Protecting Smartphone Users' Private Locations through Caching*

Asma Patel and Esther Palomar

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

Abstract. Smartphones equipped with advanced positioning technology continuously collect users' location information and make that information easily accessible to third party app and/or library developers. Users are becoming increasingly aware of the resultant privacy threats, and demanding effective privacy preserving solutions that will allow them to securely use location-based services. In addition, academic and industrial communities are paying special attention to the development of more friendly and socially-accepted approaches to location privacy. In this work, we model, design and evaluate LP-Caché, a mobile platform based service that protects locations by modifying the location resource handling process. It applies caching technique to protect users' private locations and establishes personalised location permission controls. We define the design decisions and implementation requirements towards the viability and feasibility of the model deployment. We also evaluate resources and storage requirements in order to minimise the computational and communication overheads. Empirical results of 2 months comparative study show a 2.26% change in the network fingerprints at 34 distinct places that required only 2.07% change in the overall cache storage. Both these results demonstrate feasibility of the model.

Keywords: Location Privacy, Location-based Services, Smartphones, Caching, Location-based Applications

1 Introduction

The explosive growth of context-aware mobile apps has leveraged tremendous opportunities for a whole new class of Location-Based Services (LBS) [32]. Geomarketing and geo-social networking, location-based games, monitoring, assisted eHealth, and energy consumption 3D maps represent a small subset of the third-party apps nowadays available as LBS and can certainly pose a serious threat to the users' privacy [26, 33].

* Updated and extended version of SECRYPT 2016 conference paper with title "LP-Caché: Privacy-aware Cache Model for Location-based Apps."

Currently, approaches to privacy settings of user location on smartphones[1] are based on a binary process[2]. Users are forced to rely on third party service providers that in many cases continuously collect, use and share their location data, and in some cases even prompt the user to give away their position on page load [2, 26, 16, 33]. Moreover, both academia and industry agree on the urgent need of adopting a Privacy-by-Design (PbD) approach for the development of more user-friendly and socially-accepted solutions to location privacy preservation on their mobile products and services [9].

To encounter these challenges, in [31] the authors introduced the model called *Location Privacy Caché* (LP-Caché). LP-Caché envisions beyond the simple grant/deny access method and provides the user with advanced mechanisms to decide the extent of disclosing location data with service providers. Several caching based solutions [40, 3, 29] have been proposed to minimise the risk of major location privacy threats, but lacking of deployment feasibility. They rely on unrealistic assumptions such as vast cache data storage requirements, or on the app developers modifying the code to incorporate their cached databases. LP-Caché incorporates caching technique to determine users' geographical location in a privacy preserving manner, and with minimum cache storage requirements.

In this paper we overview the main contributions presented in [31] and, further prove LP-Caché's features in an extended experimental setting. In particular, we describe

- A detailed analysis of the current location computation process deployed in smartphones when running location-based apps.
- A detailed definition of the LP-Caché model and architecture as well as its main implementation requirements.
- A complete performance evaluation of LP-Caché, analising the wireless access point data availability and consistency, and the estimated user resource and storage requirements. We will also show that LP-Caché is feasible without modifying installed apps. Estimated storage requirements and monthly datasets of wireless acess points have been analysed. Results from the extended experimental setting help us to determine the scalability of LP-Caché.

The rest of the paper is organized as follows. Section 2 outlines the current location computation process and its evaluation. Section 3 reviews the related work. Section 4 presents the design and architecture of LP-Caché, and Section 5 fully elaborates on design decisions and implementation requirements. We evaluate the feasibility of WiFi APs availability, resources and storage requirements in Section 6. Finally, Section 7 concludes and describes current work as well as sets future research plans.

[1] Throughout this paper, we use the terms smartphones and mobile interchangeably

[2] Data protection directives and acts [14, 20] across the globe state that personal data should not be disclosed or shared with third parties without consent from subject(s). Such a consent is typically obtained by mandatory acceptance of the conditions mentioned in the End User License Agreement (EULA), or through opt-out possibilities and other regulations[25].

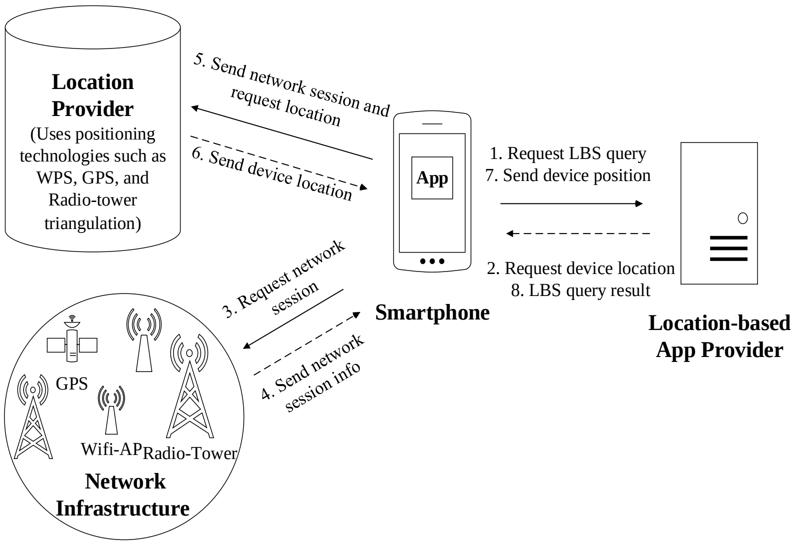

Fig. 1. Current location computation architecture

2 Overview of the Current Location Computation Process

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

2.1 Current Architecture

The current location computation architecture to use location-based apps on smartphones comprises four main entities: 1. Smartphones with installed apps, 2. App Provider, 3. Network Infrastructure, and 4. Location Provider. This architecture (Figure 1) mainly relies on third party location providers, e.g., Google Location Service [18], Skyhook [34], and Navizon [27]. The location provider represents the central database, which maps the received signatures of nearby wireless access points to the geo-coordinates, i.e., latitude and longitude, so handling every geo-location request. Therefore, the location provider has constant access to the user's location as well as to the trajectory data. To respond to any location request, the location provider maintains a database of surrounding network infrastructure, including WiFi Access Points (APs), cellular-towers, and IP addresses, which must be mapped to their exact geographical co-ordinates. Compared to GPS and cell-tower based positioning, WiFi Positioning Systems (WPS) is nowadays considered as a very accurate method for location calculation [34]. Location providers rather use enhanced WPS than GPS, primarily due to current smart-mobile devices benefit from built-in WiFi clients that perform faster than most expensive GPS receivers. This enables the service provider to get user's precise location at all times and, as a result, more effective privacy preservation measures are needed in the current process to mitigate privacy threats.

WiFi APs continuously announce their existence in the way of network frames/beacons and transmit their Service Set Identifier (SSID) and Basic Service Set Identifier (BSSID)/MAC addresses. Location providers use these WiFi APs identifers to create network signatures and map them with geo-coordinates, also called geolocation. IEEE 802.11 states two standardised ways to collect beacons from WiFi APs: 1. Active scanning, and 2. Passive scanning. Location providers are capable of deploying systems with either active scanning, passive scanning, or both together. Location providers use three different ways to collect geo-location of WiFi APs:

1. *Statically*- They collect WiFi beacons by the so called *wardiving* process. Basically, they map the equipped vehicle's exact geo-coordinates along with the signal strength of the captured beacons from surrounded APs.
2. *Dynamically*- They can collect data from WiFi APs automatically once the user device uses location services, e.g. Maps and Navigation applications. The user device as configured to be geolocated acquires unique identifiers from the surrounding WiFi APs, even if the network is encrypted, and then sends it over to the location provider in order to perform geolocation calculation. The collected information is utilised to build and update the database autonomously, for example, by applying crowdsourcing [41].
3. *User input*- They encourage users to manually input the WiFi APs' information, i.e., BSSID and the geo-coordinates, into their databases, e.g., Skyhook[3] to register WiFi APs.

2.2 Evaluation of Current Location Computation Process

We conducted a series of experiments on different mobile devices installed with *Android*, *Windows Phone*, and *iOS* operating systems to categorise the data flow in the current location computation process. With the assistance of sniffers, such as *Wireshark* [39] and *tPacketCapture* [36], we captured and analysed sequence and location data transmission when using location-based apps, e.g., Navigation and Friend Finder.

Observation. These experiments were designed to understand whether there is any difference on the location calculation process on each of these three mobile operating systems. Based on the results, all of them display common patterns of location data retrieval. The user device collects the unique identifiers from the surrounding network along with GPS data, and sends it to the location

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

```
"wifiAccessPoints": [
  {
   "macAddress": "01:23:45:67:89:XY",
   "signalStrength": 10,
   "age": 0,
   "signalToNoiseRatio": -65,
   "channel": 8
  },
  {
   "macAddress": "01:23:45:67:89:YZ",
   "signalStrength": 5,
   "age": 0
  }
 ]
```


(a)


```
"CellTowers": [
  {
    "cellId": 42,
    "locationAreaCode": 415,
    "mobileCountryCode": 310,
    "mobileNetworkCode": 410,
    "age": 0,
    "signalStrength": -60,
    "timingAdvance": 15
  }
]
```


(b)

Fig. 2. Structure of (a) WiFi AP object and (b) cell-tower object sent to the location provider [31].


```
{
  "location": {
    "lat": 58.0,
    "lng": -0.123
  },
  "accuracy": 1200.4
}
```


Fig. 3. Structure of the location object received from the location provider [31].

provider to get the exact device location. Figure 2 shows the structure of the WiFi and Cell-tower objects sent to the location provider. Once calculation is performed, the location provider sends to the device the precise location in the way of a geo-location object containing geo-coordinates. Figure 3 represents the structure of the location object received from the location provider. In short, the app developer over any mobile platform can utilize this location object to get the user's geo-location with no need of focusing on the details of the underlying location technology. In the following section, we give the detailed description of the process sequence.

Process Sequence. Figure 4 illustrates the sequence of processes and messages involved in the current location computation architecture. Note that, on a smartphone, location sharing service settings must be 'ON' while using any location-based app. If the location sharing is 'OFF', then the device prompts for changing the setting from 'OFF' to 'ON'; otherwise, user cannot use the service. Once the app obtains the location object from OS, it is then used by the app provider to send the corresponding reply to LBS query via the standard programming interface/API [5].

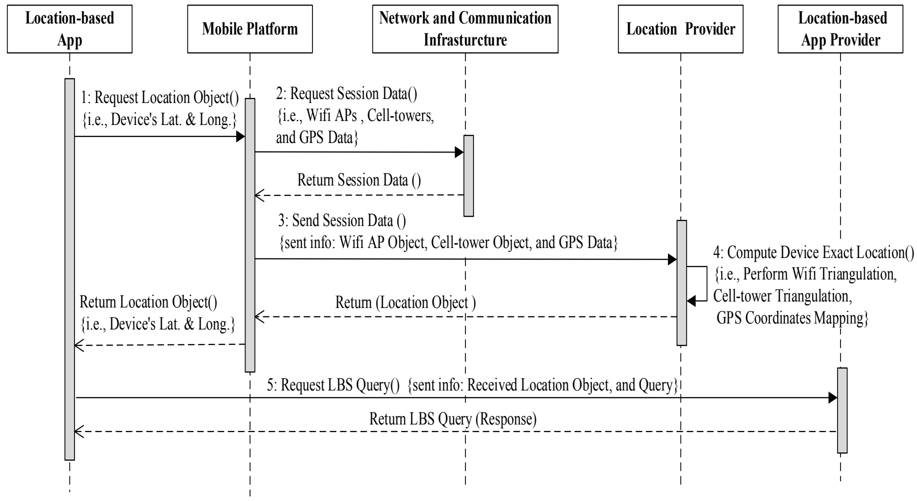

Fig. 4. Sequence diagram of current location computation process [31].

3 Related Work

Existing approaches to preservation of the location privacy can be classified into three categories: 1. Mobile Platform[4], 2. Location Query, and 3. Privacy-aware Network Communication.

3.1 Mobile Platform

A few studies have proposed static and dynamic methods to detect privacy leaks in mobile platforms. The former method statistically analyses apps by creating permission mapping, generating call graphs and data flow analysis to report privacy leaks for further auditing, e.g, AndroidLeaks [17] and PiOS [12] for Android and Apple iOS, respectively. The application of dynamic methods involves modification of the existing mobile platform. For example, TaintDroid [13] adds taint tracking information to sensitive sources calls from apps, and it tracks location data flow as it generated through applications during execution. MockDroid [8] relies on instrumenting `Android's` manifest permission mechanism to mock sensitive data from OS resource, including location data, which can affect apps' usability and functionality. LP-Caché not only monitors the location sources but also modifies, if required, the generated location data based on defined user permissions. In another attempt[15], *indistinguishability* technique is applied as location privacy preservation mechanism into the advertising and analytics libraries as well as on installed apps; however, it does not give control

[4] Throughout this paper, we use the terms mobile platform and operating system interchangeably

on the amount of WiFi and location data that is being shared with the location provider. Moreover, *indistinguishability* technique increases computational overhead on smartphones.

3.2 Location Query

Apps share location information with the provider in the form of LBS queries. The transmission of such queries to the location server may allow attackers to gain access to user location data. Privacy Enhancing Techniques (PET) like k-anonymity, dummy locations, region cloaking, location perturbation and obfuscation, and mix-zone pseudonyms have been applied to different architectures for location query formation and privacy preservation from LBS providers [30, 37, 22]. Most of these techniques rely on theoretical assumptions - like trusted infrastructure to provide the privacy protection, requiring a group of similar app users to be at the same time and same place. The main issue with PETs and cryptographic schemes is that it relies entirely on the data collection servers to comply with location privacy.

Caching. Several authors have used caching scheme along with PETs to build to a database consisting of different contents/datatypes used within location based queries to be re-used in future LBS queries. MobiCaché [40] applies k-anonymity for caching location based queries. Similarly, Niu et al. [29] attempt to improve k-anonymity based caching by adding dummy locations. Both proposals require a trusted infrastructure to maintain privacy. Caché [3] maintains a local cache within the device to reuse the data types available from applications in future location based queries; however, storing entire LBS query data increases the cache storage requirements. Besides, Caché also requires app developer to modify the way app access location data. By contrast, LP-Caché caches the network fingerprints and geo-coordinates, which reduces the storage overhead drastically; it considers installed apps as black box, and therefore, does not require app developer to modify the code, it works as a middleware between the app and the mobile platform. All these cache-based systems either intent to generalise or obfuscate the LBS query or minimise the number of queries sent to the app providers, but they do not provide privacy from WiFi content distributors. Besides, mobile devices not only send vast amounts of location data to app providers but also to location providers creating different location privacy shortcomings [2, 33]. In this regard, limited work has been published on privacy preservation from the location provider's perspective [10, 11]. Damiani (2011) proposes a theoretical approach for privacy-aware geolocation-based web services to encourage further research to minimise the amount of location data being shared with the location provider. This is mainly due to that the location provider is considered as the only source to get the user location when developing any location-based app. In LP-Caché, we minimise the process of wireless AP data collection by the WiFi content distributors or location providers, and we control information disclosure within the generated LBS query (e.g., points of interests (POIs) and nearest neighbor) since it will be sent to the third-party app provider.

3.3 Privacy-aware Network Communication

Besides location queries, device's IP address can also reveal user's private locations. To this regard, anonymous communication protocols, e.g., Anonymizer [6] and TOR [35], deal with anonymous service usage at the network layer while communicating over Internet (i.e., the server cannot infer user's location via received device's IP address along with the location query), and they are most prominent and commonly used network layer solutions.

4 LP-Caché Model

In this section, we describe LP-Caché's threat model, design goals, architecture and main processes' sequence diagram.

4.1 Threat Model

Apps deliberately collect user's sensitive data, including location and other sensitive information as part of their operations. User tracking, identification and profiling (i.e. personal habits, movement patterns, etc.) are fundamental threats to location privacy [16, 37]. Furthermore, the current direct link of smartphones to the location provider and the continuous flow of LBS queries that include device's exact geo-coordinates over network create a serious risk to the protection of users' sensitive information, even more challenging, in the presence of a malicious location provider and via advanced network sniffing practices.

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

4.2 LP-Caché Control Flow Architecture

LP-Caché's three main design goal are: 1) the third-party app provider will not be able to infer the device's exact location without getting uses's consent; 2) the user can set distinct privacy preferences for different apps and private places; and 3) the model works independently without the need of modifying the app's code. Figure 5 depicts the block diagram for LP-Caché architecture; its main components are:

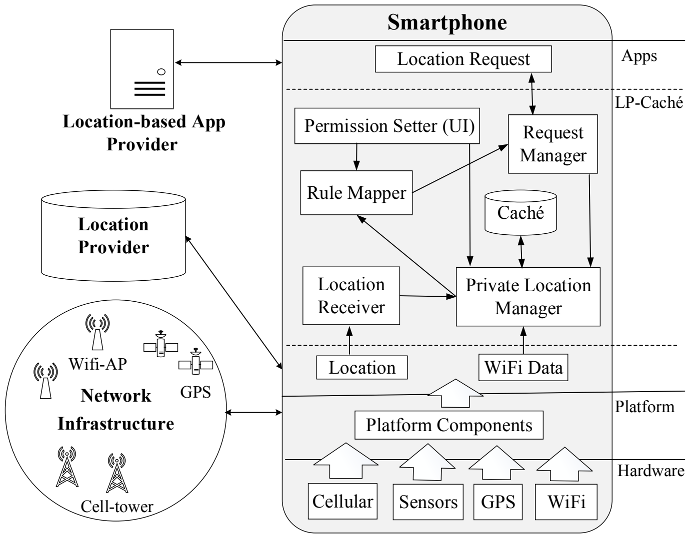

Fig. 5. LP-Caché architecture [31].

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

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

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

Rule Mapper dynamically collects and checks set permissions from *Permission Setter*. Once the representative location object is received from the *Private Location Manager*, it applies the user permissions on the location co-ordinates, alters them (if required), and outputs the processed location

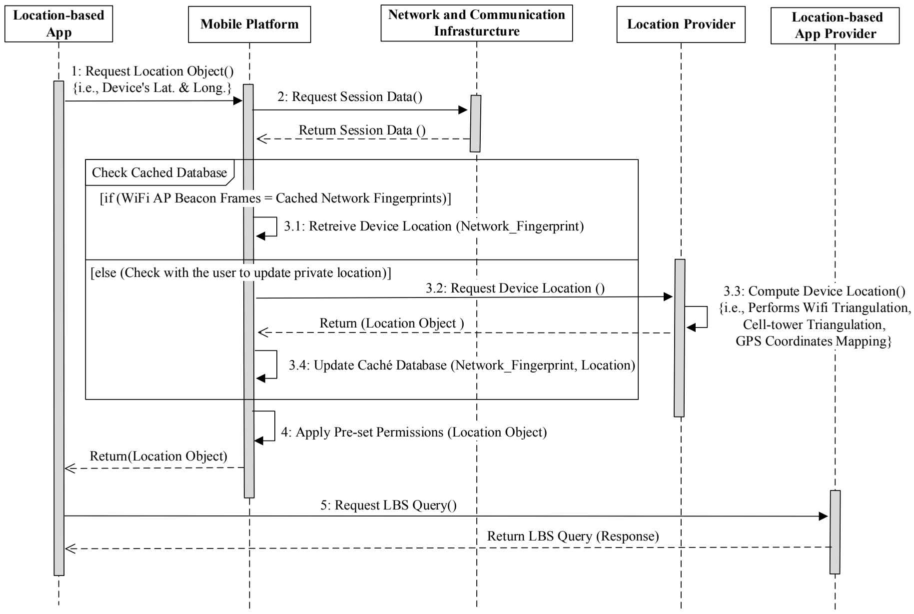

Fig. 6. Sequence diagram of location computation process using proposed model [31].

to the *Request Manager* module. If the flag is negative, then it forwards the exact location.

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

Location Receiver module receives a location object, which includes the user device's geo-coordinates (as in Figure 3), from location providers and sends it over to the *Private Location Manager* for further processing.

4.3 Process Sequence

LP-Caché modifies the current location resource handling process; however, the involved entities (as in Section 2) remain the same. Figure 6 illustrates the sequence of processes and messages involved in LP-Caché:

1. At the event of app requesting the device location, our service will intercept the request to get the location from the cache database instead of sending the request to the location provider.
2. Upon receiving the location request, our service will scan the surrounding network infrastructure.

3. Using observed network frames our service will execute as follows:
   (a) Our service compares the collected beacons with the stored network fingerprints to retrieve corresponding stored representative location coordinates.
   (b) In the case of an unmatched entry on the database, the LP-Caché prompts the user two options either input the location using UI, or allow the query to be sent to location provider that will calculate and send the current location coordinates. Note that this will only occur if the user has set the current location as private but the geo-coordinates are not cached.
   (c) The received location data for the encountered APs will be tracked within the local cache database for future use.
4. User location coordinates can be altered based on the privacy settings. LP-Caché provides three options for controlled information disclosure: (1) Adjust Location Granularity, (2) Obfuscate Location, and (3) No Change. Computed location is populated in the location object and sent to the app.
5. Once the app obtains the location object, it is then used by the app provider to send the corresponding reply to LBS query via the standard programming interface/API [5].

5 LP-Caché Implementation Requirements

In the following sections, we describe LP-Caché's implementation requirements. LP-Caché orchestrates a mobile platform based location protection service to modify the location resource handling process. For instrumenting the LP-Caché implementation, `Android` will be the best choice since it is open source; however, it can also be implemented on other permission-based mobile platforms.

5.1 Bootstrapping

LP-Caché aims to protect user's private places. Initially, LP-Caché does not have enough information to function, the two main required information are private places's network fingerprints and geo-coordinates. LP-Caché cannot collect network fingerprints and geo-cordinates for private places at runtime, as by the time we have this information, other installed apps will have access to it. Therefore, when LP-Caché first boots and before turning 'ON' location sharing settings, user will have to do the initial setup, which includes allow WiFi AP scanning, input geo-cordinates and set privacy choices (see Section 5.4). In 2013, Google presented a new service API (also works on older `Android` versions) for location-based apps that allows developers to use the new and advanced *Location and Activity API*, i.e., they changed `Location Manager` to `Fused Location Manager`, hence combining sensors, GPS, Wi-Fi, and cellular data into a single API for location-based applications [19]. As a result, separating data from `GPS_PROVIDER` and `NETWORK_PROVIDER` is no longer straight forward. LP-Caché addresses this issue by preventing app's location request to reach the `Fused Location Manager` that collects and sends the network session data to the location provider. Instead, the requested location is retrieved from the on-device cache, and then, it is sent to the requesting app (with privacy rules applied). Besides, geographic tool[5] can be incorporated in the LP-Cache's UI to get the corresponding geo-cordinates. This will allow LP-Caché to achieve effective privacy without affecting location accuracy, at the same time, prevent from the non-authorised sharing of device's exact location and network session data.

5.2 Mobile Platform

For performance evaluation, there are two possible ways of implementing LP-Caché location protection service. The first requires modifying the app's location accessing interfaces and intercepting location updates before they reach the app provider. Whereas, the second option requires modifying the platform and changing the location data before reaching the app.

App Code Modification. This comprises unpacking the app, rewriting the code to work according to the new rules, and then repackaging it again, e.g.,[21]. However, app repackaging changes the signature and stops future updates, and therefore, affects its functionality. Another way to modify app's location accessing interfaces is through the creation of an `Android` service and allowing apps to register with it. Then, Apps can use the location data provided by this service. This approach is easy to implement but relies heavily on app developers to modify their app's code, which is highly infeasible and unrealistic. Nonetheless, this approach can be used as simulated testing environment for any developed service.

Platform Modification. For the sake of experimentation, we develop LP-Caché via platform modification. One of the main task is to add a system service, where the class belongs to the location APIs; thus, it is placed in the `android.location package`, which detects private locations via APs and can also be used by other components when calling context. In Android, a context allows an app to interact with the OS resources. Another task is to make LP-Caché communicate with location requesting apps. On `Android` there are two methods to access user's location: 1) Location Manager Service (Old), and 2) Fused Location Manager Service (New) that is a part of Google Play Services. Both methods require the app to request a callback function to get regular updates by registering a location listener. The app receives a new location object when a new location is available, the callback function is invoked. Modifying these two Google services is complicated, but there is a possibility to intercept the location object before it reaches requesting apps. We will add a static context field to the location class, which will be populated when the app is invoked; this will enable us to know which app is currently requesting the location object, and also communicate with the OS [15]. The created location object will have reference to the requesting app's context, and therefore, it can interact with our external service.

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

Algorithm 1 Location Calculation Algorithm [31]

Require: n_x: Network Frames
Ensure: l_r: Representative Location

$n_x = 0$
read n_x
while $n_x \neq$ null **do**
    if $n_x = n_i$, $\forall\, i \in p$ **then**
        (step 1) retrieve the corresponding l_r
        add flag $f =$ (if private 1, else 0)
        send l_r
    else
        (step 2) request l_r from user or location provider
        set received l_r to corresponding p_i
        update c
        send l_r
    end if
end while

5.3 On-device Cache Database Creation

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

Network Fingerprinting. We can distinguish two main types of WPS techniques to determine the position of client devices with respect to APs [7]: 1) Signal trilateration and 2) Fingerprinting. The former undertakes trilateration of Received Signal Strength (RSS), Angle of Arrival (AoA), and Time of Flight (ToF) from observed APs, and the later involves mapping observed APs signatures with a stored database. LP-Caché uses fingerprinting to create cached location database; however, fingerprinting performance is highly related to the number of APs. Therefore, in Section 6 we have evaluated WiFi AP availability and consistency. The detected network management frames/beacons are mapped with the device's representative geo-location to create a network fingerprint, which is then stored in the local cached database, an example private location fingerprint is shown in Equation 1. Moreover, to reduce storage and computation overhead, our model only caches network fingerprints of private places (e.g., home, work, frequently visited places or particular stores), and it relies on user input for initial pre-set up. The user will have to select the option (via LP-Caché UI) to set current location as private place p_i, and then fingerprint will be recorded. Later, the private place will be detected automatically with respect to observed beacons $[n_x]$, such that

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

where p_i represent ith private place IDs, n is the scanned beacon, and l_r is a representative location for that private place. WiFi AP beacons $[n_x]$ consists of

Algorithm 2 Enhanced Permissions Algorithm [31]

Require: l_r: Representative Location
Ensure: l'_r: Processed Location
u_p = User Input
read l, l_g, f, u_p
if u_p = Adjust Granularity **then**
  check granularity level g_l
  truncate(l, l_g)
  replace l to l' and l_g to l'_g
  return l'_r
else if u_p = Obfuscate **then**
  randomly generate angle θ
  obfuscate(l, l_g, θ)
  replace l to l' and l_g to l'_g
  return l'_r
else
  unchanged
  return l'_r
end if

four attributes ⟨SSID, BSSID/MAC address, Signal-strength, and Timestamp⟩. The private representative location [l_r] consists of a tuple ⟨*Lattitude*, *Longitude*, and *Accuracy*⟩.

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

$$R_{l_c,x} = \frac{\sum_{i=1}^{n_{l_c}} b_{x,i}}{n_c}, b_{x,i} = \begin{cases} 1 & \text{if beacon } x \text{ found in } i\text{th scan} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $R_{l_c,x}$ is the response rate of beacon x at the current private location l_c and, n_{l_c} is the total scan count since the private place was entered. The detection count of each beacon is maintained to identify the frequently occurring beacons; and therefore, beacons with higher response rate are used to create the network fingerprint for the current private place [l_c]. $R_{l_c,x}$ will be maintained in the LP-Caché database to update the response rate of every detected beacon during a specified time interval spent at private place [l_c].

On-device Cache-based Location Calculation Algorithm. The detailed steps of privacy-aware geo-location calculation process are summarised in *Algorithm* 1. The surrounded beacons [n_x] are scanned and compared to the list of private WiFi fingerprints [n_i] to detect private place [p_i] stored in cached database [c]. Further, the representative [l_r] is altered based on set permissions (see Section 5.4).

5.4 Personalised Permissions for Location Sharing

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

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

Geo-coordinates Truncation. The geographical coordinates are represented by a tuple consisting of $\{latitude : 52°28'59.200"\ N,\ and\ longitude : 1°53'37.0001"\ W\}$, where the last digits specify more accurate geo-location. Geo-coordinate truncation method will enable us to adjust the location precision level, i.e., by removing last digits and rounding the location accuracy from street to city level or even more general. Generally, for any third party reuse, service providers or data collectors assure in the EULA that this method will be applied on the collected data since the truncated coordinates increase the ambiguity level [1]. On the contrary, LP-Caché applies this method on the user device with user's permission in order to minimise the user's sensitive data collection and privacy concerns.

Geo-coordinates Transformation. For privacy preservation, position transformation functions such as scaling, rotation and translation have been used by location data distributors or anonymisers [24, 37]. In LP-Caché, we use geo-coordinate transformation on the device to obfuscate user's private locations. Our service represents the geo-coordinate transformation using scaling and rotation, and denotes its parameters as a tuple $\langle s, \theta, (l, l_g)\rangle$, where s is the scaling factor, θ is the rotation angle, and (l, l_g) are the original coordinates. It applies Equation 3 to generate transformed or obfuscated geo-cordinates (l', l_g'), where angle θ is randomly generated.

$$\begin{aligned} l' &= \theta(s.l) \\ l_g' &= \theta(s.l_g) \end{aligned} \tag{3}$$

Table 1. WiFi measurement dataset comparative summary.

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

6 Feasibility and Usability Analysis

LP-Caché's actual performance evaluation depends on the location-based apps performance. In this section, we analise the WiFi AP data availability and consistency to measure feasibility of WiFi fingerprinting method to be included in LP-Caché's implementation. In [31], we presented a WiFi APs dataset summary for a month; we have extended the sample size for a couple of months and conducted a comparative study of the observations from both, 1st and 2nd month datasets to evaluate the scalability of the WiFi fingerprinting method. For the sake of illustration, we have maintained unique ID and sequence for all the selected 34 private places.

6.1 WiFi APs Availability and Consistency

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

1. *Data collection.* We collected beacons from fixed WiFi APs using `WiEye` [38] and `Network Info II` [28] apps on `Android` smartphones that have 802.11a/b/g/n radio feature so they can operate in both 2.4GHz and 5GHz bands at 34 different private places for a period of two months.
2. *Location categorisation.* App users are more concern about sharing their private locations[2]; therefore, in our analysis, we selected three distinct categorise of private places: 1. Home (i.e., residential place), 2. Work (i.e., commercial place), and 3. Arbitrary (i.e., any frequently visited place) to determine categorical distribution pattern of WiFi APs.
3. *Data analysis.* We collected and statistically analysed the scanned WiFi AP data. Table 1 compiles the included sample size and the measured percentage changes; whereas, Figure 7 shows the relative difference between WiFi APs density, and Figure 8 depicts the relative average accuracy distribution pattern of detected WiFi APs for each category over the 2 months period.

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

Home The results demonstrate that Wifi APs are fixed and frequent and the difference between number of constant beacons and minimum number of similar beacons is comparatively less, and therefore, it achieved highest accuracy

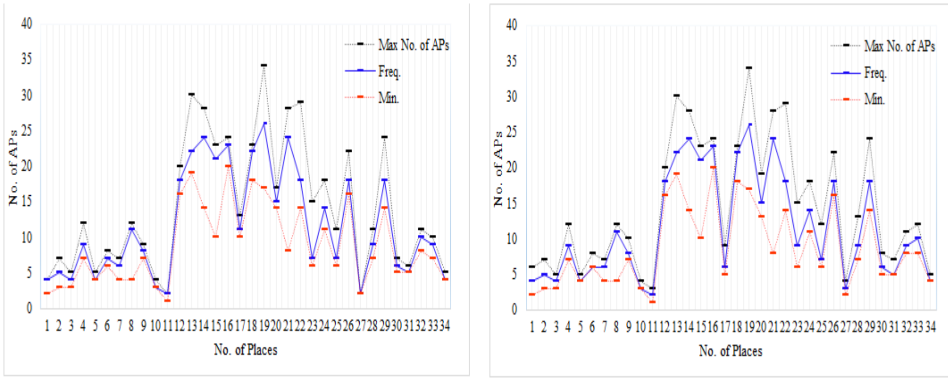

Fig. 7. Measured density of 1 month (left) and 2 months (right) detected WiFi APs at private places.

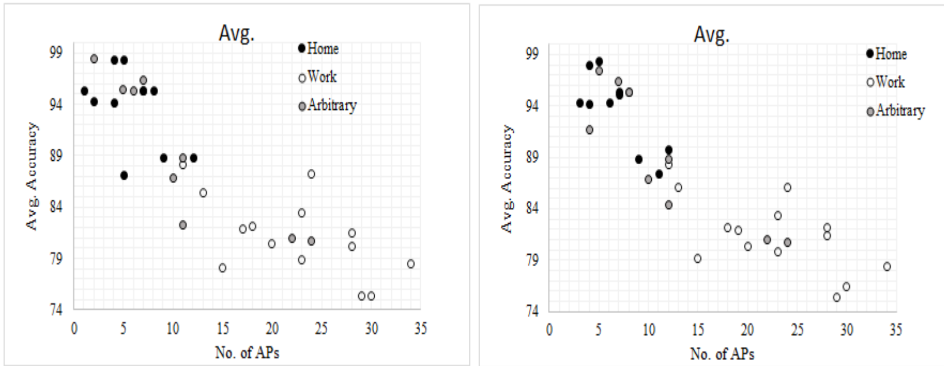

Fig. 8. Relative average accuracy distribution pattern of detected WiFi APs at private places of 1 month (left) and 2 months (right).

rate. Moreover, the ratio of SSID to BSSID is 1:1, i.e., 1 SSID (abc) has 1 BSSID (a0:12:b3:c4:56:78), this makes fingerprints distinct so improving the location detection performance.

Work This category has many fixed WiFi APs but with fluctuating signal strengths, and therefore, the sequence of available APs changes. However, the observed ratio of SSID to BSSID is many to one, i.e., 1 SSID has many BSSIDs; therefore, in this case, SSIDs along with BSSIDs can be used as unique identifiers to create a fingerprint to detect a private place dynamically.

Arbitrary In this category, the data collector could select any frequently visited locations, e.g., gym, shop, or friend's home. Figure 8 demonstrates that the outcome of this category is related to the other two categories, it either shows results similar to home or work.

The range of average accuracy for all the three categories of private places falls between 75% to 97%. Hence, it is evident that smartphones regularly detect similar beacons at frequently visited place, for place detection at least one beacon should match with the stored fingerprints. Thus, the result demonstrates that WiFi fingerprinting can be effectively used as private place detection source in LP-Caché. Nonetheless, to achieve efficient capability for place recognition via beacons, a *place discovering algorithm* like [23] can be implemented (in future work).

Observation 2. Table 1 shows comparative analysis of WiFi APs data that has been scanned and collected during both 1st and 2nd month. Considering percentage changes, the number of total detected APs has increased with 2.26% and the number of frequently detected APs remained without change, i.e, with a negligible difference of -0.76%. Equation 2 has been used statistically to identify frequently detected beacons whilst at a particular private place. Pre-set unique IDs and a sequence for all the selected 34 private places allowed us to measure the relative density and distribution pattern of the WiFi APs during both 1st and 2nd month. Figure 7 shows the relative difference between WiFi APs density, and Figure 8 depicts the relative average accuracy distribution pattern of detected WiFi APs for each category over the period on 2 months.

6.2 Estimated storage requirements

Location-based queries that are generated/received from running applications and service providers include several attributes, and their data types require vast amount of storage space. This is the case of some location privacy solutions (e.g., [3, 40]) that apply caching techniques on location-based queries as a resukt of their storage requirement. LP-Caché does not cache location-based queries, instead it stores the WiFi AP data and geo-coordinates of users' private locations. Moreover, the user's pre-set privacy rules are applied to the mapped geo-coordinates at runtime. As a result, comparatively, LP-Caché's on-device cache database does not demand massive storage requirement. By considering the 802.1 Standard and datatypes sizes, Table 2 presents the storage requirements in bytes and database components, where network fingerprint table is a tuple of $\langle no.ofbeacons, beaconfield, counter\rangle$, and permission table is a tuple of $\langle location, placeid, accuracy\ counter, no.ofprivateplaces\rangle$. Moreover, Table 3 presents the measured changes in the 1st and the 2nd month of WiFi data collected at 34 distinct private places. The results conclude that the average change has increased by 2.07%. cache storage of total 25844 bytes that includes the sum of permissions and network fingerprints for 34 distinct private places. Thus, we can anticipate that the current mobile device internal storage capacity is sufficient for the required storage [4].

Table 2. Estimated storage

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

Table 3. Relative difference of monthly storage

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

6.3 Cache hits and cache misses

In LP-Caché, up-to-date cache database and network fingerprint search result accuracy are main challenges. The three possible outcomes when looking for device's location based on the scanned beacons are:

1. *The location is cached and up-to-date* This case comes with positive result, and therefore, data can be used effectively.
2. *The location is cached but is out-of-date* This can occur if the network infrastructure changes, e.g., if router is changed then the cache data needs to be updated. The overall results of Section 6.1 and Section 6.2 prove that this case does not occur frequently. Nonetheless, for data accuracy a method that uses Equation will be incorporated to detect and measure occurrence of such situations of cache misses at runtime. Moreover, the developed method can likewise be deployed to maintain *data freshness* and *data consistency.*
3. *The location is not cached* This occurs when the observed WiFi AP is not cached and/or mapped to the private locations, then our service will have to interact with user to update the location cache. Besides, the response rate $R_{l_c,x}$ can be further extended to measure runtime occurrences of these outcomes.

6.4 Ongoing Evaluation of Caching Method

Following WiFi data availability and consistency analysis, LP-Caché's feasibility evaluation will be extended to analyse how frequently the cache needs to be updated and what are the trade-offs between the cache update frequency and location privacy and accuracy in order to measure computational and communication overheads. We also intent to conduct a thorough user study to determine usability for the users to accommodate LP-Caché's functionality. Moreover, we plan to extend the fundamental caching-related technical challenges such as cache hits and cache misses, data freshness, data consistency, and estimated bandwidth requirements in an advanced development and implementation of LP-Caché hence paying special attention to storage-efficient caching.

7 Conclusion

Secure gathering and transmission of the location data by mobile apps while preserving users' privacy is a major concern that needs reconsideration. Evaluation of current location handling process confirms that it is vulnerable to location privacy attacks; therefore, we presented a privacy-aware model that provides users with advanced location controls to mitigate major privacy threats. Within a dataset generated in 2 months of experimental setting, we observed a 2.26% change in detected WiFi APs at 34 distinct places and 2.07% change in estimated storage. These results are promising and benefit deployment of LP-Caché's. In this paper, we mainly focused on establishing the design decisions, implementation requirements, and on measuring the feasibility of LP-Caché. Work in progress is on (1) deploying our model on a mobile platform to measure its functionality and efficiency while interacting with different location-based apps; (2) carrying out run-time measurements of the cache storage over an extended period of time, and (3) performing critical analysis of the network fingerprinting and permission mapping methods with dynamic movement patterns. We plan to further assess LP-Caché's scalability in future large scale scenarios and, address end user as well as service providers concerns.

References

1. Aad, I., Niemi, V.: NRC data collection and the privacy by design principles. In: PhoneSense (2010)
2. Almuhimedi, H., Schaub, F., Sadeh, N., Adjerid, I., Acquisti, A., Others: Your Location has been Shared 5,398 Times! A Field Study on Mobile App Privacy Nudging. In: Procs. of ACM Conf. on Human Factors in Computing Systems. pp. 787–796. ACM (2015)
3. Amini, S., Lindqvist, J., Hong, J., Lin, J., Toch, E., Sadeh, N.: Caché: caching location-enhanced content to improve user privacy. In: Procs. of ACM Int. Conf. on Mobile Systems, Applications, and Services. pp. 197–210. ACM (2011)
4. Android: (March 2016), http://developer.android.com/guide/ topics/data/data-storage.html
5. Android Developer Reference: (March 2016), http://developer.android.com/reference/
6. Anonymizer: (March 2016), http://www.anonymizer.com/
7. Bell, S., Jung, W.R., Krishnakumar, V.: WiFi-based enhanced positioning systems: accuracy through mapping, calibration, and classification. In: Procs. of ACM SIGSPATIAL Int. Workshop on Indoor Spatial Awareness. pp. 3–9. ACM (2010)

8. Beresford, A.R., Rice, A., Skehin, N., Sohan, R.: Mockdroid: trading privacy for application functionality on smartphones. In: Procs. of ACM Workshop on Mobile Computing Systems and Applications. pp. 49–54. ACM (2011)
9. Cranor, L.F., Sadeh, N.: A shortage of privacy engineers. IEEE Security & Privacy (2), 77–79 (2013)
10. Damiani, M.L.: Third party geolocation services in LBS: Privacy requirements and research issues. Trans. on Data Privacy 4(2), 55–72 (2011)
11. Doty, N., Wilde, E.: Geolocation privacy and application platforms. In: Procs. of ACM SIGSPATIAL Int. Workshop on Security and Privacy in GIS and LBS. pp. 65–69. ACM (2010)
12. Egele, M., Kruegel, C., Kirda, E., Vigna, G.: PiOS: Detecting Privacy Leaks in iOS Applications. In: NDSS (2011)
13. Enck, W., Gilbert, P., Han, S., Tendulkar, V., Chun, B.G., Others: TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. TOCS 32(2), 5 (2014)
14. European Commission: Protection of personal data (2016), http://ec.europa.eu/justice/data-protection/
15. Fawaz, K., Shin, K.G.: Location Privacy Protection for Smartphone Users. In: Procs. of ACM SIGSAC Conf. on Computer and Communications Security. pp. 239–250. ACM (2014)
16. Felt, A.P., Egelman, S., Wagner, D.: I've got 99 problems, but vibration ain't one: a survey of smartphone users' concerns. In: Procs. of ACM Workshop on Security and Privacy in Smartphones and Mobile Devices. pp. 33–44. ACM (2012)
17. Gibler, C., Crussell, J., Erickson, J., Chen, H.: AndroidLeaks: Automatically detecting potential privacy leaks in Android applications on a large scale. In: TRUST 2012, pp. 291–307. Springer (2012)
18. Google Location Service: (March 2016), https://support .google.com/gmm/answer/1646140?hl=en-GB
19. Hellman, E.: Android programming: Pushing the limits. John Wiley & Sons (2013)
20. IETF: Geographic Location Privacy (March 2016), http://datatracker.ietf.org/wg/geopriv/charter/
21. Jeon, J., Micinski, K.K., Vaughan, J.A., Fogel, A., Reddy: Dr. Android and Mr. Hide: fine-grained permissions in android applications. In: In Procs. of ACM Workshop on Security and Privacy in Smartphones and Mobile Devices. pp. 3–14. ACM (2012)
22. Khoshgozaran, A., Shahabi, C., Shirani-Mehr, H.: Location privacy: going beyond K-anonymity, cloaking and anonymizers. Knowledge and Information Systems 26(3), 435–465 (2011)
23. Kim, D.H., Hightower, J., Govindan, R., Estrin, D.: Discovering semantically meaningful places from pervasive RF-beacons. In: Procs. of ACM Int. Conf. on Ubiquitous Computing. pp. 21–30. ACM (2009)
24. Lin, D., Bertino, E., Cheng, R., Prabhakar, S.: Position transformation: a location privacy protection method for moving objects. In: Procs. of the SIGSPATIAL ACM GIS 2008 Int. Workshop on Security and Privacy in GIS and LBS. pp. 62–71. ACM (2008)
25. Michael, K., Clarke, R.: Location and tracking of mobile devices: Überveillance stalks the streets. Computer Law & Security Review 29(3), 216–228 (2013)
26. Muslukhov, I., Boshmaf, Y., Kuo, C., Lester, J., Beznosov, K.: Understanding users' requirements for data protection in smartphones. In: ICDE Workshop, IEEE Int. Conf. on Secure Data Management on Smartphones and Mobiles. pp. 228–235. IEEE (2012)

27. Navizon: (March 2016), http://www.navizon.com
28. NetworkInfoIi: (March 2016), http://play.google.com/store/apps/
29. Niu, B., Li, Q., Zhu, X., Cao, G., Li, H.: Enhancing Privacy through Caching in Location-Based Services. In: Proc. of IEEE INFOCOM (2015)
30. Patel, A., Palomar, E.: Privacy Preservation in Location-Based Mobile Applications: Research Directions. In: Procs. of IEEE Int. Conf. on Availability, Reliability and Security (ARES). pp. 227–233. IEEE (2014)
31. Patel, A., Palomar, E.: LP-Caché: Privacy-aware cache model for location-based apps. In: Proceedings of the 13th International Joint Conference on e-Business and Telecommunications (ICETE 2016) - Volume 4: SECRYPT, Lisbon, Portugal, July 26-28, 2016. pp. 183–194 (2016)
32. Pontes, T., Vasconcelos, M., Almeida, J., Kumaraguru, P., Almeida, V.: We know where you live: Privacy characterization of foursquare behavior. In: Procs. of ACM Conf. on Ubiquitous Computing. pp. 898–905. ACM (2012)
33. Shklovski, I., Mainwaring, S.D., Skúladóttir, H.H., Others: Leakiness and Creepiness in App Space: Perceptions of Privacy and Mobile App Use. In: Procs. of ACM Conf. on Human factors in computing systems. pp. 2347–2356. ACM (2014)
34. Skyhook: (March 2016), http://www.skyhookwireless.com/
35. TOR: (March 2016), http://www.torproject.org/
36. tPacketcapture: (March 2016), http://play.google.com/
37. Wernke, M., Skvortsov, P., Dürr, F., Rothermel, K.: A classification of location privacy attacks and approaches. Personal and Ubiquitous Computing 18(1), 163–175 (2014)
38. WiEye: (March 2016), http://play.google.com/store/apps/
39. Wireshark: (March 2016), https://www.wireshark.org
40. Zhu, X., Chi, H., Niu, B., Zhang, W., Li, Z., Li, H.: Mobicache: When k-anonymity meets cache. In: GLOBECOM. pp. 820–825. IEEE (2013)
41. Zhuang, Y., Syed, Z., Georgy, J., El-Sheimy, N.: Autonomous smartphone-based WiFi positioning system by using access points localization and crowdsourcing. Pervasive and Mobile Computing (2015)