All Your Location are Belong to Us: Breaking Mobile Social Networks for Automated User Location Tracking

Muyuan Li[†‡], Haojin Zhu[†], Zhaoyu Gao[†], Si Chen[‡], Le Yu[†], Shangqian Hu[†], Kui Ren[‡]
[†]Department of Computer Science and Engineering, Shanghai Jiao Tong University
[‡]Department of Computer Science and Engineering, State University of New York at Buffalo
[†]{zhuhaojin, gaozy1987, lewisyu24, hu.shangq}@gmail.com
[‡]{muyuanli, schen23, kuiren}@buffalo.edu

ABSTRACT

Location-based social networks (LBSNs) feature friend discovery by location proximity that has attracted hundreds of millions of users world-wide. While leading LBSN providers claim the well-protection of their users' location privacy, for the first time we show through real world attacks that these claims do not hold. In our identified attacks, a malicious individual with the capability of no more than a regular LBSN user can easily break most LBSNs by manipulating location information fed to LBSN client apps and running them as location oracles.

We further develop an automated user location tracking system and test it on leading LBSNs including Wechat, Skout, and Momo. We demonstrate its effectiveness and efficiency via a 3 week real-world experiment on 30 volunteers and show that we could geo-locate any target with high accuracy and readily recover his/her top 5 locations. Finally, we also develop a framework that explores a grid reference system and location classifications to mitigate the attacks. Our result serves as a critical security reminder of the current LBSNs pertaining to a vast number of users.

1. INTRODUCTION

Mobile social networks have gained tremendous momentum since recent years due to both the wide proliferation of mobile devices such as smartphones and tablets as well as the ubiquitous availability of network services. Millions of users are enabled to access and interact with each other over online social networks via their mobile devices. Moreover, the positioning technologies such as GPS, and wireless localization techniques for mobile devices have made both the generation and sharing of real-time user location updates readily available. This, in turn, leads to the extreme popularity of location-based social networks (LBSNs) such as Facebook Places, Google Plus, PCube, Foursquare, Wechat, Momo, Badoo, Grindr, Blendr, and Tapmee, which boost up to hundreds of millions of users. As one of the most popular LBSNs in China, Wechat achieved more than 300 million registered user accounts in only two years, and is used in over 200 countries [22]. Another LBSN app Momo has 30 million users, 2.2 million of whom use the app on a daily basis [11, 35]. Skout, a very popular dating app in North America, draws 1.5 million new users a month who check into the app an average of nine times a day [2].

In contrast to traditional LBSNs such as Foursquare, which allow users to check-in at locations and share the information with friends within vicinity, the newer ones feature location-based social discovery. Location-based social discovery explicitly enables on-the-spot connection establishments among users based on their physical proximity. Examples of such LBSNs include Google Plus, PCube, Wechat, Momo and Skout. While services like Google Plus and PCube allow their users to control with whom they want to share the location information, popular ones like Wechat, Momo and Skout allow location-based social discovery solely based on users' physical proximity.

Along with the popularity of location-based social discovery is the increasing danger of user privacy breaches due to location information exposure. Recent studies have shown that four spatiotemporal points are sufficient to uniquely identify the individuals in an anonymized mobility data set [12, 21] and little outside or social network information is needed to re-identify a targeted individual or even discover real identities of users [12, 30, 33]. Furthermore, users' location traces can leak much information about the individuals' habits, interests, activities, and relationships as pointed out in [26]. And loss of location privacy can expose users to unwanted advertisement and location-based spams/scams, cause social reputation or economic damage, and make them victims of blackmail or even physical violence.

Recognizing the danger of user location privacy leakage due to the use of mobile device in general, various research efforts have been devoted to location privacy. Most of them focus on developing the general location privacy protection mechanisms for location-based services (LBSs) that allow users to make use of LBSs while limiting the amount of disclosed sensitive information [8, 7, 13, 23, 28, 31, 27, 14]. Existing techniques include anonymous service uses, cloaking based technique [31], mixzone or silent period [13, 27]. Mechanisms are also proposed to enable proximity testing without revealing the mobile users' real location information [34, 24] for privacy preserving distributed social discovery.

User location privacy in real-world LBSN apps, however, has not received enough attention. Current industrial practices are yet to be scrutinized for their (in)adequacy. Most of regular LBSN users have little (if any) idea of the fact that through the attack strategy proposed in our work, the amount of location privacy leak is beyond what the apps tell them. In fact, there are reports that even falsely reassure users that the location privacy protection is adequate and sound [10].

MobiHoc'14, August 11–14, 2014, Philadelphia, PA, USA.

http://dx.doi.org/10.1145/2632951.2632953.

In this paper, we ask and answer two fundamental questions regarding user privacy in the most popular LBSNs protected by the-state-of-art location hiding techniques. First, is it possible to make an involuntary localization of a random LBSN user by exploiting the public available information only? That is, without hacking into the services and using only the client side information that is publicly available through the unmodified app of LBSNs, could we accurately localize a random online user of no priori knowledge? Secondly, could we freely track a particular user within a reasonably short time period? By investigating three most popular LBSN apps (Wechat, Momo and Skout), our answers to these two questions are more than a simple "yes". Our research findings show that: 1) An attacker could perform a range-free, involuntary user localization attack with high localization accuracy; 2) Furthermore, it can successfully establish very accurate user location profile.

We implement an automated user location tracking system for mobile social networks that could track Wechat, Skout and Momo users without any awareness. To demonstrate its effectiveness, we perform a three-week real-world attack towards 30 volunteers from United States, China and Japan. By comparing the collected users' real traces with the inferred traces, it is found that the mean tracking error is 51m for 74 Wechat tests, 25m for 119 Momo tests, and 130m for 156 Skout tests. What's more, the attacker can easily identify users' Top 5 locations. According to the existing works, more than 50% of the individuals could be uniquely identified given top 2 locations [12]. Hence, the newly identified attacks pose a serious threat towards the locations privacy of hundreds of millions of LBSN users.

The rest of paper is organized as follows: Section 2 is the classification of LBSNs. Section 3 describes our attack methodology, which is followed by Section 4 describing the implementation of the attack. Section 5 presents the evaluation results. In Section 6, the mitigation approaches are discussed and Section 7 summarizes the related work. Finally, Section 8 concludes the paper.

2. LBSN: THE STATE-OF-THE-ART

2.1 Classification of LBSNs

With the wide use of mobile devices, and the increasing attention on mobile social networking, location-based social networks (LBSN) focusing on the small local social network derived from a user's geographical location become increasingly popular. In addition to the conventional location-based user check-in apps (e.g., Foursquare), more LBSN apps are exploiting the users' geographical information to achieve distance-based social discovery and location sharing. Based on how real-world LBSNs share the location information among their users in order to allow location-based social discovery, they can be classified into two main categories: I) LBSNs with Exact Location Sharing and II) LBSNs with Indirect Location Sharing. Table 1 is a summary of our surveyed 20 popular real-world LBSNs.

Category I has two subtypes. The Subtype I is *Open Access Location Sharing*. These LBSNs present the exact locations without any restriction. In Banjo, by clicking "Places" tab, the users are allowed to see people of the same city, the exact location of which are explicitly displayed on a map. Subtype II is *User Authorized Location Sharing*. For this type of LBSNs, users have the control to choose with whom they share their exact location information. For example, in Google Plus or PCube, a user decides the set of other users to share location with. In general, we believe users fully aware of potential risks of those kinds of apps.

In Category II, the exact geographic information is obfuscated by a series of location privacy protection techniques. Different from Category I which reveals users' exact locations, Category II LBSNs assure users that their exact location information is never shared by privacy protection techniques. In our investigated apps, LBSN service providers adopt the following location obfuscating techniques.
I. Relative Distance Only: This has been a very common location hiding technique adopted by many popular LBSNs, including Wechat, Skout, and Momo. Users in this case can only see others' geographic distances instead of location coordinates. From the user's point of view, revealing the distances rather than coordinates could hide the exact location but still allow the nearby strangers (or potential friends) to discover the presence of this user.
II. Setting the Minimum Accuracy Limit: Setting a safe localization accuracy limit is a traditional location obfuscation technique [4]. Most of the LBSN apps predefine a certain minimum accuracy limit for geo-localization to further protect the users' exact location. For example, Skout defines localization accuracy to 1 mile, which means that the users will be located with an accuracy no better than 1 mile. Similarly, Wechat and Momo set 100m and 10m as their localization accuracy limits.
III. Setting the Localization Coverage Limits: To prevent malicious users from abusing the geo-localization, an additional functionality, Localization Coverage limit is provided to restrict the users' localization capability to a specific region or under the maximum number of displayed users. For example, Wechat only displays the relative distance of users, the number of which is less than a predefined threshold (e.g., 1000m in Wechat for a high user density region).

In addition to above mentioned location hiding techniques, there are other factors contributing to the localization errors, which will be presented as follows.

2.2 Location Update in LBSNs

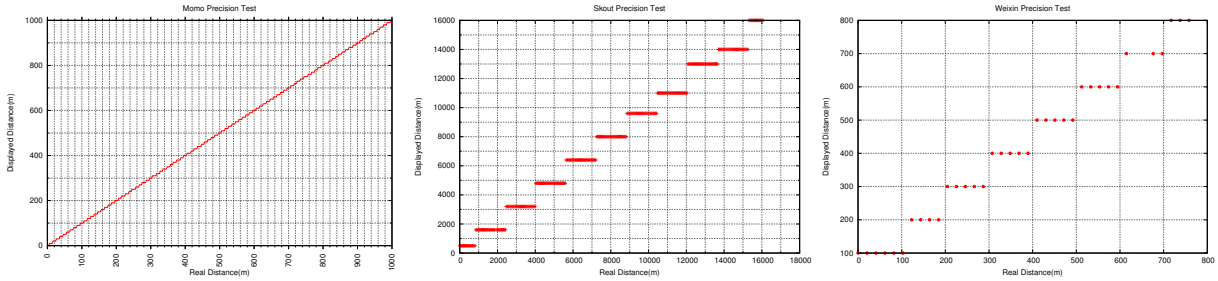
In general, the localization accuracy of smartphone relies on which kind of location data sources it uses. The location data sources (location providers) include: GPS, Wi-Fi, and Cell ID (cell tower), that corresponds to localization accuracy of 10m, 80m and 600m [32]. However, in practice, it's up to the app developers themselves to decide which location source to trust and it is always a trade-off between waiting time, precision and energy consumption[16, 3]. To have a better understanding on the updating strategy of LBSN apps, we perform the following accuracy testing experiments.

We mainly perform the accuracy testing on three apps: Wechat, Skout and Momo. We pre-define a reference point both in the physical world (akin a virtual user located in this position) and the virtual machine. We then enlarge the physical distance between our mobile device and the reference point by sending fake locations to the device and record the relative distance displayed on apps. We compare the physical distance and the distance shown in apps and obtain the accuracy testing results, which are depicted in Fig 1.

Since Momo's localization accuracy limit is set to 10m, we choose a test point for every 2m. From Fig 1a, we confirm 10m as the localization accuracy limit with a rounding every 5m. In Skout, the localization accuracy bound is approximately 0.5mile. In the experiment, we evaluate the localization accuracy every 50m. From Fig 1b, it is observed that Skout's minimum coverage is 0.5mile with distance rounding every half a mile and the distance is increased every 1.6km or 1mile. We set the reference point for every 20 meters for Wechat and have observed that the coverage bound can be up to 10km in sparsely populated area and generally 1000m in densely populated places. It is also observed that Wechat has no round-offs in its distance and the boundary is quite clear between every 100m (Fig 1c).

	Distance	Accuracy Limit	Coverage Limit	Number of Users (millions)	Platform or region	SDK	Category
Wechat	Y	100m	1km (Shanghai)	300 millions	iOS/Android/WP	Google	II
Skout	Y	0.5mile	N/A	5 millions	iOS/Android/WP	Google	II
Momo	Y	10m	N/A	30 millions	iOS/Android/WP	Baidu	II
Whoshere	Y	100m	N/A	5 millions in 2012	iOS/Android	Google	II
MiTalk	Y	100m	0.6km (Shanghai)	20 millions	iOS/Android	Baidu	II
Weibo	Y	100m	1600m	500 millions	iOS/Android/WP	Google	II
SayHi	Y	10m	1000km	500 thousands	iOS/Android	Google	I/II
iAround	Y	10m	N/A	10 millions	iOS/Android	Baidu	I/II
Duimian	Y	100m	N/A	500 thousands	iOS/Android	Google	II
Doudou Friend	Y	10m	N/A	1 million	iOS/Android	Amap	II
U+	Y	10m	N/A	10 millions	iOS/Android	Baidu	II
Topface	Y	100m	N/A	50 million	iOS/Android	Google	II
Niupai	Y	10m	N/A	61 thousands	iOS/Android	Google	II
LOVOO	Y	100m	27.8km (Shanghai)		iOS/Android	Google	II
KKtalk	Y	10m	N/A	320 thousands	iOS/Android	Google	II
Meet24	Y	0.5mile	N/A		iOS/Android	Google	II
Anywhered	Y	10m	N/A	750 thousands	Android	Baidu	II
I Part	Y	10m	1000m	8 millions	iOS/Android	Google	II
Path	N	N/A	N/A	10 millions	iOS/Android	Google	I
TweetCaster	N	N/A	N/A	10 millions	iOS/Android/WP	Google	I
Google Plus	N	N/A	N/A	10 millions	iOS/Android/WP	Google	I
eHarmony	N	N/A	N/A	5 millions	iOS/Android	Google	I
SinglesAroundMe	N	N/A	N/A	1 million	iOS/Android	Google	I

Table 1: Summary of Location-based Friend Discovery Apps



(a) Momo precision test (b) Skout precision test (c) Wechat precision test

Figure 1: Updating Strategy Evaluation Results

2.3 User Location Privacy in LBSNs

From above discussions, we can conclude that, in general, the locations reported in LBSN apps honestly reflect mobile users' real locations, though the exact locations are hidden or obfuscated by various location hiding techniques. For example, Momo only adopts the strategy of showing the relative distances (strategy I). Skout only shows the distance and, at the same time, enforces the minimum localization limit (strategy I & II). As a comparison, Wechat adopts all of location hiding strategies I, II, III.

In this paper, we argue that, relying on the above mentioned location privacy hiding techniques may introduce more dangerous location privacy leaking issues. Due to trust on these location hiding/obfuscating techniques, LBSN users are more willing to share the **protected** location information with the potential adversary, which could recover users' exact location or even traces using the methodology proposed in this paper. Without full knowledge of the potential risk, LBSN users may face the serious location privacy leaking issue, while the adversary could gain a significant advantage during the attack process by making involuntary geo-location or even tracking.

3. ATTACK METHODOLOGY

In this section, we introduce our attacker model, as well as the attack methodology in details.

3.1 Attacker Model

In this study, we consider a capability-restricted attacker aiming at geo-locating an LBSN user, who does not need to have a priori social association with him, i.e., an in-app friend. The attacker's capability is restricted in sense that I) It only has the access right no more than a normal user of a given LBSN service, which means that he can only access the publicly available information provided by the LBSN app. II) It is not allowed to hack the LBSN service by interfering its internal operations, that is, we do not consider an attacker that can compromise the LBSN servers and thus can directly access the user location information as a consequence. In summary, our attacker is a very weak one which can't gain any additional information from the LBSN services other than what is entitled to a regular service user. Specifically, the attacker will try to infer a user's location information based only on the relative distance information displayed by the LBSN apps. Note that, to obtain the relative distance, it is even not necessary for the attacker to be friend of the victim. Instead, it will automatically display the relative distance of nearby users in most of considered Category II apps. For

Momo, the attacker can obtain the distance by searching the victim via Momo ID and in Skout, the distance between the victim user and the attacker can always be displayed as long as the attacker has sent a regular message (a greeting for instance) to the victim before. We are concerned that if the LBSN under examination can't resist even such a weak attacker, the user's location privacy is obviously in a great danger as any user can be an attacker.

We further distinguish two different types of attackers, i.e., a Casual Localization Attacker and a Determined Tracking Attacker. A Casual Localization Attacker reviews the profiles of nearby users when logging in to a LBSN app as a regular user, randomly picking up a tracking target and then try to geo-localize the target. A Determined Tracking Attacker may start with a known User ID (UID) as its chosen attacking target and perform the tracking towards a specific victim for a certain duration. The goal of the tracking attacker is revealing users' Top N locations (e.g., his home or office) [33]. Note that, a tracking attacker may start with a target person in mind and exploit certain side-channel information of the target to help obtain the corresponding UID. For example, user photos shared among various social network sites can be used to establish the link for the same user, which in turn can lead to the acquisition of UID in a particular LBSN. Social engineering approaches like this have been widely studied in the literature and is not a focus of this paper [19]. We assume that a determined tracking attacker is able to start with a chosen UID he wants to locate.

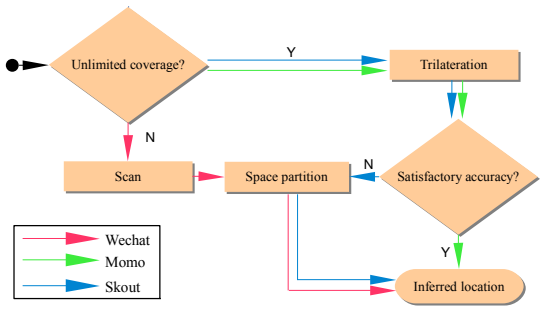


Figure 2: The Attack Flow

3.2 Methodology Overview

The security of the state-of-the-art privacy protection techniques are based on the assumption that the location cannot be faked. Under this assumption, the exact location of the mobile users is hidden/obfuscated by the above mentioned three strategies. Therefore, the intuition behind our attack is that, if the attacker could freely generate the fake anchor points with new locations, LBSN apps will be a distance oracle that always returns the relative distance with these anchor points to the attacker. By exploiting the returned information, the attacker could launch localization algorithms to geo-locate the victim and even break the accuracy limit.

As shown in Fig 2, when the attacker determines a particular victim, it generates three fake anchor locations and obtains the relative distance to the victim. With 3 anchor locations and their corresponding distances, it could trigger the iterative trilateration based localization algorithm and obtain the inferred location. After that, the attacker adopts space partition attack to further improve the accuracy until the distance reaches the predefined accuracy threshold. For those apps with the coverage limit, the attacker can scan the possible locations until the victim is shown in the "nearby list". Then, it takes advantage of space partition attack to make an accurate localization. We introduce each basic algorithm one by one in the following subsections.

3.3 Iterative Trilateration based Localization Algorithm: Skout and Momo

Our localization approach is based on the traditional Trilateration Position Problem. In our long distance tracking, we start from 3 randomly generated positions serving as the first three anchor points. In Section 4.1, we will introduce how to generate the fake locations on Android. The triggered Trilateration algorithm returns the first localization results. To minimize its distance from the target, the least squares solution can be used to solve this problem as suggested in [20]. We iteratively perform trilateration and generate the next reference point from the previous round localization results. We denote P as the list of reference points sorted by the relative distance to the target point from smaller to larger. Without loss of the generality, the first three items of P are represented by p_1, p_2 and p_3. We further define function $dist(a, b)$ to measure the distance between the point a and b, as well as function $Lsp(a, b, c)$ to return the least square estimation of the localization target based on three reference points (a, b, c). We summarize our iterative trilateration localization algorithm in Algorithm 1.

Algorithm 1: The Iterative Trilateration based Localization Algorithm

Data: List $P = \emptyset$, in which the elements are sorted by their distance to targeted node
Result: $p_0 = (x_0, y_0, z_0)$ the location of the target

Put 3 random reference points into P;
$d \leftarrow +\infty$;
while $(d > threshold)$ **do**
  $p_1, p_2, p_3 \leftarrow$ first 3 elements of P;
  $d \leftarrow$ distance between p_1 and target node;
  $t \leftarrow Lsp(p_1, p_2, p_3)$;
  Insert t into P;
end
Output p_1;

3.4 Breaking Minimum Distance Limit via Space Partition Attack: Skout and Wechat

Another best practice measure to protect location privacy is to limit the relative distance to a certain accuracy, (e.g., 800m in Skout or 100m for Wechat). In this section, we propose a space partition attack algorithm to further enhance the localization accuracy and thus breaking the minimum distance limit. The basic idea of space partition attack is similar to space partition algorithm, which is defined as the process of dividing a space (usually a Euclidean space) into two or more non-overlapping regions and thus locating any point in the space to exactly one of the regions. The basic idea of space partition attack is illustrated in Fig 3.

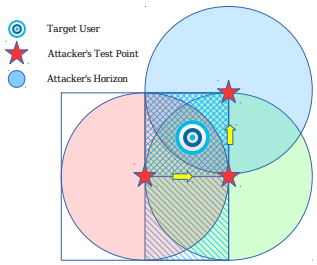


Figure 3: Illustration of Space Partition Attack

For the simplicity of problem presentation, we consider the minimum distance limit as the box rather than the circle. Given the minimum distance limit R, the edge length of the box is set to $2R$. The space partition attack could be illustrated as follows. In each round, the potential area of length r is partitioned into two regions. Then, we will check if it is within one region. If yes, it is derived that the user is in this half. Otherwise, the user is located in the other half. We could repeat this partition for multiple rounds until the expected accuracy is achieved. The whole algorithm is summarized in Algorithm 2.

Algorithm 2: Space Partition Attack Algorithm

Data: An estimated point $p_0 = (c_X, c_Y)$ and its displayed distance from target point T, given in form $dist(p_0, T) \leq R$.
Result: T', the final estimation for T

$dim = X$;
$\delta_X = R$;
$\delta_Y = R$;
while $\delta_X \geq threshold$ *or* $\delta_Y \geq threshold$ **do**
  Shift p_0 in dim dimension by R to p';
  if $dist(p', T) \leq R$ **then**
    c_{dim}=$c_{dim} + \delta_{dim}/2$;
  end
  else
    c_{dim}=$c_{dim} - \delta_{dim}/2$;
  end
  $\delta_{dim} = \delta_{dim}/2$;
  $dim = \{X, Y\}/dim$;
  $p_0 = (c_X, c_Y)$;
end
Output p_0;

3.5 Breaking Localization Coverage Bound by Correlating the Social Popularity and Geolocation: Wechat

Some apps such as Wechat set a certain coverage limit. For such kind of apps, a straightforward approach to launch the attack is to scan a certain area, and, at each sub-area, find out if the attacking target appears in the "Nearby" list of this app. In practice, the performance of such a simple scan approach may not be desirable due to the potentially large region in which the target is located. To further improve the performance of the launched attack, we propose a *Social Popularity Index* based approach to speed up the localization process. This approach is motivated by the existing research that human trajectories are regular in space and time, with each individual having a high probability of returning to a few preferred locations according to Zipf's law [29]. In other words, it is more likely for a user to be at the restaurant at 6 pm in the afternoon rather than at office. However, this statement may not hold at 10 am in the morning. The social popularity index is introduced to represent how popular a location is at a particular time slot, and thus can be used to estimate how likely a user is located at this area at this moment. In our implementations, we measure the social popularity index of different locations by collecting their user population information at different time slots. Then, based on the number of users, we could assign a higher priority to those areas with the higher user population.

In addition to the above described methodology, there are still implementation challenges, including: generating the fake locations in smartphone and fetching relative distance readings from Android LBSN apps to make the above methodology to work realistic. More importantly, all of these should be performed in an automated way. We'll introduce the implementation details in the next section.

4. IMPLEMENTATION

Besides the algorithms introduced in previous section, the system implementation involves two other key modules: the location spoofing module and the location reading module. Our system is implemented in Clojure in order to cope with MonkeyRunner to control Android virtual machines and send commands. We also implement a LocationFaker app that receives HTTP request to set the location in Android. To address problems we encounter during location faking and result reading, we tweak parts of the Android framework.

4.1 Generating Fake Locations on Android

To launch the proposed attack, we need to freely generate the anchor location points, which are used to obtain the relative distances of the victim users. To allow fast real-world automated tracking, we set our Android system on real Android X86 images running in VirtualBox. It is important to point out that, since almost all of the LBSN apps cover all of the platforms (please refer to Table 1 for our survey), including iOS/Android/WP, spoofing the android device's locations could allow the attacker to obtain the relative distances with LBSN users on iOS/WP, and thus launch the attack towards the users on all of platforms.

To enable the fast and reliable location retrieval, we implement our location spoofing component, LocationFaker, as a system service to eliminate the possibility that Android system may kill the activities to release resources, which are not supported by existing apps such as Developer Shell and FakeGPS. Further, it has embedded Jetty as a web server to provide stateless http-based interface to set locations and act as a fake location server when we redirect the network traffic. In general, most of the LBSN apps on Android either use built-in Android API (Wechat or Skout) or third-party SDKs (e.g., Momo using Baidu Location SDK), which lead to different spoofing strategies. For built-in Android API, we adopt *fake location provider based location spoofing*. However, according to the official document [6], Baidu Location SDK does not function well on virtual machines. For this case, we achieve location spoofing by using *network redirection*. In the follows, we introduce both approaches in detail.

4.1.1 Location Spoofing with Fake Location Provider

Android apps mostly acquire locations via one or more location providers (e.g., "gps" and "network") from the *location* system service. Since Android allows users to freely add location providers under certain circumstances such as debugging or providing locations from other devices, e.g. Bluetooth, it is possible to add a user-written location provider by enabling "Allow mock location" option in developer options and adopting the API "addTestProvider". It is interesting that we can set the provider's name to "gps" to make it indistinguishable from the real gps, and thus fool the system into believing that they are receiving locations from the real GPS chip. Our fake location provider is running on its thread, feeding location information every $700ms$.

One of the major challenges of spoofing the location on Android is that the provided location should satisfy certain accuracy. Otherwise, the apps will reject it. During our implementation, it is found that Wechat will return error messages if fake provides set fake locations. After checking with Google Map, we discover such

fake location provider lacks accuracy information. To address this issue, we decompiled the Android framework with ApkTool and modified the constructor of "android.location.Location" by coercing "mHasAccuracy" to "true" and enforcing "getAccuracy" to always return 70m. This enables apps to retrieve consistently accurate value under different circumstances and the location faking component starts to work as expected.

4.1.2 Location Spoofing with Network Redirection

For those LBSN apps which do not adopt Android built-in APIs for location retrieval, we introduce another approach based on network redirection. In this section, we use Momo as an example to show how it works. Basically, Momo uses Baidu Location SDK to obtain the user location. We start from analyzing the network traffic with Wireshark and Tcpdump. It is observed that the API first posts the coordinates and supplemental information, which is obtained from the device, to http://loc.map.baidu.com/sdk.php. The server returns a plaintext JSON object carrying location information as follows:

```
{"content":{"addr":{"detail":""}, "bldg":"",
 "floor":"", "point":{"y":"","x":""}, "radius":""},
 "result": {"error":,"time":""}}
```

By comparing the failed request against successful one, it is found that the key fields are the x and y coordinates in "point", "radius", the error code and the timestamp. The error code 161 indicates a *successful* query and the y and x carry the computed coordinates of the latitude and longitude. We utilize Iptables in our implementation to build a NAT that redirects all the requests originally sent to Baidu location server back to our embedded Jetty web server running by LocationFaker. LocationFaker will then construct a similar JSON object carrying fake locations to trick Baidu Location SDK to accept the received location as the real location.

4.2 Fetching Location Data Readings

The last component is location-dependent data fetching module, which retrieves the distance readings from APPs based on the fake location setting. The basic strategy is actually running the client and simulating the user's inputs to retrieve distance readings. To simulate user inputs, we adopt the MonkeyRunner library bundled with the Android SDK. With MonkeyRunner scripts provided in Jython, it simulates user inputs in apps to allow us automatically to perform various tests on apps. We integrate the API with our attacking framework to allow user defined inputs. We simulate consecutive operations in forms of touch, drag, scroll, input numbers, shell command and key press to mimic a user's behavior to the apps to trigger a location information update and scroll down the list to read out all items.

To read the distance from the apps, we have modified the TextView widget to dump text to log messages whenever a "setText" method call is made. We then retrieve text from the Adb logcat buffer and reads specific app's output by filtering log level, grepping by PID and tags then matching particular regular expression pattern.

5. REAL-WORLD EVALUATIONS

To evaluate the effectiveness of our strategy, we implement the real-world experiments by recruiting 30 volunteers for the 3 kinds of LBSN apps: Wechat, Skout and Momo. We evaluate the *Localization Accuracy* by comparing the distance between the user's *Real Locations* and *Inferred Locations*, and *Localization Efficiency* by measuring the latency of launching an attack for different apps. In the experiments of real-world tracking, we evaluate the effectiveness by measuring how many top locations could be recovered by using 3-week track.

5.1 Localization Accuracy and Efficiency

To well evaluate the localization accuracy, we set that the attack is triggered as soon as the user reports his real location obtained from location providers (e.g., GPS, Wi-Fi, or cell ID). The attack and real location reporting is set to the synchronous mode because we need to make sure that users' mobility will not impact the localization accuracy. To achieve this, we deploy a web server in which users with HTML5-capable browsers could retrieve their locations directly from location providers of their smart phones, and then submit their real location, user information to the server. The server will immediately put this request into its task queue and each idle node polls and claims a task and schedule an attack, the results of which will be reported to the server and compared with the exact location. Members of our groups regularly submit their locations to the server. We've collected in total of more than 350 location reports and attack results. The testing regions include United States, China and Japan.

5.1.1 Localization Accuracy

The evaluation on localization accuracy is shown in Fig 4. From Fig 4, it is observed that the majority of the results achieve a very high localization accuracy. For Momo, nearly 60% of the attacks can geo-locate a user at the accuracy of less than 20m and only less than 10% of the localization accuracy is more than 60m. In general, it could achieve an average localization accuracy of 25.8m for 119 evaluations. For Skout, though the minimum localization limit is 800m, most of the localization could achieve the accuracy of less than 60m while over 70% of the localization is less than 120m. The average localization accuracy could reach 129.4m for 156 tests, indicating the effectiveness of the Space Partition algorithm. For Wechat with minimum localization limit of 100m, we are able to geo-locate 50% of users in less than 40m. The average accuracy is 51.1m for 74 tests. Note that, different factors contribute to the localization errors such as inconsistent location providers / APIs, various location calculation algorithm / strategy or location cache policy.

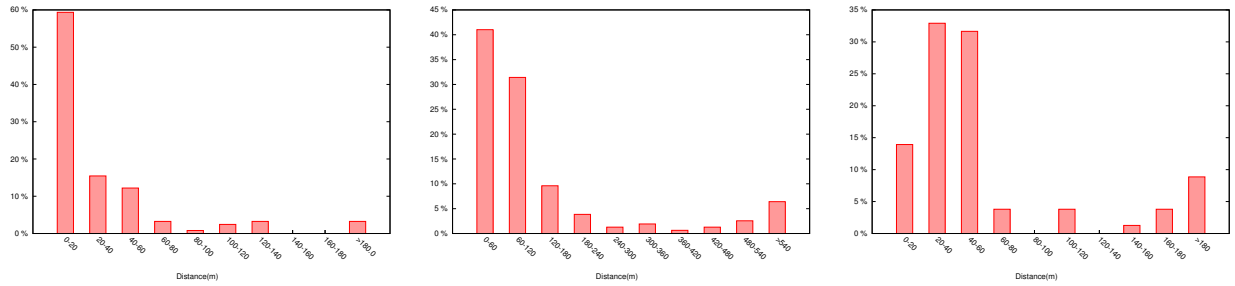
5.1.2 Localization Speed

We measure the efficiency with the average execution time of attacks. The results are shown in Fig 5a and Fig 5b, which correspond to the case of randomly setting first 3 anchor points and social popularity enhanced attacking approach.

From Fig 5a, it is shown that over 80% of the attacks for all 3 apps finish within $1200s$. It is important to point out that most of the time is spent on waiting for the app server's response. More specifically, each request for Wechat waits 40s to ensure that the user's location is fetched due to network latency and for Momo, the number increases to 55s while for Skout, it spends on 20s on queuing per query. In the evaluation, Momo has a faster localization speed as the iterative trilateration converges fast and hence requires fewer queries. From Fig 5b, it is shown that, after adding some side information such as social popularity index in Wechat or setting the initialization point in the approximate area (e.g., Shanghai) for Momo or Skout, the localization performance could be enhanced for 1.5 times.

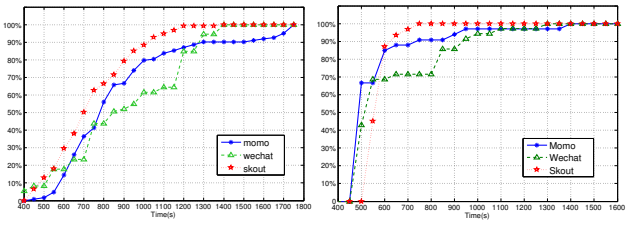
5.2 Real-world Tracking: Tracking Accuracy and Top Location Coverage

In this section, we evaluate the effectiveness in real-world tracking. The basic goal of this experiment is to compare the inferred



(a) Momo's Localization Accuracy (b) Skout's Localization Accuracy (c) Wechat's Localization Accuracy

Figure 4: Evaluation on Localization Accuracy



(a) Localization inference time on different apps (b) Improved localization inference time on different apps

Figure 5: Localization Efficiency of the Original and Enhanced Scheme

mobility traces of the mobile users with their real mobility trace to measure how much location information the attacker could obtain by tracking the users in a certain duration. In this phase, we recruit 30 volunteers from China, Japan and United States to participate in our three-week real-world experiments. The tracked Skout and Momo volunteers are scattered in all the three countries. For Wechat, due to the coverage limit, we fix a region of the size of 3km∗5km in Shanghai and 20km∗20km in Buffalo. In these three weeks, the volunteers use the LBSN apps in the same way as other typical LBSN users. To obtain the ground truth data (or user's real mobility traces), we develop an app based on Baidu Location API to record their locations every half an hour and submit the traces to the server. On the server side, we run 3 Momo tracking instances, 7 Wechat nodes and 3 Skout nodes to track Momo, Wechat and Skout users, respectively. We continuously track them for 3 weeks and collect 3395 inferred points in total.

5.2.1 *Tracking Accuracy*

In real-world tracking, synchronization of user real trace reporting and our tracking is almost infeasible due to unexpected user usage pattern as well as the randomness of the delay between victim's location updating and our tracking. Therefore, we also evaluate the tracking accuracy in the asynchronous mode. In particular, the user's real-world trace is periodically updated (e.g., 30 mins), and the tracking on users is also periodically launched (40 mins). In this case, we define the *Tracking Accuracy* as the distance of the inferred location and its closest counterpart of the reported user traces (ground truth data) in time domain. Such tracking accuracy provides the upper bound of the localization error.

The evaluation of tracking accuracy is shown in Fig 6. The experiment results demonstrate that the asynchronous tracking can also achieve a very high level of accuracy. As shown in Fig 6a, more than 80% of tracking results on Momo can geo-locate the victims in 40m, more than 90% of tracking results on Skout can break the distance limit of 800m to geo-locate the victims to $0-20$m and $80-100$m, and over half of the tracking on Wechat users can be located to the accuracy of less than 60m.

The factors which may potentially affect the tracking accuracy includes: the location providers (GPS, Wi-Fi, or Cell ID) varied in precision, cache policy defining how long the user's location is buffered at the server side. We've investigated the cache policy of Wechat by comparing the results from China and US, which have different user populations and thus different cache time. The results are indicated in Fig 7. In China, as one of the most popular LBSN apps, Wechat has a huge population of users, which makes the users' locations buffered at the server side for a shorter duration, making user tracking more difficult. It is much easier to track a Wechat user in the US due to smaller number of users and a much longer location cache time.

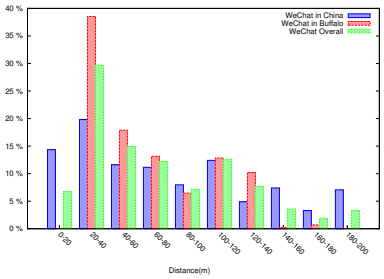


Figure 7: Wechat Accuracy Comparison

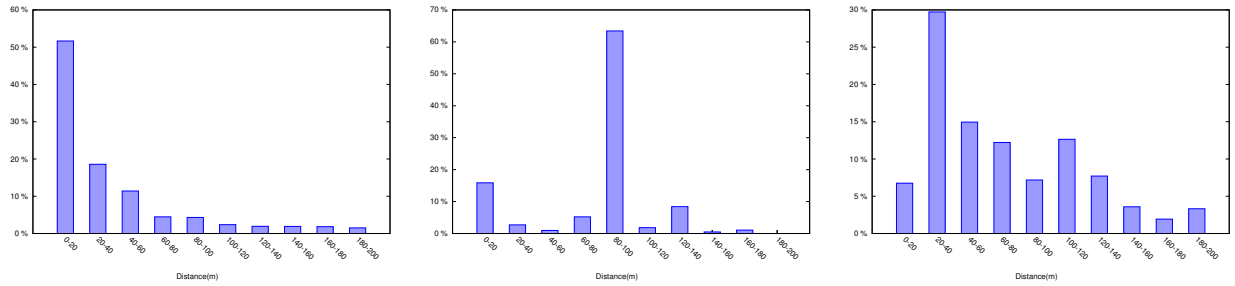
5.3 The Coverage Rate of Top N Location

According to [33], "Top N" locations refer to the locations that are most correlated to users' identities. For example, "top 2" locations likely correspond to home and work locations. In the section, we investigate how much location information the attacker could gain from tracking by introducing the concept of Top N location Coverage Rate, which is defined as follows. Given $\mathbb{G}$ as the set of reported traces (ground truth data) and $\mathbb{I}$ as the set of inferred traces, we define $Top_N()$ as the function that returns N most visited locations from a specific trace and thus define Top N Location Coverage rate as

$$TNR = \frac{|Top_N(\mathbb{G}) \cap Top_N(\mathbb{I})|}{N},$$

which refers to the percentage of locations that belongs to both of Top N locations in reported and inferred mobility traces.

Evaluation Results: Without loss of the generality, we set $N = 5$ and evaluate the top location coverage rate for three weeks, which



(a) Tracking Accuracy of Momo (b) Tracking Accuracy of Skout (c) Tracking Accuracy of WeChat

Figure 6: Evaluation Results on Tracking Accuracy

top location	one week			two weeks			three weeks		
	Momo	Wechat	Skout	Momo	Wechat	Skout	Momo	Wechat	Skout
1	92.3%	50.0%	20.0%	100.0%	57.1%	60.0%	100.0%	71.4%	60.0%
2	46.1%	21.4%	0.0%	46.1%	21.4%	40.0%	69.2%	21.4%	40.0%
3	30.7%	21.4%	20.0%	46.1%	28.5%	60.0%	38.4%	28.5%	80.0%
4	23.0%	35.7%	20.0%	30.7%	35.7%	40.0%	38.4%	35.7%	40.0%
5	23.0%	21.4%	0.0%	15.3%	21.4%	40.0%	15.3%	14.2%	40.0%

Table 2: Top 5 Location Coverage Result for 3 Weeks

is shown in Table 2. We choose the location match criteria in accordance with each app's accuracy limit, e.g., 100m for Wechat, and the cluster radius of the location region. The cluster radius of location region is the maximum radius for clustering a user's location coordinates in his real trace and it is set to 50m empirically (typical length of a building in the area where experiments are conducted). Thus the location matching precisions for all three apps are 50m (Momo), 100m (Wechat) and 0.5 mile (Skout) in our experiment. From Table 2, it is observed that Momo shows the best coverage rate. After three weeks tracking, we can obtain all the volunteers' top 1 locations and about 70% volunteers' top 2 locations. For Wechat, we could successfully infer 71.4% 21.4%, 28.5% of top $1, 2, 3$ locations after 3 week tracking. For Skout, 60.0%, 40.0%, 80.0% volunteers' top $1, 2, 3$ locations could be successfully recovered. Our evaluation results also show that the temporal factor plays an important role in Top N location recovery. In particular, the Top N location coverage rate will significantly increase along with more tracking days. In general, our attack shows a high Top 5 location coverage rate.

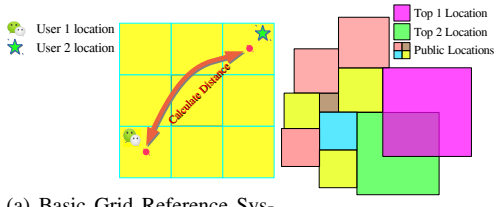
6. A USER-CENTRIC PRIVACY ENHANCEMENT FRAMEWORK

In this section, we aim to propose some suggestions to limit the attacker's capability. Certainly, effective location proof could prevent our attack, but the typical location proof techniques include using the deployed trusted infrastructures (e.g., cell tower or Wi-Fi access points) [25] or using environmental signals as the location tags [24, 34, 9]. However, they either require the presence of trusted infrastructure or are only effective in a small-scale (e.g., less than 100m) due to the spatial diversity of wireless signals. As a result, the existing location proof techniques are only feasible in a small-scale region and less practical in our scenario.

One potential approach of limiting the attacker's capability is that the service provider can compare users' location changes with their mobility patterns or behavior patterns to identify potential anomalous users (e.g, changing the locations too frequently or making too many queries within a short period). For example, from our experiment, it is observed that Wechat has put a limit on the number of queries issued at a certain duration (depending on the workload of the server) and the misbehaving account will be blocked for a specic period, which signicantly slows down attacking process. Our real-world experiments show that, though the attacker may use multiple accounts to speed up the attack, a more stringent limit on the number of queries will increase the difculty of launching the attacks since the attack should be nished between two consequent location updating events of the target.

We notice that Momo and Wechat provide an option to manually remove their locations from the public. However, with no idea about the potential risks brought by LBSN apps, few people do choose this option. This further signifies the importance of making the public more aware of the potential risk, which is one of major motivations of this paper.

Another possible approach for reducing the accuracy of the proposed attack is adding more noises to the location management module of LBSNs to achieve a better privacy protection at the cost of the decrease of users' utility. Here we present a user-centric privacy enhancement framework based on a global grid reference system and location classification to provide the tradeoff of the privacy and the utility.



(a) Basic Grid Reference System (b) Classified Grid Reference System

Figure 8: The Grid Reference System

Distance Obfuscation with Grid Reference System: We propose a distance obfuscation technique based on grid reference system, which aims to prevent the attacker from using LBSN as the location oracle to obtain the accurate location information. As shown in Fig 8a, the server maintains a grid reference system, where the

location of a mobile user can be expressed as the center of the grid cell that the user is located in. Therefore, the distance of two grid cells expresses the relative distance of two users defined as the minimum path connecting these two cells. The benefit of using grid reference system to express the relative distance of two users is that it obfuscates the real location of mobile users with the center of the cell and the attacker cannot obtain extra information of the target if the generated fake anchors are located at the same cell. It is noted that other advanced obfuscation techniques [5] or differential location privacy [1] could be applied to the proposed global grid reference system to provide the privacy enhancement for LBSNs.

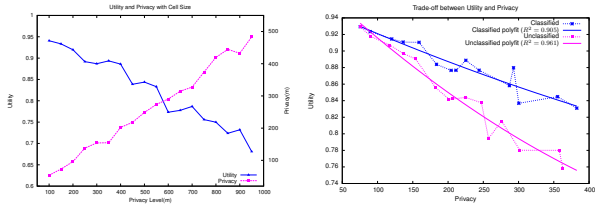
Privacy Definition vs Utility Metrics Similar to other obfuscation techniques, grid reference system will also decrease user utility. Considering the relative distance is the main metric of LBSN, we define the metric of privacy as:

$$\mathbf{Pri} = \mathbf{Dist}(\mathbf{L}_R, \mathbf{L}_O),$$

where $\mathbf{L}_R$ and $\mathbf{L}_O$ refer to the real and obfuscated location of the mobile user, respectively, and function $\mathbf{Dist}()$ returns the distance of two locations in Grid reference system. By given a specific anchor node at location $\mathbf{L}_A$, we further define the utility metric as

$$\mathbf{UT} = 1 - \frac{|\mathbf{DDist}(\mathbf{L}_R, \mathbf{L}_A) - \mathbf{DDist}(\mathbf{L}_O, \mathbf{L}_A)|}{\mathbf{Dist}_{max}},$$

where function $\mathbf{DDist}()$ returns the displayed distance in LBSN apps, $\mathbf{Dist}_{max}$ represents the maximum distance that the user could tolerate. It is obvious that, when the displayed distance between the real location and anchor point $\mathbf{DDist}(\mathbf{L}_R, \mathbf{L}_A)$ equals displayed distance between the obfuscated location and anchor point $\mathbf{DDist}(\mathbf{L}_O, \mathbf{L}_A)$, the utility achieves the maximum value 1. When $\mathbf{DDist}(\mathbf{L}_O, \mathbf{L}_A)$ is much larger or smaller than $\mathbf{DDist}(\mathbf{L}_R, \mathbf{L}_A)$ (their gap should not be larger than $\mathbf{Dist}_{max}$), the utility is close to 0. By assigning different values to the size of the cells, we could achieve different location privacy protection level as well as different utilities. We evaluate the effectiveness of location privacy protection and its impact on the utility by applying grid reference system to the data set collected from our real world experiments (ground truth data and inferred location data). Fig 9a shows the privacy gain and the utility under different settings of cell size. It is observed that the increase of privacy gain will lead to the decrease of the utility, and vice versa. We will discuss how to achieve the tradeoff of privacy and utility next.



(a) Relationship of Utility/Privacy with Cell Size (b) Comparison of Utility/Privacy Trade-offs

Figure 9: Evaluation on Countermeasure

Achieving Privacy and Utility Trade-off via Classification of Users' Locations: In the previous section, we have shown that the obfuscation techniques decrease the utility. To achieve the tradeoff between the privacy and utility, we introduce a novel user controllable location privacy protection scheme. The proposed scheme is motivated from the observation that the user has different location privacy protection preference for locations. For example, a mobile user cares more about their Top 2 location privacy (e.g., home, work place) but is willing to share locations in public regions (e.g., cafe or bars). Therefore, in a user controllable location privacy protection solution, the mobile users can classify the locations into several categories, which correspond to different privacy protection requirements with different obfuscation parameters. During the subsequent LBSN usage process, users record their location profiles ranked with their visiting frequency and could be dynamically updated along with usage. With such a location profile with different ranking, the most frequently visited locations are given more privacy protection and thus suffer from a lower utility while the less frequently visited locations could enjoy more utility with less privacy protection as indicated in Fig 8b. To implement our idea, we transform the original grid reference system of the uniform cell size to the non-uniform grid reference system, in which top locations cover a larger area while public regions cover a smaller area. Note that the proposed location classification concept could also be applied to other existing obfuscation techniques [5]. To evaluate the proposed solution, we compare the uniform grid reference system with non-uniform grid reference system based on the data set collected from our real-world experiments. In the uniform grid reference system, we tune the cell size from 200m to 1000m, which correspond to the privacy level from 50 to 400. In the non-uniform grid reference system, we fix the cell size of top locations to 1000m to provide highest privacy protection level while tune the cell size of normal location from 200m to 1000m. It is observed that the non-uniform grid reference system based on location classification has a significant advantage in privacy/utility trade-off over the uniform grid reference system as shown in Fig 9b.

7. RELATED WORK

Location Privacy Protection in location-based services is a long-standing topic and has received a lot of attentions in the last decades. The most popular approach to achieve location privacy in LBS is utilizing the obfuscation techniques to coarse the spatial or temporal granularity of the users' real locations [17, 18]. A different approach to hide the users' location is based on mix zones [13]. The third approach is to protect location privacy by adding dummy requests issued by fake location and indistinguishable from real requests [31]. A recent work [28] proposes a game-theoretic framework that enables a designer to find the optimal LPPM for a given location-based service, ensuring a satisfactory service quality for the user. Different from the location privacy issues considered in previous works, providing the relative distance is the key functionality of LBSN apps while the obfuscation will inevitably reduce the utility of LBSNs. Achieving the tradeoff between the location privacy and the utility is of the highest priority. The proposed users' location classification based approach could help to reduce the impact of obfuscation techniques on users' utility, and thus can be a compliment to various obfuscation techniques.

There are many other works on inferring the victim's trajectory and further re-identify other private information [15, 33, 30, 12]. Our work is different from the existing work in that we propose a novel attack approach that allows anyone to perform an involuntary tracking towards any specific target to collect traces for user re-identification.

8. CONCLUSION

LBSN is becoming extremely popular recently. However, most LBSN users are unaware of the location privacy leakage. We target 3 most popular LBSN apps and develop a novel automatic tracking system, which could achieve range-free, accurate, and involuntary tracking towards the target only using the public information. Our

real-world attack experiments show that the attack achieves high localization accuracy and the attacker can recover the users' top 5 locations with high possibility. We've discussed various mechanisms to mitigate such threats and analyzed the privacy and utility trade-off. Our study is expected to urge LBSN service providers to revise their location privacy protection techniques and more importantly, serve as a call for more attentions from the public to have the full knowledge of the potential risks brought by LBSN apps. Our mitigation suggestions will provide a guideline for future revisions of these LBSN apps.

9. ACKNOWLEDGEMENT

This work is supported by the National Natural Science Foundation of China (Grant 61272444) and the US National Science Foundation under grant no. CNS-1318948 and CNS-1262275.

10. REFERENCES

[1] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi. Geo-indistinguishability: Differential privacy for location-based systems. In *CCS'13*, 2013.

[2] Apk Apps. Skout 3.4.3 Apk – Meet, Chat, Friend. http://www.apk4.net/applications/social-applications/skout-3-4-3-apk-meet-chat-friend/.

[3] Apple. Location Awareness Programming Guide. http://developer.apple.com/library/ios/#documentation/userexperience/conceptual/LocationAwarenessPG/CoreLocation/CoreLocation.html.

[4] C. A. Ardagna, M. Cremonini, E. Damiani, S. D. C. di Vimercati, and P. Samarati. Location privacy protection through obfuscation-based techniques. In *Data and Applications Security XXI*, pages 47–60. Springer, 2007.

[5] C. A. Ardagna, M. Cremonini, S. De Capitani di Vimercati, and P. Samarati. An obfuscation-based approach for protecting location privacy. *Dependable and Secure Computing, IEEE Transactions on*, 8(1):13–27, 2011.

[6] Baidu. FAQ - Geolocation. http://developer.baidu.com/map/geosdk-android-qa.htm.

[7] A. R. Beresford and F. Stajano. Location privacy in pervasive computing. *Pervasive Computing, IEEE*, 2(1):46–55, 2003.

[8] L. Bindschaedler, M. Jadliwala, I. Bilogrevic, I. Aad, P. Ginzboorg, V. Niemi, and J.-P. Hubaux. Track me if you can: on the effectiveness of context-based identifier changes in deployed mobile networks. In *NDSS'12*, 2012.

[9] J. Brassil, P. Manadhata, and R. Netravali. Traffic signature-based mobile device location authentication. *IEEE Transactions on Mobile Computing*, 2013.

[10] CNET. How Skout attracted millions with its flirty ways. http://news.cnet.com/8301-1035_3-57407367-94/how-skout-attracted-millions-with-its-flirty-ways/.

[11] Craig Smith. How Many People Use the Top Social Media, Apps & Services? http://expandedramblings.com/index.php/resource-how-many-people-use-the-top-social-media/.

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

[13] J. Freudiger, R. Shokri, and J.-P. Hubaux. On the optimal placement of mix zones. In *Privacy Enhancing Technologies*. Springer, 2009.

[14] B. Gedik and L. Liu. Location privacy in mobile systems: A personalized anonymization model. In *ICDCS'05*. IEEE, 2005.

[15] P. Golle and K. Partridge. On the anonymity of home/work location pairs. In *Pervasive Computing*, pages 390–397. Springer, 2009.

[16] Google. Location Source and Accuracy. http://support.google.com/gmm/bin/ans-wer.py?answer=81873.

[17] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *MobiSys'03*. ACM, 2003.

[18] B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady. Preserving privacy in gps traces via uncertainty-aware path cloaking. In *CCS'07*. ACM, 2007.

[19] M. Iasonas, F. George, A. Sotiris, and S. Zanero. All your face are belong to us: Breaking facebook's social authentication. In *ACSAC'12*, 2012.

[20] J. Liu, Y. Zhang, and F. Zhao. Robust distributed node localization with error management. In *MobiHoc'06*. ACM, 2006.

[21] C. Y. Ma, D. K. Yau, N. K. Yip, and N. S. Rao. Privacy vulnerability of published anonymous mobility traces. In *MobiCom'10*. ACM, 2010.

[22] Manila Standard Today. Whats APP? WeChat. http://manilastandardtoday.com/2013/04/06/whats-app/.

[23] J. Meyerowitz and R. Roy Choudhury. Hiding stars with fireworks: location privacy through camouflage. In *MobiCom'09*. ACM, 2009.

[24] A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, and D. Boneh. Location privacy via private proximity testing. In *NDSS'11*, 2011.

[25] S. Saroiu and A. Wolman. Enabling new mobile applications with location proofs. In *MobiSys'09*. ACM, 2009.

[26] B. Schilit, J. Hong, and M. Gruteser. Wireless location privacy protection. *Computer*, 36(12):135–137, 2003.

[27] R. Shokri, G. Theodorakopoulos, J. Le Boudec, and J. Hubaux. Quantifying location privacy. In *Security and Privacy 2011*. IEEE, 2011.

[28] R. Shokri, G. Theodorakopoulos, C. Troncoso, J.-P. Hubaux, and J.-Y. Le Boudec. Protecting location privacy: Optimal strategy against localization attacks. In *CCS'12*. ACM, 2012.

[29] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási. Limits of predictability in human mobility. *Science*, 327(5968):1018–1021, 2010.

[30] M. Srivatsa and M. Hicks. Deanonymizing mobility traces: Using social network as a side-channel. In *CCS'12*. ACM, 2012.

[31] T. Xu and Y. Cai. Feeling-based location privacy protection for location-based services. In *CCS'09*. ACM, 2009.

[32] P. A. Zandbergen. Accuracy of iphone locations: A comparison of assisted gps, wifi and cellular positioning. *Transactions in GIS*, 13(s1):5–25, 2009.

[33] H. Zang and J. Bolot. Anonymization of location data does not work: A large-scale measurement study. In *MobiCom'11*. ACM, 2011.

[34] Y. Zheng, M. Li, W. Lou, and Y. T. Hou. Sharp: Private proximity test and secure handshake with cheat-proof location tags. In *ESORICS'12*, 2012.

[35] Zhu Feng. Momo Announced Number of Users Exceeded 30 Million, Active 450 Million. http://tech.ifeng.com/mi/detail_2013_03/13/23047454_0.shtml.