Location Privacy Protection for Smartphone Users

Kassem Fawaz and Kang G. Shin
Dept. of Electrical Engineering and Computer Science, University of Michigan
Ann Arbor, Michigan, USA
{kmfawaz,kgshin}@umich.edu

ABSTRACT

As smartphones are increasingly used to run apps that provide users with location-based services, the users' location privacy has become a major concern. Existing solutions to this concern are deficient in terms of practicality, efficiency, and effectiveness. To address this problem, we design, implement, and evaluate *LP-Guardian*, a novel and comprehensive framework for location privacy protection for Android smartphone users. LP-Guardian overcomes the shortcomings of existing approaches by addressing the tracking, profiling, and identification threats while maintaining app functionality. We have implemented and evaluated LP-Guardian on Android 4.3.1. Our evaluation results show that LP-Guardian effectively thwarts the privacy threats, without deteriorating the user's experience (less than 10% overhead in delay and energy). Also, LP-Guardian's privacy protection is shown to be achieved at a tolerable loss in app functionality.

Categories and Subject Descriptors

C.2.0 [**COMPUTER-COMMUNICATION NETWORKS**]: General—*Security and protection*; K.4.1 [**COMPUTERS AND SOCIETY**]: Public Policy Issues—*Privacy*

General Terms

Algorithms, Design, Measurement

Keywords

Location Privacy; Anonymity; Indistinguishability; Location-Based Services; Android

1. INTRODUCTION

Location privacy has been a hot topic in research and media over the last decade or so [3,9,21,27,35,43]. The popularity of location-aware smartphones has led to the prevalence of apps that access users' location in order to provide them personalized/customized services. Nevertheless, location access has introduced a new class of privacy threats that users are increasingly becoming aware of. These threats range from an adversary's ability to localize an individual to profiling and identifying him based on the places he visits.

CCS'14, November 3–7, 2014, Scottsdale, Arizona, USA.

http://dx.doi.org/10.1145/2660267.2660270 .

Motivation:

To assess users' perceptions of location privacy and location-aware apps, we surveyed[1] 180 smartphone users. We recruited 70 participants through social network announcements and the rest through Amazon Mechanical Turk. We chose Mechanical Turk workers who have achieved "master qualification," i.e., those who have shown high competency of performing tasks.

We find the survey results supporting the deployment of a location privacy protection mechanism. 78% of the participants believe that apps accessing their location can pose privacy threats. Also, 85% of them reported that they care about who accesses their location information, compared to 87% reported by a Microsoft survey [27] two years ago. Users are even expected to be more sensitive towards this issue in relation to the recent revelations on government accessing their location as collected by apps [3]. Interestingly, 52% of the surveyed individuals stated no problem in supplying apps with imprecise location information to protect their privacy. Only 18% of the surveyed people objected to supplying apps with imprecise location information. Finally, 77% of the users included the term "privacy" as a factor affecting their choice in installing a privacy protection mechanism.

There have been numerous research proposals for location privacy protection from various angles and in various scenarios. Unfortunately, the vast majority of them have not found their way to the common users. Existing mechanisms (e.g., see surveys by Krumm [21] and Shin [35]) suffer several shortcomings that hinder their deployment in the real world. These shortcomings can be best described in terms of effectiveness, efficiency, and practicality as will be more evident in Section 2. Existing mechanisms address the tracking threats without guaranteeing protection against profiling or identification threats. Also, they impose the same protection measures regardless of the app and the privacy threat it poses. Finally, most of these mechanisms rely on unrealistic assumptions, making their real-world deployment difficult.

In this paper, we present the design, implementation, and evaluation of a new location privacy protection framework, called the *Location Privacy Guardian* (LP-Guardian), that addresses the shortcomings of the existing mechanisms. We show that privacy protection can be brought to the masses through a client-side solution at a minimal cost. Although we focus on the Android platform, LP-Guardian is applicable to other platforms that utilize the permission model to authorize location access (e.g., Windows Phone,

[1] https://docs.google.com/forms/d/1VFKlSa3Heq7Wz_mY4MmL7YNu8D74Gtl-lDNlnXesUTo/viewform

BlackBerry OS) as well as those that rely on explicit user authorization for every location access (iOS).

Our design philosophy is based on seven main features as discussed below.

A. The app only accesses location when the user expects it to do so: A user expects the app to access his location only when a location-based functionality is required, e.g., localized-search, place check-in, etc. Most Android apps, however, engage in location acquisition without explicit user authorization/awareness. LP-Guardian addresses this issue by anonymizing location access in the background and enabling the user to choose the appropriate anonymization strategy for foreground location access. Finally, LP-Guardian feeds Advertising and Analytics (A&A) libraries anonymized location samples to prevent tracking through third parties.

B. The app only accesses location with a granularity sufficient to produce the location-based functionality: A majority of Android apps request location with a finer granularity than actually required to deliver necessary service to the user. Our analysis of the top 1150 location-aware apps in Google Play revealed that 68% of them can accommodate coarser location without significant losses in quality of service. As a result, LP-Guardian feeds every app with the location granularity necessary to serve the user. LP-Guardian can thus safely anonymize location for the majority of the apps without hindering their functionality.

C. An anonymous app can't identify the user based on his frequently visited places: As not all apps can afford an anonymized location, feeding them with an accurate location might lead to user identification from frequently visited places (e.g., home and work) [6, 13, 19]. This is applicable even in apps that don't require explicit user identity to function, such as games, search apps, etc. In this paper, we formalize this identification threat and propose a novel mechanism that addresses this threat.

D. A single app alone poses no significant profiling threats based on the collected location information: Some places the user visits are not sensitive to his identity/privacy but might assist in profiling him (e.g., health clinics, religious places, bars, etc.). LP-Guardian relies on the user to learn these places and anonymize the location, if needed, by applying the mechanism of Andrés *et al.* [2].

E. An app can't track the user all the time even when tracking is required to perform functionality: Some apps might require constant monitoring of the user's location, such as fitness or speed-monitoring apps. The absolute location matters less than relative mobility. LP-Guardian reacts by replacing the real location samples with dummy ones that belong to a synthetic route. This route preserves the actual route properties (mainly speed) and does not reveal the actual user's location to thwart tracking.

F. Privacy protection fits within the existing mobile ecosystem: We implemented LP-Guardian in the Android core platform by instrumenting the `Location` object. Our implementation can be easily incorporated in custom ROM or even in a rooted device through the Xposed framework [33]. Moreover, it neither requires modification of the apps nor it relies on additional entities. Most importantly, it is app-aware as it applies different anonymization strategies independently for different apps.

G. Privacy protection comes at a minimal cost in usability and app functionality: Anonymization naturally comes at a cost in terms of usability, delay, energy, and loss of app functionality. LP-Guardian minimizes the interaction with the user and makes the anonymization decisions on his behalf. It also incurs minimal delay and energy overhead (less than 10%). Finally, LP-Guardian minimizes the instances of anonymization to preserve app functionality (more than 60% of the sessions are not anonymized). According to a user study that we performed, such loss of app functionality is tolerable as it comes in places where users don't usually require location-based functionality. This is the first time that a location privacy protection mechanism is evaluated on real-world app usage data.

Contributions:

This paper makes the following main contributions: LP-Guardian

1. provides privacy protection on a per-app basis; the protection level is proportionate to the threat posed and location granularity requirements of the app;
2. packs a novel mechanism to prevent identification by leveraging the notion of indistinguishability; and
3. is practical to deploy and use as it achieves protection for each app independently without modifying the apps and with minimum user interaction.

Organization:

The paper is organized as follows. Section 3 defines the threat model, while Section 2 reviews the related work. Section 4 gives an overview and then details the design of the components of LP-Guardian. Section 5 details the architecture of LP-Guardian, while Section 6 describes its implementation. We evaluate LP-Guardian in Section 7 and make concluding remarks and discuss future work in Section 8.

2. RELATED WORK

Approaches addressing location privacy fall into two categories: theoretical and practical.

Theoretical Approaches

These are the approaches that have been evaluated on traces, but were neither implemented on mobile platforms nor tested with actual apps. Most of these mechanisms address the tracking threat [20, 23–25, 36, 37, 42] in that they hide the user's raw location while still revealing the high-level features of the user's mobility [43], thus becoming not or less effective. These mobility patterns could eventually lead to user profiling and even identification. LP-Guardian protects user's privacy at three levels: *tracking*, *profiling*, and *identification* as will be evident later.

Moreover, some of these mechanisms hinge on unrealistic assumptions, such as trusted infrastructure to provide the privacy protection [12, 25], requiring a set of users of the same app at the same time and same place (e.g., mixzones [11, 29]), or focusing on a subset of location accessing apps [32].

Practical Approaches

Researchers have also proposed more practical approaches that fit within the existing mobile platforms. MockDroid [5] provides users with OS-based controls to disable access to certain resources in Android, including location. The app will never receive location updates. This is a solution that provides full privacy but zero utility. Similarly, Micinski *et al.* [26] coarsen the location supplied to the apps without considering the threat level or the location granularity required by the app. Last but not least, PlaceMask [31] allows users to supply fake locations for apps rendering it unusable. LP-Guardian has the advantage of balancing between privacy requirements, usability, and quality of service.

Other proposed systems require changes to the existing mobile ecosystem to provide the privacy guarantees. Koi [16] relies on a

cloud-based service to achieve location privacy protection. It also requires developers to use a different API for location access that is based on a location matching criterion, rather than the raw location. Similarly, Caché [1] requires developers to change the way they access location in the apps. LP-Guardian requires no modification to the existing apps or infrastructure (it is a completely device-based solution), facilitating its deployment.

Finally, several researchers have studied the problem of private information leakage in mobile devices. For example, TaintDroid [10] tracks the information propagation in mobile devices and detects whether private information (including location) has been leaked. LP-Guardian is complementary to such approaches; it controls what location information the app gets access to, while taint tracking reveals how the apps are managing the accessed location information.

3. THREAT MODEL

We assume an honest-but-curious and passive adversary who is interested in inferring more information about the user from collected location information. Apps constitute the only mechanism by which the adversary can access the user's location through the available location APIs. The adversary won't attempt to hack into the system or circumvent any privacy controls. We view the security challenges as orthogonal to our work. Our objective is not to implement a solution that prevents a determined adversary from overriding the operating system's controls. The existence of such a solution will further strengthen LP-Guardian.

The adversary will collect user's location as part of the app's operation. The collected location information will enable the adversary to pose the following three types of threats:

- *Tracking Threat*: the adversary might receive continuous location updates that enable him to locate the user in real time. The adversary might also be able to identify the user's mobility patterns (frequently traveled routes) and predict his future location with high accuracy by leveraging the typical consistency of people's mobility patterns [18].

- *Identification Threat*: Even if the adversary sporadically accesses user's location, he might still be able to isolate the user's frequently visited locations, such as home and work. The adversary can use these places as quasi-identifiers to reveal the user's identity from anonymous location traces [13, 19].

- *Profiling Threat*: The user's mobility trace might not include places that would reveal his identity, but places that the adversary can use to profile him. Examples include some health clinics, places with religious significance, etc.

We treat all apps belonging to the same developer or having the same signature as one sink of location information. We choose to trust the underlying operating system, since no practical solution can be implemented without such a trust. The user also trusts his own device, including the underlying OS, to manage and store all his personal information.

4. DESIGN

4.1 High-Level Overview

Before delving into the inner-workings of LP-Guardian, we present a high-level overview as shown in Fig. 1. This diagram highlights the main operations performed whenever a new location sample is to be delivered to an app (Section 6.1).

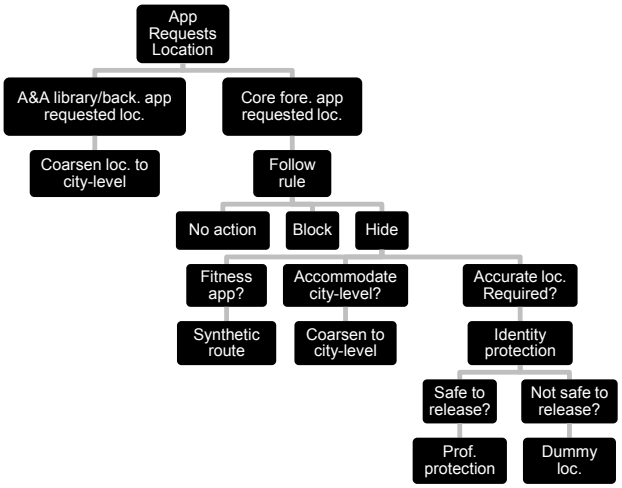

Figure 1: The decision diagram highlighting LP-Guardian's main operations when an app receives a new location update

LP-Guardian first determines if an A&A library or the core app is receiving the location update (Section 4.4). If the A&A library receives the location update, then the location is automatically coarsened to the city-level. If the app is running in the background, the location is also coarsened to the city-level (Section 4.3). We base this design decision on our analysis of the location-accessing apps. The analysis revealed that only 3% of the apps access user's location while running in the background. Thus, location coarsening in the background has little effect on the functionality of most apps. If the core app, while running in the foreground or perceptible states, is receiving the location update, we rely on the user's preference. The location can be released without modification, completely blocked, or anonymized. In the case of anonymization, there are three available options as follows.

1. The app can accommodate coarsening without loss of service (weather apps), in which case the location is automatically coarsened to the city-level (the location is replaced by a pair of coordinates representing the center of the city).

2. The app is monitoring the user's mobility (fitness app), where LP-Guardian feeds the app a synthetic route that preserves some features of the user's actual route (Section 4.7).

3. The app requires location with high granularity (e.g., geo-search app). LP-Guardian applies a novel mechanism to control release of the location to prevent any possible identification (Section 4.5).

If the location is safe to be released, LP-Guardian consults with the user to check if he is comfortable with release of the location. If the user isn't comfortable, the location is obfuscated (noise added) to hide the visited place (Section 4.6), else the location is released as is. On the other hand, if LP-Guardian decides that it is not safe to release the location, it replaces the real location with a fake location as described in Section 4.5. Finally, LP-Guardian minimizes user interaction as much as possible, in order not to hinder the user experience. At the same time, it keeps the user in the loop by informing

him of the anonymization process and asking for his feedback only when needed (Section 6.2).

In what follows, we elaborate on the main design decisions that address the various location privacy threats.

4.2 Location Sources

The location APIs are not the only way by which apps can access location information. Scanning the nearby WiFi access points (APs) and cellular towers might help locate the user. Skyhook, for example, is an online service that maps the signature of nearby APs to location coordinates. Such location can be fairly accurate (<30 m accuracy). Google localization service relies on a similar concept to provide network-based location samples. LP-Guardian addresses this issue by preventing apps from scanning nearby APs to limit the apps' location access to Android's location APIs.

4.3 Foreground vs. Background

In Android, an app can assume multiple states depending on its execution status. Android recognizes four app states, three of which are important to us: running in the foreground, background, and perceptible to the user. A foreground app is the one that occupies the screen and the user can interact with. When the user exits the app, Android caches it for faster re-execution and is thus moved to the background. An app can also run persistently and show the user a notification indicating that it is perceptible. For example, apps like Google Now run persistently as a service and are not considered background apps.

Because of elongated periods of location access, tracking threats are more pronounced in apps accessing location in the background or when running as a persistent service. We handle the former case of background location access by coarsening the location to the city-level. That way, the app will still receive relevant location updates, but won't be able to pose any viable location privacy threats. We will also specify how we handle location access in the foreground and as a persistent service.

4.4 A&A Libraries

Most free apps pack A&A libraries to generate revenue by displaying targeted ads to the user running the app. Hence, the apps need to feed these libraries some information about the user, including location [38]. As the number of A&A libraries is limited as compared to the number of apps, an A&A library is most certainly to be packed in multiple apps.

Instead of looking at apps as independent sources of the user's location trace, A&A libraries can aggregate location traces from multiple apps. This implies that the location privacy threats posed by these libraries is more critical. However, apps can thus easily accommodate feeding these libraries with coarsened location samples to the city-level. These libraries will still receive relevant location information to display ads to users, but won't be able to pose any viable tracking/identification/profiling threats.

Theoretically, it is plausible to coarsen location for A&A libraries, but the challenge is how to separate between location requests coming from the A&A library and those coming from the core app. To deal with this challenge, we studied more than 100 A&A libraries in Android. We collected these libraries from the top 1100 apps in Google Play and from different literature sources [7, 15, 30, 38]. It turns out that 98% of these libraries access location information from their code space through two mechanisms: the app (1) enables location collection, or (2) passes them a location object that they can access. In both cases, the stack trace of every location access request should reveal whether the request is coming from the app or the packed library. Fig. 2 shows an example of the stack trace of a location access request from WebMD app; it is evident how the request originates from Medialets Analytics.

```
dalvik.system.vmstack
java.lang.thread
android.location.location
com.medialets.analytics.e
com.medialets.analytics.mmanalyticsmanager
com.medialets.analytics.mmanalyticsmanager
com.medialets.analytics.mmanalyticsmanager
com.medialets.analytics.mmanalyticsmanager
com.medialets.advertising.admanagerservice
android.app.intentservice$servicehandler
android.os.handler
android.os.looper
android.os.handlerthread
```

Figure 2: The stack trace for a location request by WebMD app with the analytics library method calls highlighted.

4.5 Identification Protection

As LP-Guardian coarsens location accessed in the background, accurate location access is limited to the foreground. The apps will thus only sample the user's location only when running in the foreground. Foreground sessions are short, sporadic, and occur mostly within the same place. These facts will be shown later in Section 7 based on three datasets that measure app usage patterns. This implies that there is a one-to-one mapping between an app session and a visited location. For these apps, the mobility information can be best viewed in the form of a histogram of visited places. We model the city which the user is visiting as divided into a 2D grid, where every cell refers to a city block. The set of city blocks is $Bl = \{bl_1, bl_2, bl_3, \ldots\}$. Every resident has a probability distribution of visiting the blocks in the city as $p_i = P(bl_i)$; this probability will be 0 for blocks the user never visits. After a period of app usage, the app records the number of user visits to every block, thus forming the histogram. Each bin in the histogram is the number of times, c_{bl_i}, the app observes that the user was at the block bl_i.

Even if the location information is anonymous, an adversary can map the user's histogram to the user's identity given the background information at the adversary's side. This is what is widely referred to as the *identification* or *inference* attack in literature [6, 19]. In what follows, we provide the first formal treatment of this attack. We assume that the adversary has access to background information in the form of a mapping between a set of individuals' identities and the probability distribution of visiting each block in the city. The adversary aims to match the anonymous mobility histogram from the app to one of the individuals in his database.

Specifically, the adversary is interested in the probability of the histogram belonging to any of the individuals, x, in his database, $P(h_{app}|x)$. If the app records N user sessions, $P(h_{app}|x)$ is the probability of observing the individual x being at each bl_i for c_{bl_i} times out of a total of N observations, where this individual has a probability p_i of visiting block bl_i. As a result, the probability distribution of the histogram follows the multinomial distribution, assuming that different app sessions are independent, and is given as:

$$P\left(h_{app}|x\right) = \frac{N!}{\prod_{i=1}^{Bl} c_{bli}!} \prod_{i=1}^{Bl} p_i^{c_{bli}}. \quad (1)$$

This model holds when $c_{bli} = 0$ as $c_{bli}! = 1$ and $p_i^{c_{bli}} = 1$.

If the app can't accommodate coarsened location (e.g., geo-search app), then LP-Guardian releases the user's location as long as the user's histogram can't be mapped to his identity. In other words, the

probability of the histogram originating from the real user should be close to the probability of the histogram originating from another individual within the same city. We define the "closeness" of probability distributions in terms of Eq. 2 which is similar to the δ disclosure criterion [8]. For two individuals x and y, the histogram maps to both individuals with a close probability:

$$e^{-\epsilon} \leq \frac{P(h_{app}|x)}{P(h_{app}|y)} \leq e^{\epsilon}. \quad (2)$$

Plugging the probability expression of Eq. 1 into Eq. 2 we get for two individuals x and y:

$$e^{-\epsilon} \leq \frac{\prod_{i=1}^{|Bl|} (P(bl_i|x))^{c_{bl_i}}}{\prod_{i=1}^{|Bl|} (P(bl_i|y))^{c_{bl_i}}} \leq e^{\epsilon}. \quad (3)$$

For the rest of this paper, we assume $\epsilon = 0.5$. The user's histogram must obey the property of Eq. 3 to satisfy the privacy criterion. As LP-Guardian operates solely on the client side, it has no information about other individuals in the city, and thus can't fill in the probabilities for other potential individuals. Alternatively, we consider a criterion that other individual must satisfy in the form of a minimum probability of visiting the blocks that the user visits. So, we replace the $P(bl_i|y)$ values in Eq. 3 with a value of p_{min}. Now, the privacy criterion states:

The user is indistinguishable within a theoretical set of individuals who have a minimum probability of p_{min} of visiting the places in the user's histogram.

The value of p_{min} thus controls the level of the user's privacy; the higher p_{min} the lower the privacy guarantee, as less people will visit the same places as the user with high probability. On the other hand, a lower value of p_{min} will indicate a stricter privacy guarantee as the user will be potentially indistinguishable within a larger set of individuals. We rearrange Eq. 3, after applying the logarithm, to satisfy the following inequality:

$$-\epsilon \leq \sum_{i=1}^{Bl} c_{bl_i} \left(\ln\left(P(bl_i|x)\right) - \ln\left(p_{min}\right)\right) \leq \epsilon. \quad (4)$$

This inequality provides a test for releasing the location from a session or not. For every new app session, the expression of Eq. 4 is evaluated given the past released histogram, the user's mobility model, and the value of p_{min}. If the summation value in Eq. 4 exceeds ϵ or falls below $-\epsilon$, then the location can be released; otherwise, a dummy location within the city is released. Eq. 4 provides an important insight on locations the user visits. If we view ϵ as a privacy budget, places that users frequently visit will exhaust part of the budget. On the other hand, if the user visits a place with a probability lower than p_{min}, this will increase the available privacy budget. The released dummy location is one that the user visits with a very low probability which helps increase the available privacy budget.

So far we have assumed the protection mechanism is blind of background information regarding other users. The availability of such information, however, can assist in improving the trade-off between app functionality and privacy. Our idea is based on hiding the user among a theoretical set of people who have to satisfy a minimum probability constraint of the visiting the places that the user visit. If we know that an actual set of people satisfy a more relaxed constraint, then we can achieve the same privacy level with much improved usability.

Our mechanism relies on the census data that specifies the population in every city block [39]. It identifies the user's home location as the most fequently visited place during night time. It then assigns the home block location a value of p_{min} consistent with the population in the same block. If the number of block residents is above 500, we automatically assign p_{min} for the home location the same value of the user's probability of visiting the home location. The privacy criteria then transforms to the subset of the home block residents who have a minimum probability of p_{min} of visiting the other user places of interest. In that sense, the user enjoys a natural protection level resulting from the fact that he lives in a crowded place. Our mechanism can be relaxed more in releasing the home location. This insures protection against the identification threat; the user will still have the option of hiding his location (or any other location) as will be evident later.

If a user lives in a sparsely populated area, then he naturally suffers lower privacy guarantees. LP-Guardian reacts by assigning very low p_{min} values for these users, meaning they won't be able to release locations from frequently visited places (e.g., home or work) for apps that require location with high precision (e.g., Yelp). Nevertheless, one could also argue that people who live in sparsely populated areas have a lower need for location-based services at home or work as they will tend to be more familiar with the area.

4.6 Profiling Protection

A user could be profiled based on the places that he visits, albeit at a low frequency. A user might not be willing to reveal that he is visiting a particular church, a health clinic, a certain bar, or some hotel. Revealing these places might enable an entity with access to the user's location to profile him. LP-Guardian addresses these threats by putting the user in control, as he is the best judge of determining the places which profile him. Each time the user invokes an app from a new place, the user has to decide whether he is willing to hide the place he is currently visiting (more details in Section 6). If the user opts to hide his place, we leverage the solution of Andrés *et al.* [2] to anonymize his location.

This approach is well-suited for sporadic location access. It adds noise drawn from a polar Laplacian distribution to make the reported location provably indistinguishable from the actual location within a given radius. Given the user's actual location as a pair $< x, y >$ and an anoymization radius r, the added noise can be computed in terms of a pair $< rad, \theta >$ as follows [2]:

- Draw θ from the uniform distribution $[0, 2\pi)$;
- Draw p from the uniform distribution $[0, 1]$, and set $rad = -\frac{r}{l}(W_{-1}(\frac{r(p-1)}{l}) + 1)$, where W_{-1} is the -1 branch of the Lambert W function and l is a privacy level typically chosen close to 1.

The new location is just a translation of the original location by rad and θ. The noise level, r, determines the underlying privacy – utility trade-off. In LP-Guardian, we rely on a noise value of 200m; the user's actual location is hidden among location within a range of 200m. Still, the location will be of some utility to maintain some app functionality. As long as the user is visiting the same place, the anonymized location reported to the app is kept the same to prevent any additional leakage of information.

Finally, the user's IP address might reveal the place he is visiting, especially if he accesses the Internet from a public hotspot [40]. Although not included as part of LP-Guardian, TOR (Android app is available on Google Play) could be used to anonymize the user's IP address and protect his location effectively.

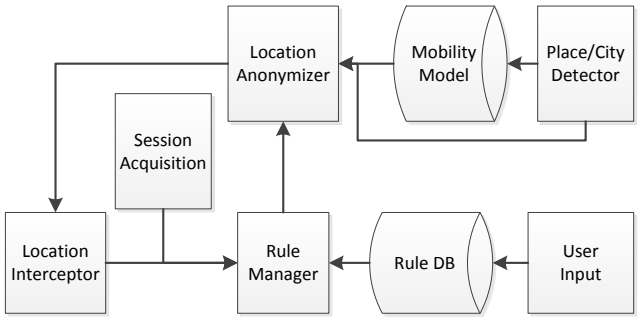

Figure 3: LP-Guardian's architecture and interactions of its components

4.7 Synthetic Route

Fitness apps track the exercising activity of the users. They provide the user with feedback on the path and distance covered during an exercise session. These apps monitor the user's location in the background for elongated periods of time and with high location precision, thus posing a tracking threat. We approach this type of apps by separating between the two objectives: distance/speed and path of the exercise. Instead of crudely anonymizing location which will render these apps useless, we sacrifice the second objective to provide privacy while maintaining the first.

LP-Guardian anonymizes the location for these apps by keeping the distance the user covered the same while modifying the actual route. It essentially feeds the fitness app a synthetic route that has the same length of the actual route. Whenever a new fitness tracking session starts (such an app is running as persistent service and location update rate is high), LP-Guardian feeds the app with a random location within the current city. Then, as the app receives new location updates, LP-Guardian reacts by calculating the distance covered since the last update, and calculates a new location sample based on this distance and the last reported location.

4.8 Navigation Apps

Navigation apps are the most challenging. They require elongated location access periods with high precision. The current version of LP-Guardian doesn't handle this case, but we provide some remedies that balance usability and privacy in navigation apps. Offline navigation (e.g., Garmin, Google Maps) can handle part of the problem by preventing the app from sharing the user's location in real time. However, there are no guarantees against an app leaking this information when it comes online. A possible remedy is to run the navigation app in a "private mode", where it's disbarred from connecting to the Internet, and the stored data is wiped after the session ends. The downside is that the app will be deprived of access to the real-time traffic information that is useful for navigation.

5. ARCHITECTURE

Fig. 3 shows the block diagram that implements LP-Guardian. Described below are the main components of LP-Guardian and their interactions.

Location Interceptor

The location interceptor, as its name implies, is responsible for diverting the app's control flow to our service in the event of a location access. This module blocks the app, extracts the newly acquired/created location, and sends it (along with the app's package name) to our service for further processing. It is also responsible for delivering the anonymized location to the app so that the control flow can be resumed. We describe the implementation of this module in Section 6.1.

Rule Manager

This module selects the appropriate rule—via interaction with the user—to apply depending on the situation. The rule indicates an appropriate strategy for the location anonymizer module to follow. The rule manager module takes as input the app, location sample, app state (foreground or background), and whether this is a new or ongoing session from the session acquisition module. A foreground/background session is the time period during which the app is continuously executing in the foreground/background. The rule manager also takes input from the place detector module to know which place/city/block the user is currently visiting. This module caches the rule from the database for use in subsequent location accesses of the same session. We define two types of rules as follows.

- *Global Rules*: indicate the protection mechanisms invoked for both the foreground and background states. There is only one global rule per app.
- *Per-place Rules*: control the protection level of individual places the user might be visiting. There is one per-place rule for every app–place combination.

The global rule is always given preference over the per-place rule as it considers the big picture of the user's released location traces. Invoking this rule will determine whether the current location is safe to be released. If it is to be released, the per-place rule is invoked to decide if any further protection is needed.

Place/City Detector

This module is responsible for identifying the current place the user is visiting and for maintaining the user's mobility model. It performs online processing of the user's real location trace to identify spatio-temporal clusters, in a manner similar to that proposed by Bamis *et al.* [4]. Formally, we define a place as a cluster of locations within a radius of 100m and over a minimum duration of 5 minutes. Whenever the place detector receives a new location sample, it tries to map it to one of the existing places in the database. If the mapping is successful, it updates the total visiting time of that place, else it creates a new place. The mobility model is the set of places the user visited along with the total visiting time of every place. This enables computation of the probability of visiting every place. This module also maps the user's location to the nearest city/block.

Location Anonymizer

The location anonymization module is the cornerstone of LP-Guardian. It takes as an input the location, the app state (foreground or background), and the rule from the rule manager module, and outputs the anonymized location back to the location interceptor module. The rule dictates the appropriate anonymization strategy to be followed as indicated in Section 4.

6. IMPLEMENTATION

6.1 Core Implementation

Android provides apps with two mechanisms with which they can access the user's location: the older Location Manager Service and the newer Google Play Services. In both mechanisms, the apps request regular updates by registering a location listener which acts

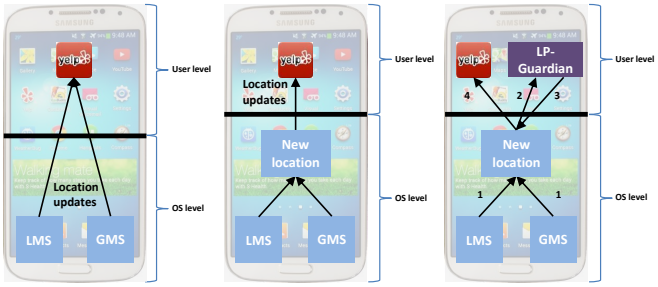

Figure 4: Location access mechanisms in Android (left & middle) and LP-Guardian's deployment within Android (right)

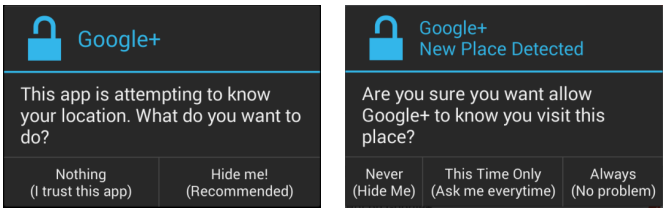

Figure 5: The displayed prompts to set the global (left) and per-place (right) rules

as a callback function (Fig. 4 – left). Whenever a new location is ready, the callback function is invoked, and the app receives a new location object.

To enforce any kind of location protection mechanism, the location passed to the app has to be modified. Moreover, such a mechanism has to be app-aware so that it may modify the location on a per-app basis. There are two options for modifying the location on a per-app basis. The first involves instrumenting the app's location accessing interfaces and intercepting location updates before they reach the app. While the other option involves instrumenting the platform and changing the location object before reaching the app.

The first option could be implemented with a mechanism similar to Aurasium [41]. Albeit effective, this mechanism requires unpacking, instrumenting, and then repackaging the app. Repackaging the app will change its signature. This will break any future updates to the app, and affect any functionality requiring an authentic app signature. Moreover, the users will have to download the app from a different app store that requires an entity to manage and keep it up-to-date. Instead, we opted to implement a platform-based solution that treats the apps as black boxes and maintains their full functionality.

Android relies on two mechanisms to relay location updates to the apps. So, any platform-based solution has to instrument both mechanisms to intercept and then modify the location object before reaching the app. Nevertheless, Android ships Google Play Services as a closed-source app, and hence, its instrumentation is not straightforward. Fortunately, both mechanisms create a location object prior to propagating it to the app. The location class is included in Android code and can be altered easily (Fig. 4 – middle).

The location object is merely a data holder, but has to be created within the app process space before delivering it to the app. We added a static context field to the location class that is populated when the app is invoked, particularly when a context is created for the app. A context is what enables an app in Android to interact with the OS resources. Enabling the location object with a context enabled us to (1) know what app is currently creating a location object, and (2) communicate with OS or other processes, if needed. We instrumented the location object and pushed the location privacy logic to an independent system app that can be easily updated, if needed. Since the location object has a reference to the app's context, it can communicate with that external service (Fig. 4 – right) whenever a new location object is created.

Anonymization incurs a processing cost. It might be prohibitive subjecting every location update to the anonymization operation, especially in case of very high location update rates. It is unlikely that the user's position will change within a second, even with speeds up to 100 mph. Consequently, LP-Guardian only communicates with the anonymization service once every 750ms.

6.2 User Interface

Our design philosophy is to rely on the user's input as infrequently as possible. LP-Guardian makes decisions on his behalf to control different configuration and anonymization parameters. Nevertheless, there are some situations where the user needs to interact with LP-Guardian to enable three features: bootstrapping, per-place, and per-session controls. Each of these features is elaborated next.

6.2.1 Bootstrapping

Initially, the mechanism is blind and doesn't have enough information to function. The two main missing pieces of information are user mobility data and per-app anonymization rules. As explained earlier, our identification protection mechanism requires mobility data to function. LP-Guardian can't collect mobility data at runtime, as by the time we have this information, other apps will have access to it. At the very first time when the LP-Guardian boots, the user has to set his top N visited places by tapping on a map (the user still has the option of adding places later through an advanced settings menu). We then set the initial probabilities of visiting these places according to the model proposed by González *et al.* [14]. Specifically, LP-Guardian assigns every place a probability of visit inversely proportional to its rank of visit, then normalizes the probabilities to form a PDF.

The second piece of the bootstrapping includes setting the rule for every app. The first time the app accesses location, LP-Guardian present prompts (Fig. 5 – left) the user to set the app's anonymization rule. The user only gets to choose one of two self-explanatory options: "Do nothing" and "Hide Me." When the user chooses the second option, LP-Guardian chooses the appropriate anonymization mechanism depending on the app. It applies city-level coarsening for apps that accommodate such granularity. For the rest of the apps, it chooses identification protection with a default value of p_{min} = 0.0005. Additionally, LP-Guardian sets the per-place rule for the location the user is currently visiting. We use 0.0005 as the default value, which amounts to spending 30 minutes at a certain place each 40 days. LP-Guardian also has an advanced settings menu that allows the user to set fine-grained p_{min} for each app.

6.2.2 Per-place/session controls

The per-place rules control the profiling protection level. As LP-Guardian can't infer places sensitive to the user, it relies on explicit user input to set these rules. Whenever an app attempts to access the user's location from a new place (one the app hasn't seen before), LP-Guardian will prompt the user to ask for his decision (Fig. 5 – right). The user has to choose one of three options: hide the place every time (Section 4.6, reveal this place only during the current

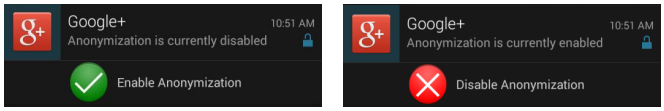

Figure 6: The displayed notifications when anonymization is disabled (left) and enabled (right)

session, and reveal this place always. If the user chooses the second option, LP-Guardian will keep on issuing the prompt from the same place until the user chooses a permanent option (either the first or third option).

Finally, LP-Guardian always ensures that the user is aware of the anonymization if it takes place. Whenever the user is running an app and anonymization takes place, a notification is displayed (Android notifications are placed in the top left corner and are non-intrusive). This notification, shown in Fig. 6, displays the app, a note to the user, and an option to temporarily disable the anonymization during the current session. If the user disables anonymization, the notification will include an option to re-enable location anonymization.

6.2.3 *Properties of the prompts*

Authorizing resource access in software has been studied before. Livshits [22] proposed a set of four requirements for valid placement of user prompts that control for resource access (including location) in mobile platforms. They are: safety, frugality, visibility, and non-repetitiveness. To achieve safety, a user prompt must precede every resource access. A prompt is frugal if it is *only* displayed in the event of a resource access. Visibility indicates that a user prompt must only be displayed if the app currently executing in the foreground is attempting to access a resource. Finally, a user prompt is not repetitive if it is never displayed for a resource access when a more critical resource of the same type has already been authorized.

LP-Guardian satisfies the last three constraints by prompting the user only in the event of a location request and while running in the foreground thus satisfying both frugality and visibility. As for non-repetitiveness, it is already ensured through Android's permission model. If an app is granted fine location permission, it is automatically allowed access for coarse location, but the inverse doesn't hold.

The safety requirement ensures that no location access goes unchecked. Theoretically, this is an optimal requirement to protect the user's privacy. Nevertheless, the users can't be expected to assess the implications of every resource access and thus can't always make informed decisions. Second, protecting every resource access with a user prompt will affect the user experience negatively as the app execution will be interrupted frequently. One option, as implemented in iOS, is to prompt the user at the first time location access with an option to remember the decision. Still, that doesn't appropriately address the two issues. If the user chooses to remember the decision, this will certainly reduce the frequency of the prompts but will not necessarily protect the user's privacy.

LP-Guardian addresses the above two issues by balancing between the frequency of prompts and privacy guarantees. Only the first location access from every newly visited place is preceded by a prompt to allow the user to make a choice. LP-Guardian then makes the subsequent decisions on the behalf of the user. It asks users to make decisions in terms of hiding or revealing places rather than asking them to authorize location accesses. Needless to say,

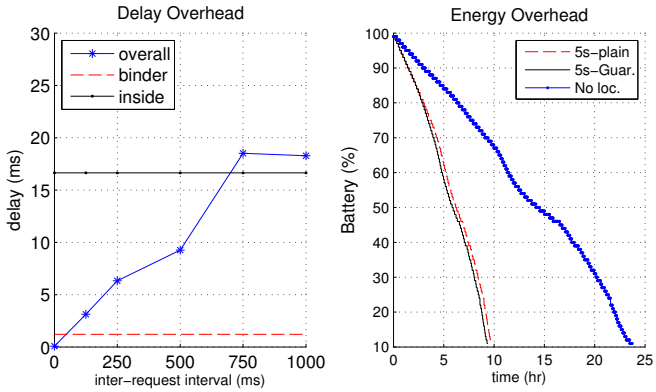

Figure 7: The delay overhead (left) and battery depletion rate (right) of LP-Guardian for Galaxy Nexus

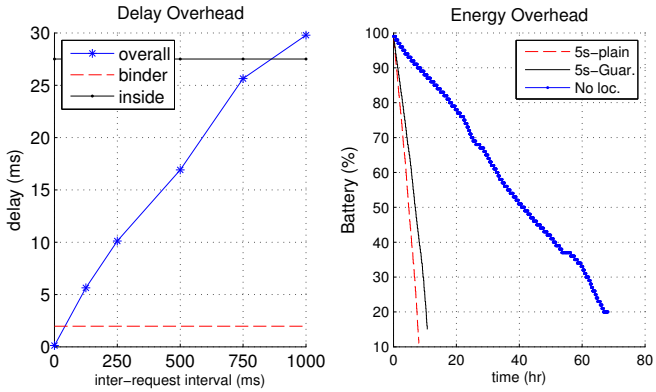

Figure 8: The delay overhead (left) and battery depletion rate (right) of LP-Guardian for Galaxy S3

users can relate better to places rather than raw location samples. It is worth noting that the frequency of prompting might be high at the beginning. It should, however, decrease after LP-Guardian learns the places the user visits as people tend to exhibit consistency in their mobility.

7. EVALUATION

We now evaluate LP-Guardian in terms of performance, privacy guarantees, and usability.

7.1 Performance

First, we validated that LP-Guardian can effectively obfuscate the location delivered to the apps and verified that it doesn't cause the apps to crash. We manually installed, ran, and verified 40 representative location-accessing apps. We then evaluated delay and energy overheads on three devices: Google Galaxy Nexus, Samsung Galaxy S3, and Samsung Galaxy S4 running Android 4.3.1.

To evaluate the delay overhead, we simulated location access through an app that accesses location with a varying frequency. We then measured the delay between the time the location is created and the time the modified location is delivered to the app. The left plots of Figs. 7–9 show the average delay for each of the three devices versus the inter-request interval. Bearing in mind that LP-Guardian temporarily caches location, the delay value increases with the increase of the inter-request interval.

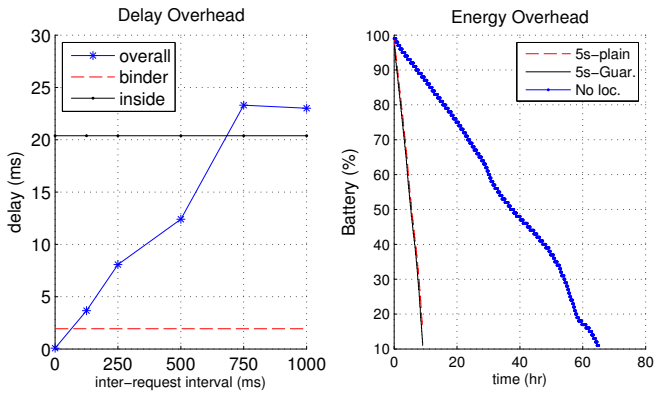

Figure 9: The delay overhead (left) and battery depletion rate (right) of LP-Guardian for Galaxy S4

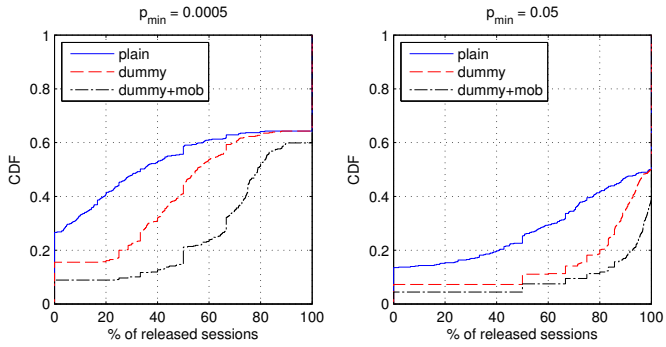

Figure 10: The distribution of the released sessions for our dataset

We also measured the anonymization delay for the worst-case scenario where the anonymization actually takes place (i.e., communication with the anonymization service). The delay was less than 20ms for two devices and less than 30ms for the Galaxy S3 (the curve labeled "inside" in every plot of the delay overhead plots (left plots of Figs. 7–9). LP-Guardian imposes a maximum delay of 30ms every 750ms which shouldn't impact the app usability.

To evaluate the energy overhead, we considered two scenarios: no location access, and a load of location access of one request per 5 seconds. We measured the rate of battery depletion for each scenario with LP-Guardian running and compared it with the rate when the framework is not running. The right plots of Figs. 7–9 show the rate of battery depletion for each of the three devices and for the two loads.

For a location access rate of 1/5 Hz (one request per 5 sec), the battery depletion rate when LP-Guardian is running is very close to the case when it is not. This is evident from the the curves close to the y-axis in the energy overhead plots (the right plots of Figs. 7–9) for each of the three devices, with an energy overhead less than 10% for the three devices. In the case of no location access, while LP-Guardian was running, the battery lasted more than 24 hours for Galaxy Nexus (Fig. 7 – right) and more than 60 hours for both Samsung Galaxy S3 (Fig. 8 – right) and S4 (Fig. 9 – right). Actually, LP-Guardian doesn't do any processing in the background except for location acquisition to maintain the mobility model. All other processes are invoked in response to a location request by the app.

7.2 Privacy

In Section 4, we proposed a set of mechanisms to cope with the identification and tracking threats. In what follows, we evaluate the privacy protection mechanisms and their impact on quality of service.

7.2.1 Datasets

We evaluated LP-Guardian over three datasets that contain app usage information for 100 users over a period varying between 1 week and 1 year in three different cities. The first dataset includes 25 Android users (running Android 4.0.3 or higher) that we recruited from our institution over the period of 8 months. We collected app sessions and whether the app collected location while running. The second dataset consists of 49 Android users whom we recruited through PhoneLab [28], a smartphone measurement testbed from the State University of New York, Buffalo. We also collected app sessions from these users over 4 months. Finally, we relied on the dataset from the LiveLab project from Rice University [34]. They collected app sessions from 30 iPhone users for a period of one year. The subsequent analysis assumes a worst-case scenario of an app accessing location whenever it runs and then sharing it with the service provider.

7.2.2 Identification Protection

We evaluate the loss in app functionality when LP-Guardian is applied in two scenarios: a conservative privacy requirement of p_{min} = 0.0005, and more relaxed one of 0.05. In what follows, we report the percentage of sessions in which app functionality is negatively affected. For every scenario, we report the distribution of the percentage loss of sessions for every app–user combination. We report this distribution when the mechanism is applied plainly, when we add dummy queries, and when we add the mobility data from the US census data [39].

Figs. 10–12 show the potential loss in app functionality for each of the three datasets, ours, LiveLab's and PhoneLab's datasets consecutively. We can draw the following conclusions from observing these figures.

- More relaxed privacy constraints will enable releasing more of the user's location (comparing the left and right plots in every figure).
- Adding dummy locations and including the mobility data greatly increases the number of released sessions (comparing the dashed and dotted lines in every plot).
- Users with more diversity in their mobility patterns (visit more places) enjoy naturally higher privacy protection and can thus release more sessions (comparing our and LiveLab's datasets with PhoneLab's).
- The number of unreleased sessions with potential loss in functionality is low. More than 60% of the sessions are released for more than 60% of the users in the more privacy-constraint scenario of $p_{min} = 0.0005$.
- Although not shown in the figures, most of the unreleased sessions belong to top visited places (mostly home and work) for each user. These places are more critical to the user's identity.

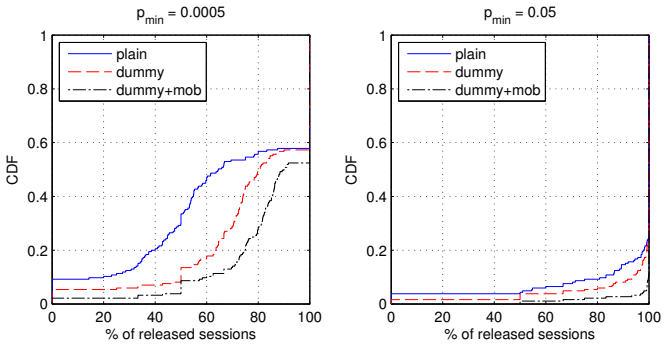

Figure 11: The distribution of the released sessions for LiveLab's dataset

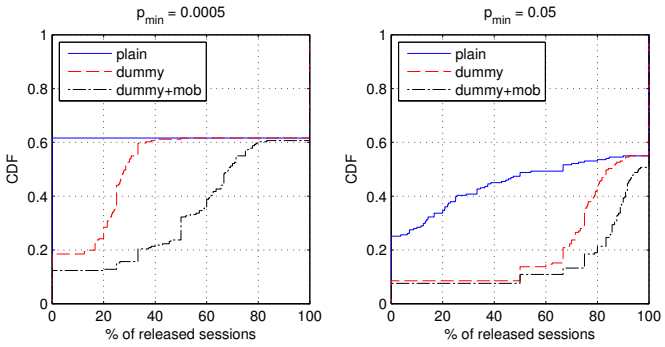

Figure 12: The distribution of the released sessions for PhoneLab's dataset

7.2.3 Tracking

As for the tracking threat, our mechanism blocks location access in the background, which leaves tracking limited to foreground sessions that are sporadic (mostly invoked at most once a day), short (average session length less than 5 minutes) and invoked from the same location for more than 98% of the time. Furthermore, our high-level privacy protection schemes hide locations in some sessions that are critical to the user's privacy. However, as noticed above, a considerable number of the sessions will be released. We evaluated the tracking threat each app could pose through the time tracked per day (similar to the time-to-confusion metric [17]). We evaluate the total time (in seconds) per day during which an app receives accurate location updates. We focus only on the case where most of the sessions will be released (p_{min} = 0.05) as it constitutes the worst-case scenario in terms of tracking.

Figs. 13–15 show the distribution of the time tracked per day for each of the three datasets (ours, LiveLab, and PhoneLab respectively). In all three plots, the approach with dummy and mobility data (labeled dummy+mob) releases the most of the sessions and thus has the most tracking threat. In all of the cases, 90% of the user–app combinations have tracked time per day of less than 500sec/day; most apps can't track the user for more than 8 minutes a day. It is important to note that most of the foreground sessions happen when the user is stationary, which prevents the adversary from tracking the user while moving. This limits tracking to counting the places that the user visits. LP-Guardian's identification and profiling prevention schemes handle threats resulting from the inference attacks based on the patterns of visited places.

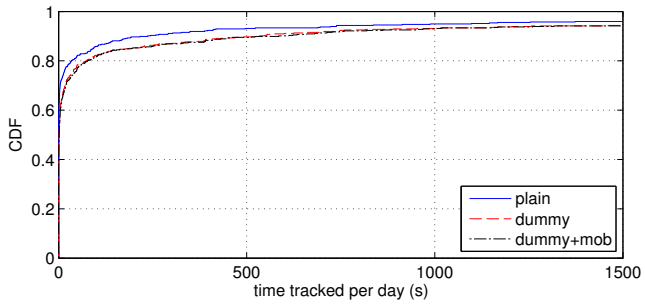

Figure 13: The distribution of the time tracked per day metric for our dataset

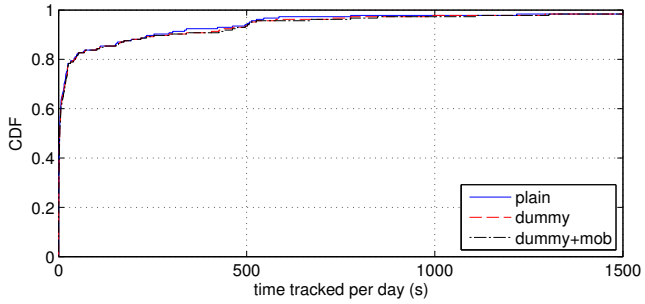

Figure 14: The distribution of the time tracked per day metric for LiveLab's dataset

7.3 User Study

LP-Guardian achieves theoretical guarantees on privacy with an inevitable loss of quality of service, albeit kept to a minimum. Most of these hidden sessions belong to home and work locations. To assess the user's perception of such loss in app functionality, we asked the participants (same as those of Section 3) whether they could accommodate reduced functionality of their apps from home and work locations for six classes of apps.

- Geo-search (e.g., Yelp): We asked participants (107 have such an app installed) about the level of comfort (scale 1–5) if they receive inaccurate search results while performing a search from either home or work. The majority of the participants (57%) indicated a comfort level larger than or equal to 3.

- Social networking (e.g., Facebook): We asked the participants (156 have such an app installed) whether they share location while being either at home or work. Most of the participants (62%) answered that they don't, 29% answered that would sometimes invoke this feature from either places. Also, 82% of the participants answered that they have **no** problem with sharing a city-level location instead of the actual home or work locations. The rest of participants either had no idea (5%), or prefer sharing the actual location (13%).

- Messaging/chatting (e.g., Whatsapp): The participants who reported installing such apps (123) don't normally invoke the location sharing feature from home and work. Only 6% perform such location sharing regularly, and 14% perform it sometimes. Also, most participants (78%) indicated that they have no problem in sharing a city-level location instead of home or work actual locations.

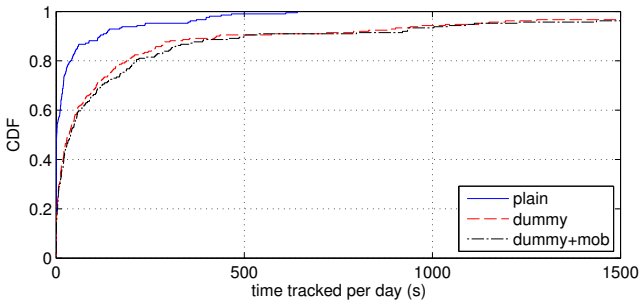

Figure 15: The distribution of the time tracked per day metric for PhoneLab's dataset

- Sports/fitness tracking (e.g., Runkeeper): Users rely on these apps to monitor their exercising activity. Thirty participants have reported installing and running a fitness tracking app. Only 6% of the participants care about viewing their actual tracks, while the rest either care more about the distance (36%) or both the distance and viewing the tracks (58%).
- Gaming (e.g., Angry Birds): Most of these apps access location to feed A&A libraries in order to generate revenue. Of the participants who play games on their smartphones (131/180), 88% reported that their gaming experience won't be different despite possible location anonymization.
- Weather (e.g., Weather Bug): More than half the participants reported that weather information won't differ inside the same city, indicating that coarsening location information supplied for weather apps won't hinder their functionality.

It is evident from the survey that users generally won't mind some loss of app functionality at home and work. This is actually expected as users would rely more on location based functionality at unfamiliar places as opposed to places that they live or work at. For example, the user study shows that users would have no problem searching for relatively far places from their home or work locations when utilizing geo-search apps. Also, users seldom share their home or work locations while using social networking or chatting apps. Moreover, users are relatively open to the possibility of loss of quality of service when using sports tracking apps. In summary, LP-Guardian provides users with privacy at a tolerable loss of app functionality.

8. CONCLUSION & FUTURE WORK

In this paper, we proposed LP-Guardian, a novel location privacy protection framework for Android smartphone users that is practical, effective, and efficient.

- *Practical*: We fully implemented LP-Guardian on Android 4.3. It is compatible with all the apps, and is not difficult to deploy. LP-Guardian also incurs acceptable energy and delay overheads.
- *Effective*: LP-Guardian addresses location privacy threats over three levels. It addresses the tracking threats through reducing the time tracked per day. It addresses the profiling threat by enabling the user to hide sensitive places. Finally, it addresses identification threats through a novel mechanism that makes the user's mobility pattern indistinguishable.
- *Efficient*: LP-Guardian provides privacy guarantees at a tolerable loss in app functionality. User's location is hidden for a small number of sessions and at places where location-based functionality is less needed.

In the future we plan to pursue the deployment challenges related to location privacy protection. We are planning to deploy LP-Guardian in the phones of a set of diverse participants with limited technical background for a period of six months. This usability study will allow us to fine-tune LP-Guardian and identify areas where the privacy–usability tradeoff could be improved. Finally, we will consider incorporating LP-Guardian as a part of a custom ROM (e.g., CyanogenMod) to reach a larger audience.

Acknowledgments

The work reported in this paper was supported in part by the NSF under Grants CNS-0905143 and CNS-1114837, by the AFOSR under Grant FA9550-10-1-0393, by the ARO under Grant W811NF-12-1-0530, and under a Google Faculty Research Award. We would like also to thank Carmen Ruiz Vicente and Xinxin Liu of Google for useful discussions and comments on this work.

9. REFERENCES

[1] S. Amini, J. Lindqvist, J. Hong, J. Lin, E. Toch, and N. Sadeh. Caché: Caching location-enhanced content to improve user privacy. In *Proceedings of MobiSys '11*, pages 197–210, New York, NY, USA, 2011. ACM.

[2] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi. Geo-indistinguishability: Differential privacy for location-based systems. In *Proceedings of CCS '13*, pages 901–914, New York, NY, USA, 2013. ACM.

[3] J. Ball. Angry birds and 'leaky' phone apps targeted by NSA and GCHQ for user data. http://www.theguardian.com /world/2014/jan/27/nsa-gchq-smartphone-app-angry-birds-personal-data, January 2014.

[4] A. Bamis and A. Savvides. Lightweight extraction of frequent spatio-temporal activities from GPS traces. In *Proceedings of RTSS '10*, pages 281 –291. IEEE, December 2010.

[5] A. R. Beresford, A. Rice, N. Skehin, and R. Sohan. Mockdroid: Trading privacy for application functionality on smartphones. In *Proceedings of HotMobile '11*, pages 49–54, New York, NY, USA, 2011. ACM.

[6] C. Bettini, X. Wang, and S. Jajodia. Protecting privacy against location-based personal identification. *Secure Data Management*, pages 185–199, 2005.

[7] T. Book, A. Pridgen, and D. S. Wallach. Longitudinal analysis of android ad library permissions. In *Mobile Security Technologies (MoST '13)*, San Francisco, CA, May 2013.

[8] J. Brickell and V. Shmatikov. The cost of privacy: Destruction of data-mining utility in anonymized data publishing. In *Proceedings of KDD '08*, pages 70–78, New York, NY, USA, 2008. ACM.

[9] Y.-A. de Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel. Unique in the crowd: The privacy bounds of human mobility. *Sci. Rep.*, 3, Mar 2013.

[10] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In *Proceedings of OSDI '10*, pages 1–6, Berkeley, CA, USA, 2010. USENIX Association.

[11] J. Freudiger, M. Manshaei, J.-P. Hubaux, and D. Parkes. Non-cooperative location privacy. *IEEE TDSC*, 10(2):84–98, March 2013.

[12] B. Gedik and L. Liu. Protecting location privacy with personalized k-anonymity: Architecture and algorithms. *IEEE TMC*, 7(1):1–18, January 2008.

[13] P. Golle and K. Partridge. On the anonymity of home/work location pairs. 5538:390–397, 2009. 10.1007/978-3-642-01516-8_26.

[14] M. C. González, C. A. Hidalgo, and A.-L. Barabási. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, June 2008.

[15] M. C. Grace, W. Zhou, X. Jiang, and A.-R. Sadeghi. Unsafe exposure analysis of mobile in-app advertisements. In *Proceedings of WISEC '12*, pages 101–112, New York, NY, USA, 2012. ACM.

[16] S. Guha, M. Jain, and V. N. Padmanabhan. Koi: A location-privacy platform for smartphone apps. In *Proceedings of NSDI'12*, pages 14–14, Berkeley, CA, USA, 2012. USENIX Association.

[17] B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady. Achieving guaranteed anonymity in gps traces via uncertainty-aware path cloaking. *IEEE Transcations on Mobile Computing*, 9(8):1089–1107, August 2010.

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

[19] J. Krumm. Inference attacks on location tracks. In *In Proceedings of the Fifth International Conference on Pervasive Computing (Pervasive), volume 4480 of LNCS*, pages 127–143. Springer-Verlag, 2007.

[20] J. Krumm. Realistic driving trips for location privacy. In *Proceedings of Pervasive '09*, pages 25–41, Berlin, Heidelberg, 2009. Springer-Verlag.

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

[22] B. Livshits and J. Jung. Automatic mediation of privacy-sensitive resource access in smartphone applications. In *Proceedings of USENIX Security '13*, pages 113–130, Berkeley, CA, USA, 2013. USENIX Association.

[23] H. Lu, C. S. Jensen, and M. L. Yiu. PAD: privacy-area aware, dummy-based location privacy in mobile services. In *Proceedings of MobiDE '08*, pages 16–23, New York, NY, USA, 2008. ACM.

[24] J. Meyerowitz and R. R. Choudhury. Realtime location privacy via mobility prediction: Creating confusion at crossroads. In *HotMobile*, 2009.

[25] J. Meyerowitz and R. Roy Choudhury. Hiding stars with fireworks: location privacy through camouflage. In *Proceedings of MobiCom '09*, pages 345–356, New York, NY, USA, 2009. ACM.

[26] K. Micinski, P. Phelps, and J. S. Foster. An Empirical Study of Location Truncation on Android. In *Mobile Security Technologies (MoST '13)*, San Francisco, CA, May 2013.

[27] Microsoft Trustworthy Computing. Location based services and privacy. http://www.microsoft.com/en-us/download/confirmation.aspx?id=3250, January 2011.

[28] A. Nandugudi, A. Maiti, T. Ki, F. Bulut, M. Demirbas, T. Kosar, C. Qiao, S. Y. Ko, and G. Challen. PhoneLab: A large programmable smartphone testbed. In *Proceedings of SENSEMINE '13*, pages 4:1–4:6, New York, NY, USA, 2013. ACM.

[29] B. Palanisamy and L. Liu. Mobimix: Protecting location privacy with mix-zones over road networks. In *Proceedings of ICDE '11*, pages 494 –505, april 2011.

[30] P. Pearce, A. P. Felt, G. Nunez, and D. Wagner. Addroid: Privilege separation for applications and advertisers in android. In *Proceedings of ASIACCS '12*, pages 71–72, New York, NY, USA, 2012. ACM.

[31] PlaceMask. Placemask location privacy, May 2014.

[32] K. Puttaswamy, S. Wang, T. Steinbauer, D. Agrawal, A. El Abbadi, C. Kruegel, and B. Zhao. Preserving location privacy in geosocial applications. *IEEE TMC*, 13(1):159–173, Jan 2014.

[33] rovo89. Xposed module repository, May 2014.

[34] C. Shepard, A. Rahmati, C. Tossell, L. Zhong, and P. Kortum. Livelab: Measuring wireless networks and smartphone users in the field. In *HotMetrics*, 2010.

[35] K. Shin, X. Ju, Z. Chen, and X. Hu. Privacy protection for users of location-based services. *Wireless Communications, IEEE*, 19(1):30 –39, february 2012.

[36] R. Shokri, G. Theodorakopoulos, G. Danezis, J.-P. Hubaux, and J.-Y. Le Boudec. Quantifying location privacy: the case of sporadic location exposure. In *Proceedings of PETS '11*, pages 57–76, Berlin, Heidelberg, 2011. Springer-Verlag.

[37] R. Shokri, G. Theodorakopoulos, J. Le Boudec, and J. Hubaux. Quantifying location privacy. In *IEEE Symposium on Security and Privacy (SP),2011*, pages 247 –262, May 2011.

[38] R. Stevens, C. Gibler, J. Crussell, J. Erickson, and H. Chen. Investigating user privacy in android ad libraries. In *Mobile Security Technologies (MoST '12)*, May 2012.

[39] U.S. Census Bureau. US Census Bureau 2010 Census Interactive Population Map. http://www.census.gov/2010census/popmap/, 2014.

[40] N. Vratonjic, K. Huguenin, V. Bindschaedler, and J.-P. Hubaux. How others compromise your location privacy: The case of shared public ips at hotspots. In E. Cristofaro and M. Wright, editors, *Privacy Enhancing Technologies*, volume 7981 of *Lecture Notes in Computer Science*, pages 123–142. Springer Berlin Heidelberg, 2013.

[41] R. Xu, H. Saïdi, and R. Anderson. Aurasium: Practical policy enforcement for android applications. In *Proceedings of USENIX Security '12*, pages 27–27, Berkeley, CA, USA, 2012. USENIX Association.

[42] T.-H. You, W.-C. Peng, and W.-C. Lee. Protecting moving trajectories with dummies. In *Mobile Data Management, 2007 International Conference on*, pages 278 –282, may 2007.

[43] H. Zang and J. Bolot. Anonymization of location data does not work: a large-scale measurement study. In *Proceedings of MobiCom '11*, pages 145–156, New York, NY, USA, 2011. ACM.