User to apply standard policy settings for each pre-marked private places that are displayed on the map.

Once processed geo-coordinates (l' l'_g) that comply with pre-set privacy rule are generated, we measure achieved level of privacy on per-session basis using values of both Pl_d and Pl_s (as defined in Section 3.4). Listing 5 contains pseudocode for calculating haversine distance (Pl_d) between two positions with longitude and latitude (ϕ, λ) and the radius r of the Earth.

```
initialise static final int EARTH_RADIUS = 6371;
request double haversin(double val)
return  Math.pow(Math.sin(val / 2), 2);
request HaverDistance(double originalLat, double originalLong,
                         double observedLat, double observedLong)
    double dLat  = Math.toRadians((observedLat - originalLat));
    double dLong = Math.toRadians((observedLong - originalLong));
    originalLat  = Math.toRadians(originalLat);
    observedLat  = Math.toRadians(observedLat);
    double a     = haversin(dLat) + Math.cos(originalLat)
                 * Math.cos(observedLat) * haversin(dLong);
    double c     = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
return EARTH_RADIUS * c;
```

Listing 5 Pseudocode for calculating Haversine Distance (Pl_d) between two positions with longitude and latitude (ϕ = *original*, λ = *observed*)

Cache DB & Location Receiver *Cache DB* is the established on-device cached database, and it is routinely queried by the *Location Manager* module, which can add, update and delete the cached location data. The locations in *Cache DB* are those which are to be protected, and they can also represent regions of space. In Android, *Cache DB* stores the network fingerprints in an SQLite table that contains the mapping of each observed WiFi APs signature to their representative geo-coordinates. Each SQLite table entry is recorded along with a network fingerprint and geo-location that are acquired either from UI and/or the *Location Receiver*. When the location update request is sent to the *Location Receiver* component, it receives the location object, which includes the user device's geo-coordinates (as in Figure 3), from location providers and sends it over to the *Location Manager* for further processing.

4.2. Middleware Implementation

PL-Protector orchestrates a mobile platform based privacy protection service on Android to modify the location resource handling process. PL-Protector's communication operations only require process isolation and IPC services; hence, minimising the requirements placed on hardware or OS modifications. In Android, there are two methods to access user's location: 1) Location Manager Service (Old), and 2) Fused Location Manager Service (New) that is a part of Google Play Services. However, both methods require the app to request a callback function to get regular updates by registering a location listener. The app receives a new location object when a new location is available, the callback function is invoked (Figure 3 (left)). Modifying these two Google services is complicated, but we make PL-Protector communicate with the location requesting apps by intercepting the location object before it reaches requesting apps (Figure 3-(right)). One of the main

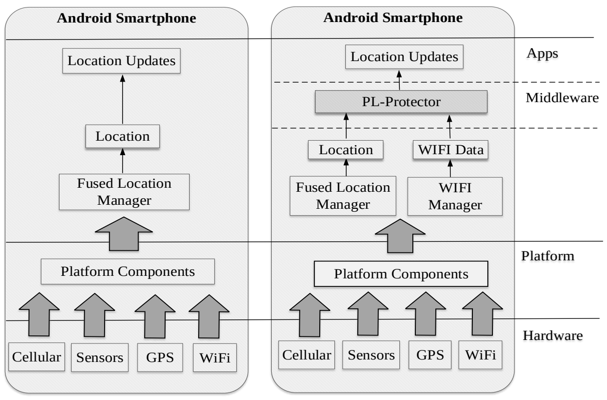
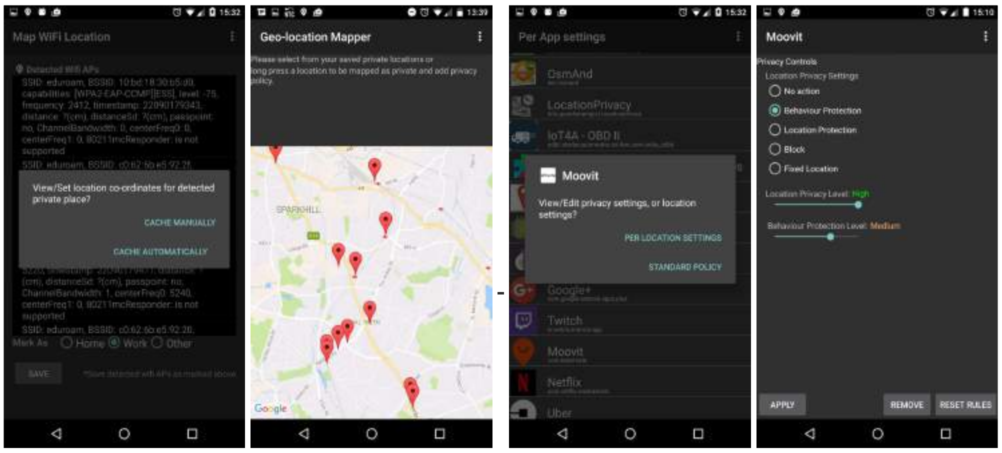


Figure 3: Computation Mechanism.

task is to add a system service, where the class belongs to the location APIs; thus, the new service is placed in the `android.location package`, which detects private locations via APs and can also be used by other components when calling context. In Android, a context allows an app to interact with the OS resources. Similar to [26], we add a static context field to the location class, which will be populated when the app is invoked; this enables PL-Protector to know which app is currently requesting the location object, and also communicate with the OS. Besides, `Fused Location Manager` combines sensors, GPS, Wi-Fi, and cellular data into a single API for location-based applications [27], hence separating data from `GPS_PROVIDER` and `NETWORK_PROVIDER` is no longer straight forward. PL-Protector addresses this issue by preventing app's location request to reach the `Fused Location Manager` that collects and sends the network session data to the location provider. Instead, the requested location is retrieved from the on-device cache, and then, it is sent to the requesting app (with privacy rules applied).

User Privacy Preferences. Complex policies, fine-grained configurations and explicit technical details in the UI discourage users from fully exploiting the provided functionalities. To this end, we designed PL-Protector's UI (see Figure 4) in an intuitive and straightforward manner that maintains balance between the usability and expressiveness of users privacy preferences.

Bootstrapping. When PL-Protector first boots and before turning 'ON' location sharing settings, the user will have to perform the initial setup. This will allow WiFi APs scanning, input geo-coordinates and set privacy choices using *User Interfaces* (UI) (Figure 4 - 1st & 2nd). PL-Protector's UI incorporates a map to get the corresponding geo-coordinates so achieving an effective privacy without affecting the location accuracy. At the same time, this prevents non-authorised sharing of device's exact location and network session data.



Map WiFi Location

Detected WiFi APs

SSID: eduroam, BSSID: 10:bd:18:30:b5:d0, capabilities: [WPA2-EAP-CCMP][ESS], level: -75, frequency: 2412, timestamp: 22090179343, distance: ?(cm), distanceSd: ?(cm), passpoint: no, ChannelBandwidth: 0, centerFreq0: 0, centerFreq1: 0, 80211mcResponder: is not supported

SSID: eduroam, BSSID: c0:62:6b:e5:92:2f,

View/Set location co-ordinates for detected private place?

CACHE MANUALLY

CACHE AUTOMATICALLY

5240, timestamp: [illegible], distance: ?(cm), distanceSd: ?(cm), passpoint: no, ChannelBandwidth: 1, centerFreq0: 5240, centerFreq1: 0, 80211mcResponder: is not supported

SSID: eduroam, BSSID: c0:62:6b:e5:92:20,

Mark As ( ) Home [✓ Work] ( ) Other

SAVE

Geo-location Mapper

Please select from your saved private locations or long press a location to be mapped as private and add privacy policy.

Per App settings

Moovit

View/Edit privacy settings, or location settings?

PER LOCATION SETTINGS

STANDARD POLICY

Moovit

Privacy Controls

Location Privacy Settings

( ) No action [✓ Behaviour Protection] ( ) Location Protection ( ) Block ( ) Fixed Location

Location Privacy Level: High

Behaviour Protection Level: Medium

APPLY REMOVE RESET RULES

Figure 4: User Interface to manage: WiFi/location input settings (1st & 2nd screens), and per-app/location privacy rule settings (3rd & 4th screens)

The UI (Figure 4 - 3rd & 4th) enables users to set and manage their private locations and apps distinctly.

5. Evaluation

We evaluate PL-Protector considering three different research questions: RQ1. What is the overall performance overhead of PL-Protector while interacting with real-world apps over time? RQ2. How well the on-device cache perform in practice? Can it find accurate location data corresponding to the cached network fingerprints and apply permission rule in real-time? RQ3. How well PL-Protector can perform with respect to user privacy/data leakage using actual mobility traces collected from real-world apps?

5.1. *Experimental setup*

We deployed PL-Protector, a middleware on a Nexus 6 with Android 6.0 (API 23) that have 802.11a/b/g/n radio feature so it can operate in both 2.4GHz and 5GHz bands at 34 different private places. We considered two different experimental set-up for PL-Protector's performance evaluation:

1. *Location-based apps performance.* Experiment 1 studies the efficiency of PL-Protector in terms of QoS and usability on operations relevant to the location-based apps and privacy leakage tests. For this purpose, we ported real apps of five different LBS queries categories: 1) Social Networking (e.g., Facebook), 2) Instant Messaging/Chatting (e.g., Whatsapp), 3) Tracking (e.g., Fitness), 4) Utilities (e.g., Weather, Alarm, etc.) and 5) Finder (PoI Finder/Geo-search). Based on app operations we assume that both types 1) and 3) require continuous access to location data; whereas, types 2), 4) and 5) involve sporadic access.

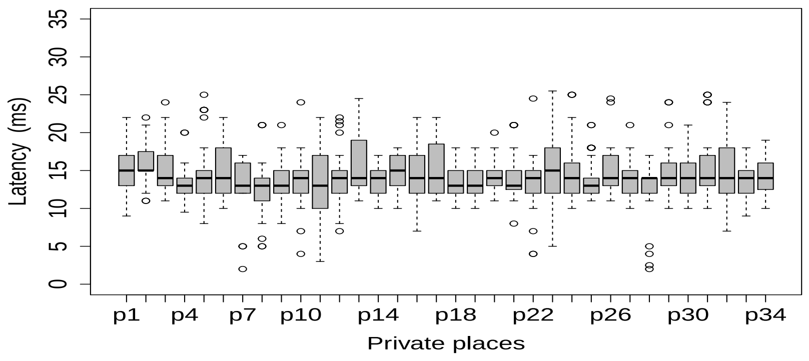


Figure 5: PL-Protector's overall computation latency caused at 34 distinct private places.

2. *WiFi Fingerprinting and caching performance.* The statistical study [10] on WiFi AP data availability and consistency demonstrates smartphones regularly detect similar beacons at frequently visited place, for place detection at least one beacon should match with the stored WiFi fingerprints. Whereas, in Experiment 2, we investigate the performance of WiFi fingerprinting method, which is used in our middleware as private place detection source, at runtime. Using 2 or more different LBS apps at 34 distinct private places, we sent a sequence of location update requests to our middleware in order to measure its dynamic response rate and cache accuracy over time.

Data collection and analysis. We have collected empirical data from a number of sessions running at different time intervals over a period from 3 to 9 months. We then created two datasets: In a first dataset, we include the session data of ported apps that runs over the conventional Android environment without interacting with PL-Protector. Henceforth, we call the conventional Android environment as *Baseline*. The second dataset consists of the same apps but running in the presence of PL-Protector. Based on the two datasets, we can draw the following observations and conclusions.

5.2. Results for RQ1: Impact on the quality of service

In this section, we assess developer effort and complexity of implementing PL-Protector directly into `Android` platform core, highlighting its efficiency and feasibility.

Impact on the underlying OS: computational effort and memory consumption. Android provides a flexible model of process isolation, inter-component communication (ICC) and IPC services. Since Android development architecture is comprised of multiple components, PL-Protector used ICC calls known as *Intents* and IPC services to communicate messages with benchmark apps' components. We assess PL-Protector's computational overhead posed on the base OS (i.e., `Android`) while responding to apps' location calls. We used

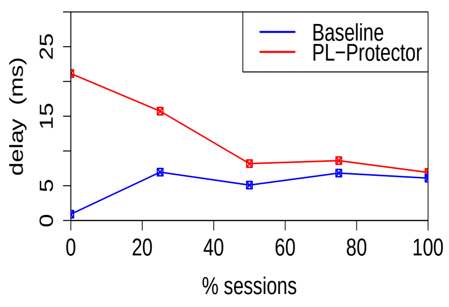


Figure 6: Total communication overhead for both environment Baseline and PL-Protector.

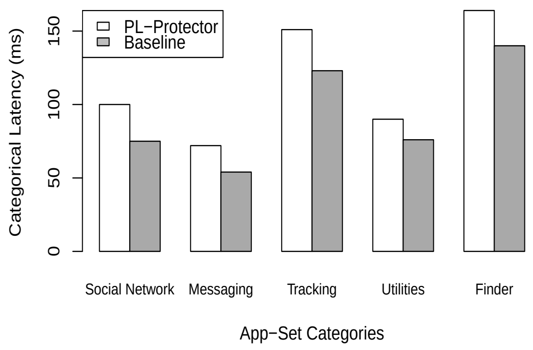


Figure 7: Categorical latency (ms) on different app-sets for baseline versus with PL-Protector.

Android's `MemoryInfo` API and `Android Monitor` tool to log observed computational and memory overhead caused by PL-Protector's presence. Equation 7 is used to calculate total memory consumption required for the computational lifecycle of PL-Protector, where C_{comp} represents total memory consumption by the LLPMs, C_i is a location computation operation, M_i is a rule mapping operation, and l_n is the number of received location update calls from apps.

$$C_{comp} = \sum_{i=1}^{l_n} C_i + M_i \tag{7}$$

We sent 1 to 10 location calls to the middlware using benchmarked apps and recorded the memory consumption. Our results show that PL-Protector's core functionality requires 7.3 MB of device memory while interactions with each app requires 2.8 MB of memory on an average. To justify this overhead, we argue that *NEXUS 6P* comes with built-in 3GB memory RAM (< RAM size in later versions) and the official Google Chrome browser app on page load with an empty page consumes around 97 MB of the memory. Thus, PL-Protector's consumption of 21.3 MB of memory to communicate with 5 apps (at once) falls within acceptable limits of the base OS.
For improving the response time, while the user is stationary the location result is cached in the device memory and shared with other apps that has similar permissions. PL-Protector requires 125 bytes for caching a single location result as an `object` in cache memory. The current built-in caching limit for Android is 1 MB [28]. Therefore, the maximum number of location query result saved as cached messages must be < 8388, which is sufficient for PL-Protector's functionality since it only considers user's private locations.

Table 2: Observed difference in monthly dynamic DB storage

Storage Overhead	**1 Month**	**3 Month**	**Observed Change in DB size**
Network Fingerprint	54 KB	108 KB	44 KB Increase in size
Permissions	21 KB	28 KB	7 KB increase in size
Total DB Storage	75 KB	136 KB	51 KB Total increase in size

Impact on location-based apps functionality. Crucial for its functionality, we measure latency as the time PL-Protector takes to interact with the app and perform an entire computational cycle, i.e., to compute the location on-device and to apply the privacy rules. To measure the overall functionality overhead for each app, we varied the range of location calls – over 10 trials of 2 to 5 types of LBS queries – in collected databases for both baseline and PL-Protector. For instance, PL-Protector took 187ms to successfully reply to the location based query requested by the app, compared to 179ms on baseline, i.e., 8ms increase. On average, PL-Protector presents a latency lower than 22 milliseconds upon all the location-access calls executing PL-Protector's privacy controls at runtime for all the 34 private places (as shown in Figure 5). The reason for increased latency is due to PL-Protector's load time, and cross-process/IPC service transfers of location updates. However, this latency is smaller than 100 ms and, thus, small enough to not cause user-noticeable delays while utilising apps on the device. Furthermore, Figure 6 and Figure 7 show the communication overhead during different sessions and the overall app functionality overhead for the 5 app categories and compares both baseline and PL-Protector execution environments. In per-location access sessions, we found < 19ms delays when continuous location updates, and less than 8 ms delay for sporadic location updates (see Figure 7). Figure 6 shows PL-Protector delay decreases after a number of repeated sessions and it remains well within the bounds throughout the sessions. Thus, PL-Protector is suitable to run all the existing apps of aforementioned five LBS categories since their core functionality already accepts delays in this range.

5.3. Results for RQ2: On-device cache performance

Cache accuracy. To analyse the accuracy of the on-device cache method at runtime, we measured occurrence of cache hits and misses that includes three possible outcomes: (a) *The location is cached and up-to-date*, (b) *The location is cached but is out-of-date*, and (c) *The location is not cached*. Figure 8 indicates the performance of on-device cache improves over time. Initially, for upto 8 hrs duration, the result accuracy of on-device cache range from 40% to 60% that gradually increases to 90%. This indicates PL-Protector's on-device cache update frequency is within practical limits, and it provides accurate location data at runtime to requesting apps requiring both sporadic and continuous location-updates.

Cache storage Overhead. Location privacy solutions (e.g., [13, 12]) that apply caching techniques on location-based queries that are generated/received from running applications

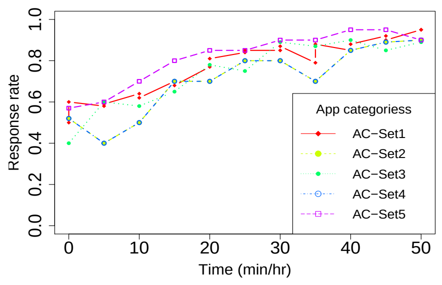


Figure 8: On-device cache accurary of the inter-request interval over time (hr)

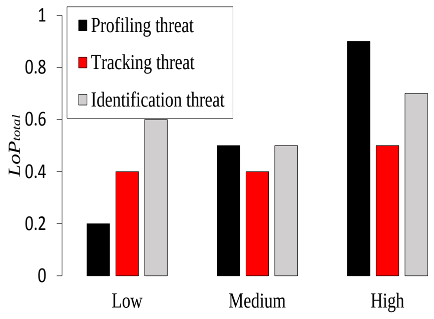


Figure 9: LoP_{total} achieved by PL-Protector to mitigate 3 privacy threats

and service providers include several attributes, and their data types require vast amount of storage space. PL-Protector does not cache location-based queries, instead it stores the WiFi AP data and geo-coordinates of users' private locations and the user's pre-set privacy rules are applied to the mapped geo-coordinates at runtime. As a result, comparatively, PL-Protector's on-device cache database does not demand massive storage requirement. Considering the 802.1 standards and datatypes sizes, we created PL-Protector's embedded database structure. The SQLite table attributes of the network fingerprint consists of a tuple of $\langle no.of beacons, beaconfield, counter\rangle$, and the permission table is a tuple of $\langle location, placeid, accuracy\ counter, no.of private places\rangle$. While PL-Protector is installed and running on the NEXUS 6P, we dynamically collected regular versions of its database for a period of 1 to 3 months by implementing built-in datasets collection mechanism. Table 2 presents the observed monthly increase in the database size. The results evident that PL-Protector does not have a massive on-device cache storage requirements. PL-Protector's DB size of 136 KB include dynamically collected network fingerprints for 34 private places, privacy rules for 5 location-based apps, and other database attributes needed during the development, implementation and data collection stages. Since the internal storage limit for Nexus 6P range from 32 GB, 64 GB to 128 GB [29], we argue that PL-Protector's DB size of 136 KB for 3 months is within acceptable internal storage limit and that the current mobile device internal storage capacity is sufficient for PL-Protector's overall functionality.

6. Security Analysis

In this section, we analyse user privacy/data leakage that is likely to affect PL-Protector by an adversary's access to user location in and out the device and can launch an attack.

6.1. Security

We discuss data security risks that each of the five apps pose when running on existing mobile platforms, and find that PL-Protector mitigates those risks successfully under privacy leakage tests.

Continuous access. Apps (e.g., Social Networking and Tracking) have higher potential to leak location information to attackers, e.g., unauthorized 3rd party service providers and content distributors, since users access such apps for longer duration and in frequent manner. The social networking app has constant Internet access for ads and sending troubleshoot/crash reporting. PL-Protector separates private locations from public/less-sensitive locations and enforces privacy rules.

Sporadic access. Apps (e.g., Instant Messaging, Utilities or Finder) can leak location data along with other sensitive user information as we note that these apps require Internet access for core functionality. Therefore, under current OS controls it is very easy for this app to leak location data. PL-Protector isolates the private location flow to be restricted within device, eliminating any possibility of cross-flows between apps and 3rd party service providers over Internet.

Covert access. A potential malicious apps leverages vulnerabilities in other already installed benign apps to perform actions that are beyond its individual privileges such as sending device location through text messages. Current mobile OS access control model is per app and it cannot detect such location attacks as it cannot check security posture of the entire system. PL-Protector, enforces location privacy policies before releasing the data that mitigates most, if not all, of such covert access exploits.

6.2. Results for RQ3: Privacy Threats Mitigation

Here we present an informal analysis concerning correctness of our proposal in terms of three fundamental threats: identification, profiling, and tracking. We use the aforementioned privacy metrics (Section 3.4) to evaluate achieved privacy. To compute LoP_{per} and (overall) LoP_{total}, We identify applied privacy settings using (labeled) value of Pl_s in the collected location traces of apps' sessions and compare them with the privacy rule. We measured LoP_{total} against observed value of Pl_d for every location-access session, higher the value of LoP more protected is the private location. We can draw the following conclusions from observing Figure 9.

User Tracking Threat. This requires protection of private location anonymity and sensitive behaviour pattern from the adversary. We observed that the continuous location updates can pose high risk to locate the user in real time. PL-Protector blocks app's background access to location updates that leaves tracking limited to foreground sessions, which are mainly sporadic (once/twice a day) and initiated from same private place for about 96%. By preventing consistent user's mobility patterns, PL-Protector mitigated tracking threats by preventing sensitive behaviour pattern and location prediction in 40%, 40% and 50% of released location-access sessions at P_{low}, P_{medium}, and P_{high} settings, respectively.

Identification Threat. Even sporadic location access can allow an adversary to isolate the user's private locations, such as home and work. These places can be used as quasi-identifiers to reveal the user's identity from anonymous location traces. We report that locations released in the sessions at P_l (Equation 2) follow privacy rules and prevent location identify in 60% (P_{low}), 50% (P_{medium}) and 78% (P_{high}) of released location traces.

Profiling Threat. This requires protection of sensitive behaviour profiling such as health clinics, religious places, shopping habits, etc. Compared to the first dataset, the user's mobility traces in the second dataset did not include places that would reveal his identity; but, places that the adversary could use to profile him were observed. Figure 9 reports that in the released apps sessions profiling threat increases with more relaxed privacy rule (P_{low}= 20%, P_{medium} = 50% and P_{high} = 90%) since this will enable releasing more of the user's private locations $\subseteq (P_l)$.

OS's Middleware Layer Threats. Android security model considers all apps as potentially malicious and, therefore, runs each app in its own process, known as process isolation, and access its own files by default. This security practice protects apps with sensitive information from malware. Despite this process isolation mechanism, apps can optionally communicate via inter-message passing, which can become an attack vector. Further, location data can be stolen by eavesdroppers and permissions can be accidentally transferred between applications. If a developer sends location data to the wrong recipient (intentionally or intentionally), then it might leak sensitive user information. PL-Protector's enhanced permission mechanism enables robust and efficient source (OS) to sink (apps) flow control to user's sensitive location data before it is sent to app providers. Hence, this enhanced mechanism protects user's private locations from both over-privileged, malicious 3rd party apps, and inter app communication-based attacks.

6.3. Field study: Users Perspectives

Results of field study indicate that 82.18% of smartphone users rely on location-based app functionality when they are anywhere outside or unknown places. 77.03 % of smartphone users think not all apps need continuous access to their locations; whereas, 10.81% were not sure. We gave users to chose their preferred accuracy level while sharing their private locations with 5 different categories of location-based apps: 1. Social networking (e.g., Facebook), 2. Instant messaging (e.g., Whatsapp), 3. Sports/Fitness Tracking (e.g., Fitness/Workout), 4. Utilities (e.g., Weather, Alarm) and 5. Finder (e.g., PoI-search, Geo-search). Out of the 5 given location-based app categories, users where more relaxed to share their private location with (3) Sports and (5) Finder both app categories. However, (1) Social, (2) Messaging and (3) Utilities app categories scored the lowest scale that indicates users are more reluctant to share their private locations with these app categories.
To summarise, it is evident from the aforementioned results and empirical findings of the performance and security analysis that PL-Protector provides users with location privacy at a tolerable loss of app functionality and acceptable overheads on the underlying OS.

7. Related Work

We have categorised existing approaches to preservation of the location privacy in mobile devices as:

Policy-based approaches. Service providers' privacy policies are defined in compliance with the existing privacy regulations. Standards for location-based products and services exist in both the 3GPP [30] and IETF [5] arenas. Although such privacy policies aims to be flexible and support the personalisation of privacy preferences, they only provide a deterrent against privacy violations. Therefore, if a third party decides to violate those norms, in spite of the risk of penalties, the user's privacy cannot be protected. Moreover, manual enforcement is expensive and thus policies are often ignored by users, particularly when users are unaware of privacy regulations. And, automating privacy policy enforcement has been the main objective of early research on privacy protection. Our approach automates recommended privacy policy that enforces private location protection.

Theoretical approaches. Apps share location information with the provider in the form of LBS queries. The transmission of such queries to the location server may allow attackers to gain access to user location data. Privacy Enhancing Techniques (PETs) like k-anonymity, dummy locations, region cloaking, location perturbation and obfuscation, pseudonyms, and differential privacy have been applied to different architectures for location query formation and privacy preservation from LBS providers [31, 23, 32]. Most of these techniques rely on theoretical assumptions - like trusted infrastructure to provide the privacy protection, requiring a group of similar app users to be at the same time and same place. The main issue with PETs and cryptographic schemes is that it relies entirely on the data collection servers to comply with location privacy. Besides, mobile devices not only send vast amounts of location data to app providers but also to location providers creating different location privacy shortcomings [7, 3]. In this regard, limited work has been published on privacy preservation from the location provider's perspective [33, 34]. Damiani [33] proposes a theoretical approach for privacy-aware geolocation-based web services to encourage further research to minimise the amount of location data being shared with the location provider. This is mainly due to that the location provider is considered as the only source to get the user location when developing any location-based app.

Practical approaches. A few studies have proposed static and dynamic methods to detect privacy leaks in mobile platforms. The former method statistically analyses apps by creating permission mapping, generating call graphs and data flow analysis to report privacy leaks for further auditing, e.g, AndroidLeaks [35] and PiOS [36] for Android and Apple iOS, respectively. The application of dynamic methods involves modification of the existing mobile platform. For example, TaintDroid [18] adds taint tracking information to sensitive sources calls from apps, and it tracks location data flow as it generated through applications during execution. MockDroid [37] relies on instrumenting `Android's` manifest permission mechanism to mock sensitive data from OS resource, including location data, which can affect apps' usability and functionality. LP-Caché not only monitors the location sources but also modifies, if required, the generated location data based on defined user permissions. In another attempt[26], *indistinguishability* technique is applied as location privacy preservation mechanism into the advertising and analytics libraries as well as on installed apps; however, it does not give control on the amount of WiFi and location data that is being shared with the location provider. Moreover, *indistinguishability* technique increases computational overhead on smartphones.

Cache-based approaches. Several authors have used caching scheme along with PETs to build to a database consisting of different contents/datatypes used within location based queries to be re-used in future LBS queries. MobiCaché [12] applies k-anonymity for caching location based queries. Similarly, Niu et al. [14] attempt to improve k-anonymity based caching by adding dummy locations. Both proposals require a trusted infrastructure to maintain privacy. Caché [13] maintains a local cache within the device to reuse the data types available from applications in future location based queries; however, storing entire LBS query data increases the cache storage requirements. Besides, Caché also requires app developer to modify the way app access location data. By contrast, LP-Caché caches the network fingerprints and geo-coordinates, which reduces the storage overhead drastically; it considers installed apps as black box, and therefore, does not require app developer to modify the code, it works as a middleware between the app and the mobile platform. All these cache-based systems either intent to generalise or obfuscate the LBS query or minimise the number of queries sent to the app providers, but they do not provide privacy from WiFi content distributors. In PL-Protector, we minimise the process of wireless AP data collection by the WiFi content distributors or location providers, and we control information disclosure within the generated LBS query (e.g., PoIs and nearest neighbor) since it will be sent to the third-party app provider.

Network layer approaches. Besides location queries, device's IP address can also reveal user's private locations. To this regard, anonymous communication protocols, e.g., Anonymizer [38] and TOR [39], deal with anonymous service usage at the network layer while communicating over Internet (i.e., the server cannot infer user's location via received device's IP address along with the location query), and they are most prominent and commonly used network layer solutions.

8. Conclusion

To summarise, we present the design and implementation of PL-Protector, a location privacy-enhancing middleware, which is a prototype system developed to validate the theoretical model (LP-Caché). We mainly focused on supporting our design goals, justifying the design decisions and elaborated on PL-Protector's functionality. We have successfully implemented PL-Protector on Android Platform (`Android version 6`) to enforce the privacy rules over both the information and control flows occurred between sources and sinks. Through the implementation of the middleware, we proved the deployment feasibility of a new series of privacy controls on a mobile platform to prevent private location disclosure during the formation of LBS queries. This also minimises the interaction and data collection from wireless access points, content distributors and location providers. Later, we assessed developer efforts and complexity of implementing

PL-Protector directly into Android platform core. We comprehensively described PL-Protector's evaluation in terms of performance and security. Followed by presentation of results and finding in terms of usability and efficiency of PL-Protector when interacting with real apps in real-time. We also present security analysis based on the threat model to test its compliance with the three privacy settings and achieved privacy guarantees. In future work, we are also plan to conduct a usability study that will allow us to enhance PL-Protector's privacy and usability rates.

References

[1] T. Pontes, M. Vasconcelos, J. Almeida, P. Kumaraguru, V. Almeida, We know where you live: Privacy characterization of foursquare behavior, in: Procs. of ACM Conf. on Ubiquitous Computing, ACM, 2012, pp. 898–905.

[2] I. Muslukhov, Y. Boshmaf, C. Kuo, J. Lester, K. Beznosov, Understanding users' requirements for data protection in smartphones, in: IEEE Int. Conf. on Secure Data Management on Smartphones and Mobiles, IEEE, 2012, pp. 228–235.

[3] I. Shklovski, S. D. Mainwaring, H. H. Skúladóttir, Others, Leakiness and Creepiness in App Space: Perceptions of Privacy and Mobile App Use, in: Procs. of ACM Conf. on Human factors in computing systems, ACM, 2014, pp. 2347–2356.

[4] European Commission, Protection of personal data, http://ec.europa.eu/justice/data-protection/ (2016).

[5] IETF, Geographic Location Privacy, http://datatracker.ietf.org/wg/geopriv/charter/ (March 2016).

[6] K. Michael, R. Clarke, Location and tracking of mobile devices: Überveillance stalks the streets, Computer Law & Security Review 29 (3) (2013) 216–228.

[7] H. Almuhimedi, F. Schaub, N. Sadeh, Others, Your Location has been Shared 5,398 Times! A Field Study on Mobile App Privacy Nudging, in: Procs. of ACM Conf. on Human Factors in Computing Systems, ACM, 2015, pp. 787–796.

[8] A. P. Felt, S. Egelman, D. Wagner, I've got 99 problems, but vibration ain't one: a survey of smartphone users' concerns, in: Procs. of ACM on Security and Privacy in Smartphones and Mobile Devices, ACM, 2012, pp. 33–44.

[9] L. F. Cranor, N. Sadeh, A shortage of privacy engineers, IEEE Security & Privacy (2) (2013) 77–79.

[10] A. Patel, E. Palomar, LP-Caché: Privacy-aware cache model for location-based apps, in: Procs. of the 13th Int. Conf. on Security and Cryptography (SECRYPT), 2016, pp. 183–194.

[11] A. Patel, E. Palomar, A middleware enforcing location privacy in mobile platform, in: International Conference on Trust and Privacy in Digital Business, Springer, 2017, pp. 32–45.

[12] X. Zhu, H. Chi, B. Niu, W. Zhang, Z. Li, H. Li, Mobicache: When k-anonymity meets cache, in: GLOBECOM, IEEE, 2013, pp. 820–825.

[13] S. Amini, J. Lindqvist, J. Hong, J. Lin, E. Toch, N. Sadeh, Caché: caching location-enhanced content to improve user privacy, in: Procs. of ACM Int. Conf. on Mobile Systems, Applications, and Services, ACM, 2011, pp. 197–210.

[14] B. Niu, Q. Li, X. Zhu, G. Cao, H. Li, Enhancing Privacy through Caching in Location-Based Services, in: Proc. of IEEE INFOCOM, 2015.

[15] Android Developer Reference, http://developer.android.com/reference/ (March 2016).

[16] J. Xie, B. P. Knijnenburg, H. Jin, Location sharing privacy preference: analysis and personalized recommendation, in: Procs. of the 19th Int. Conf. on Intelligent User Interfaces, ACM, 2014, pp. 189–198.

[17] R. Montoliu, J. Blom, D. Gatica-Perez, Discovering places of interest in everyday life from smartphone data, Multimedia tools and applications 62 (1) (2013) 179–207.

[18] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, Others, TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones, TOCS 32 (2) (2014) 5.

[19] P. Faruki, A. Bharmal, V. Laxmi, V. Ganmoor, M. S. Gaur, M. Conti, M. Rajarajan, Android security: a survey of issues, malware penetration, and defenses, IEEE communications surveys & tutorials 17 (2) (2015) 998–1022.

[20] K. Fawaz, H. Feng, K. G. Shin, Anatomization and protection of mobile apps' location privacy threats, in: 24th USENIX Security Symposium (USENIX Security 15), USENIX Association, 2015, pp. 753–768.

[21] S. Bugiel, S. Heuser, A.-R. Sadeghi, Flexible and fine-grained mandatory access control on android for diverse security and privacy policies., in: Usenix security, 2013, pp. 131–146.

[22] M. Backes, S. Bugiel, E. Derr, P. McDaniel, D. Octeau, S. Weisgerber, On demystifying the android application framework: Re-visiting android permission specification analysis, in: USENIX Security, 2016.

[23] M. Wernke, P. Skvortsov, Others, A classification of location privacy attacks and approaches, Personal and Ubiquitous Computing 18 (1) (2014) 163–175.

[24] C. C. Robusto, The cosine-haversine formula, The American Mathematical Monthly 64 (1) (1957) 38–40.

[25] J. Freudiger, R. Shokri, J.-P. Hubaux, Evaluating the privacy risk of location-based services., in: Financial Cryptography, Vol. 7035, Springer, 2011, pp. 31–46.

[26] K. Fawaz, K. G. Shin, Location Privacy Protection for Smartphone Users, in: Procs. of ACM Conf. on Computer and Comm. Securty, ACM, 2014, pp. 239–250.

[27] E. Hellman, Android programming: Pushing the limits, John Wiley & Sons, 2013.

[28] Android, http://developer.android.com/guide/ topics/data/data-storage.html (March 2016).

[29] NEXUS 6P, https://www.google.com/nexus/6p/ (August 2017).

[30] 3GPP, The 3rd Generation Partnership Project, http://www.3gpp.org/about-3gpp (April 2017).

[31] A. Patel, E. Palomar, Privacy Preservation in Location-Based Mobile Applications: Research Directions, in: Procs. of IEEE Int. Conf. on Availability, Reliability and Security (ARES), IEEE, 2014, pp. 227–233.

[32] A. Khoshgozaran, C. Shahabi, H. Shirani-Mehr, Location privacy: going beyond K-anonymity, cloaking and anonymizers, Knowledge and Information Systems 26 (3) (2011) 435–465.

[33] M. L. Damiani, Third party geolocation services in LBS: Privacy requirements and research issues, Trans. on Data Privacy 4 (2) (2011) 55–72.

[34] N. Doty, E. Wilde, Geolocation privacy and application platforms, in: Procs. of ACM SIGSPATIAL Int. Workshop on Security and Privacy in GIS and LBS, ACM, 2010, pp. 65–69.

[35] C. Gibler, J. Crussell, J. Erickson, H. Chen, AndroidLeaks: Automatically detecting potential privacy leaks in Android applications on a large scale, in: TRUST 2012, Springer, 2012, pp. 291–307.

[36] M. Egele, C. Kruegel, E. Kirda, G. Vigna, PiOS: Detecting Privacy Leaks in iOS Applications., in: NDSS, 2011.

[37] A. R. Beresford, A. Rice, N. Skehin, R. Sohan, Mockdroid: trading privacy for application functionality on smartphones, in: Procs. of ACM Workshop on Mobile Computing Systems and Applications, ACM, 2011, pp. 49–54.

[38] Anonymizer, http://www.anonymizer.com/ (March 2016).

[39] TOR, http://www.torproject.org/ (March 2016).