# A Middleware Enforcing Location Privacy in Mobile Platform

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.** 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 milliseconds) 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 [24] and Google Weave/Android of Things [13], 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) [22]. 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 [18, 25].

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, 25]. 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 [7].

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 [21]. 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

[1] Throughout this paper, we use the terms Smartphone and Mobile interchangeably
[2] Data protection directives and acts [9, 16] 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[17].

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. Section 4 fully elaborates on the design decisions, architecture and implementation of the middleware. We evaluate PL-Protector's performance overheads in Section 5. Finally, Section 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 [21], 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

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

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 [27]. Moreover, existing OS's location access controls by system services respond inadequately to major privacy threats [1, 11].

### 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 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 [8, 10]. 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 [20, 26] 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* [28], [19] and [4], .

[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.

*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 [14] 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 [12] 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. Additionally, MockDroid [5] 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. Our middleware not only monitors the location sources but also modifies, if required, the generated location data based on defined user permissions.

LP-Caché introduced in [21] 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[6]. 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 [26]. 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 $i^{th}$ private place identification, which is derived from a series

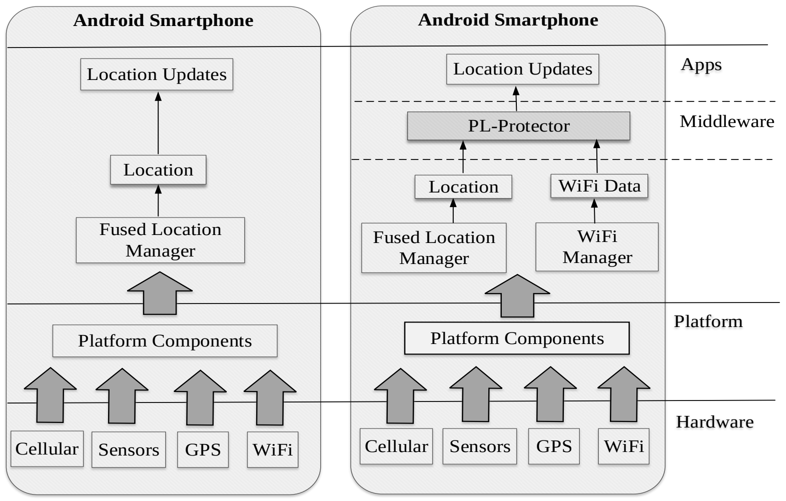


**Fig. 1.** Mechanism to access user's location in Android (left), and PL-Protector's deployment in Android (right).

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 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 $U_{p_i} \subseteq p_1, p_2, \cdots, p_j$, 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 [23] to quantify tracking and profiling threats as the distance $Pl_d$ (Eq. 4) between two positions with longitude and latitude $(\phi, \lambda)$ and the radius $r$ of the Earth:

$$Pl_d = 2r \sin^{-1}(\sqrt{\sin^2(\frac{\phi 1 - \phi 2)}{2} + \cos(\phi 1)\cos(\phi 2)\sin^2(\frac{\lambda 2 - \lambda 1}{2}))} \quad (4)$$

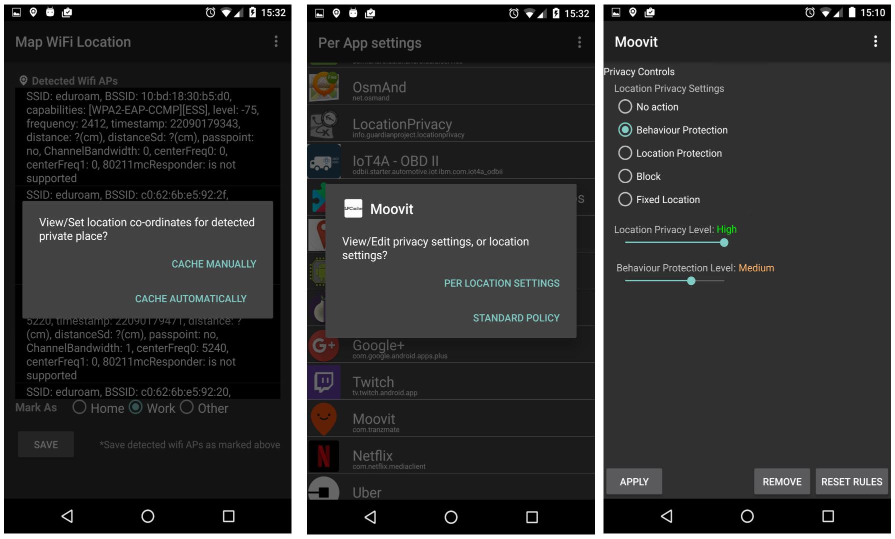
where the haversine function is given by $Hsin(\theta) = sin^2(\frac{\theta}{2})$, $\phi 1$ & $\phi 2$ are the original geo-coordinates, and $\lambda 1$ & $\lambda 2$ are the observed geo-coordinates. Secondly, the privacy rules (see more details in Section 4.1) pre-set by user will, later, be used to measure achieved privacy using the distance scale $\langle P_{high}, P_{medium}, P_{low} \rangle$.

## 4 PL-Protector on Android

In this section we describe the architecture and implementation decisions for PL-Protector middleware on Android.

**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 $>$ 111.32km |
| $P_{medium}$ | Distance is $>$ 11.132km |
| $P_{low}$ | Distance is $>$ 1.1132km |



15:32

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

15:32

Per App settings

OsmAnd net.osmand

LocationPrivacy info.guardianproject.locationprivacy

IoT4A - OBD II odbii.starter.automotive.iot.ibm.com.iot4a_odbii

Moovit

View/Edit privacy settings, or location settings?

PER LOCATION SETTINGS

STANDARD POLICY

Google+ com.google.android.apps.plus

Twitch tv.twitch.android.app

Moovit com.tranzmate

Netflix com.netflix.mediaclient

Uber

15:10

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

**Fig. 2.** User Interface: (left) WiFi settings screen, and (middle and right) screens to manage per-app/location rule settings

## 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 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 [21] 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, $\theta$ 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 [21] 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 3.3).

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.

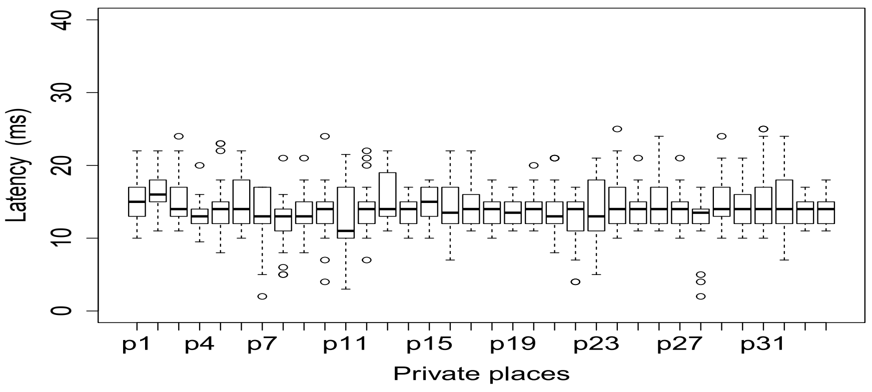


**Fig. 3.** PL-Protector's overall computation latency caused at 34 distinct private places.

## 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 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 [12], 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,

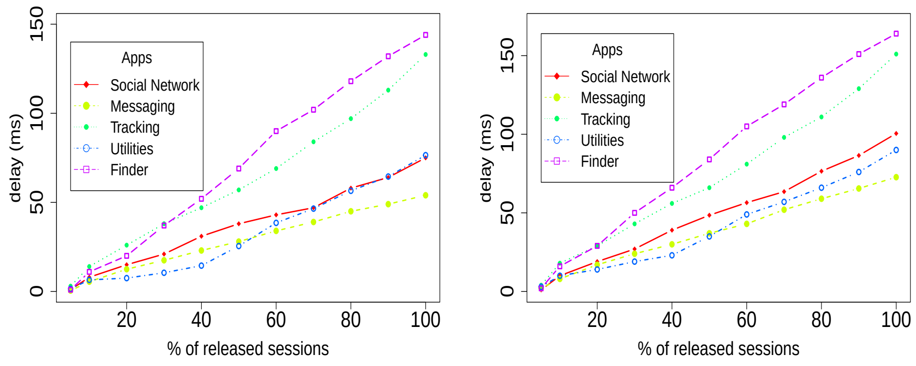


**Fig. 4.** Apps communication overhead without (left) and with (right) PL-Protector.

`Fused Location Manager` combines sensors, GPS, Wi-Fi, and cellular data into a single API for location-based applications [15], 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 of Quality of Service (QoS), communication and computational overheads.

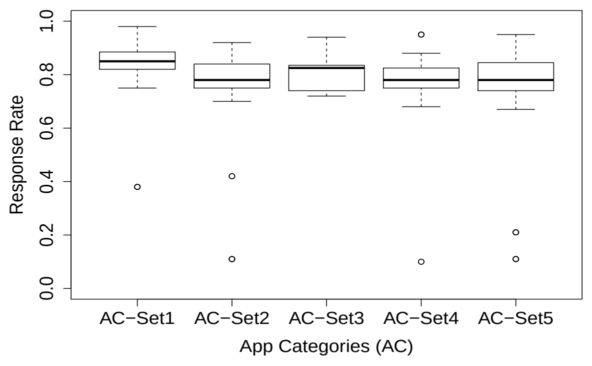


**Fig. 5.** Total response rate by PL-Protector when interacting with apps

## 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 milliseconds 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 an 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., 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)
2. 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)
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: Computational Science and Engineering (CSE), 2012 IEEE 15th International Conference on. pp. 162–169. IEEE (2012)
5. 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)
6. 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)
7. Cranor, L.F., Sadeh, N.: A shortage of privacy engineers. IEEE Security & Privacy (2), 77–79 (2013)
8. 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)
9. European Commission: Protection of personal data (2016), http://ec.europa.eu/justice/data-protection/
10. 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 communications surveys & tutorials 17(2), 998–1022 (2015)
11. 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)
12. Fawaz, K., Shin, K.G.: Location Privacy Protection for Smartphone Users. In: Procs. of ACM Conf. on Computer and Comm. Securty. pp. 239–250. ACM (2014)
13. Google Weave: Weave, https://developers.google.com/weave/
14. Guha, S., Jain, M., Padmanabhan, V.N.: Koi: A location-privacy platform for smartphone apps. In: Procs. of the 9th USENIX conference on Networked Systems Design and Implementation. pp. 14–14. USENIX Association (2012)
15. Hellman, E.: Android programming: Pushing the limits. John Wiley & Sons (2013)
16. IETF: Geographic Location Privacy (March 2016), http://datatracker.ietf.org/wg/geopriv/charter/
17. Michael, K., Clarke, R.: Location and tracking of mobile devices: Überveillance stalks the streets. Computer Law & Security Review 29(3), 216–228 (2013)
18. Muslukhov, I., Boshmaf, Y., Kuo, C., Lester, J., Beznosov, K.: Understanding users' requirements for data protection in smartphones. In: IEEE Int. Conf. on Secure Data Management on Smartphones and Mobiles. pp. 228–235. IEEE (2012)
19. Niu, B., Li, Q., Zhu, X., Cao, G., Li, H.: Enhancing Privacy through Caching in Location-Based Services. In: Proc. of IEEE INFOCOM (2015)
20. 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)

21. Patel, A., Palomar, E.: LP-Caché: Privacy-aware cache model for location-based apps. In: Procs. of the 13th Int. Conf. on Security and Cryptography (SECRYPT). pp. 183–194 (2016)
22. 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)
23. Robusto, C.C.: The cosine-haversine formula. The American Mathematical Monthly 64(1), 38–40 (1957)
24. Samsung: smartthings, http://www.samsung.com/uk/smartthings/
25. 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)
26. Wernke, M., Skvortsov, P., Others: A classification of location privacy attacks and approaches. Personal and Ubiquitous Computing 18(1), 163–175 (2014)
27. Xie, J., Knijnenburg, B.P., Jin, H.: Location sharing privacy preference: analysis and personalized recommendation. In: Procs. of the 19th Int. Conf. on Intelligent User Interfaces. pp. 189–198. ACM (2014)
28. 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)